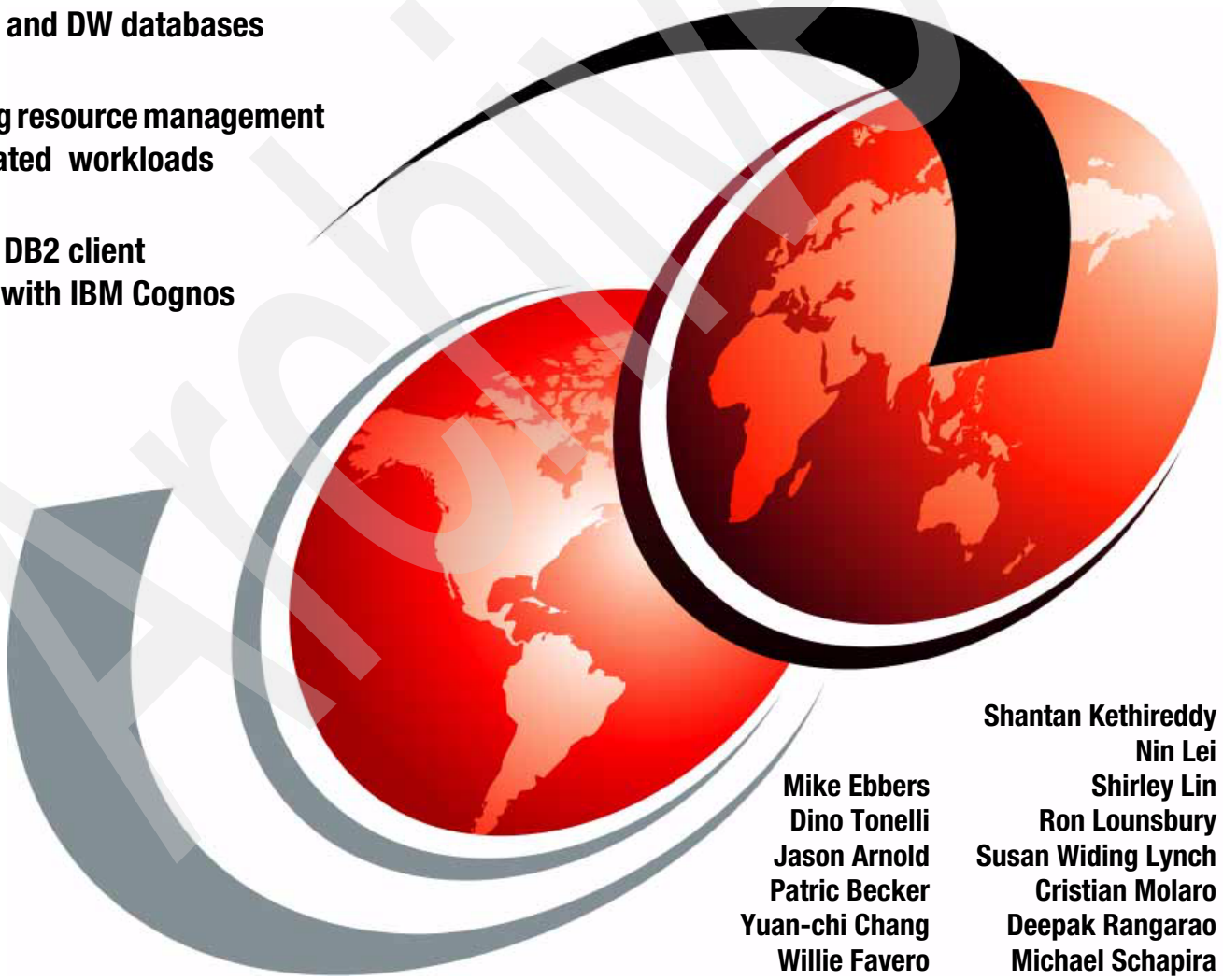
Information Management

IBM

Co-locating Transactional and Data Warehouse Workloads on System z

- Reaping the benefits of data-sharing your OLTP and DW databases
- Facilitating resource management for co-located workloads
- Exploiting DB2 client attributes with IBM Cognos



Mike Ebbers
Dino Tonelli
Jason Arnold
Patric Becker
Yuan-chi Chang
Willie Favero
Shantan Kethireddy
Nin Lei
Shirley Lin
Ron Lounsbury
Susan Widing Lynch
Cristian Molaro
Deepak Rangarao
Michael Schapira

Redbooks

ibm.com/redbooks

Archive

Archived

International Technical Support Organization

Co-locating Transactional and Data Warehouse Workloads on System z

November 2010

Archived

SG24-7726-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

Archived

First Edition (November 2010)

This edition applies to Version 1 Release 10 of IBM z/OS, DB2 9 for z/OS, and WebSphere Application Server for z/OS Version 7.0.

© Copyright International Business Machines Corporation 2010. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices xi
Trademarks xii

Preface xiii
The team that wrote this book xiv
Now you can become a published author, too! xvi
Comments welcome xvii
Stay connected to IBM Redbooks xvii

Part 1. Introducing the co-location project 1

Chapter 1. Executive summary and highlights 3
1.1 The project 4
1.2 The environment 4
1.3 Experiment results 5
1.4 Conclusions 6
1.5 The highlights 6

Chapter 2. Why System z for data warehousing 9
2.1 Architecture 10
2.1.1 z/Architecture and the z/OS operating system 10
2.2 DB2 for z/OS 11
2.2.1 DB2's effect on your warehouse 11
2.2.2 How DB2 for z/OS Version 8 helped 13
2.2.3 How DB2 9 for z/OS helped 14
2.3 Resource management 19
2.3.1 Logical partitions 19
2.3.2 z/OS Workload Manager 20
2.3.3 z/VM - System z Virtual Machine 20
2.4 Unique I/O capabilities 21
2.5 Security 24
2.6 The specialty engines available on System z 26
2.6.1 General purpose CPs 27
2.6.2 System Assist Processor 27
2.6.3 Integrated Coupling Facility 27
2.6.4 Integrated Facility for Linux 28
2.6.5 System z Application Assist Processor 28
2.6.6 System z Integrated Information Processor 29
2.6.7 zAAP on zIIP capabilities 31
2.7 IBM Smart Analytics Optimizer 31
2.8 Bandwidth resources and co-location 38
2.9 Continuous availability and disaster recovery 39
2.10 Power consumption 40
2.11 Storage management (SMS) 41
2.12 Data auditability and compliance 42
2.13 Risk avoidance and data consolidation 42
2.14 Managed reporting 43

Chapter 3. Co-location topologies 45

© Copyright IBM Corp. 2010. All rights reserved.

3.1 Configuration options 46
3.1.1 Single LPAR, OLTP, and DW share single DB2 subsystem 47
3.1.2 Single LPAR, separate OLTP and DB2 DW subsystems 48
3.1.3 Multi-LPAR - OLTP LPAR and DW LPAR 48
3.1.4 Single LPAR - two-member data sharing 49
3.1.5 Multi-LPAR - two-member data sharing 50
3.1.6 Multi-LPAR - OLTP LPARs and OLTP and DW LPAR 51
3.1.7 Multi-LPAR - data sharing with OLTP and DW LPARs 52
3.1.8 Multi-LPAR - data sharing with multiple OTLP LPARs and multiple DW LPARs . 53
3.1.9 Standalone data warehouse DB2 subsystem 54
3.1.10 Front-end topologies 54
3.2 Floating DW and DataStage with BatchPipes 54

Part 2. Project environment 57

Chapter 4. Co-location data models 59
4.1 Overview 60
4.2 The transactional data model 60
4.2.1 Logical schema 61
4.2.2 Physical schema 61
4.3 The data mart data model 63
4.3.1 Physical schema 63
4.3.2 Query speedup through Materialized Query Table (MQT) 64
4.4 Data refresh to build inventory history 65
4.5 Data modeling using IBM Industry Models 66

Chapter 5. BookStore workload 1 - transactional 67
5.1 Online transaction processing workload 68
5.1.1 Components and architecture 68
5.2 BookStore transactions 71
5.3 WebSphere WLM setup 72
5.3.1 WLM classification of BookStore transactions 72
5.3.2 Verifying the WebSphere WLM classification implementation 74
5.3.3 Monitor the WebSphere WLM work queues 75
5.4 WebSphere Application Server setup topology 76
5.4.1 Distributing incoming work to application servers 77
5.5 WebSphere environment variables 79

Chapter 6. BookStore workload 2 - IBM Cognos BI reporting 81
6.1 Business Intelligence with IBM Cognos BI 82
6.1.1 IBM Cognos BI and Linux on System z 84
6.1.2 IBM Cognos BI software components 85
6.2 System environment and configuration options 87
6.2.1 Product components 87
6.2.2 Topology and setup 88
6.2.3 Tuning IBM Cognos BI gateway 92
6.2.4 Tuning IBM Cognos BI report service 93
6.2.5 Tuning WebSphere Application Server 94
6.2.6 Tuning DB2 for Linux on System z 97
6.2.7 General Linux OS tuning 98
6.2.8 z/VM tuning 98
6.2.9 IBM Cognos BI and DB2 client information 99
6.3 Reporting with IBM Cognos BI 100
6.3.1 Building IBM Cognos BI data models 100

6.3.2 Designing reports with Report Studio 102
6.3.3 Tuning reports using IBM Optim Query Tuner 106
6.3.4 Operational BI reporting 110
6.4 Performance and load testing using Rational Performance Tester 112
6.4.1 Rational Performance Tester - introduction 113

Chapter 7. BookStore workload 3 - data warehouse refreshing 121
7.1 Overview of data refreshing 122
7.2 Batch refresh with InfoSphere Warehouse SQW 123
7.2.1 InfoSphere Warehouse components 123
7.2.2 Physical data modeling using Design Studio 124
7.2.3 InfoSphere Warehouse refresh scenario in a non-data sharing environment . . . 140
7.2.4 InfoSphere Warehouse refresh scenario in a data sharing environment 142
7.2.5 Other possible refresh scenarios specific to data sharing environments 145
7.3 Overview of InfoSphere Change Data Capture 146
7.3.1 Introduction to InfoSphere Change Data Capture 146
7.3.2 Configuration of InfoSphere Change Data Capture for z/OS 149
7.3.3 Tuning InfoSphere Change Data Capture for z/OS 166
7.4 Using InfoSphere Change Data Capture for z/OS 184
7.4.1 Building replication definitions 184
7.4.2 Operations and monitoring 198
7.4.3 InfoSphere CDC as a DB2 workload 204

Chapter 8. Our test configuration and infrastructure 207
8.1 Hardware and software 208
8.1.1 Hardware configuration 210
8.1.2 Software configuration 211
8.2 IBM Cognos BI for Linux on System z 213
8.2.1 Our test setup 214
8.2.2 Move content store to z/OS 219
8.2.3 WebSphere for Linux on z for IBM Cognos BI 221

Part 3. Implementation considerations 225

Chapter 9. Performance monitoring and management of queries overview 227
9.1 Overview 228
9.2 Performance management 228
9.2.1 Defining performance 228
9.2.2 Performance metrics 229
9.3 Monitoring facilities and tools 233
9.3.1 Primary tools 233
9.4 Resource management of DB2 data warehouse queries 238
9.4.1 Steps for resource management 239
9.4.2 Distributed query flow 243

Chapter 10. Considerations for data warehousing with DB2 on System z 247
10.1 Overview 248
10.2 Parallelism 248
10.3 Co-location data-sharing 250
10.3.1 ETL 251
10.3.2 Accessing OLTP and data warehouse data with a single query 251
10.3.3 Global buffer pools 252
10.3.4 Lock structure 253
10.4 DB2 work files 253

10.5 Buffer pools 254
10.5.1 Buffer pool page fixing 255
10.6 Coupling Facility 256
10.7 DSNZPARM recommendations 256
10.7.1 Highly recommended parameters 257
10.7.2 Recommended parameters 257
10.7.3 Buffer pool setting recommendations 258
10.7.4 DBM1 virtual storage consumption below the 2 GB bar 258
10.8 Automatic identification of latch contention and DBM1 below-the-bar virtual storage 259

Chapter 11. Resource management of data warehouse mixed workloads 263
11.1 Introduction 264
11.2 WLM service definition with data warehouse workloads 264
11.2.1 Determining the warehouse workloads, sources, users, and relative importance . 265
11.2.2 Determining WLM workloads 267
11.2.3 Determining WLM service classes 268
11.2.4 Determining data warehouse service class goals and period durations 271
11.2.5 Sample data warehouse query service classes 272
11.2.6 Classification 272
11.2.7 Report classes 275
11.2.8 Protecting work 276
11.2.9 Multiple policies 277
11.2.10 Service class relative importance 278
11.3 Analysis and verification of a working WLM policy 279
11.3.1 Resource Management Facility (RMF) Workload Activity report 279
11.3.2 Resource Management Facility (RMF) Monitor III 281
11.3.3 Resource Management Facility (RMF) spreadsheet reporter 282
11.3.4 Recommendations for maintaining the WLM service definition 284
11.3.5 Process for review of WLM service definitions 285
11.4 Additional resource management considerations 287
11.4.1 zIIP specialty engines 287
11.4.2 zAAP on zIIP 288
11.4.3 LPAR group capacity limit 288
11.4.4 Blocked workload support 289

Chapter 12. DB2 data sharing workload balancing and fault tolerant configuration 291
12.1 Introduction 292
12.2 Configuration 293
12.3 Data sharing subgroup attach 295
12.4 Dynamic VIPA and Distributed DVIPA (Sysplex Distributor) 299
12.5 Stress tests and workload distribution 302
12.5.1 Failover example 308
12.5.2 Installation of Data Server Runtime Client 311
12.5.3 Cataloging the data warehouse subsystem into the Data Server Client 313
12.5.4 The db2dsdriver configuration file 315
12.5.5 How to verify that the Client WLB is working 316

Chapter 13. Utilizing DB2 client information for resource management and monitoring 321
13.1 DB2 client information 322
13.1.1 Using JDBC Universal Driver methods 323
13.1.2 Using the WLM_SET_CLIENT_INFO stored procedure 324
13.2 Mapping of special registers to DB2 and WLM 326

13.3 WLM_SET_CLIENT_INFO in IBM Cognos BI 326
13.4 Leveraging WLM set client information in IBM Cognos BI 329
13.4.1 Background and objective 329
13.4.2 IBM Cognos BI command blocks to call WLM set client information 330
13.4.3 WLM classification using DB2 client information 331
13.5 Monitor views of work requests using DB2 client information 332

Chapter 14. I/O considerations 337
14.1 Introduction 338
14.2 DB2 workloads 338
14.3 IBM DS8000 storage subsystem 339
14.3.1 Introduction to the IBM DS8000 series features and functions 339
14.3.2 Performance for System z 340
14.3.3 Considerations for mixed workloads 341

Part 4. Project experiment results 345

Chapter 15. Single z/OS LPAR topology experiments 347
15.1 Introduction 348
15.2 WLM implementation for experiments 348
15.3 Measurement and data collection methodology 353
15.4 Key performance metrics 354
15.4.1 BookStore data warehouse DB2 query profile 356
15.4.2 Key configuration settings 357
15.5 Sophisticated management with DB2 client attributes 357
15.5.1 Objectives 357
15.5.2 Results 357
15.6 Spike in transactional workload 359
15.6.1 Objective 359
15.6.2 Results 359
15.6.3 Observations and conclusions 359
15.7 Spike in data warehouse workload 360
15.7.1 Objective 360
15.7.2 Results 360
15.7.3 Observations and conclusions 361
15.8 Spike in data warehouse workload - protect transactional 362
15.8.1 Objective 362
15.8.2 Results 362
15.8.3 Observations 363
15.9 zIIP-eligible work to only run on zIIP processors 364
15.9.1 Objective 364
15.9.2 Preparation 364
15.9.3 Results 365
15.9.4 Observations and conclusions 365
15.10 Dynamic WLM management of SQW refresh 366
15.11 Summary 367

Chapter 16. Multiple z/OS LPAR experiments 369
16.1 Introduction 370
16.2 Topology 370
16.3 Key performance metrics 371
16.3.1 TPS 371
16.3.2 Queries per minute (QPM) 371
16.3.3 Response time 371

Archived

16.3.4 Processor utilizations by workloads . . . 372
16.3.5 Total processor utilization . . . 372
16.4 Measurements . . . 372
16.4.1 OLTP only . . . 372
16.4.2 OLTP with data warehouse . . . 374
16.4.3 Spike OLTP . . . 378
16.5 Summary . . . 383

Part 5. Appendixes . . . 385

Appendix A. InfoSphere Change Data Capture DDL and binds . . . 387

Appendix B. Additional performance data . . . 413
B.1 Linux monitoring . . . 414
B.2 Linux performance results . . . 417

Appendix C. Using DB2 accounting data for setting WLM period durations . . . 421
C.1 Sizing SC periods using DB2 accounting information . . . 422

Appendix D. Sample WLM service definition . . . 433
Sample service definition . . . 434
D.1 Service policies . . . 434
D.1.1 Service Policy CL_OPTML - Optimal classification for management . . . 434
D.1.2 Service Policy CL_POWR - Optimal policy for power users . . . 439
D.1.3 Service Policy CL_RFRSH - Policy to favor SQW refresh . . . 444
D.1.4 Service Policy CL_WCRIT - WASTCHI is Critical CPU = YES . . . 450
D.2 Classification by subsystem type . . . 455
D.2.1 Subsystem Type ASCH - Use Modify to enter *your* rules . . . 455
D.2.2 Subsystem Type CB - Rules for WebSphere Application Server (transaction only) . . . 455
D.2.3 Subsystem Type CICS - Rules for CTS (transaction only) . . . 456
D.2.4 Subsystem Type DB2 - Use Modify to enter *your* rules . . . 456
D.2.5 Subsystem Type DDF - Rules for DDF (transaction only) . . . 456
D.2.6 Subsystem Type EWLM - Use Modify to enter *your* rules . . . 457
D.2.7 Subsystem Type IMS - Rules for IMS (transaction only) . . . 458
D.2.8 Subsystem Type IWEB - Use Modify to enter *your* rules . . . 458
D.2.9 Subsystem Type JES - Rules for batch jobs . . . 458
D.2.10 Subsystem Type MQ - Use Modify to enter *your* rules . . . 458
D.2.11 Subsystem Type OMVS - Rules for forked OMVS work . . . 459
D.2.12 Subsystem Type SOM - Use Modify to enter *your* rules . . . 459
D.2.13 Subsystem Type STC - Rules for started tasks . . . 459
D.2.14 Subsystem Type TCP - Rules for TCP (transaction only) . . . 462
D.2.15 Subsystem Type TSO - Rules for TSO usage . . . 462
D.3 Report classes . . . 462
D.4 Application environmentsl . . . 464

Appendix E. Setting up the transactional workload . . . 469
E.1 Build BookStore EJB mapping . . . 470
E.2 Application deployment . . . 475
E.3 JDBC configuration . . . 477
E.3.1 JDBC connection types . . . 477
E.3.2 DB2_jcc property file . . . 479
WebSphere environment variable for JDBC . . . 479

Define J2C authentication user ID 479
Configure JDBC data source 480
WebSphere Application Server setup topology 483
Distributing incoming work to application servers 484
WebSphere clusters 487
Issue when running two application servers 488
16.5.1 WebSphere administrative command examples 489

Appendix F. WLM refresher 491
F.1 How WLM works 492
F.1.1 WLM components 492
F.1.2 The performance index 497
F.2 Considerations for WLM and DB2 499

Appendix G. Additional material 505
Locating the Web material 505
Using the Web material 505
How to use the web material 505

Related publications 507
IBM Redbooks 507
Other publications 507
Online resources 508
How to get Redbooks 508
Help from IBM 508

Index 509

Archived

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2010. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX 5L™
AIX®
BatchPipes®
CICS®
Cognos®
DataStage®
DB2 Connect™
DB2®
Distributed Relational Database Architecture™
DRDA®
DS8000®
FICON®
FlashCopy®
GDPS®
Geographically Dispersed Parallel Sysplex™
HiperSockets™
IBM®
IMS™
InfoSphere™
Language Environment®
LiveAudit™
MVS™
OMEGAMON®
Optim™
Parallel Sysplex®
Passport Advantage®
POWER5+™
PowerPlay®
POWER®
PR/SM™
Processor Resource/Systems Manager™
QMF™
RACF®
Rational®
Redbooks®
Redpaper™
Redbooks (logo) ®
Resource Measurement Facility™
RMF™
Solid®
System i®
System p®
System Storage®
System x®
System z10®
System z9®
System z®
Tivoli®
VTAM®
WebSphere®
z/Architecture®
z/OS®
z/VM®
z/VSE™
z10™
z9®
zSeries®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Archive

Preface

As business cycles speed up, many clients gain significant competitive advantage from quicker and more accurate business decision-making by using real data. For many, choosing the path to co-locate their transactional and analytical workloads on System z better leverages their existing investment in hardware, software, and skills. We created a project to address a number of best practice questions on how to manage these newer, analytical type workloads, especially when co-located with traditional transactional workloads.

The goal of this IBM Redbooks publication is to provide technical guidance and performance trade-offs associated with resource management and potentially DB2 data-sharing in a variety of mixed transactional and data warehouse System z topologies. The term co-location used here is specifically defined as the practice of housing both transactional (OLTP) and data warehouse (analytical) workloads within the same System z configuration. We also assumed that key portions of the transactional and data warehouse databases would reside on DB2 for z/OS. The databases may or may not reside in a DB2 data-sharing environment; we discuss pros and cons in this book.

The intended audience includes DB2 data warehouse architects and practitioners who are facing choices in resource management and system topologies in the data warehouse arena. This specifically includes Business Intelligence (BI) administrators, DB2 database administrators (DBAs) and z/OS performance administrators and systems programmers. In addition, decision makers and architects can utilize this book to assist in making platform and database topology decisions. The book is divided into four parts.

- Part I, “Introducing the co-location project” covers the System z value proposition and why one should consider System z as the central platform for their data warehousing and business analytics needs. Some topics are risk avoidance via data consolidation, continuous availability, simplified disaster recovery, IBM Smart Analytics Optimizer, reduced network bandwidth requirements, and the unique virtualization and resource management capabilities of System z LPAR, z/VM® and WLM. Part I also provides some of the common System z co-location topologies along with an explanation of the general pros and cons of each. This would be useful input for an architect to understand where a customer is today and where they might consider moving to.
- Part II, “Project environment” covers the environment, products, workloads, workload drivers, and data models implemented for this study. The environment consisted of a logically partitioned z10™ 32way, running z/VM, Linux®, and z/OS operating system instances. On those instances we ran products such as z/OS DB2 V9, IBM Cognos® Business Intelligence Version 8.4 for Linux on System z, InfoSphere™ Warehouse for System z, InfoSphere Change Data Capture, z/OS WebSphere® V7, Tivoli® Omegamon for DB2 Performance expert. Utilizing these products we created transactional (OLTP), data warehouse query, and data warehouse refresh workloads. All the workloads were based on an existing web-based transactional BookStore workload, that's currently utilized for internal testing within the System p® and z labs. While some IBM Cognos BI and ISWz product usage and experiences information is covered in this book, we do not go into the depth typically found in IBM Redbooks publications, since there’s another book focused specifically on that. One exception to this is the InfoSphere Change Data Capture product, in which we did include some step-by-step implementation details, as this information was less readily available at the time of this project.
- Part III, “Implementation considerations” is the core of the book and covers the resource allocation, management and monitoring co-location implementation considerations for z/OS and DB2 for data warehousing. This includes both single z/OS system

© Copyright IBM Corp. 2010. All rights reserved.

implementation as well as DB2 data-sharing between the transactional and data warehouse DB2s. It starts out with an overview to help bridge perspectives of the various administrators. It then covers DB2, WLM, and I/O resource considerations, then provides guidance on bridging the DB2 and WLM views of resource usage. Finally, it provides experimental data covering several resource management facets in two of the key co-location topologies (Single LPAR / separate DB2 sub-systems, Multi-LPAR DB2 data-sharing).

- Part IV, “Project experiment results” describes the results of our experiments and provides guidance for others to be able to co-locate their own workloads in a System z environment.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Mike Ebbers is a Consulting IT Specialist and Project Leader at the International Technical Support Organization, Poughkeepsie Center. He has worked with IBM mainframe hardware and software products since 1974 in the field, in education, and in the ITSO.

Dino Tonelli is a senior software analyst working in IBM Poughkeepsie. He has over 15 years experience with System z performance working on WLM, DB2, data warehousing, and Parallel Sysplex®. Dino's current focus is performance of new System z solutions. He holds an MS degree in Computer Science from Polytechnic University.

Jason Arnold is an IBM IT Specialist for information, integration, and data warehousing on System z in the United States. He has seven years of experience with System z, including three years of experience with data warehousing and Business Intelligence. His areas of interest include integration of legacy data into System z data warehouses and data synchronization. Jason holds Bachelor of Science degrees in Computer Science and Mathematics from Northern Illinois University.

Patric Becker is a Software Architect in the Data Warehousing on System z Center of Excellence at IBM Boeblingen Lab. The team conducts Proof of Concepts for large and complex DWH implementations and supports customers in all areas of DWH topics on System z. Before joining IBM, Patric worked for one of the largest DB2 for z/OS customers in Europe. He has over 12 years experience with DB2 for z/OS. In the past, Patric has been responsible for developing several high availability DB2 and IMS™ applications. He is also co-author of these IBM Redbooks publications: *DB2 for z/OS Using Large Objects*, *DB2 UDB for z/OS: Application Design for High Performance and Availability*, and *LOBs with DB2 for z/OS: Stronger and Faster*.

Yuan-Chi Chang is a Research Staff Member and Manager at IBM Thomas J. Watson Research Center. He has 10 years of industrial experience in data management. He has performed research and development work in the areas of digital library, electronic commerce, data warehouse and most recently data integration using InfoSphere DataStage®. He published over a dozen papers and has over 20 patents issued worldwide. Yuan-Chi joined IBM in 1999 after receiving his Ph.D. from the University of California, Berkeley.

Willie Favero is an IBM Senior Certified IT Software Specialist and DB2 SME for IBM’s Silicon Valley Lab Data Warehouse on System z Swat Team. He has over 30 years of experience working with databases with more than 25 years of that working with DB2. He is a

sought-after international speaker for conferences, user groups, and seminars, publishes articles, white papers, and Redbooks, and has one of the top technical blogs on the Internet.

Shantan Kethireddy joined IBM in 2001 after earning his Master of Science degree from the University of Iowa in Electrical and Computer Engineering. Shantan is a DB2 for System z Field Technical Sales Specialist (FTSS) supporting information integration, data warehousing, and Business Intelligence products and solutions. Prior to his current position, he spent six years working in DB2 development with a focus on SQL query optimization and data warehousing. In 2009, Shantan was named an IBM Master Inventor. Shantan can be reached at shantank@us.ibm.com.

Nin Lei is a recognized expert in DB2 performance and database design with interests in Very Large Data Bases (VLDB). He is a frequent speaker at conferences presenting data warehouse topics. He is a Distinguished Engineer at the World Wide Client Centers conducting customer studies. He drives performance analysis work for major projects, and provides architectural guidance to customers for designing their applications to meet high performance and scalability requirements. He also advises customers in new technology exploitation issues in the areas of System z and database performance. He is a member of the IBM IT Specialist Certification Board.

Shirley Lin has been working at IBM as a software engineer for 25 years. Early in her career, she worked as developer on products such as LAN Server, DCE, and LDAP design. In recent years, Shirley has worked on System z software performance test projects, including configuring and tuning WebSphere, DB2, CICS®, and the latest hardware.

Ron Lounsbury is a Software Performance Analyst for IBM. After graduating from RPI with an MS in Computer Science in 1982, Ron joined IBM to work on hardware and software development projects such as System z hardware microcode and WebSphere for z/OS.

Susan Widing Lynch is a Senior Software Engineer working in the z/OS performance organization in Poughkeepsie, NY. She has worked with z/OS for over thirty years and during that time has held a variety of different positions including development, quality assurance and performance. She holds an AB degree in biology from Vassar College, an MS in computer science from Union College and is nearing completion of an MSIS from SUNY Albany. She loves large databases and is especially interested in digital libraries.

Cristian Molaro is an independent consultant and instructor based in Belgium. He is IBM Information Champion and an IBM Certified DBA and Application Developer for DB2 for z/OS V7, V8, and V9. His main activity is linked to DB2 for z/OS administration and performance. Cristian is co-author of these IBM Redbooks publications: *Enterprise Data Warehousing with DB2 9 for z/OS*, *50 TB Data Warehouse Benchmark on IBM System z*, and *DB2 9 for z/OS: Distributed Functions*. He holds a Chemical Engineering degree and a Master in Management Sciences. He can be reached at cristian@molaro.be.

Deepak Rangarao is a Technical Sales Specialist on the IBM Americas Advanced Warehousing team, focusing on analytics. He has over 10 years of cross-industry experience in data warehousing and analytics. He has worked for customers and vendors in both a pre-sales and post-sales capacity in retail, banking, telecommunications, and public services. Deepak has also taught in universities on electrical engineering, Java™ network programming, and multimedia and web development. Deepak holds a Masters degree in Information Technology from R.M.I.T. in Melbourne, Australia.

Michael Schapira is an IBM IT Specialist in Montpellier PSSC Customer Center, France. He has four years of experience in System z and Business Intelligence on System z. He holds a “Computer science for decision support” engineering degree from the Ecole des Mines de Nantes, France. His areas of expertise include DB2 on z/OS, Linux on System z, and IBM Cognos BI on System z. He is interested in all related activities to Business Intelligence and

data warehousing on the mainframe. He has written extensively on Linux monitoring, IBM Cognos BI, and z/OS integration in this book. It is his first publication.

Thanks to the following people for their contributions to this project:

Jaime Anaya
IBM San Jose

Harry Batten
IBM San Francisco

John Campbell
IBM Silicon Valley Laboratory

Victor Chao
IBM Poughkeepsie Center

Gary Crupi
IBM Milwaukee

Syed Ashfaq
Cindy Chang
Ron Paul
IBM Chicago

Michael Dewert
Frank Neumann
IBM Boeblingen Laboratory

Ann Jackson
IBM Chicago

Andy Perkins
IBM Dallas

Emma Jacobs
Ann Lund
Alfred Schwab
International Technical Support Organization

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

`ibm.com/redbooks/residencies.html`

Archived

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Archived

Archived

Part 1

Introducing the co-location project

In this part we introduce our project. We provide technical guidance and performance trade-offs associated with resource management and DB2 data-sharing in a variety of mixed transactional and data warehouse System z topologies.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

1

Executive summary and highlights

As business cycles speed up, many clients gain significant competitive advantage from quicker and more accurate business decision-making by using up-to-date data. The landscape of the business intelligence world is shifting and its IT applications are becoming more mission-critical. To cope with these changing environments, many customers choose to place both their transactional and their analytical workloads on a single System z. This better leverages their existing investment in hardware, software, and skills. It enables more efficient utilization of the processor capacity and shortens ETL processing. Also, a growing set of clients are running their DW applications on multiple LPARs to improve availability.

We created a project to capture the value of co-location and examine associated techniques and considerations for housing both workloads on System z. We are documenting our findings in this book.

© Copyright IBM Corp. 2010. All rights reserved.

1.1 The project

Co-locating two different workload/data types (transactional and data warehouse) can bring up considerations about resource management and IBM DB2 data-sharing. We discussed a variety of System z topologies. The goal of this IBM Redbooks publication is to provide a combination of technical guidance and performance trade-offs based on our findings.

1.2 The environment

The IBM mainframe has been processing two-thirds of the world's business data for many decades. Its features, which provide reliability, availability, and serviceability, are well-known to many. The System z features and functions make it an excellent choice for an environment that handles both data warehouse and transactional workloads.

Based on client interest, we built two DB2 test configurations for this study. One configuration was a single z/OS V1.10 z10 LPAR, housing two unrelated DB2 V9 subsystems, one for transactional processing and the second for data warehouse processing. The primary focus of this configuration was Workload Manager resource management between these diverse workloads.

The second configuration was a 4-LPAR 4-way DB2 data sharing sysplex, all part of the same z10 server. Two of the DB2s were for transactional processing, and two were for data warehouse and analytics processing.

The primary focus of this configuration was data sharing and workload balancing implementation considerations and understanding potential resource constraints, such as processing overheads or response time delays.

Both configurations were front-ended by a zVM LPAR, on the same z10 processor, which housed SLES10 SP2 zLinux guests running IBM Cognos BI V8.4 SP2 and InfoSphere Warehouse on System z (ISWz)

Additionally, InfoSphere Change Data Capture (CDC) was also run on z/OS.

Two (one transactional, one data warehouse) 1-TB databases were built for this study.

The following four workloads were created to access one or both of these 1-TB databases:

- BookStore Transactional Web Browse/Buy application. Over 1000 users simulated, driving transaction rates upward of 2000 transactions/sec.
- BookStore Data Warehouse IBM Cognos Analysis, Reporting, Operational BI application. Over 100 novice to high importance knowledge workers simulated, generating a mix of trivial to large DB2 queries.
- Continuous Refresh of portion of the data warehouse via InfoSphere CDC.
- Batch Refresh of portions of the data warehouse via the SQW function within InfoSphere Warehouse on System z

The details of our environment are described in Chapter 8.

Archived

1.3 Experiment results

In our project, we executed a set of experiments in two key topologies that System z clients are already in or are considering migrating to.

- Co-locating transactional and data warehouse query workloads in a single LPAR.
- Co-locating transactional and data warehouse query workloads in a 4-way DB2 data-sharing group, isolating the transactional workload to two LPARs and data warehouse query workloads to the other two LPARs.

Our primary objective in the single LPAR topology was to understand WLM's ability to effectively manage system resources. The key focus areas were:

- Examine WLM methods to ensure our transactional workload continues to meet its performance objectives while at the same time meeting the performance objectives and tolerating the much more dynamic resource consumption behavior of data warehouse query workloads.
- Examine new DB2 client attribute functionality, using IBM Cognos BI applications, for more granular means to differentiate and provide a higher level of service to the critical-to-the-business knowledge workers.
- Manage short-running small consumption queries, such as Operational BI, to WLM response time goals.
- Respond to sudden large increases in demand (spikes) for processing resources from either transactional or data warehouse query workloads.
- Dynamically manage data warehouse refreshes concurrently with query workloads.

The workload manager (WLM) component of z/OS handled all types of workloads in a variety of environments. It was able to give the needed resources to each workload to enable them to meet their performance and throughput goals.

The results of our measurements confirmed our confidence in using WLM to effectively manage system resources for a mix of transactional and data warehouse query workloads within a single LPAR. In order to obtain satisfactory performance for both workloads, our experience suggests that the single LPAR environment will likely take more workload analysis and policy iterations than running the two workloads in separate LPARs. Additionally, in the single LPAR environment, there is potentially more of a need to go beyond the traditional WLM goal management, and make use of WLM's "protection of work" functions, such as CPU Critical.

Our primary objective in the multiple LPAR DB2 data-sharing topology was to understand the performance impact of adding two data warehouse LPARs running query workloads to the existing two transactional workload LPARs. We found that co-locating the OLTP and data warehouse databases in a data sharing group spread across four LPARs did not yield any performance degradation to the OLTP workloads. Furthermore, this 4-LPAR configuration supported the growth of the OLTP workload with linear scalability.

To increase availability of your applications, multiple LPARs are recommended. This added availability benefit does not seem to have a performance impact; measurement results from this study show very similar performance when deploying the applications in single vs. multiple LPARs.

Archive

1.4 Conclusions

System z can bring our clients competitive advantages in their business processing through the effective deployment of both transactional and analytics workloads under a common set of shared resources. The unique architecture of System z hardware and software allows these disparate types of workloads to coexist on the same system without interference and permits our clients to implement two key strategies: (1) To more broadly deploy analytic capabilities to their staff, and (2) To allow the deep integration of transaction and analytic workloads to improve business outcomes.

Although data sharing may add a small degree of complexity to a DB2 environment, it is an excellent solution for separating transactional workloads from data warehouse or business intelligence workloads, yet providing integration between them as desired. It provides ease of access to real-time transaction data and the data warehouse from a single application connection. It opens up opportunities for more efficient ETL processing using in-database transformations (ELT) as well as data replication strategies. It is also a tremendous high availability option for both the transactional and data warehouse DBs. With data sharing, especially when members are separated by different LPARs or different CPCs, fallback is always available to the other data sharing member (or members).

We found that adding two data warehouse DB2s (two LPARs) to an existing 2-LPAR data sharing transactional system, forming a larger 4-way data sharing group, did not lead to any performance interference with the OLTP applications.

By their nature, data warehouse and business intelligence workloads tend to require many processor, I/O, and potentially memory resources. Additionally, the growth of users and applications in this arena is dynamic. Hence, if deciding to co-locate this work with transactional workloads on the same LPAR, you must ensure that there are enough resources available to satisfy the important transactional and data warehouse workloads. Additionally, housing both in a single LPAR will likely require more iterations and maintenance of your Workload Manager service definition, honing the right level of service for each of the workloads.

1.5 The highlights

Highlights of this book include:

- Key advantages of housing data warehouse and transactional systems on System z (Chapter 2)
  - Proven high performance z/OS and DB2 for z/OS in one system
  - Data security and privacy for data which never leaves the system enclosure
  - High availability and disaster recovery for both mission-critical transactional data and decision-critical analytical data
  - High throughput and low latency data movement from transactional to data warehouse leveraging high-speed communications to enable real-time decision making
  - Dynamic workload management to optimize resource utilization
  - Specialized accelerators to further improve total system value

Archive

- Discussing trade-offs of the various potential System z topologies (Chapter 3)
  - Storing both transactional and data warehouse data on the same single DB2 subsystem requires lower management cost but makes DB2 resource tuning challenging.
  - Placing transactional data on a separate DB2 subsystem from the data warehouse DB2 while both sit on the same logical partition (LPAR). Tailoring buffer pools and DB2 parameters becomes independent of each workload. However, moving data from transactional to data warehouse becomes less efficient.
  - Host transactional and data warehouse DB2 subsystems in the same LPAR through data sharing. Belonging to the same data sharing group means DB2 catalog, directory and user data is available to both. Therefore, data can move most efficiently from transactional to data warehouse using DB2 processes.
  - The two DB2 subsystems can reside on multiple, different LPARs and still maintain data sharing. This topology further improves scalability.
- Guidance on utilizing DB2 for parallel query and the associated resource considerations (Chapter 10).

  Here we explain DB2 parallelism and the advantages of exploiting it. We also provide guidance on setting DB2 zparms and buffer pool thresholds to take advantage of it.
- Guidance on implementing data warehouse refresh strategies (Chapter 7)
  - In this book, we use both continuous refresh and batch refresh of data warehouse tables. We discuss the factors involved in deciding whether to use continuous versus batch refresh.
  - We discuss performance considerations for continuous refresh, including controlling processor utilization and meeting latency requirements.
  - Co-location of transactional data and warehouse data in a single data sharing group has performance implications for your refresh strategy.
    - When replicating data within a single data sharing group, a single InfoSphere Change Data Capture started task is capable of acting as both source and target. This offers processor savings versus replicating between distinct DB2 z/OS subsystems, each with its own InfoSphere Change Data Capture started task.
    - Data sharing provides the capability of moving data to your data warehouse tables using SQL (this is commonly referred to as ELT). While this can improve refresh performance, one must understand the volume of data being updated and the locking considerations involved. We provide guidance to you on when this is applicable and we include implementation considerations.
- Guidance for a more effective resource management and monitoring relationship between the various personnel that make up a company's Business Intelligence Competency Center (BICC) (Chapter 9).
  - Managing a data warehouse or business intelligence system requires the cooperation of BI, DB2 and system administrators, each of which had a different resource monitoring and management perspective. We provide some common performance terminology, the high-level steps associated with resource management, suggested areas for collaboration, and high-level descriptions of some of the common monitoring and management tools for System z.
- Guidance on building, analyzing and verifying System z/OS resource management of data warehouse and mixed transactional or data warehouse workloads (Chapters 11 and 15).
  - Here we provide a process for building a WLM service definition, specifically targeted for data warehouse workloads. This encompasses from early preparation to the building of the key WLM constructs: *workloads, service classes, classification rules.*

Considerations are provided for managing a dedicated data warehouse system, such as the IBM Smart Analytics System 9600, or a mixed transactional and data warehouse workload.

  - We provide guidance for analyzing performance and maintenance of the WLM service definition.
  - We take you through some experiments, exhibiting WLM's behavior, under common workload fluctuations. We also cover other resource management influencers, such as specialty engines.
  - We provide a technique, based on DB2 accounting data, to help hone a set of service class period durations for one's DDF query workloads.
  - Included is a sample WLM service definition, which is the basis for the IBM Smart Analytics System 9600 service definition.

- Detailed information on utilizing *DB2 client information* attributes for qualifying business intelligence queries for management, monitoring and reporting (Chapter 6).
  - Optimal classification and differentiation of data warehouse query workloads is critical to overall performance of the business intelligence environment. Here we provide how to utilize DB2 client information attributes for more granular and sophisticated resource management and monitoring. Specifically, we exhibit how to implement the use of this for our IBM Cognos BI solution.
- Value and implementation considerations for DB2 data-sharing the operational and data warehouse databases (Chapters 10 and 16).
  - With data sharing, both real-time operational and data warehouse data can be accessed from a single application connection. We provide you with a sample IBM Cognos "Operational BI" exploitation of this benefit (section 6.3.4).
  - Here we describe the reasoning and the benefits of configuring OLTP and data warehouse databases in a data sharing group. We quantify the impact of co-location in a data sharing group based on measurement results. Observations about co-location operations are also given.
  - We take you through some experiments running operational workloads in a 2-way data sharing environment. We compare those results to transaction and DW workloads in a 4-way data sharing environment.
- Implementing DVIPA and Sysplex Distributor in the data-sharing environment and limiting the query workload activity to a subset via DB2 location alias (Chapter 12).

  Here we provide guidance and examples on implementing a DB2 Data Sharing Workload Balancing and Fault Tolerant configuration, including client side sysplex support. Also how to set a DB2 Location Alias within a data sharing group in order to limit Data Warehouse reporting to specific members.
- Throughout the book, we recommend that certain PTFs/APARs be applied for improved resource management and monitoring.

Archived

2

Why System z for data warehousing

The IBM mainframe has been processing over two-thirds of the world's business data for many decades. Its features provide reliability, availability, and serviceability; they are well-known to many users. This chapter describes the System z features and functions that make it an excellent choice for an environment that handles both data warehouse and transactional workloads.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

2.1 Architecture

The z/OS operating system and the IBM System z196, z10, System z9®, and zSeries® 890 and 990 systems offer architectures that provide qualities of service that are critical for data warehousing.

2.1.1 z/Architecture and the z/OS operating system

z/OS is a highly secure, scalable, and open operating system that offers high performance while supporting a diverse application execution environment. The z/OS operating system is based on 64-bit z/Architecture®. The robustness of z/OS powers the most advanced features of the IBM System z technology, enabling you to manage unpredictable business workloads.

DB2 gains a tremendous benefit from z/Architecture. The tight integration that the IBM DB2 database management subsystem has with the System z architecture and the z/OS environment creates a synergy that allows DB2 to exploit advanced z/OS functions. The following z/Architecture features benefit DB2:

64-bit storage

Increased capacity of central memory from 2 GB to 16 exabytes eliminates most storage constraints. 64-bit storage also allows for 16 exabytes of virtual address space, a huge step in the continuing evolution of increased virtual storage. In addition to improving DB2 performance, 64-bit storage improves availability and scalability, and it simplifies storage management.

High-speed communication

HiperSockets™ enables high-speed TCP/IP communication across partitions of the same System z server; for example, between Linux on System z and DB2 for z/OS.

Dynamic workload management

One of the strengths of the System z platform and the z/OS operating system is the ability to run multiple workloads at the same time within one z/OS image or across multiple images. The evolution of data warehousing has spawned a diverse set of workloads, each having unique service level requirements. Some examples are operational BI (or tactical queries), analytics, scheduled reporting, refresh processing, and data mining. Additionally, the user base has grown beyond senior executives, mid-level management and business analysts to include customer-facing personnel, such as service representatives. The coexistence of this mixed workload and how to distribute resources has been identified as one of the most important issues in the DW design.

The function that makes this possible on z/OS is dynamic workload management, which is implemented in the Workload Manager (WLM) component of the z/OS operating system. The idea of z/OS Workload Manager is to make a contract between the applications and the operating system. The installation classifies the work running on the z/OS operating system into distinct service classes and defines goals for them that describe how the work should perform.

WLM uses these goal definitions to manage the work across all systems of a Sysplex environment. WLM adjusts dispatch priorities and resource allocations to meet the goals of the service class definitions. It does this in order of importance. Resources include processors, memory, and I/O processing.

Intelligent Resource Director

The Intelligent Resource Director (IRD) allows you to group logical partitions that are resident on the same physical server, and in the same sysplex, into an LPAR cluster. This gives Workload Management the ability to manage resources across the entire cluster of logical partitions.

Faster processors

With more powerful, faster processors, along with the System z Integrated Information Processor (zIIP) specialty engine, DB2 achieves higher degrees of query parallelism and higher levels of transaction throughput. The zIIP is designed to improve resource optimization and lower the cost of eligible workloads, enhancing the role of the mainframe as the data hub of the enterprise.

2.2 DB2 for z/OS

At the heart of your data warehouse, before you can even consider running a BI application, you need a database management system (DBMS). Fortunately, System z has DB2 for z/OS. In the beginning of data warehousing, when it was still referred to as decision support, DB2 for z/OS was at its center. In fact, throughout DB2's long history, it has always delivered product enhancements that championed decision support. Many customers continue to use DB2 for z/OS today for their warehouses. To meet the changing database landscape of today's challenging warehousing world, IBM is delivering even more significant DB2 for z/OS enhancements than ever before in direct support for data warehousing and BI. The last few DB2 releases have been rich with capabilities to improve your warehouse and BI experience.

DB2's renewed presence in the warehouse world is not coincidental. Warehousing has been changing. Rather than determining what happened in the past, customers want to use all that information to make immediate decisions today. Instead of allowing only a few access to the invaluable information being kept in the warehouse, today it is being leveraged by a larger number of individuals, including their clients. The focus is on getting the information to the correct person when that person needs it. Information must arrive rapidly and accurately. It is starting to sound a lot like OLTP, is it not? What better RDBMS than DB2 for z/OS to satisfy these modern data warehouse challenges?

2.2.1 DB2's effect on your warehouse

Just how has DB2 for z/OS supported data warehousing over the last 25 years? DB2 has continually delivered features and functions in direct or indirect support of warehousing and its associated business intelligence applications. Some of DB2's more significant features that warehousing can and should take advantage of (this list is far from being a complete list) are:

- Partitioning - table-based partitioning; clustering can be different from the partitioning order.
- Resource Limit Facility (RLF), introduced in DB2 V2.1, allows for the control of the amount of processor resource that a task, in this case a query, can actually use. RLF affects dynamic SQL, which can make up a significant portion of a warehouse's SQL workload. This can be critical in controlling system resources. RLF can also help you control the degree of parallelism obtained by a query.
- Hardware-assisted data compression was delivered in DB2 V3, and still has a major, immediate effect on data warehousing. Enabling compression for table spaces can yield significant disk savings. In testing, numbers as high as 80 percent have been observed.

- I/O parallelism, CP parallelism, and sysplex query parallelism became available in DB2 Version 3, Version 4, and Version 5, respectively. A discussion of this feature follows.
- Data sharing was delivered along with CP parallelism in DB2 Version 4. High availability for a data warehouse has become the norm rather than the exception. Data sharing is capable of giving a data warehouse that kind of high availability. DB2's data sharing allows access to the transactional data by the warehouse and analytics, yet still lets you separate those applications into their own DB2, reducing the chances of the warehouse activity impacting transactions.

Partitioning

The large volume of data stored in DW environments can introduce challenges to database management and query performance. The table space partitioning feature of DB2 for z/OS currently has the following characteristics to aid in reducing those challenges:

- Maximizes availability or minimizing runtime for specific queries by allowing queries and utilities to work at the partition level.
- Can grow to 4096 partitions, with a partition being a separate physical data set.
- Allows loading and refreshing activities, including the extraction, cleansing, and transformation of data in a fixed transactional window.
- Increases parallelism for queries and utilities. Parallelism can be maximized by running parallel work across multiple partitions.
- Accommodates data growth; a universal table space can grow automatically up to 128 TB and has the functionality of segmented table spaces while retaining the size and partition independence allowed by a partition table space.
- If data should become damaged or otherwise unavailable, data recovery or restoration can be performed at the partition level, improving availability and reducing elapsed time.

Compression

DB2's compression is specified at the table space level, is based on the Lempel-Ziv lossless compression algorithm, uses a dictionary, and is assisted by the System z hardware. Compressed data is also carried through into the buffer pools. This means compression could have a positive effect on reducing the amount of logging you do because the compressed information is carried into the logs. This will reduce your active log size and the amount of archive log space needed. Compression can also improve your buffer pool hit ratios. With more rows in a single page after compression, fewer pages need to be brought into the buffer pool to satisfy a query's get page request. One of the additional advantages of DB2's hardware compression is the hard speed. As hardware processor speeds increase, so does the speed of the compression built into the hardware's chipset.

When implementing a data warehouse, the size is often considered problematic. Regardless of the platform, DB2's hardware compression can help address that concern by reducing the amount of disk needed to fulfill your data warehouse storage requirements.

Parallelism

One method of reducing the elapsed time of a long-running query is to split that query across more than one processor. This is what DB2's parallelism does. Parallelism allows a query to run across two or more CPs. A query is broken into multiple parts, with each part running under its own service request block (SRB) and performing its own I/O. Although there's some additional processor used when DB2 first decides to take advantage of query parallelism for its setup, there is a close correlation between the degree of parallelism achieved and the query's elapsed time reduction. There are also DSNZPARMs and bind parameters that need to be set before parallelism can be used.

Star schema

There is a specialized case of parallelism called a star schema—a relational database's way of representing multi-dimensional data—that is often popular with data warehousing applications. A star schema is usually a large fact table with lots of smaller dimension tables. For example, you might have a fact table for sales information. This sales table would hold most of your data. The dimension tables could represent products that were sold, the stores where those products were sold, the date the sale occurred, any promotional data associated with the sale, and the employee responsible for the sale. Using star joins in DB2 requires enabling the feature through a DSNZPARM keyword. You should also check a few other ZPARMs before using star joins because they can affect a star join's performance.

2.2.2 How DB2 for z/OS Version 8 helped

Many enhancements in DB2 V8 can directly impact a warehouse implementation. The following are just a few major enhancements in DB2 V8.

Backward index scan

Indexes are a huge performance asset to a data warehouse. Having the ability to read an index backward lets you avoid building both an ascending index and a descending index structure. Reducing the number of indexes improves the cost of doing inserts and deletes. Every insert or delete must update every index on the table being changed. Backward index scan also can reduce the cost of doing updates if the update occurs against a column participating in the index. In addition, backward index scan reduces the amount of disk storage required to build all those indexes. With backward index scan, you will need to use only one index; you previously needed two.

Multi-row FETCH and INSERT

This enhancement lets you read or insert multiple rows with a single SQL statement via an array. This reduces the processor cost of running a FETCH or INSERT. This feature is completely usable in distributed applications processing using Open Database Connectivity (ODBC) with arrays and dynamic SQL. This offers significant performance advantages: it could increase the performance of FETCH processing by 50 percent and INSERT processing by 20 percent. In fact, in some customer testing, this feature averaged 76 percent improvement for FETCH and 20 percent improvement for INSERT. Improving INSERT performance and reducing INSERT processor consumption could be a significant help to your Extract, Transform, Load (ETL) processing.

Indexable unlike types

Mismatched data types can now be stage 1 and indexable. This is key for those applications that do not support all data types available in DB2 for z/OS.

Sparse index in memory work files

For star join processing, sparse indexes can use many work files. DB2 V8 will attempt to put these work files in memory. This can result in a significant performance improvement for warehouse queries using star joins.

More partitions (4,096) and automatic space management

Growth is one of those inherent qualities of a warehouse, something you just expect to happen. With 4,096 partitions, a warehouse could grow to 16 TB for a 4 KB page size; 128 TB for a 32 KB page. This is for just one table. DB2 V8 also gives you automatic space management, the ability to let DB2 manage your primary and secondary space allocations. With DSNZPARM MGEXTSZ activated, DB2 will manage the allocation of a table space's

extents, ensuring that the table space can grow to its maximum size without running out of extents.

Materialized Query Table (MQT)

An MQT is a DB2 table that contains the results of a query, along with the query's definition.

An MQT can be thought of as a materialized view or automatic summary table that is based on an underlying table or set of tables. These underlying tables are referred to as the base tables. MQTs are a powerful way to improve response time for complex SQL queries, especially in queries that involve some of the following situations:

- A commonly accessed subset of rows
- Joined and aggregated data over a set of base tables
- Aggregated or summarized data that covers one or more subject areas

MQTs can effectively eliminate overlapping work among queries by doing the computation once when the MQTs are built and refreshed, and then reusing their content for many queries. In many workloads, users will frequently issue queries over similar sets of large volume data. Moreover, this data is often aggregated along similar dimensions (for example, time or region). Though MQTs can be directly specified in a user's query, their real power comes from the query optimizer's ability to recognize the existence of an appropriate MQT implicitly, and to rewrite the user's query to use that MQT. The query accesses the MQT (instead of accessing one or more of the specified base tables). This shortcut can drastically minimize the amount of data read and processed.

For example, suppose that you have a large table named SALES that contains one row for each transaction that a certain company processes. You want to compute the total transactional revenue along the time dimension. Although the table contains many columns, you are most interested in these four columns:

- YEAR, MONTH, and DAY, which represent the date of a transaction
- REVENUE, which represents the revenue gained from the transaction

To total the amount of all transactions between 2001 and 2008 by year, you would use this query:

```
SELECT YEAR, SUM (AMOUNT)
FROM TRANS
WHERE YEAR >= '2001' AND YEAR <= '2008'
GROUP BY YEAR
ORDER BY YEAR ;
```

This query might be expensive to run, particularly if the TRANS table is a large table with millions of rows and many columns. Suppose that you define a system maintained MQT that contains one row for each day of each month and year in the TRANS table. Using the automatic query rewrite process, DB2 could rewrite the original query into a new query that uses the MQT instead of the original base table TRANS. The performance benefits of the MQT increase as the number of queries that can consume the MQT increases. However, users must understand the associated maintenance cost of ensuring the proper data currency in these MQTs. Therefore, the creation of effective and efficient MQTs is both a science and an art.

2.2.3 How DB2 9 for z/OS helped

DB2 9 for z/OS delivered more changes that could directly impact your warehouse and application analytics. Details on just a few of those enhancements follow.

Universal table space

This is at the top of the list of DB2 enhancements for data warehousing. Consider the unpredictable growth of a warehouse and the high possibility that many tables could be frequently refreshed. A universal table space is a cross between a partitioned table space and a segmented tablespace, giving you many of both of its parents' best features. When using a universal tablespace, you get the size and growth of partitioning while retaining the space management, mass delete performance, and insert performance of a segmented table space. It is like having a segmented table space that can grow to 128 TB of data, assuming the right DSSIZE and right number of partitions are specified, and that also gives you partition independence.

Randomized index key

You want your keys spread throughout the index with no hotspots, a task sometimes easier to explain than accomplish. Sometimes stuff just all ends up in an incorrect place, no matter how hard you try to pick a key that will spread the data around. Enter DB2 9 for z/OS with a new option for the CREATE and ALTER INDEX SQL statements. In the past, you could specify ascending for forward index scans or descending for backward index scans on each column in an index key. In DB2 9, you have the additional choice of specifying RANDOM for an index column. This causes the index entries to be put in random order by that column. Randomly inserting keys could reduce the chances of contention caused by ascending index inserts or index hot spots. Indexes created with the RANDOM option do not support range scans.

There are instances when you cannot use RANDOM:

- If the key column is VARCHAR and the index uses the NOT PADDED option.
- The index was created with the GENERATE KEY USING clause.
- The index is part of the partitioning key.

The RANDOM clause on an index is available only after you have upgraded your DB2 9 subsystem to New Function Mode (NFM).

Index usage tracking

For years, clients asked for an easy way to determine whether an index is no longer being used. If they knew there was no access against a particular index, it would make it much easier to delete the index, cleaning up the DB2 catalog, and maybe avoiding some unnecessary updates and deletes to that index. There is now a way to accomplish this. This can be especially important for a data warehouse that frequently refreshes its tables. By removing unused indexes, DB2 avoids updating those indexes during INSERT or LOAD processing.

DB2 9 for z/OS added the column LASTUSED to the Real-Time Statistics (RTS) table SYSIBM.SYSINDEXSPACESTATS. LASTUSED contains the date when the index, described by this row in the catalog table, was last used for a SELECT, FETCH, searched UPDATE, searched DELETE, or used to enforce a referential integrity constraint. This column is not updated when the index is changed because of an INSERT or UPDATE. LASTUSED defaults to a value of 1/1/0001. If you find an index with the default date, that index has never been used since moving to DB2 9.

You should wait a while after upgrading to DB2 9 before taking advantage of this feature to make sure the column gets the opportunity to get updated.

There is also an APAR, a somewhat older hiper PTF, APAR PK44579 (maintain SYSIBM.SYSINDEXSPACESTATS.LASTUSED for all cases) that needs to be applied if you plan to take advantage of this column.

The tables for RTS are now part of the DB2 catalog.

Index usage tracking is available only after you've upgraded your DB2 9 subsystem to NFM.

SQL

Two new SQL statements, MERGE and TRUNCATE, can be highly effective when used with a data warehouse. MERGE, sometimes called “upsert,” lets you change data without needing to know whether the row already exists. With MERGE, if it finds an existing row, it updates it. If it does not, then if performs an INSERT. No more doing a SELECT first to see whether the row exists or doing an INSERT or UPDATE and checking to see whether it failed. TRUNCATE is an easy way to remove all the rows from a table with a single SQL statement. It is especially handy if you are using DELETE triggers. Also, there is now an APPEND option on the CREATE/ALTER table that tells DB2 to ignore clustering during INSERT processing. This should improve INSERT performance by eliminating the need to figure out where the row should go and just placing it at the end of the table.

APPEND clause

A data warehouse can have a tremendous amount of SQL INSERT and online LOAD utility processing. Normally, DB2 makes every attempt to place a row in the correct place in a table. To accomplish this, it must perform a search. The more it has to search, the more expensive the INSERT or LOAD process becomes. Some warehouses may refresh one or more tables using INSERT or LOAD on a nightly basis—all expensive, time-consuming operations.

DB2 9 provides a solution. DB2 has added a new APPEND clause to the CREATE and ALTER TABLE SQL statements. If APPEND is set to YES, DB2 places a new row at the end of the table, ignoring any forms of clustering defined to that table. The end of a table can be defined several ways, depending on the type of object being accessed. For example:

- For a segmented table space, it is placed at the end of the table.
- For partitioned table spaces and partition by range universal table spaces, the row is placed at the end of the appropriate partition.
- For partition by range universal table spaces, the row is placed in any partition with space at the end.

The APPEND clause cannot be used with tables created in LOB, XML, and workfile table spaces. It is also ignored by the REORG utility, so REORG can be used to maintain cluster order if that order is necessary.

The APPEND clause is available only after you have upgraded your DB2 9 subsystem to NFM.

Clone tables

Tables can often be completely replaced on a weekly (even daily) basis in a warehouse. Replacing a table can cause an outage, even if that outage seems short. Clone table support in DB2 9 gives you an easy way to create a replacement table while still accessing the original table and using a command to switch which table SQL actually accesses. The concept is a lot like an online LOAD REPLACE, something unavailable in DB2.

RANK

DB2 9 gives you a little bit of Online Analytical Processing (OLAP) functionality with RANK, DENSE_RANK, and ROW_NUMBER. When used in an SQL statement, they return the ranking and row number as a scalar value. Rank is the ordinal value of a row in a defined set of rows. You can specify that the result be returned with (RANK) and without (DENSE_RANK) gaps. ROW_NUMBER is a sequential row number assigned a result row.

Index compression

Your first line of defense against a warehouse performance problem, after a well-written query, is creating lots of indexes. With warehousing, and for some types of OLTP, it is possible to use as much (or more) disk space for indexes as for the data. DB2 9 for z/OS index compression can make a huge difference when it comes to saving disk space. The implementation of index compression is nothing like data compression. It does not use a dictionary and there is no hardware assist. However, the lack of a dictionary could be a plus. With no dictionary, there is no need to run the REORG or LOAD utilities before compressing your index data. When compression is turned on for an index, key and Record Identifier (RID) compression immediately begins.

One of the more popular solutions to a query performance dilemma is an index. Adding an index can fix many SQL issues. However, adding an index has a cost; the additional index will consume additional disk space. You can try to decide which is the lesser of two evils, the disk space consumption or the poorly running SQL statement. Moreover, DB2 9 now supports some powerful index enhancements. You may suddenly find yourself using even more disk storage for indexes in DB2 9 than you ever thought possible. Of course, DB2 9 does come with a near-perfect solution: index compression.

Even though some query types used in a data warehouse environment can significantly benefit from the addition of an index, it is possible for a warehouse to reach a point were the indexes have storage requirements that are equal to, if not sometimes greater than, your table data storage requirements.

Index compression in DB2 9 is a possible solution to an index disk storage issue. Early testing and information obtained from first implementers indicates that significant disk savings can be achieved by using index compression. With index compression being measured as high as 75 percent, you can expect to achieve, on average, about a 50 percent index compression rate. Of course, compression carries some cost. In certain test cases, there was a slight decrease in class 1 processor time accessing a compressed index, but you can expect your total processor time, class 1 and class 2 SRB processor time combined, to increase.

However, it is important to remember that processor cost is realized only during an index page I/O. Once the index has been decompressed into a buffer pool or compressed during the write to disk, compression adds zero cost.

Unlike data compression, with index compression there is no performance advantage directly from the use of compression. Index compression is strictly for reducing your index disk storage. If any performance gain occurs, it will show up when the optimizer can take advantage of one of the additional indexes that now exist—an index that may never have been created because of disk space constraints prior to the introduction of index compression.

Index compression is available only after you have upgraded your DB2 9 subsystem to NFM.

Data-partitioned secondary index

A data warehouse usually means large amounts of data. That translates to partitioned table spaces, and partitioned table spaces often call for Data-Partitioned Secondary Indexes (DPSIs). DB2 9 has made a few improvements to DPSIs that should make them more usable and popular. For example, a DPSI can now get index-only access and a DPSI index can be defined as unique when columns in the DPSI key are a superset of partitioning columns. A few more subtle DPSI improvements are: enhanced page range screening to help avoid hitting every partition, more parallelism, and more index look-asides.

Remember the lessons learned in DB2 Version 8 if you use DPSIs. Always include a local predicate on the leading columns of the DPSI index to avoid scanning the entire index

structure. Also, remember that when using a DPSI, you will see an increase in the number of VSAM data sets that DB2 will need to use. Make sure you appropriately adjust your DSMAX value.

All these DPSI index improvements are available in Conversion Mode (CM).

Prefetch improvements

In a data warehousing environment, some queries use prefetch; it is just normal. As mentioned earlier, there are situations where a stage 2 predicate is your only option; stage 2 means prefetch. The good news for all of those stage 2 warehouse queries is that prefetch has improved in DB2 9. Prefetch quantity has increased. When a pool size becomes larger than 40,000 pages, a larger prefetch quantity is now used. If the buffer pool is greater than 40,000 and less than or equal to 80,000 pages, the prefetch quantity is 64 pages for SQL processing and 64 pages for utilities. The utility page quantity jumps to 128 pages when the VPSIZE * VPSEQT is greater than 80,000 pages. Of course, this description of prefetch quantities is based on 4K page size. These numbers must be adjusted for other page sizes.

When DB2 chooses to use dynamic prefetch, there is a significant change made in DB2 9. Dynamic prefetch is used for index scans and table access via table scans. When dynamic prefetch is picked at bind time, it can switch between sequential prefetch and index access based on the data pages accessed using something called sequential detection. In addition, dynamic prefetch does not use a triggering page and can scan forward or backward, something that is handy for a backward index scan. Finally, dynamic prefetch engines also can run in parallel. A few utilities available via APAR PK44026 also are recent converts to dynamic prefetch. For example, dynamic prefetch is enabled for:

- The UNLOAD phase of REORG INDEX
- The UNLOAD and BUILD phases of REORG TABLESPACE PART
- The BUILD phase of LOAD REPLACE PART
- The RUNSTATS phase of RUNSTATS INDEX

The changes to dynamic prefetch in DB2 9 apply to single table access, multitable join, outer join, subquery, and union. Sequential prefetch, as of DB2 9, is used only for table space scans.

Dynamic prefetch could have all kinds of performance advantages over sequential prefetch because it is not dependent on the optimizer making the correct decision the first time and allows the access path to be changed on the fly as data patterns change.

The prefetch changes are available in DB2 9 CM.

Reordered Row Format (RRF) is one of those significant changes that most would not even know about if not for a couple of user group presentations. This change affects how a table's variable length columns are ordered when written to disk. A data warehouse, by its nature, can have lots of variable length columns.

A challenge that has existed for DB2 professionals since DB2's beginnings is placement of the fixed length (CHAR) columns vs. the placement of variable length (VARCHAR) columns in a newly created table definition. There have been conflicting views on what is correct, efficient, and the best coding practice. Most have agreed that the VARCHAR columns should go at the end of the row, or at least infrequently updated variable length columns should go last. However, no matter what order you use to define a table's columns, once DB2 hits a VARCHAR column when retrieving a row, DB2 must calculate the starting position of subsequent columns in that row. Until DB2 reads the length of the VARCHAR column (the two-byte prefix on every VARCHAR column), it has no idea where to find the beginning of the next column that needs to be retrieved.

DB2 9 provides a potential solution to this quandary. Once you get to DB2 9 for z/OS NFM, all VARCHAR columns on newly created table spaces are placed at the end of the row. DB2 9 can make this type of decision because the row format in DB2 9 has changed. DB2 9 introduces RRF, a straightforward concept in which a DB2 9 NFM row will have all the fixed length columns first, followed by pointers to the beginning of each VARCHAR row. The pointers are followed by the actual variable length data. Rather than scan what could be some lengthy columns just to find the beginning of the column you are looking for, DB2 only needs to scan the list of pointers to find the location for the beginning of the column you want.

You will get RRF for any table space created in DB2 9 NFM; additionally, any table spaces, or table space partitions, will be converted to RRF when a REORG or LOAD REPLACE is run against that table space or table space partition. Be cautious when working with partitioned table spaces. If you only REORG selected partitions of a table space, then you will end up with some partitions, the ones that have been reorganized, in the new RRF. The remaining partitions that have not yet been reorganized will stay in *basic row format*. Basic row format is the phrase used to describe the row format prior to DB2 9 for z/OS NFM.

RRF is available only after you have upgraded your DB2 9 subsystem to NFM.

2.3 Resource management

System z is a unique platform, especially when it comes to resource management. It was the first platform to run multiple operating systems on the same physical machine at the same time, through LPAR technology. Virtualization and resource management are part of the IBM mainframe's architecture.

A key value of z/OS is its workload manager, WLM, which makes it possible to dynamically reassign resources to a specific task that is not meeting its performance objectives. WLM can work with the LPAR technology through a module called Intelligent Resource Director (IRD).

The mainframe offers another level of virtualization with a product called z/VM, a hypervisor operating system.

Now let us define these terms and concepts to give a basic view of the resource management features of System z.

2.3.1 Logical partitions

Logical partitions (LPARs) were introduced in 1988 with the ESA/390 architecture. It is a mature technology for the mainframe, but has more recently been copied by other hardware platforms. The function of LPAR is to physically create multiple computers within one physical hardware system. It was the first level of virtualization, which makes it possible to run multiple operating systems on the same physical hardware. This can be any mainframe operating system; there is no need to run z/OS, for example, in each LPAR. The installation planners may elect to share I/O devices across several LPARs, but this is a local decision.

Important: Logical partitions are equivalent to separate mainframes.

The system administrator can assign one or more system processors for the exclusive use of an LPAR. Alternately, the administrator can allow all processors to be used on some or all LPARs. Here, the system control functions (often known as microcode or firmware) provide a dispatcher to share the processors among the selected LPARs. The administrator can specify a maximum number of concurrent processors executing in each LPAR. The administrator can

also provide weightings for different LPARs; for example, specifying that LPAR1 should receive twice as much processor time as LPAR2.

These techniques make resource management very flexible. There is no need to use all of them; select the ones that give an administrator the desired control of the resources.

2.3.2 z/OS Workload Manager

The Workload Manager (WLM) is part of z/OS. Each z/OS system has its own WLM policy, which is the way to classify workloads or tasks. What is great about WLM is that you are not allocating resources to a task, but goals in a business oriented manner. For example, one can define a policy where the transaction processing will have to sustain a load of at least 250 transactions per second with an average response time of 0.1 second. As WLM manages the operating system task run list, it will allocate resources and dispatch tasks according to the policy that was defined by the system administrator.

WLM is also able to work with the underlying hardware with a module called Intelligent Resource Director (IRD). Imagine if one z/OS LPAR did not achieve its WLM objectives because of a constraint on the resources whereas another LPAR overachieved and had unused resources. This would be a loss of computing power. This is why z/OS moves resources (such as processor) from one LPAR to another, transparent to the application. This is done by WLM, IRD, and LPARs (through PR/SM™).

Important: The main purpose of this book is to discuss co-location of transactional and data warehouse data. We give you insight and a methodology on how to define the WLM policy that will fit your installation.

2.3.3 z/VM - System z Virtual Machine

The IBM mainframe with IBM Virtual Machine (VM) has been performing virtualization for four decades. Today, z/VM is the most functionally rich virtualization platform available. When Linux came to the IBM mainframe in 2000, it was a natural fit to run under z/VM. You can run dozens of Linux images on the same system logical partition. Some clients are running hundreds of images in production mode.

With a z/VM and Linux infrastructure, you can reduce the time between deciding on the acquisition of new servers and then implementing them because new servers can be deployed in a matter of minutes. This powerful build and clone capability can enable you to launch new products and services without the exhaustive planning, purchasing, installing and configuring of new hardware and software that can be associated with conventional discrete hardware servers. Development groups who need test environments built and rebuilt rapidly to enable them to efficiently deliver their projects, while handling change management in the process, can also benefit from this unique advantage.

Listed here are some of the most significant strengths offered by the IBM mainframe and z/VM:

- Their virtualization capabilities are more mature and robust than any other hardware and hypervisor combination.
- z/VM provides a rich, functional, and sophisticated level of system management, which can greatly benefit running large numbers of Linux servers.
- z/VM's virtual switch (VSWITCH) makes networking Linux much simpler.

- Full volume backup of systems allows for complete disaster recovery when another data center is available.
- z/VM is one of the easiest operating systems to customize at the base installation level, with only a relatively small number of configuration files. After it is set up, z/VM will run for months with little maintenance and administration required.

z/VM 5.3

z/VM 5.3 became generally available in June 2007. Scalability was extended to allow 256 GB of real memory, a total of 8 TB of virtual storage, and 32 real processors. z/VM V5.3 also added support for the Collaborative Memory Management Assist (CMMA) on the z9 EC and the z9 BC processors or later. Virtual Machine Resource Manager (VMRM) detects when memory is constrained and notifies the Linux guests, which can then adjust their memory consumption to help relieve the memory constraint. In the previous major release (z/VM 5.2), many memory contention issues were removed with the Control Program (CP) now using memory above 2 GB for a much broader set of operations. Previously, guest pages had to be moved below 2 GB for many reasons, for example in both standard I/O and Queued Direct I/O (QDIO). Now I/O can be performed using buffers anywhere in real memory, and QDIO structures can reside above 2 GB, as can most CP control blocks. These improvements offer constraint relief for large-real-memory virtual server environments that are memory-intensive.

z/VM 5.4

z/VM 5.4, available in August 2008, provides major improvements when operating on System z servers with large memory configurations. It improves scalability and can help support increased workloads on IBM System z servers. This release exploits new capabilities of System z10, including:

- Greater flexibility, with support for the new z/VM-mode logical partitions, allowing all System z processor-types (CPs, IFLs, zIIPs, zAAPs, and ICFs) to be defined in the same z/VM LPAR for use by various guest operating systems.
- The capability to install Linux on System z from the HMC, which eliminates network setup or a connection between an LPAR and the HMC.
- Enhanced physical connectivity by exploiting all OSA-Express3 ports, thus helping to service the network and reduce the number of required resources.

z/VM 5.4 dynamic memory upgrade support allows real memory to be added to a running z/VM system, thereby avoiding the need to shut down z/VM and its guests, deactivate the LPAR, change its memory allocation, reactivate the LPAR, re-IPL z/VM, and restart its guests. Memory can be added non-disruptively to individual guests that support the dynamic memory reconfiguration architecture.

Read more about System z virtualization capabilities on the web at:

`http://www.vm.ibm.com/`

2.4 Unique I/O capabilities

Because the amount of data in a data warehouse environment increases daily, there is a requirement for providing fast data access to the processor unit. High I/O delays from the data storage devices can lead to performance problems in DW queries, which sometimes perform a large number of multiple reads from one particular volume.

Environments hosting a large number of users and the traditionally huge amounts of mixed workloads coming into the system play in the sweet spot of the System z architecture. These

mainframes have an I/O architecture with multiple paths to storage devices, for both performance and availability. In addition, parallel access volumes (PAV) enable a single System z server to simultaneously process multiple I/O operations to the same logical volume, which can significantly reduce device queue delays (IOSQ time). This is achieved by defining multiple addresses per volume. With dynamic PAV, the assignment of addresses to volumes can be automatically managed to help the workload meet its performance objectives and reduce overall queuing.

For example, while building an ad hoc report that reads a large table space, fast data access to the CPU will determine how quickly that report can be built. System z offers FICON® Express4, which is a new generation of FICON and Fibre Channel Protocol (FCP) features providing fast data access with 1, 2, and 4 Gbps auto-negotiating links. FICON Express4 supports increased CPU performance and provides increased application performance while providing a manageable migration to higher speed. FICON Express4 continues the tradition of a robust balanced I/O system design on IBM System z.

FICON Express4 and other System z9 channel enhancements improve channel performance, provide support for more devices, and support standards-based FCP enhancements that help improve resource sharing and access control for Linux on System z environments.

FICON distance and bandwidth capabilities also make it an essential and cost effective component of data high availability and disaster recovery solutions when combined with System z Parallel Sysplex and GDPS® technology. Parallel Sysplex provides resource sharing, workload balancing, and continuous availability benefits while GDPS provides system level automation enabling the most advanced, application-transparent, multi-site disaster recovery solution with fast recovery time. It offers two unrepeated distance options (4 KM and 10 KM) when using single mode fiber optic cabling All FICON Express4 and FICON Express2 features support the Modified Indirect Data Address Word (MIDAW) facility. MIDAW is a new system architecture with software exploitation that improves channel use, reduces channel overhead, and potentially reduces I/O response times. AMP and zHPF are new to the FICON arena.

The combination of IBM System z10 and IBM System Storage® DS8000® delivers unique I/O capabilities for large environments. Among other significant innovations, DS8000 offers the following I/O capabilities, contributing to large and complex installations in terms of performance and increased efficiency for huge amounts of mixed workloads:

- Storage Pool Striping - New volume configuration option to maximize performance without special tuning.
- Adaptive Multi-Stream Pre-Fetch (AMP) - Breakthrough caching technology can dramatically improve sequential read performance to reduce backup times, processing for Business Intelligence and Data Warehousing.
- z/OS Global Mirror Multiple Reader - IBM unique innovation to improve throughput for z/OS remote mirroring.
- z/OS Global Mirror enabled for zIIP - Can help provide better price performance and improved utilization of resources at mirrored site.
- HyperPAVs - Reduce the number of alias device addresses, PAVs switched as required by DS8000 on each write I/O.
- zHPF - Improved performance for small block transfers.
- Intelligent Write Cache (IWC) - Improvements in destage, keeping data in NVS that needs to be in NVS longer.
- Parallel Access Volumes (PAVs) - Allowing for parallel I/Os on the same disk.

- DS8000 Disk Encryption Integrated with TLKM (same key manager for disk and tape).
- z/OS Metro/Global Mirror - Incremental resynchronization innovation that reduces the amount of data transmitted.
- IBM z/OS Basic Hyperswap - An integrated solution to help enable cost effective data availability protection.
- Extended Address Volumes - Allowing for management of up to four times more information in mainframe environments than with any other storage system.
- Dynamic Volume Expansion - Easier, online, volume expansion to support growth.

Among other innovations, PAVs are among the major contributors for performance enhancements of large workloads. A PAV is the ability to read or write on a single disk volume in parallel, which eliminates many of the data placement issues. By accessing data in parallel, you can gain significant performance increases in terms of I/O throughput.

Simply stated, Parallel Access Volumes (PAVs) allow multiple concurrent I/Os to the same volume at the same time from applications running on the same z/OS system image. Read and write operations can be accessed simultaneously to different domains. (The domain of an I/O operation is the specified extents to which the I/O operation applies.) This concurrency helps applications better share the same logical volumes with reduced contention. The ability to send multiple concurrent I/O requests to the same volume nearly eliminates I/O queuing in the operating system, thus reducing I/O response times.

Traditionally, access to highly active volumes has involved manual tuning, splitting data across multiple volumes, and more actions in order to avoid those hot spots. With PAV and the z/OS Workload Manager, you can now almost forget about manual device level performance tuning or optimizing. The Workload Manager is able to automatically tune your PAV configuration and adjust it to workload changes. The DS8000 in conjunction with z/OS has the ability to meet the highest performance requirements.

The DS8000, thanks to Multiple Allegiance (MA), can accept multiple parallel I/O requests from different hosts to the same device address, increasing parallelism and reducing channel overhead. The requests are accepted by the DS8000 and all requests are processed in parallel, unless there is a conflict when writing data to the same extent of the count key data (CKD) logical volume. Still, good application access patterns can improve the global parallelism by avoiding reserves, limiting the extent scope to a minimum, and setting an appropriate file mask, for example, if no write is intended.

In particular, different workloads (for example, batch and online) running in parallel on different systems can have an unfavorable impact on each other. In such cases, Multiple Allegiance can dramatically improve the overall throughput.

For additional information, see 14.3.1, “Introduction to the IBM DS8000 series features and functions” on page 339. To increase the I/O throughput in your system, the following publications contain excellent material in much more detail:

- High Performance FICON (zHPF), which is a FICON protocol and system I/O architecture that results in improvements for small block transfers.

  More information can be found at:

  `http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FQ127122`

- Modified Indirect Data Address Word (MIDAW), which is a modification to a channel programming technique reduce the number of frames and sequences flowing across the link, which makes the channel more efficient.

  More information can be found at:

http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp4201.html?Open

For more detailed information about PAVs, refer to section 15.2., “Parallel Access Volumes” in *DS8000 Performance Monitoring and Tuning,* SG24-7146.

2.5 Security

The focus of the co-location project is on the benefits of combining transactional and data warehouse workloads and databases on a single IBM System z and managing that system's resources as a single entity. But it is also important to highlight a few points about the security benefits of combining the transactional and data warehouse workloads and databases on a single System z. Among one of the many strengths of the System z are its security and integrity.

The configuration documented for the co-location project has the main infrastructure residing on a System z. That infrastructure includes DB2 for the transactional and data warehouse running on z/OS and IBM WebSphere for z/OS handling the workload for the transactional database. Depending upon the scenario, the databases can be in a single LPAR or multiple LPARs. z/VM is running on its own LPAR supporting multiple Linux guests. IBM Cognos BI is running on one of those Linux guests and handling the data warehouse queries submitted by the users. Those queries are then run against DB2 on z/OS. Finally the Extract, Transform, and Load (ETL) processing is handled by ISWz running on another Linux guest. The ETL processing extracts the data from the transactional database in DB2 for z/OS, transforms the data to match the warehouse schema and loads the data into the data warehouse database also in DB2 for z/OS.

All of the z/OS and z/VM images are running on z LPARs on a single System z. Why is this important? The main network infrastructure for these LPARs is a technology called HiperSockets. HiperSockets is a logical LAN technology. What this means is that TCP/IP is still used to transfer the data, but z microcode detects that data is being sent through a HiperSocket between z LPARs, and then transfers the data in-memory. Because the data is moved between LPARs in-memory, this data never leaves System z. Therefore, it cannot be compromised as it might be when the data is transferred outside of System z on other systems in the network. Also, since it does not leave the system during the transfer, there is no real need to encrypt the data when transferring, thus speeding up the transfer and reducing encryption key management issues.

So, in addition to performance, simplicity, network availability, and cost effectiveness, HiperSockets provides a secure means of transferring data between LPARs.

Encryption of the database data stored on storage devices is critical to the security and confidentiality of that data. There are a couple of different means of encrypting data when using DB2 on z/OS. First is to use the built-in encryption mechanisms that come with the DS8000 line of storage devices. Data for all DB2 objects goes through encryption when it is read or written to the device. The encryption of the raw data occurs on the DS8000, not on System z.

The second means of encryption is to use the Data Encryption for DB2 and IMS product. This product uses the Cryptographic services that ship with z/OS to encrypt and decrypt the data. The data is encrypted or decrypted at the DB2 entity level before it is written to the DS8000.

Whichever means of encryption is chosen, what is important is that by co-locating your transactional and data warehouse data on a System z, both the transactional and data warehouse data can be equally secure. Also, a single point of administering the encryption

setup can eliminate differences in the encryption policy by using a single person or a small group to administer that policy.

IBM Cognos BI can interact with an LDAP server for authentication of users when they log in to IBM Cognos BI to issue queries. The user database that supports this LDAP server can be RACF running on z/OS. The LDAP server allows for the authentication of users to IBM Cognos BI, but not the authorization of resources within IBM Cognos BI. By using RACF for the user database, the user database can be administered by a single z/OS RACF administrator, who would also control the RACF database for z/OS (this user database could be one and the same). The benefit of using RACF as the user database is a reduction in the number of user databases that need to be administered and also helps to have a single point of administration.

Trusted context with DB2 for z/OS is another means of securing your transactional and data warehouse DB2 environments. With TCP/IP clients coming into DB2 for z/OS, you can specify specific IP addresses that are allowed to connect to your DB2 subsystem. The trusted context is associated with a user role, and this role is then granted privileges in DB2 using the appropriate DB2 DDL. When a trusted connection (as defined by the trusted context) is made, it is associated with a role and that role is used to determine what privileges that connection has. So for example, with IBM Cognos BI running on Linux, the DB2 for z/OS administrator would specify the IBM Cognos BI Server's IP address in a trusted context. The role for that trusted context would be used to grant privileges to the data warehouse data. If IBM Cognos BI queries or any other type of data warehouse query comes in from a different IP address into DB2, DB2 does not allow access to the data specified in the query because there is no role associated with the request that has been granted privileges for the data warehouse data.

Although trusted contexts work for any network client, in this instance the IBM Cognos BI Server's TCP/IP connection, running on Linux, can be classified as trusted, thus helping to ensure the security of your data warehouse data and not allowing rogue applications to connect and access that data.

Using trusted contexts along with RACF's SERVAUTH and TCP/IP's NetAccess functionality, applications accessing DB2 through a network, subnetwork, or hosts can be granted privileges in DB2 using rules that are defined in DB2. This support requires functionality in multiple products, first TCP/IP's NetAccess must be able to map an IP address, network, subnetwork, or host to a RACF SERVAUTH name. The SERVAUTH class must be activated in RACF and the appropriate permissions given. And finally the SERVAUTH name must be identified to DB2 through a trusted context definition. So, instead of granting DB2 privileges to a single IBM Cognos BI Server, a group of IBM Cognos BI Servers on a network can be given access, allowing for more throughput and easier security administration.

Another means of securing your DB2 data from rogue applications on the network is through IP filtering. IP filtering is a means for a TCP/IP administrator to set up rules for incoming packets. IP filters are rules defined to either discard or permit packets into the system. IP filtering matches a filter rule to data traffic based on any combination of IP source or destination address (or masked address), protocol, source or destination port, direction of flow, or time. IP filtering can control traffic being routed, or control access at the host that has the communication endpoint. In this case the communications endpoint would be DB2 on z/OS.

When managing multiple DB2 subsystems in a data sharing group, using the RACF access control module can help significantly by reducing the security setup. The co-location study was done with some scenarios having four DB2 subsystems in a data sharing group. By letting RACF do the authorization of users for access to DB2 entities, you may grant access at the data sharing group level, thus eliminating the need to grant access for the DB2 entities for

each DB2 subsystem. This helps in a couple ways, first by reducing the authorization administration that needs to happen because this administration is done for the data sharing group, not for each DB2 subsystem in the data sharing group. Second, security access for the system and DB2 is through RACF and can be managed with a single security policy and possibly a reduced number of administrators.

The emphasis, for some of the items in the security section, is on putting the security for your DB2 subsystems under the control of RACF. By doing so, this allows you to monitor a single audit trail for security issues with your system, as opposed to spreading the audit trail across multiple security products possibly running on multiple systems. This makes it much easier to monitor the systems for security issues and making the systems more secure and requiring less work to audit.

For additional information on these topics, refer to the following publications:

- Data Encryption for IMS and DB2. IBM Techdoc “Data Encryption for IMS and DB2 and zSeries Crypto interaction: Overview and Short Explanation on Use of Crypto Keys”, Document number: PRS2890.
- DS8000 disk encryption. *IBM System Storage DS8700 Disk Encryption Implementation and Usage Guidelines*, REDP-4500.
- RACF Access Control Module. IBM SHARE presentation at:

  ftp://ftp.software.ibm.com/eserver/zseries/zos/racf/pdf/r07_racf_db2_overview.pdf

2.6 The specialty engines available on System z

The System z machine can be configured with a maximum of 80 processors. All of the existing processors are physically identical. In general terms, a central processor (CP) is referred to as a general purpose CP, meaning that all kinds of workloads on a System z machine can be processed by those CPs. However, each of the processors can be classified as a specialty engine (SE). There are four types of SEs available in a System z machine:

- System Assist Processor (SAP), introduced in 1994
- Integrated Coupling Facility (ICF), introduced in 1997
- Integrated Facility for Linux (IFL), introduces in 2000
- System z Application Assist Processor (zAAP), introduced in 2004
- System z Integrated Information Processor (zIIP), introduced in 2006

The number of SEs in a System z machine cannot exceed the number of general processors in a physical server. However, an LPAR can be configured to contain more SEs than general purpose CPs. For example, you might want to use three IFLs to run your Linux on the System z LPAR, but no general purpose CPs are required.

It is important to understand that there is neither a technical difference in a processor classified as an SE nor a difference in performance if data is processed on a general purpose CP or on an SE that is eligible for a specific workload. The main reason for using Specialty Engines is to cut costs. While general purpose CPs are charged based on the average usage, SEs are purchased once, therefore allowing you to run all eligible workloads without any limitations. Furthermore, general purpose CPs can be capped (in terms of MIPS) whereas SEs are always running at full speed.

Not all types of workloads are eligible for being processed on an SE. Certain workloads are designed for those types of processors, freeing up general purpose CP capacity for other workloads.

2.6.1 General purpose CPs

All types of workloads can be processed by using a general purpose CP. However, be aware that utilizing general purpose CPs to process workloads that are eligible to be processed on one of the available SEs can result in higher costs because general purpose CPs are basically paid by usage. This is also the biggest differentiator from the SEs, which are purchased once and can be utilized as much as possible using eligible workloads without increasing costs. The other good news is that once you purchase an SE, upgrading a machine provides you with the increased performance for the SE without any additional charges and without purchasing the SEs again.

2.6.2 System Assist Processor

The System Assist Processor (SAP) was the first specialty engine. It manages the starting and ending of I/O operations for all LPARs and all attached I/O subsystems. The SAP is standard on IBM System z servers (there is at least one per CPC) and is a dedicated I/O processor to help improve efficiencies and reduce the overhead of I/O processing of every IBM System z logical partition regardless of the operating system (z/OS, z/VM, Linux, z/VSE, or z/TPF). For I/O intensive workloads, an additional one or more SAPs can be purchased (model dependent). Technically, a SAP only runs Licensed Internal Code (LIC) and is part of all System z servers.

2.6.3 Integrated Coupling Facility

The Integrated Coupling Facility is the second SE that was introduced, in 1997. The initial Parallel Sysplex implementation required external Coupling Facilities (CFs) in separate machines. Most installations incorporated two different external Coupling Facilities to avoid a single point of failure. See Figure 2-1.

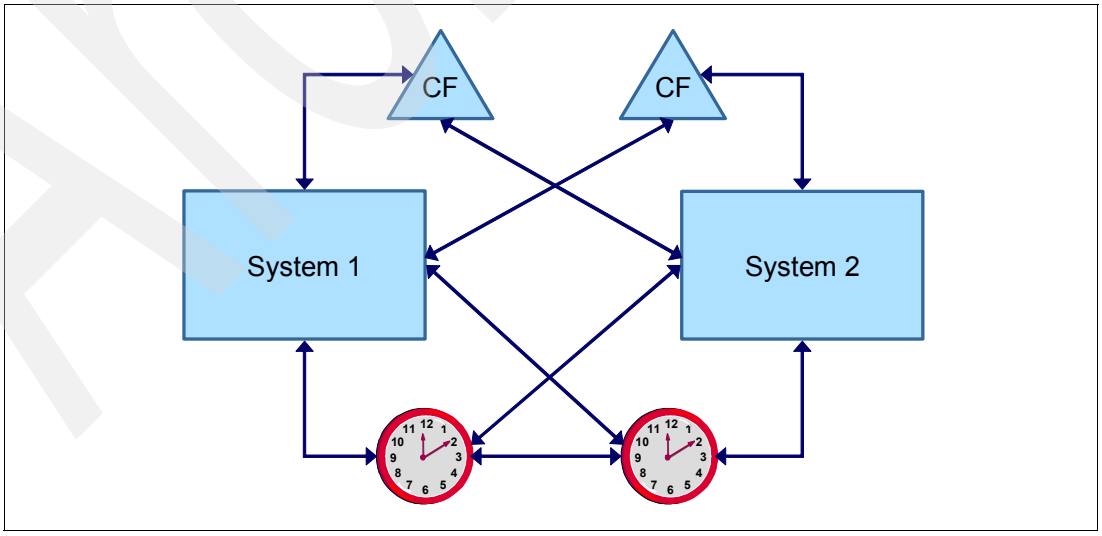


Figure 2-1 Parallel Sysplex environment with two External Coupling Facilities

With the introduction of the ICF it became possible to use ICFs on central processor complexes (CPCs) and to eliminate additional hardware. See Figure 2-2.

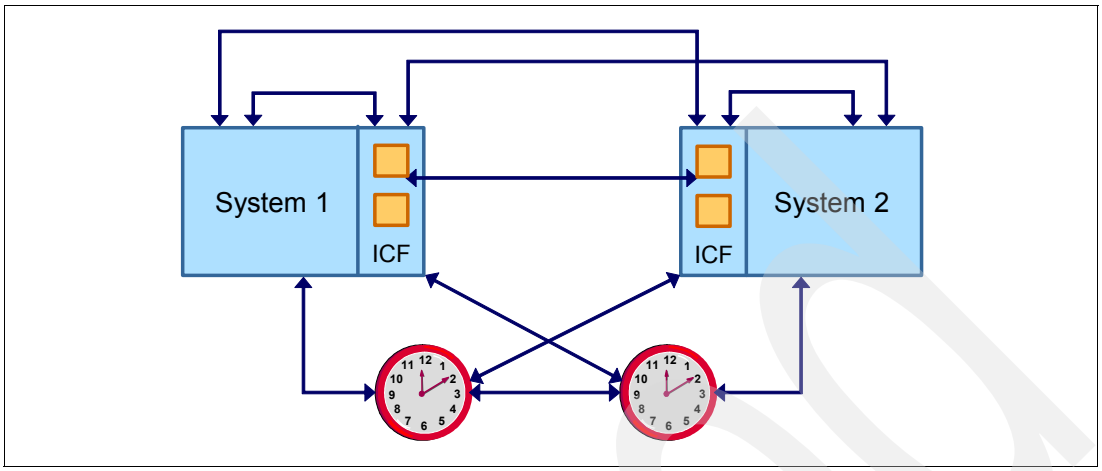


Figure 2-2 Parallel Sysplex environment with two Internal Coupling Facilities

An ICF provides additional processing capacity for the execution of the Coupling Facility Control Code (CFCC) in a CF within a System z machine.

2.6.4 Integrated Facility for Linux

The Integrated Facility for Linux (IFL) is a processor dedicated to Linux on System z workloads, introduced in 2001. The IFL processor allows you to purchase additional processing capacity for Linux workloads without increasing IBM software charges for the existing System z environment. No other workloads can be processed on an IFL. However, Linux on System z workloads can also be processed by general purpose CPs at a higher cost.

Note: IFLs cannot be mixed with standard processors in a partition.

2.6.5 System z Application Assist Processor

The System z Application Assist Processor can run Java and z/OS XML workloads such as WebSphere Portal workloads. It was first introduced in 2004. Utilizing a zAAP can lead to huge parts of the workload being routed to this type of SE. Dedicated to z/OS Java execution, it enables you to integrate e-business Java workloads with core business applications and data, safely, securely efficiently and cost effectively. When configured with central processors within logical partitions running z/OS, zAAPs may enable you to simplify and reduce server infrastructures and integrate e-business Java web applications next to mission-critical data for high performance, reliability, availability, and security.

Note: Some restrictions apply to zAAPs. They cannot be IPLed and they can only execute z/Architecture mode instructions. They do not support all manual operator controls such as PSW Restart, LOAD, or LOAD derivatives from files, CDROM, or servers.

2.6.6 System z Integrated Information Processor

The System z Integrated Information processor is the latest specialty engine, introduced in 2006. It is the most important processor for DB2 for z/OS workloads, since huge parts of DB2 for z/OS workloads can be routed to a zIIP processor for execution. The technical reason for workloads being eligible to be processed on a zIIP is determined by the Service Request Block (SRB) used to execute a workload. Whenever a workload is executed using an enclave SRB, this part of the workload is eligible to be offloaded to a zIIP processor.

In DB2 9 for z/OS, four different types of workload are eligible for being routed to a zIIP:

- DB2 remote access via TCP/IP using Distributed Relational Data Architecture (DRDA)
- Complex parallel queries
- DB2 utilities for index maintenance
- Native remote SQL procedures

Enabling parallelism is the major contributor to exploitation of your zIIP capacity. Requests that utilize large parallel queries, including star schema queries, may have portions of this SQL request directed to a zIIP. If you are curious about how many parts of your existing workload could be zIIP eligible, the parameter PROJECTCPU in member IEAOPT00 in PARMLIB allows you to monitor your possible zIIP usage in a what-if scenario. Even if you already have a zIIP processor configured in your system, this can give you an indication of whether your installation could benefit from more zIIP processors. See Figure 2-3.

Query	Elapsed Time	CP Time	zIIP Time	Redirected to CP	CPU Total
Q1	02:14:14,61	08:26:11,17	04:25:53,15	05:51:45,92	12:52:04,32
Q2	00:01:44,30	00:01:41,87	00:02:34,19	00:00:50,10	00:04:16,06
Q3	00:31:42,70	00:50:10,12	00:38:16,69	00:24:18,61	01:28:26,81
Q4	00:17:37,01	00:15:56,27	00:33:13,09	00:06:06,28	00:49:09,36
Q5	00:25:31,53	00:35:23,82	00:26:20,46	00:11:56,23	01:01:44,28
Q6	01:26:04,08	01:19:44,30	00:00:00,00	00:00:00,00	01:19:44,30
Q7	00:52:30,47	02:55:58,58	01:35:43,10	01:55:42,09	04:31:41,68
Q8	06:28:36,31	00:13:28,71	00:23:55,90	00:03:52,64	00:37:24,61
Q9	00:30:11,04	00:34:27,22	00:33:53,57	00:14:55,42	01:08:20,79
Q10	00:23:30,64	00:34:48,41	00:23:20,02	00:15:39,99	00:58:08,43
Q11	00:17:02,56	00:45:05,20	00:33:40,40	00:29:19,97	01:18:45,60
Q12	00:55:01,11	02:06:01,93	01:01:05,61	01:10:27,33	03:07:07,54
Q13	00:28:28,40	00:54:47,28	00:30:23,01	00:25:51,30	01:25:10,29
Q14	00:29:50,58	01:16:35,39	00:47:39,54	00:47:19,48	02:04:14,93
Q15	00:11:45,02	00:22:04,82	00:09:48,92	00:08:21,78	00:31:53,74
Q16	09:51:42,72	00:03:31,45	00:00:00,60	00:00:00,00	00:03:32,05
Q17	01:02:42,38	03:06:55,74	01:44:46,46	02:08:31,17	04:51:42,20
Q18	02:26:23,95	00:01:20,98	00:04:49,10	00:00:00,02	00:06:10,08
Q19	90:56:56,00	01:44:45,58	06:58:02,43	00:00:01,98	08:42:48,01
Q20	00:02:58,30	00:10:23,33	00:05:52,62	00:06:58,90	00:16:15,95

Figure 2-3 General purpose CP and zIIP times for tested queries

Looking at query Q1, the information you can derive from our experiments tells you that the elapsed time for query Q1 was 2 hours, 14 minutes and 14.61 seconds. The total CP time for this query is noted as 12 hours, 52 minutes and 04.32 seconds. Due to huge exploitation of parallelism, the CP time dramatically exceeds the elapsed time to complete the request. While 08 hours, 26 minutes and 11.17 seconds of CP time have been spent on a general

purpose CP for this query, 04 hours, 25 minutes and 53.15 seconds have been spent on a zIIP processor for this query.

The column labeled Redirected to CP contains another important piece of information: 05 hours, 51 minutes and 45.92 seconds of CP time have been redirected from the zIIP processor to a general purpose CP, meaning that 05 hours, 61 minutes and 45.92 minutes of CP time have been zIIP eligible, but all zIIP processors were busy so that the work had to be routed to a general purpose CP. This projection gives you a good estimate whether you could benefit from additional zIIP processors in your system, even if some zIIPs are already installed.

Note: A single query with a high degree of parallelism can consume all the processor capacity of a system. To avoid queries monopolizing your system, it is very important to tune your WLM policy to control parallel query processor consumption.

Once you enable zIIP processors in your environment, eligible workloads are automatically routed to the zIIP processor. However, there can be situations, depending on the workload in your system, when all available zIIP processors are busy. In this case, zIIP eligible work can either be redirected to a general purpose CP (which is the default beginning in z/OS x.y) or join the queue waiting for the next zIIP processor to become available. The decision which option is in effect is derived from the parameter IIPHONORPRIORITY; the following options are available:

- YES to allow work to be redispatched to a general purpose CP
- NO to ensure that all portions of zIIP eligible work will be processed on a zIIP

Important: The value of the parameter IIPHONORPRIORITY can have a dramatic impact on performance of your zIIP eligible work. Set it to a value that will satisfy your processing requirements in terms of performance and throughput.

Once zIIP processors are installed in your system and IIPHONORPRIORITY is set to YES, counters will be available in both WLM reports as well as in Omegamon Performance Expert reports showing the amount of processor time that was rerouted from a zIIP processor to a general purpose CP, as shown in Example 2-1.

Example 2-1 CPU time distributions extract from OMEGAMON® XE for DB2 PE on z/OS

```
CP CPU TIME   13:28.7168
[...]
 IIPCP CPU    3:52.64263
[...]
IIP CPU TIME  23:55.9036
```

For this example, a zIIP eligible query has been traced and zIIP processors have been enabled in the system. The example shows a standard CPU time in field CP CPU TIME of 13:28.7168 minutes, a zIIP CPU time in field IIP CPU TIME is 23:55.9036 minutes, and the zIIP-eligible CPU time, which was rerouted to a general purpose CP for this query, is shown as IIPCP CPU with a value of 3:52.64263 minutes.

If you have enabled the redirection of zIIP eligible workloads to your general purpose CPs and a high value of IIPCP CPU is regularly shown in either your WLM or OMPE reports, this can be a good indicator whether one or more additional zIIP processors can pay off very quickly in your system.

IBM continues to improve specialty engines and makes it possible to redirect even more work to them. For example, since z10 and z/OS 1.10, Hipersockets interface can move multiple output data buffers in a single write operation. This new feature can be redirected to zIIPs. Go to:

http://publib.boulder.ibm.com/infocenter/zos/v1r10/index.jsp?topic=/com.ibm.zos.r10.halz002/hipercc.htm

2.6.7 zAAP on zIIP capabilities

Especially in DB2 for z/OS subsystems dedicated to data warehousing, many installations can greatly benefit from zIIP specialty engines, while there might not be enough Java workload on the same machine to justify the purchase of a zAAP specialty engine. Beginning with z/OS 1.11, a new capability was provided that can enable zAAP-eligible workloads to run on zIIP processors when no zAAP processors are installed on the machine. This capability allows you to run zIIP- and zAAP-eligible workloads on only one type of specialty engine: the zIIP processor.

APAR OA27495 introduces a new IEASYSxx parameter, ZZ, with the available options to enable your system for zAAP on zIIP processing.

Note: The capability of zAAP on zIIP was retrofitted to both z/OS 1.9 and 1.10 with a PTF for APAR OA27495 and is available for System z9 and z10 machines.

The ZZ parameter determines whether the system can run zAAP-eligible work on zIIP processors when no zAAP processors are installed on the machine. The two options are:

- ZZ=YES
- ZZ=NO

When you specify ZZ=YES, the system can run zAAP-eligible work on zIIP processors when no zAAP processors are installed on the machine. When you specify ZZ=NO, the system cannot run the zAAP-eligible work on zIIP processors when no zAAP processors are installed on the system. The default value is NO.

Note: The default of ZZ is NO in z/OS V1R9 and z/OS V1R10; the default is YES as of V1R11.

The main intention of this feature is to address clients without enough zAAP- or zIIP-eligible workloads to justify the purchase of a specialty engine today. Additionally, this can also be beneficial for clients that have only zIIPs installed in their environment today, planning to bring additional Java workloads to System z.

Tip: For any questions regarding zAAP on zIIP, contact your local IBM representative to find out about the details regarding your specific situation. For additional details on zIIPs and zAAPs, see 11.4.1, “zIIP specialty engines” on page 287 and 11.4.2, “zAAP on zIIP” on page 288.

2.7 IBM Smart Analytics Optimizer

System z was created more than 40 years ago, and in the beginning there were a lot of constraints in terms of processor capacity and I/O throughput. All systems have been tuned to

get the most out of them in terms of performance, regarding both processor utilization and I/O throughput. Tuning an application can still be a very time-consuming task, especially because systems tend to increase in complexity rather than simplifying the existing infrastructure. Even if the vast majority of all systems out there are well tuned, there are some challenges in modern Business Intelligence (BI) and Data Warehouse environments which have only been solved partially. One of the major challenges we face today are dynamic queries coming into the system, with an unpredictable amount of time to complete the request—more importantly, with an unpredictable amount of resource consumption.

Since BI applications increasingly play a vital role in many companies, it is essential to decrease the response time of those dynamic queries and to prevent those queries from consuming all your system resources and, in a worst case, block these resources from being used by your OLTP workloads driving your transactional business. Since the resource consumption can be addressed by ensuring that appropriate WLM policies are in place, minimizing the effect of gobbling resources for those queries will lead to longer query execution times. For further details, see 11.3, “Analysis and verification of a working WLM policy” on page 279.

While there are plenty of reports where the immediate availability of the result set is not most likely going to affect your stock quote, there are also an increasing number of reports that are needed by upper management that could potentially influence a business decision. For those business critical queries, it can be very important to have the shortest response time possible. WLM policies can influence the priority of the workload of both processor performance and I/O throughput, but they cannot accelerate the spinning of the underlying DASD devices if large amounts of data have to be scanned.

Relational Database Management Systems (RDBMS) are most widely used as sophisticated data repositories that usually have a huge ecosystem of different applications built on top of them. In many cases, the infrastructure most widely utilized is optimized for OLTP processing. Even if a well designed data warehouse data model is in place, scanning gigabytes or even terabytes of data to complete a BI query request can take a large amount of time to complete.

Traditional RDBMSs attempt to address those performance and scalability challenges with their standard tools of trade such as indexing, prebuild aggregates as with materialized query tables (MQT), and SQL optimization to favor available indexes for access paths. All these attempts require very sophisticated tools and highly skilled database administrators and application programmers to optimize performance and solve performance problems.

Unfortunately, due to the ad hoc and unpredictable nature of data warehouse and business intelligence query requests, this approach can increasingly fail to address all possible performance bottlenecks in advance, still keeping the laws of physics for I/O-bound queries in mind. To overcome the current limitations of spinning disks, a new approach is required to store large amounts of data if it has to be queried within a short time.

To address these issues, IBM has conducted a research project to enable query acceleration for long-running online analytical processing (OLAP) queries by orders of magnitude, the Blink project. For more details, refer to:

`http://www.almaden.ibm.com/cs/projects/blink/`

One of the main goals of the researchers in this project was to provide consistent query response times regardless of specific query structures accessing the underlying data. This performance-oriented goal is addressed through implementing leading technology trends: hybrid row/column stores in main memory and predicate evaluation on compressed data are combined with multicore and vector-optimized algorithms. In fact, the Blink project laid the foundation for the development of the IBM Smart Analytics Optimizer.

Technically, the IBM Smart Analytics Optimizer consists of one or more blade centers, each equipped with hundreds of processors and main memory to store thousands of GBs of raw data, while being attached to a System z machine. It keeps a snapshot of data stored in a data warehouse in a highly compressed and scan-optimized form in main memory of the blades, allowing dynamic queries fulfilling certain criteria to be accelerated by accessing the data in main memory of the IBM Smart Analytics Optimizer rather than accessing the data on DASD. It is important to understand that those blades being connected to a System z machine will not touch your existing Service Level Agreements (SLAs) for your System z environment. The IBM Smart Analytics Optimizer is a highly reliable and very robust solution, providing its own internal failover mechanisms as well as fencing DB2 for z/OS against possible and unexpected failures in the blade center.

The IBM Smart Analytics Optimizer acts as another resource manager for DB2 for z/OS, such as Data Manager (DM), Buffer Manager (BM), Internal Resource Lock Manager (IRLM) or the Log Manager. Therefore, it is a highly integrated solution within DB2 for z/OS and the users. Neither required are any changes to existing applications which already connect to DB2 for z/OS. Instead, the DB2 for z/OS optimizer is aware of an IBM Smart Analytics Optimizer existence in a given environment and can route a given query either to the IBM Smart Analytics Optimizer or to DB2 for z/OS for execution. This mechanism guarantees that queries are executed in the most efficient way irrespective of their type (OLTP or OLAP). See Figure 2-4 for a brief overview of the IBM Smart Analytics Optimizer's integration into a DB2 for z/OS environment.

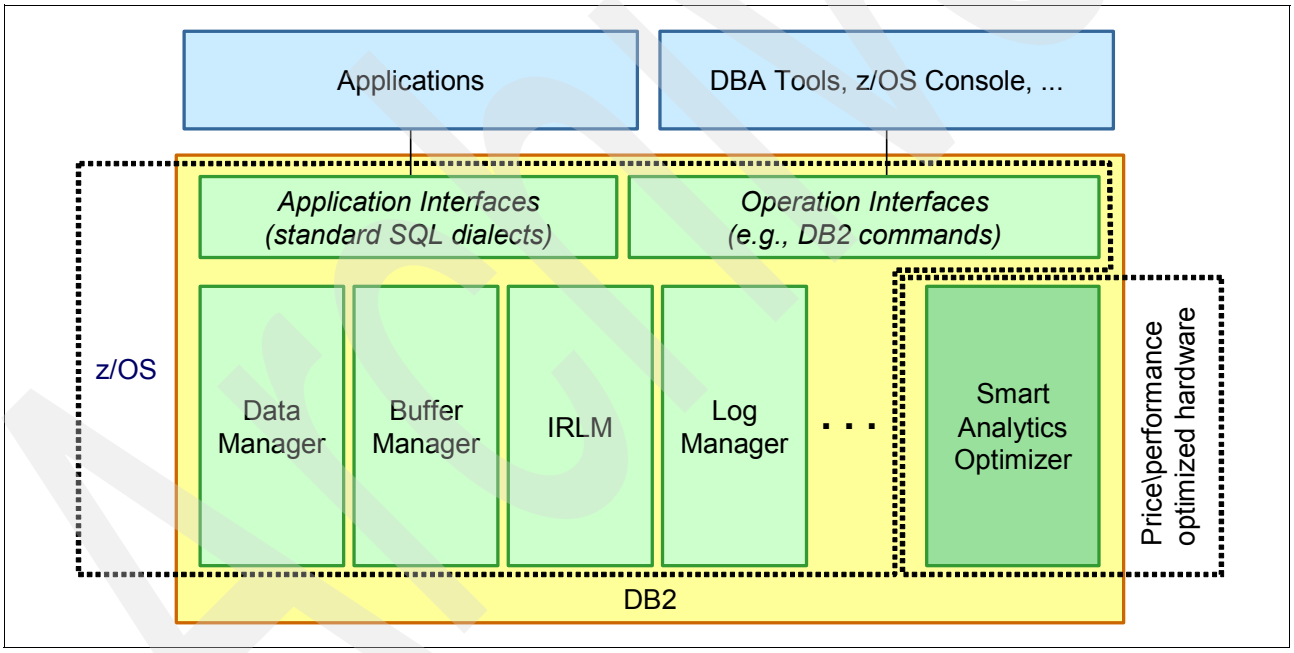


Figure 2-4 IBM Smart Analytics Optimizer platform view

Depending on the configuration of the IBM Smart Analytics Optimizer, each IBM Smart Analytics Optimizer consists of a small number of coordinator nodes and a larger number of worker nodes. The IP addresses of the coordinator nodes are listed after the IPNAME. It is recommended that those IP addresses only be reached by the system that the IBM Smart Analytics Optimizer is connected to. In Figure 2-4, we use a configuration of 14 blades, three of them being entitled as coordinator nodes and 11 as worker nodes. Besides the processing capacity of the IBM Smart Analytics Optimizer, more information is available about the current status of the blades, including the average processor utilization and the memory being available for further data storage in the IBM Smart Analytics Optimizer.

Once an IBM Smart Analytics Optimizer is installed in a system, its integration can be shown using the new DB2 for z/OS command DISPLAY ACCELERATOR. See Example 2-2 for the information provided by this command.

Example 2-2 Smart Analytics Optimizer information available using -DIS ACCEL(*) DETAIL

```
DSNX810I  -DWA1 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DWA1 DSNX8CDA
ACCELERATOR                      MEMB  STATUS  REQUESTS QUED MAXQ FAIL
-------------------------------- ---- -------- -------- ---- ---- ----
H104                             DWA1 STARTED         0    0    0    0
PORT=1400 IPNAME=00010C01
IPADDR=10.104.1.242 HEALTHY
IPADDR=10.104.1.243 HEALTHY
IPADDR=10.104.1.244 HEALTHY
INCREMENTAL STATISTICS SINCE 15:35:34 OCT 30, 2009
  STATUS = FULLY OPERATIONAL
  AVERAGE QUEUE WAIT                                =        0
  MAXIMUM QUEUE WAIT                                =        0
  PROCESSING CAPACITY                               =  1314278
  TOTAL NUMBER OF PROCESSORS                        =      224
  AVERAGE CPU UTILIZATION ON COORDINATOR NODES      =        0
  NUMBER OF ACTIVE COORDINATOR NODES                =        3
  PHYSICAL MEMORY AVAILABLE ON COORDINATOR NODES    =    47748 MB
  PHYSICAL MEMORY AVERAGE USAGE ON COORDINATOR NODES =      526 MB
  AVERAGE CPU UTILIZATION ON WORKER NODES           =       59
  NUMBER OF ACTIVE WORKER NODES                     =       11
  PHYSICAL MEMORY AVAILABLE ON WORKER NODES         =    47504 MB
  PHYSICAL MEMORY AVERAGE USAGE ON WORKER NODES     =      403 MB
  SHARED MEMORY DATA AVAILABLE ON WORKER NODES      =    16038 MB
  SHARED MEMORY DATA AVERAGE USAGE ON WORKER NODES  =     7916 MB
  MAXIMUM SHARED MEMORY DATA IN USE ON WORKER NODES =     8171 MB
DISPLAY ACCEL REPORT COMPLETE
DSN9022I  -DWA1 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION
```

The output provides important information about the status of the IBM Smart Analytics Optimizer and the current content of the heartbeat of the IBM Smart Analytics Optimizer, which is stated as HEALTHY in our example for all available coordinator nodes. A status of HEALTHY indicates that the IBM Smart Analytics Optimizer can be pinged on the IP addresses provided and is ready to execute queries that are routed to the IBM Smart Analytics Optimizer for execution. The IBM Smart Analytics Optimizer operates based on Linux and the processing capacity shown above is stated in BogoMips, which is basically a measurement of processor speed performed by a Linux kernel at boot time. The CPU utilization ranges from 0 to 10000, where 0 means no utilization and 10000 full CPU utilization. Note that the value states the average utilization within the last 60 seconds and not the current utilization of the IBM Smart Analytics Optimizer. Additionally, the value of physical memory available in the nodes gives you an indication of how much more data you can offload to the IBM Smart Analytics Optimizer for query acceleration.

It is important to understand that not all data of a data warehouse is stored in main memory of the IBM Smart Analytics Optimizer, but only those tables that have been identified as targeted by queries that qualify to execute on the IBM Smart Analytics Optimizer. Eligible queries are mostly long-running OLAP queries, not any short-running OLTP queries. Tables being accessed by long-running OLAP queries will be offloaded as a snapshot to the IBM Smart Analytics Optimizer. What is identified and stored in the IBM Smart Analytics Optimizer is a set of tables that are related to each other. One example of those related tables is a star

schema. In data warehouse environments, you typically have a large fact table and a lot of smaller dimension tables. One set of those tables is referred to as MART, and a MART is technically what is going to be offloaded to the IBM Smart Analytics Optimizer. See Figure 2-5 for an example about a MART as it would be stored in the IBM Smart Analytics Optimizer.

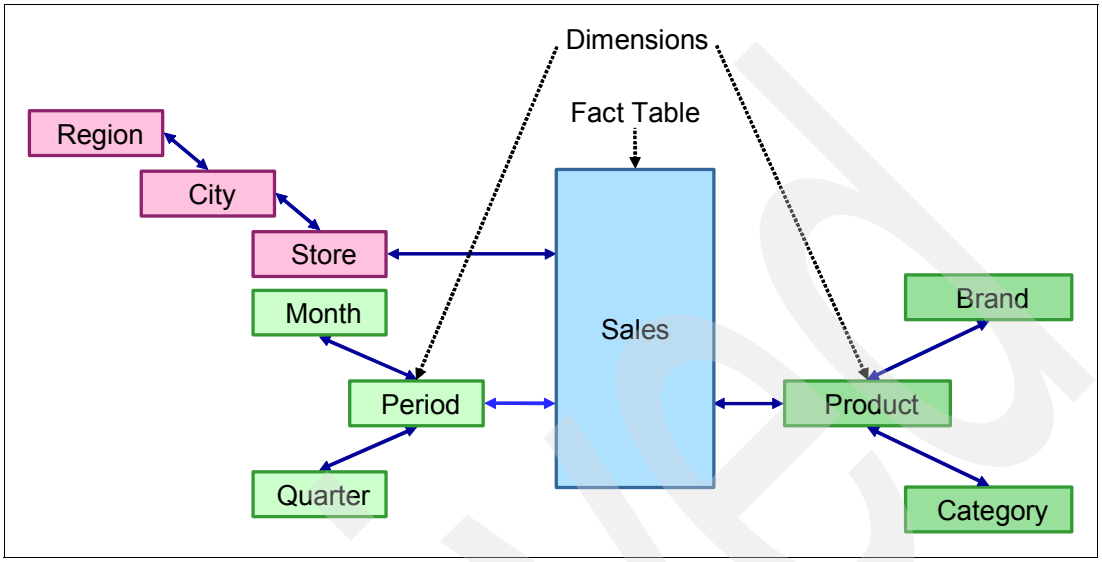


Figure 2-5 MART example for the IBM Smart Analytics Optimizer

A set of tables belonging to the same MART can be identified using a graphical user interface (GUI). After identifying the relationships between those tables either by automatically detecting referential constraints or by manually defining the relationships between those tables, a MART can be loaded into the IBM Smart Analytics Optimizer. Unloading the data is performed by using the UNLOAD utility and the utilization of USS pipes to send the data to the IBM Smart Analytics Optimizer.

Usually, only a subset of all data stored in a data warehouse is related to long-running OLAP queries. Therefore, the total size of all data warehouse data does not necessarily indicate how many tables can be stored in the IBM Smart Analytics Optimizer and how many queries of a given installation can be accelerated. It is recommended that only those portions of data are defined and offloaded as a MART to the IBM Smart Analytics Optimizer which can greatly benefit from a dramatic increase of performance.

Note: Data stored in a data warehouse within DB2 for z/OS is usually compressed. Because the IBM Smart Analytics Optimizer uses a compression algorithm other than DB2 for z/OS, you will observe more data being sent via USS pipes during an unload process to the IBM Smart Analytics Optimizer than is currently used on DASD in your DB2 for z/OS subsystem. Additionally, you may observe a different compression ratio for the same data within the IBM Smart Analytics Optimizer compared to DB2 for z/OS.

Loading data into the IBM Smart Analytics optimizer can exploit parallelism. The stored procedure used to trigger the UNLOAD utility executes on a partition level. The maximum degree of parallelism is therefore dictated by the number of partitions of a tablespace. In the JCL for the WLM address spaces for the stored procedures for this product you can find a line similar to the following:

```
Eaxmple n-m: Controlling the maximum number of UNLOAD utilities executing in
parallel
//AQTENV   DD DISP=SHR,DSN=DB91.NEW.SDSNSAMP(AQTENV)
```

The following line controls the maximum degree of parallelism used during the LOAD process of the accelerator:

```
MAX_UNLOAD_IN_PARALLEL=8
   # maximum number of parallel UNLOAD invocations
   # (1 for each partition in a table is theoretically possible,
   # but requires significant CPU availability)
```

Note: Loading data into the accelerator can be processor intensive. Choose a degree of parallelism that does not interfere with the resources needed for non-load processing.

The data being unloaded is stored in an uncompressed form in temporary files on DASD, occupying the space needed for the amount of data being unloaded in an uncompressed form.

Important: Make sure that enough temporary space is available to store the data sets that are unloaded to the IBM Smart Analytics Optimizer.

Once the process is started to unload the data to the IBM Smart Analytics Optimizer, chunks of data are stored among all available blades, thus allowing for massive parallel processing on the blades. The IBM Smart Analytics Optimizer can store as many MARTs as capacity allows, and within a given query only data residing in one MART is allowed to be accessed by a query which is routed to the IBM Smart Analytics Optimizer. This means that no SQL statements utilizing the IBM Smart Analytics Optimizer are allowed that are crossing a MART's boundary. Figure 2-6 shows a definition of two MARTs using the same dimension tables but different fact tables. In this example, if a query would access both fact tables, the query would be eligible to be routed to the IBM Smart Analytics Optimizer.

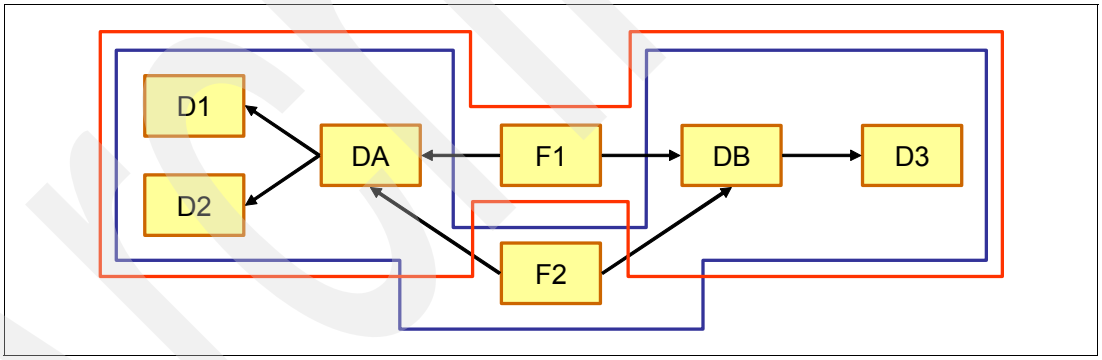


Figure 2-6 Multiple MART definitions

Whenever a MART is defined, the information about it is stored in the DB2 for z/OS catalog, enabling the optimizer to take these definitions into account when calculating the most efficient access path for a given query. The decisions made by the optimizer for routing queries accessing data that is also stored in the IBM Smart Analytics Optimizer are based on a cost estimate as for traditional DB2 for z/OS workloads. From an IBM Smart Analytics Optimizer point of view, a MART is also called an Accelerated Query Table (AQT).

Important: The IBM Smart Analytics Optimizer is designed to accelerate long-running OLAP queries. It is not designed to accelerate short-running OLTP queries. It is important to understand that a snapshot of data is stored and queried in the IBM Smart Analytics Optimizer as is very common for data warehousing environments. The original data will still remain in DB2 for z/OS. Accessing data either in the IBM Smart Analytics Optimizer or from disk is controlled by the special register CURRENT REFRESH AGE.

Since data warehouse and BI data in most cases is aggregated before providing the final results of a query, depending on your specific situation, it can be very likely that the result of a query is not going to be affected in a way such that the result changes significantly if non-snapshot data is queried. But this depends on the type of query and type of data in your SQL statements. If small deviations in the result set could be ignored, resulting from incremental changes to your data warehouse data before the MART is refreshed with the most current data, it is recommended to have long-running OLAP SQL statements enabled for being routed to the IBM Smart Analytics Optimizer.

Because the IBM Smart Analytics Optimizer stores a snapshot of parts of your data warehouse data, it can be important for some queries to access the data as it is stored within DB2 for z/OS since most current data can be mission-critical for specific queries. However, same as with MQTs, it is also possible to query the data as it is stored within DB2 for z/OS even if the query would be eligible to be routed to the IBM Smart Analytics Optimizer. Whether or not the optimizer will consider an existing MART definition within the IBM Smart Analytics Optimizer for query optimization is determined by the value of the special register CURRENT REFRESH AGE (same as with MQTs):

- ANY enables the IBM Smart Analytics Optimizer usage for the optimizer
- 0 disables the IBM Smart Analytics Optimizer usage for the optimizer

Example 2-3 shows both the syntax to enable and disable IBM Smart Analytics Optimizer usage for a query succeeding the statement shown within the same thread.

Example 2-3 Enabling or disabling IBM Smart Analytics Optimizer usage for a given query

```
Enable IBM Smart Analytics Optimizer usage:     SET CURRENT REFRESH AGE = ANY;
Disable IBM Smart Analytics Optimizer usage:    SET CURRENT REFRESH AGE = 0;
```

Additionally, you can retrieve the information if a query will be executed using a MART stored inside the IBM Smart Analytics Optimizer by obtaining the new information, which is now available in a new explain table named DSN_QUERYINFO_TABLE. See Example 2-4 for the DDL of DSN_QUERYINFO_TABLE where the qualifier needs to be adjusted to your current environment.

Example 2-4 New explain table added for the IBM Smart Analytics Optimizer

```
CREATE TABLE qualifier.DSN_QUERYINFO_TABLE (
      QUERYNO INTEGER NOT NULL WITH DEFAULT ,
      QBLOCKNO SMALLINT NOT NULL WITH DEFAULT ,
      QINAME1 VARCHAR(128) NOT NULL WITH DEFAULT ,
      QINAME2 VARCHAR(128) NOT NULL WITH DEFAULT ,
      APPLNAME VARCHAR(24) NOT NULL WITH DEFAULT ,
      PROGNAME VARCHAR(128) NOT NULL WITH DEFAULT ,
      VERSION VARCHAR(122) NOT NULL WITH DEFAULT ,
      COLLID VARCHAR(128) NOT NULL WITH DEFAULT ,
      GROUP_MEMBER VARCHAR(24) NOT NULL WITH DEFAULT ,
      SECTNOI INTEGER NOT NULL WITH DEFAULT ,
```

```
    SEQNO INTEGER NOT NULL WITH DEFAULT ,
    EXPLAIN_TIME TIMESTAMP NOT NULL WITH DEFAULT ,
    TYPE CHAR(8) NOT NULL WITH DEFAULT ,
    REASON_CODE SMALLINT NOT NULL WITH DEFAULT ,
    QI_DATA CLOB(2097152) NOT NULL WITH DEFAULT ,
    SERVICE_INFO BLOB(2097152) NOT NULL WITH DEFAULT ,
    QB_INFO_ROWID ROWID NOT NULL GENERATED ALWAYS
)
AUDIT NONE
DATA CAPTURE NONE
CCSID UNICODE;
```

If column TYPE = A and the REASON_CODE = 0, a query block successfully qualifies for offloading with the AQT specified in QINAME1 and QINAME2.

Important: The prerequisites for the IBM Smart Analytics Optimizer include System z196 or newer, DB2 9 for z/OS, and z/OS 1.10.

2.8 Bandwidth resources and co-location

Running a data warehouse requires a number of resources. Key among these are the pipelines required to load and access the data.

In the traditional warehousing approach, data has to be extracted from our data source of record (such as DB2) and moved to an LUW repository, where it is transformed and loaded into a data store capable of analytical processing. This analytical processing requires that users have network access to the application servers capable of executing the BI software, and these servers in turn require network access to the data servers containing the transformed data. Moving and accessing large amounts of data using this method are both time consuming and costly in hardware resources and CPU cycles. An example of this flow follows:

1. Data is extracted from the DB2 Database and placed in a format to transfer and transform.
2. This data must then be transmitted (usually over a TCP/IP network) to a processor that will transform it into a required format to match the client's warehouse layout.
3. The transformed data is then loaded into a database on the warehouse server.
4. If data in the warehouse needs to be viewed in real time, there will be a fairly continuous flow along these lines.

It should be noted that these extract, transform, and load (ETL) processes can occur in parallel. However, each and every server must have access to both disk storage and the network. This means that each server will require multiple fiber channel or SCSI-type connections as well as multiple TCP/IP connections. This is because redundancy is normally a requirement, along with the ancillary equipment such as routers, switches, and hubs. A significant amount of bandwidth has to be dedicated to this model of data warehouse. Much of this bandwidth would be specific to this application and so would effectively be superfluous for normal operations.

Using the ELT process and data warehouse co-location on System z, we can dramatically reduce the amount of physical bandwidth required to enable our warehouse.

Archived

First, let us look at the model where the data warehouse resides on the same LPAR as the OLTP DB2 database but as a separate subsystem. From a physical hardware point of view we will probably require additional disk storage to house the OLAP data. This can normally be added to existing DASD controllers, so in most cases the existing FICON infrastructure can be used. WLM allows us to prioritize I/O as well.

There is effectively no networking requirement for data in this scenario as all data movement would take place under the covers of the LPAR and as such would only utilize the FICON DASD infrastructure. With this model there could also be a reduction in the amount of physical I/O required to complete the load, because data would only need to be read once and written once to populate the OLAP database.

Secondly, if we place the OLAP database on a separate LPAR, we then require network connectivity to load the database. If the OLAP LPAR resides on a CPC that has an LPAR with a subsystem of the OLTP resource on it, we can utilize hipersockets to communicate between LPARs. Hipersockets provide us with a high-speed TCP/IP link between LPARs while not requiring any additional networking hardware or bandwidth. If the LPAR is remote from any of the OLTP subsystems, we would then use high-speed OSA adapters, giving us transfer speeds of up to 10 Gb. Normally CECS would be networked together, so we should be able to utilize the existing network infrastructure. By ensuring that we have proper quality of service prioritization, there is no risk of this data movement flooding the network.

User queries flowing into both models would utilize the existing System z network infrastructure, also reducing the need for unique network environments.

Co-location can not only drastically reduce the hardware requirements for both I/O and networking but also drastically reduce or even eliminate the bandwidth that needs to be reserved for this activity.

2.9 Continuous availability and disaster recovery

DB2 for z/OS is the database of choice for critical data for many enterprises. This is especially true in terms of serving as a data hub for many enterprises, acting both globally and locally. Providing a 24x7 availability throughout the whole year is very likely to be essential to guarantee business continuity for most companies. System z hardware and z/OS as the operating system, as well as DB2 for z/OS as a database management subsystem, lay the foundation for continuous availability. Additionally, high availability definitely needs to be designed into the applications as well to exploit the capabilities a System z infrastructure can offer.

On the other hand, developing applications to ensure high availability is only one part of the story. Leveraging DB2 for z/OS in a System z Parallel Sysplex environment provides your company with an availability of your IT infrastructure that is unmatched in the industry, thus allowing for taking only parts of a system down for scheduled maintenance while all other components of your system continue to process the current workload. This feature is essential to avoid outages for business-critical applications such as providing back-end services for automated teller machines.

System z can even take it one step further: Building a Parallel Sysplex environment and designing continuous availability into your application does not usually cover the aspects of an information disaster, following unplanned events such as floods or earthquakes when the entire system can go down. For these instances, a careful disaster recovery plan is needed to get the system up and running again in the shortest amount of time possible with a consistent copy and with minimal losses. Features such as system-level backup, system level recovery

and Geographically Dispersed Parallel Sysplex™ (GDPS) are designed to allow for switching computer operation from one site to another, with minimal impact.

It is not the intention of this book to cover all flavors of high availability and disaster recovery scenarios, but to point you to the right resources regarding this complex topic. Regarding the complex topic of high availability, *DB2 UDB for z/OS: Application Design for High Performance and Availability,* SG24-7134 covers a large variety of recommendations to increase the availability of your data, both including data modeling and application design considerations. A great source of information for dealing with disaster recovery scenarios is *Disaster Recovery with DB2 UDB for z/OS,* SG24-6370. See this IBM Redbooks publication for a detailed investigation into the different disaster recovery scenarios and possible solutions provided by System z and DB2 for z/OS to complete the picture of high availability and disaster recovery in mission-critical DB2 for z/OS environments.

2.10 Power consumption

Power consumed by equipment in the data center is one of the more critical issues facing IBM clients today. While the cost of the power is important, many clients must manage to power limits in their data centers, both limits in the physical power distribution capacity within the data center and possible limits imposed by power suppliers.

Although it is not the focus of this project, it is important to note that co-locating the transactional and data warehouse databases can reduce overall power consumption. The co-location project system configuration consisted of a single System z with multiple LPARS supporting several z/OS, z/VM, and Linux images. A single DS8300 storage device was used to store the transactional and data warehouse databases. Hipersockets were used for network traffic between z/OS and Linux images.

There is a common theme in the system configuration: virtualization. Virtualization is a technology designed to enable multiple application workloads, each having an independent computing environment and service level objectives, to run on a single machine. This eliminates the approach of dedicating a single workload to a single server, a practice that yields low utilization rates, and allows virtualized servers to function near maximum capacity.

The most energy-efficient equipment is equipment that is no longer in use, whether it is a server, a router or a storage device. So by allowing a client to combine several standalone systems onto fewer, more efficient pieces of equipment, reducing equipment footprint and power consumption, virtualization allows realization of savings and efficiencies that have been difficult to achieve through the design of even the greenest systems or buildings.

Virtualization was used in both the System z and the I/O devices used to support the configuration.

In System z, the configuration employed both hardware and software virtualization to support multiple independent operating system images. The hardware hypervisor is implemented in firmware and is called Processor Resource/Systems Manager™ (PR/SM). PR/SM is always active. PR/SM technology has received Common Criteria EAL51 security certification, so each logical partition is as secure as an isolated server.

Software virtualization is provided by the z/VM product. z/VM uses the resources of the LPAR it is running in to create functional equivalents of real System z servers, which are known as *virtual machines*. In addition, z/VM is able to emulate I/O peripherals including printers, by using spooling techniques, and LAN switches and disks by exploiting memory. The z/VM virtualized z/Architecture servers support all operating systems and other software supported on a logical partition. In fact, a z/VM virtual machine is the functional equivalent of a real

server. The extreme virtualization capabilities of z/VM, which have been perfected since its start in 1967, make it possible to virtualize thousands of distributed servers on a single z10 server.

LAN virtualization is provided by HiperSockets, which uses microcode to transfer data between LPARs on the same z system. For the co-location project, HiperSockets was used to transfer data between all of the z/VM, Linux, and z/OS images on the z system. With HiperSockets, the transferred data never leaves the z system.

Storage virtualization can combine storage capacity from multiple vendors and multiple DASD volumes onto a single reservoir of capacity that can be managed from a central point. Just as server virtualization reduces the number of servers needed, storage virtualization reduces the number of spindles required, increasing the total amount of available disk space and optimizing utilization rates. Storage virtualization can also improve application availability by insulating host applications from changes to the physical storage infrastructure. Refer to *IBM Systems Storage DS8000: Architecture and Implementation*, SG24-6786 for more details on virtualization and the DS8000.

Virtualization can be a tremendous ally in reducing heat and expense simply because it means that you will need less equipment. All servers use energy and give off heat whether they are in use 100 percent of the time or 15 percent of the time, and on System z the actual difference in electrical consumption and heat generated between those two points is not significant. This means a System z server that is only 15 percent utilized will cost about as much to run as a server that is fully utilized. Similarly, disks use energy when they spin, and latency requirements mean that they are always spinning when they have active data.

Several IBM studies using System z demonstrate that the higher utilized the installed hardware is on the system, the more efficient the power consumption per unit of work will be. These studies used different workloads, but since System z power is determined by configuration rather than workloads, there should be no significant difference between the co-location workload and these other workloads with regards to power consumption. For additional information, refer to the following IBM techdocs:

```
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101110
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101265
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101493
```

Increasing processor utilization and taking advantage of processor scalability on System z and the DS8000, through the use of virtualization and adding new workloads to existing hardware, can result in lowering the total power consumption for the data center by combining underutilized hardware onto a single piece of hardware. Although showing a reduction in power consumption by combining the OLTP and data warehouse workloads on a single System z was not the goal of this project, it appears to be one of the positive results.

2.11 Storage management (SMS)

Databases and their data can benefit from system-managed storage (SMS) on z/OS in the following ways:

- Database data can benefit from data isolation achieved through the use of SMS storage groups.

  SMS services can help you automate allocation of OLTP and OLAP data sets on separately managed volumes. This isolation may be desirable because of different performance and availability requirements or special backup and recovery management procedures. This could also benefit the separation of data and indexes.

- Databases can benefit from SMS allocation algorithms.

  SMS tries to balance allocations in storage groups that have large numbers of volumes. With SMS allocation, users do not have to worry about specifying volume serial numbers or ensuring that the specified volumes have adequate free space. Additionally, with SMS allocation algorithms, volumes are less fragmented when space requests are satisfied.
- Management of database data is improved by implementing standards enforcement through the use of data classes and the data class ACS routine.

  Storage and database administrators can jointly develop data classes for commonly used types of database data. With data classes, you can standardize the allocation parameters used to create new database data sets.
- Database applications that perform sequential data access can benefit from striping. Striping is especially beneficial for the DB2 active log data sets.
- Applications, such as data warehousing, that use large data sets can benefit from extended addressability. Extended addressability allows creation and addressing of data sets over 4 GB in size.

2.12 Data auditability and compliance

Protecting the sensitive data of employees and customers, ranging from credit information to health information and social security numbers, is crucial to maintaining trust between the corporation and the individuals it services. As regulatory bodies have become more aware of this fact, they have passed laws such as Sarbanes-Oxley and Hipaa that makes protecting sensitive information not only wise but also a legal necessity.

Within an enterprise this compliance takes many forms, from the security of physical data to the need to report to the regulating organizations on a regular basis, as well as complying with the audit requirements of the legislation.

When data is extracted from an OLTP production database and loaded into an OLAP warehouse, it is still considered by legislation to be sensitive data and as such bound by the same rules and regulations that affect traditional data. Among these regulations is the need for data retention, data security, audit trails and audit data. For many non-z platforms this has resulted in the need to acquire specialized software and hardware as well as developing policies and procedures to ensure these requirements are met.

Data lifecycle management and security has been part of the System z infrastructure for many years. Products such as the DF suite and RACF along with state of the art long-term retention hardware devices, for example encrypted tape, are pervasive within the z client base. Along with the skills and knowledge gained from traditional OLTP and batch environments, these make it extremely simple to extend these facilities to the warehouse environment without incurring any major additional expenses. The built-in data lifecycle management facilities of DFSMS can guarantee that all retention regulatory requirements are met while ensuring that expired data is safely erased.

2.13 Risk avoidance and data consolidation

Why is having transactional data and data warehouse on the same system a good idea from a data management point of view? Because it makes it easier to manage risk. Here is an overview of the benefits:

- Data management
  - Avoid multiple copies to be maintained
  - Reduce latency in data propagation, which speeds up time to action
  - Simplicity vs. complexity
  - Reduce needed resources to maintain several copies for backing up and restoring, updating the copies, and so on
- Security
  - Your mainframe is a secure platform you can trust. Do you really want to copy the most valuable information to another system that you trust less?
  - Transfers are usually done over a network, which makes it possible to be intercepted. If you keep all the data at the same place, there is no need to worry about this.

Data consolidation reduces risk, or makes it possible to manage it. The question would be, is a single machine able to provide the required I/O bandwidth to serve all the data consumers, including applications and people?

As we saw in 2.4, “Unique I/O capabilities”, System z with its exceptional I/O capabilities is the only platform that can sustain such a mixed workload from the data storage perspective.

2.14 Managed reporting

The recent financial crisis has exposed the problem that the world's largest financial institutions cannot adequately account for and report on liquidity, positions, currency exposure, credit, market, and interest rate risk and product, customer and organizational performance. The CFO plays a critical role in correcting this problem by leveraging the financial data, as well as leveraging scale to reduce cost. But even industry insiders do not realize that financial institutions suffer a unique set of problems when it comes to financial reporting.

Some financial reporting systems are antiquated and inefficient. They were designed decades ago to simply track the flow of capital, revenue, and expenses at the company and department levels. The lack of transparency is evident in the increasing costs of the finance function, with few benefits to show for the investment. Sarbanes Oxley and other regulations have proven largely ineffective at getting at the root of the problem, and the resulting financial meltdown regulations may well prove similarly ineffective. These pressures affect the largest institutions the most, with their complex finances.

For the most part, existing systems deliver accurate results in summary, but the increase in transparency requires investigation into the underlying causes of those results. Consider if your personal bank account statement or credit card bill only presented the change in balance between periods, but provided no list of transactions. When the statement is as expected, further detail may not be needed. But when the balance is in question, your first response is “why” and you immediately want to see the transaction details. The same issues are at stake when managing the finances of the enterprise, with the associated cost and consequences considerably higher. A single instance of financial restatement has cost some organizations hundreds of millions of dollars to correct, not counting lost market valuation.

Currently 90% of the money supply in mature markets is represented by digital records of transactions and not hard currency. It is no wonder that the volume of electronic finance records being kept has exploded compared to when the systems were first created. Yet our approach to these demands has not been to automate the process of keeping and accessing

Archived

the details of the transactions. Almost all employees in today's financial institutions are involved in capturing and coding financial details in some way, and a large number of non-finance employees are involved in the investigative process to locate the additional detail so often required. The effort for this manual intervention is incredibly inefficient and costly.

As we see all around us, computing capacities have increased by several orders of magnitude since these finance systems were designed. However, reporting processes have grown organically as a system of transaction summaries in order to continue to bridge multiple financial systems, but have lacked a single unified approach. This has meant that for the most part the business of financial reporting has not benefited from the increase of computing capacities available today.

Much of the data required for such transparency has been contained on z/OS data bases, most notably DB2. Previously customers would extract this data to an external data store in order to perform analytics or produce the required reports. This method has the unfortunate side effect that the data tends to be out-of-date the minute it is extracted, leading to the demand from large clients for transactional data stores and more recently dynamic data warehousing. z/OS-based products such as the IBM Scalable Architecture for Financial Reporting (SAFR) system are uniquely placed to provide managed reporting to co-located OLAP databases.

By harnessing today's computing power, the details behind summary data can be made available in seconds rather than days or weeks. Providing nearly instant access to the highest quality financial data at any level of granularity can eliminate the duplicative reporting systems that tend to capture and produce summaries of the same business events for many stakeholders and reporting requirements.

More importantly, it can automate the hidden work of armies of people who are required to investigate details and attempt to explain results, or attempt to reconcile the disparate results of these many reporting systems—a truly wasteful activity caused by the systems themselves. Keeping the details in a finance system that can serve these needs allows for increased control, quality, and integrity of audit efforts rather than dissipating them.

Archive

3

Co-location topologies

This chapter describes some of the topologies that are possible to assist in co-locating data.

- Single vs. multiple LPARs
- DB2 data sharing or not

Archived

© Copyright IBM Corp. 2010. All rights reserved.

3.1 Configuration options

What configuration options are available for co-locating online transaction processing (OLTP) and a data warehouse? There are quite a few, each with its own set of advantages. In this chapter, we examine some of the topologies available for co-locating these workloads.

In all cases, processor capacity can be dynamically shifted between the OLTP and DW workloads based on application demand and priority assignment.

Prior to exploring the various topology options, we first briefly discuss two key considerations for several of the topologies. These are also two key focus areas of this book.

Resource management

Do I house transactional and data warehouse workloads together in a single LPAR, yielding resource management based purely on the WLM service definitions and goals? Or do I isolate the workloads into separate LPARs, using more coarse-level resource management such as LPAR specifications (# of processors, weights, caps, and so on), Intelligent Resource Director (IRD), and perhaps Group Capacity controls? The latter approach is generally a well-understood, proven, efficient means to share resources between workloads. The use of multiple LPARs also lends itself to allocating resources to a given workload. This is typically desired by BI knowledge workers wanting to have their own system.

On the other hand, some IT shops desire to keep their number of LPARs to a minimum, and would like to run both traditional transactional and the newer data warehouse and BI type workloads in the same LPAR. Though clearly WLM is one of, if not *the* most sophisticated resource managers in the industry, managing this diverse mix of work is more challenging. This was one of the key influencers for the “single LPAR” experiments in this book. Our conclusion was that efficient and controlled resource management can be achieved in the single LPAR configuration as well, though it likely will take more WLM policy iterations and the maintenance of same. See Chapter 11, “Resource management of data warehouse mixed workloads” on page 263 for the how. See Chapter 15, “Single z/OS LPAR topology experiments” on page 347 for the experiment results.

DB2 data sharing

Do I put my data warehouse DB2s in the same data-sharing group with my Transactional DB2s? What is the value and what are the associated implementation hurdles and performance cost?

One of the key advantages of data sharing is ease of access to real-time transactional data and the data warehouse from the same application or application connection. A single query can access (recommendation is light access) the transactional data and join it with data directly from the data warehouse (see 6.3.4, “Operational BI reporting” on page 110). Additionally, the refreshing of the data warehouse from the transactional DB can potentially be expedited with in-database transformations. When a data warehouse requires the same high level of availability of the transactional system, data sharing can automatically provide the additional availability, as well as simplifying high availability and disaster recovery (HADR).

The implementation considerations involve minimizing group buffer pool dependencies with the operational DB2s and any additional global locking activities. These are discussed in 10.3, “Co-location data-sharing”. We examined the performance cost in multiple z/OS LPAR experiments in Chapter 16, “Multiple z/OS LPAR experiments” on page 369. Our measurement results yielded very similar performance data when deploying the data warehouse applications in a DB2 with data-sharing vs. not.

3.1.1 Single LPAR, OLTP, and DW share single DB2 subsystem

The first topology, and probably the simplest, is having an OLTP and data warehouse sharing the same single DB2 subsystem (Figure 3-1). At first, this may seem like a choice with minimum management cost; however, this is usually not the case. Having just a single DB2 might require fewer people but that is where the savings may stop. Achieving the best possible performance for both the OLTP workload and data warehouse workloads could be the challenge. Often the tuning requirements, such as the buffer pool strategy, are different for the two environments. Where a high buffer reuse for random reads would be the target for OLTP, warehousing would favor more prefetch activity, with possibly little or no buffer pool reuse because of the size of the tables used in data warehousing. Even the DSNZPARM members may have differences. OLTP and data warehousing would be highly unlikely to share the same needs for star joins and parallelism. These each require system-wide DSNZPARM settings that would be set in opposite ways for the two differing workloads.

When it comes to actually tuning the DB2 subsystem for the best possible system and application performance, again the different workload characteristics would make this extremely difficult. Separating the warehouse work from the transaction processing work would be time consuming, if separation were possible at all. Also, having both workloads running in a single DB2 subsystem would reduce the concurrency levels of each workload due to virtual storage constraints.

Application availability (for both OLTP and DW) could also be impacted. For example, in an OLTP environment, frequent data reorganization may be necessary, a process that could be of little use to a warehousing environment. Even storage concerns could come into play. With a data warehouse workload there is a high likelihood that parallelism could be used. Overuse of parallelism or a poorly designed parallel using strategy could have a direct effect on available virtual storage. This scenario is discussed in Chapter 15, “Single z/OS LPAR topology experiments” on page 347.

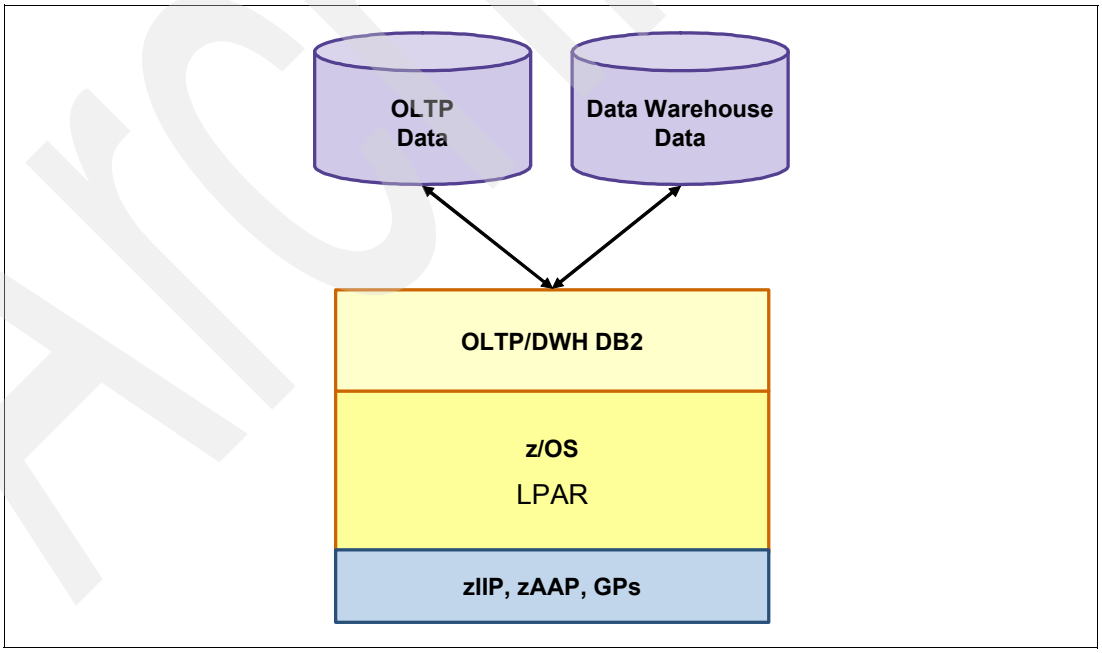


Figure 3-1 A single DB2 subsystem

3.1.2 Single LPAR, separate OLTP and DB2 DW subsystems

A second possible configuration would be placing the OLTP work into a separate DB2 subsystem on a single logical partition (LPAR) from the data warehouse workload (Figure 3-2). With each of the two work types in their own subsystems, tailoring pools, and system parameters becomes a much easier task. However, the work is completely separated. Something extra would have to be put in place for data from the OLTP subsystem to be used in the data warehouse subsystem. For example, the data movement could be achieved with using the SQW component of InfoSphere Warehouse on System z or Information Server.

Even though both subsystems are housed in the same LPAR, z/OS will always treat them as two distinct and non-sharing tasks, each with their own set of data.

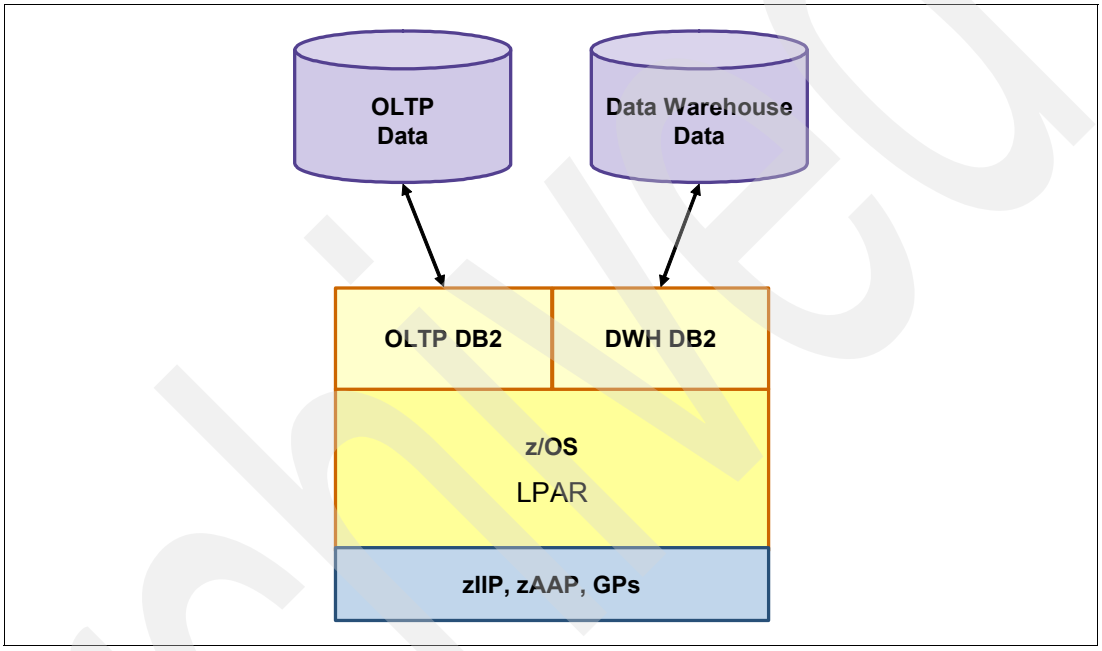


Figure 3-2 Single LPAR: separate OLTP and DW DB2 subsystems

3.1.3 Multi-LPAR - OLTP LPAR and DW LPAR

This configuration best aligns with the current IBM Smart Analytics System 9600 offering. The DB2 Data Warehouse is maintained in its own LPAR, which provides better isolation of work and leads to a higher level of availability. With this setup it becomes very simple to add a data warehousing LPAR to an existing z/OS environment. Though there is a cost to build and maintain an additional LPAR, the advantages far outweigh the cost in this situation; see Figure 3-3 on page 49.

Archived

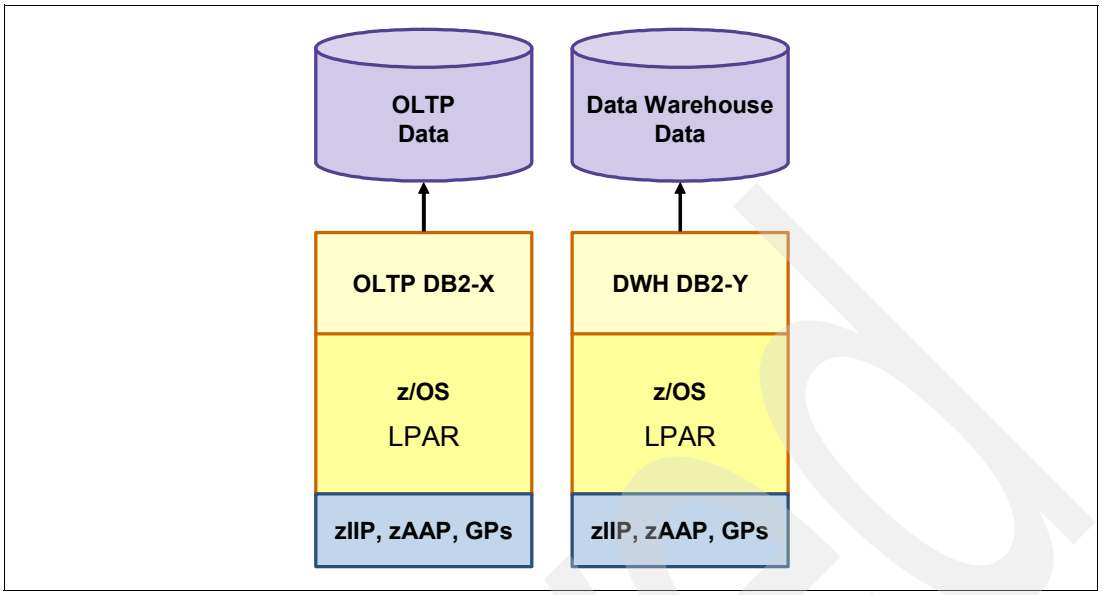


Figure 3-3 Multi LPAR: one OLTP LPAR and one DW LPAR

The multiple LPAR topology can have cost advantages. A new workload, including data warehousing, added to an existing system but in its own LPAR can be available for special pricing. DB2 Value Added Edition (VUE) can also be considered for this topology.

3.1.4 Single LPAR - two-member data sharing

Another possibility is to keep both the OLTP and data warehouse subsystems in the same LPAR, as in Figure 3-4 on page 50. However, in this scenario they are both participating in a data sharing group. As such, the OLTP and warehouse DB2s become members of a single data sharing group. Belonging to the same data sharing group means the DB2 catalog, DB2 directory, and all user data is available to both the OLTP and DW DB2 members. Data can move easily from the OLTP side to the warehouse side using simple DB2 processes, simplifying the entire ETL procedure. With shared user data, an application can extract data from an OLTP table, perform whatever transformations are necessary for use with the data warehouse, and then load that data into a warehouse table. Business analytics can run on the data warehouse DB2 member without interfering with or affecting the performance of the OLTP DB2 member.

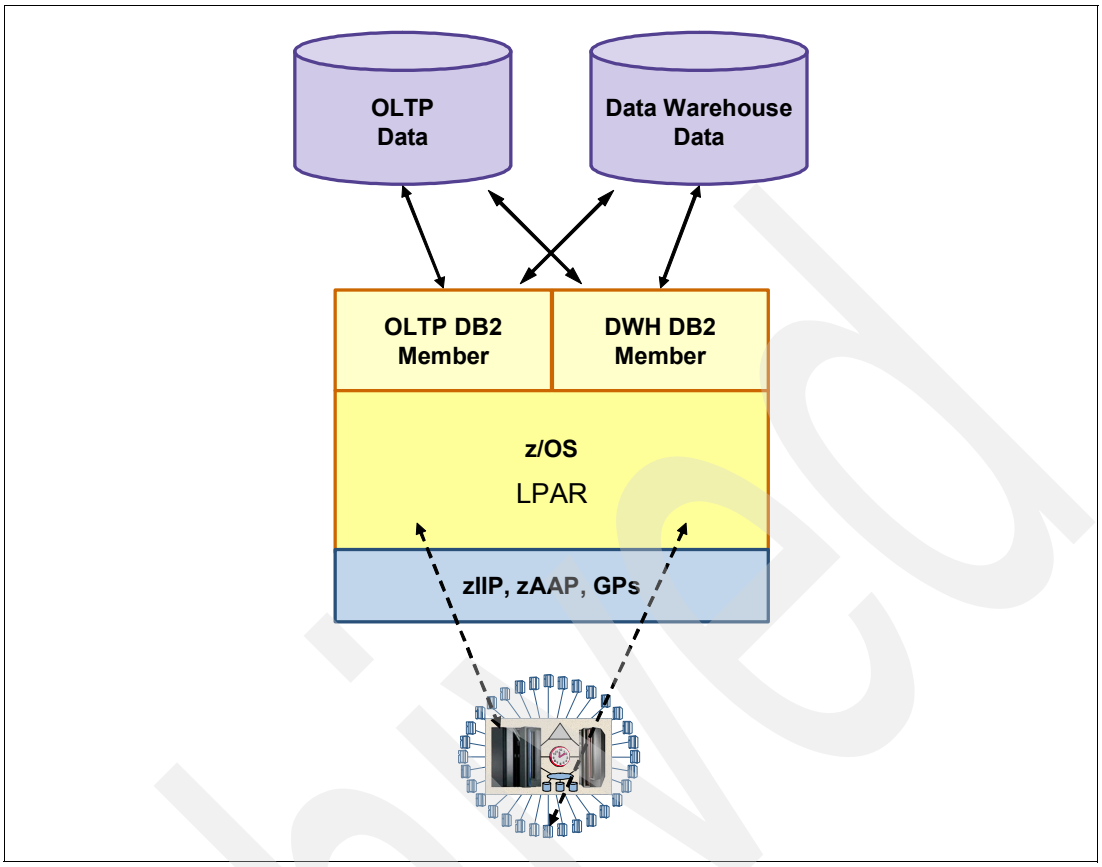


Figure 3-4 Two member data sharing in a single LPAR

If business requirements call for it, OLTP data is now accessible to data warehouse queries. If a table should be shared by both members and changes are made to that table, the Coupling Facility (see 2.6.3, “Integrated Coupling Facility” for more details on ICF) ensures that all members of the data sharing group are aware of those changes.

Another potential advantage to the use of the data sharing setup is that a single program could connect to just one place and have access to both OLTP and DW tables. For example, a banking application could show both transactional data and historical spending patterns. In addition, join operations could be performed, if necessary, between OLTP and DW tables.

Tip: To avoid potential performance issues, share only what needs to be shared.

3.1.5 Multi-LPAR - two-member data sharing

The previous topology can be improved with minimal effort by moving the OLTP members and the data warehouse members into separate LPARs. With this configuration, any LPAR resources would be available to only the DB2 member using that LPAR. This configuration could be further enhanced by running each of the LPARs on their own central processor complex (CPC). See Figure 3-5 on page 51.

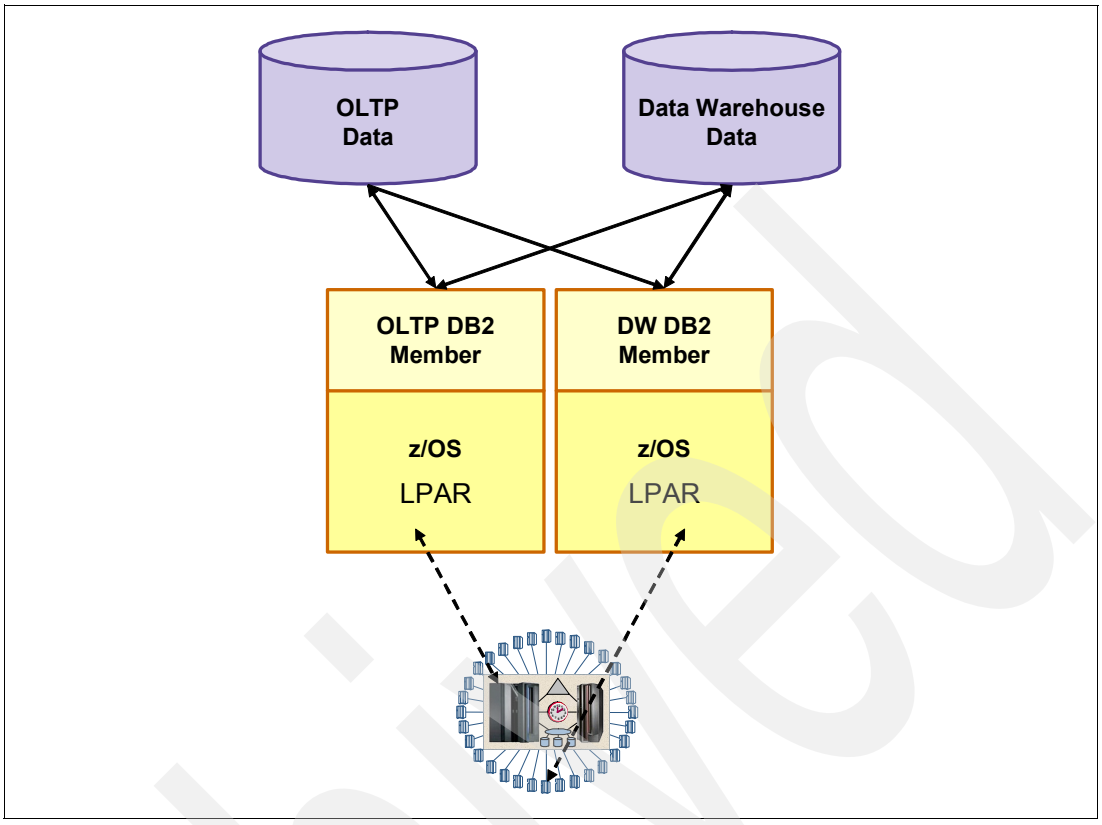


Figure 3-5 Multi-LPAR two-member data sharing group

In this topology, OLTP and DW are still running as members of a data sharing group, each as their own member. However, now each member resides in a different LPAR, so each LPAR could potentially be on a different CPC. This option and the above option, 3.1.3, "Multi-LPAR - OLTP LPAR and DW LPAR" on page 48, could be the preferred ones of all the choices in the chapter. They both allow separation of the OLTP and DW workloads at the LPAR level.

A multiple LPAR data sharing group can also improve the availability for both the OLTP and DW members. Fail-over scenarios with this configuration are easily designed.

If both LPARs are on the same CPC, available processor resources (CPs, zIIPs, and zAAPs) are shared across the LPARs. If the LPARs exist in different CPCs, processor resources become independent.

3.1.6 Multi-LPAR - OLTP LPARs and OLTP and DW LPAR

Another configuration could use multiple LPARs with an OLTP member in one LPAR along with a DW member and a second OLTP member sharing a second LPAR (Figure 3-6 on page 52). Preferably, each DB2 subsystem would reside in its own LPAR, but there are situations where the IT staff would prefer not to add LPARs for their DW applications. In those cases, this configuration is a viable alternative.

Archived

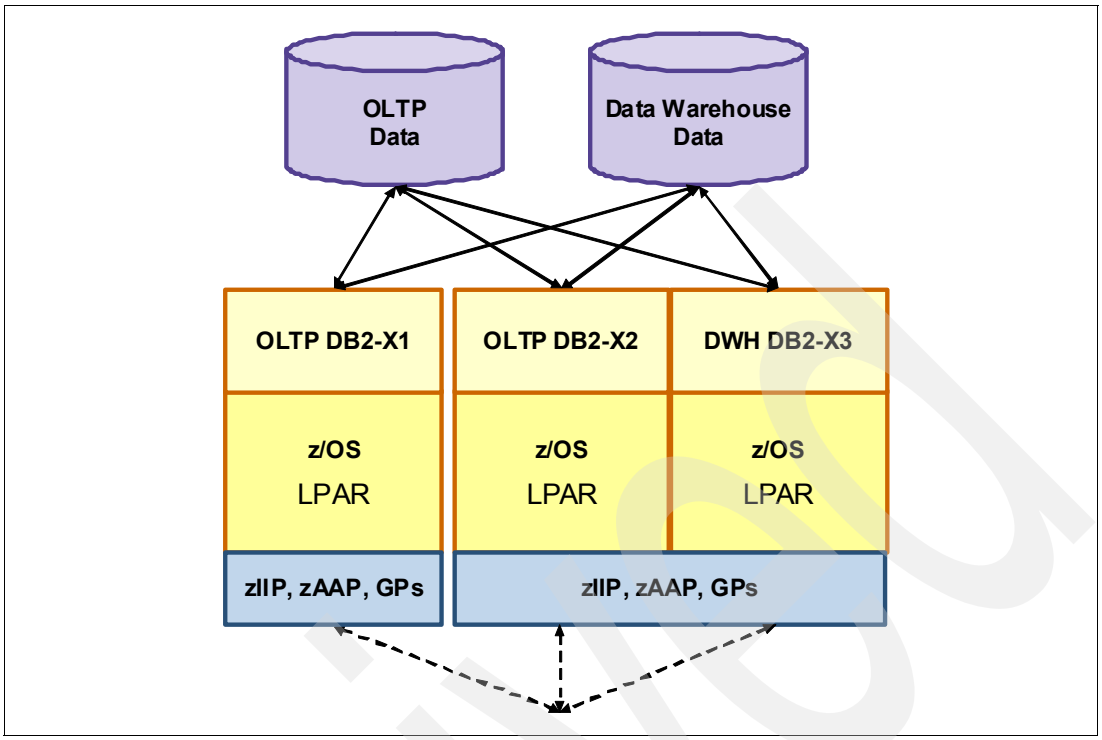


Figure 3-6 Multi-LPAR, one OLTP LPAR and one DW OLTP LPAR

This topology moves us up to three LPARs. Two LPARs are running OLPT workloads in two data sharing members. Two other LPARs are running DW workloads in two data sharing members, each member running in its own LPAR.

3.1.7 Multi-LPAR - data sharing with OLTP and DW LPARs

Figure 3-7 on page 53 shows another multiple LPAR example. Here, we are back down to a two LPAR configuration. However, this time we are combining an OLTP and data warehouse DB2 member in each LPAR. Our configuration is now up to four-way data sharing. Fortunately, after doing two-way data sharing, the cost to add additional data sharing members is minimal.

Archive

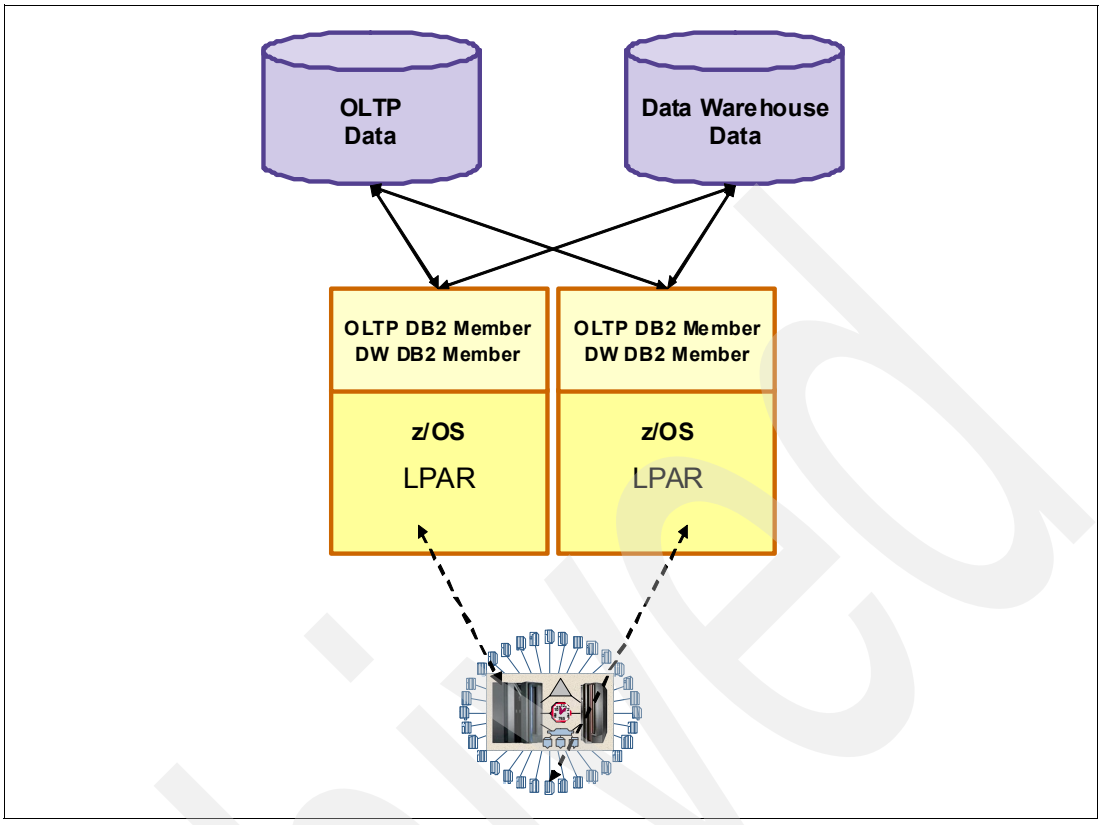


Figure 3-7 Multi LPAR - two OLTP and DW LPARs using data sharing

Using only two LPARs does remove the overhead of managing a third LPAR. However, the OLTP and warehouse data sharing members now need to share the same LPAR resources. The advantages of separation of resources required for OLTP and those required for data warehousing performance are lost. Reduced LPAR management is traded for less DB2 performance tuning granularity.

3.1.8 Multi-LPAR - data sharing with multiple OTLP LPARs and multiple DW LPARs

This configuration utilizes at least four LPARs with one DB2 member on each LPAR. In a four LPAR configuration, two of the LPARs run OLTP applications while the other two LPARs run DW workloads. This setup requires additional system management effort, but in return it delivers the highest level of availability. Neither the OLTP nor the DW workloads stop completely with the failure of a DB2 member or an LPAR. See Figure 3-8 on page 54.

This multiple LPAR configuration is discussed in Chapter 16, “Multiple z/OS LPAR experiments” on page 369.

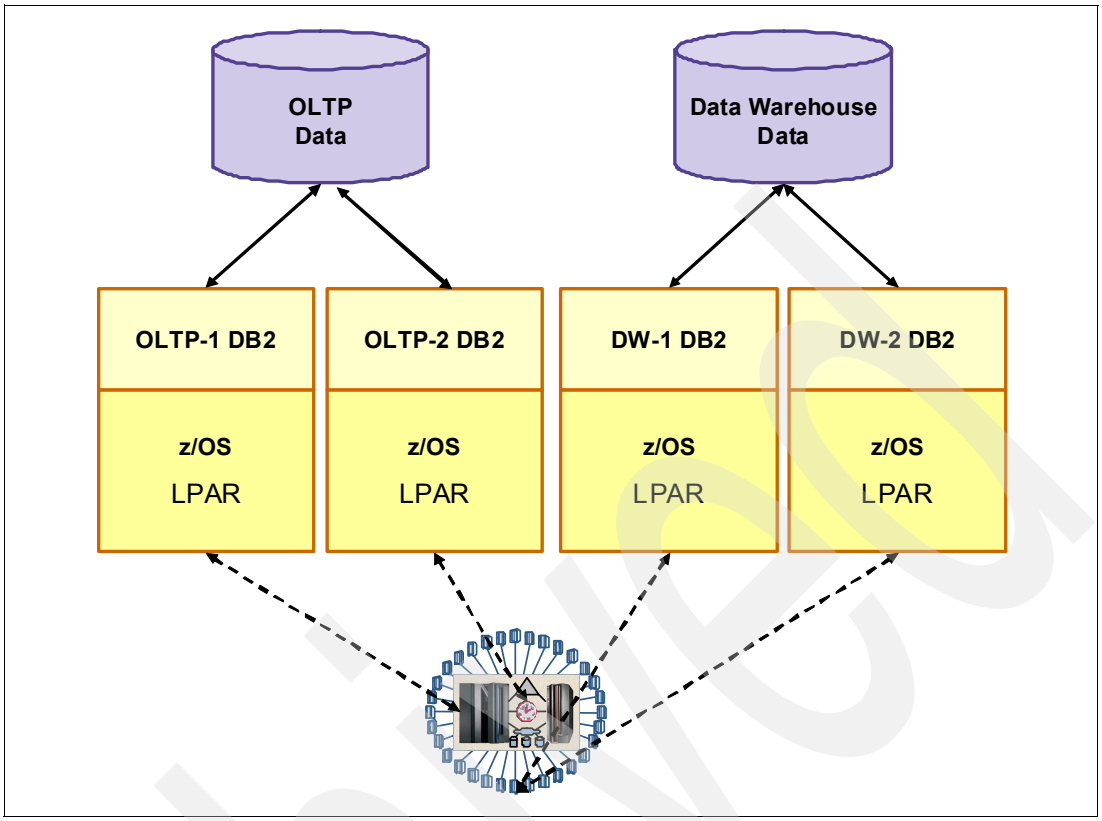


Figure 3-8 Multi-LPAR - data sharing with multiple OLTP LPARs and multiple DW LPARs

3.1.9 Standalone data warehouse DB2 subsystem

This is an almost identical scenario to the configuration discussed in 3.1.3, “Multi-LPAR - OLTP LPAR and DW LPAR” on page 48. However, in this topology an OLTP DB2 subsystem does not exist. This topology is also similar to how an operational data store (ODS) might be set up.

3.1.10 Front-end topologies

Until now, we have only mentioned the data warehouse backend with DB2 on z/OS topologies. In this project we used IBM Cognos BI on Linux for System z as the user front end for report generation, which becomes a DB2 for z/OS workload. Although IBM Cognos BI can have multiple topologies, the purpose of this project was to focus on z/OS resource management. We did not explore or test IBM Cognos BI configuration options; rather, IBM Cognos BI was used to generate a workload into DB2. We implemented IBM Cognos Business Intelligence Version 8.4 SP2 as a WebSphere 31-bit J2EE application, but portions of IBM Cognos BI for Linux on System z can also be implemented in WebSphere as a 64-bit J2EE application.

3.2 Floating DW and DataStage with BatchPipes

As a helpful sidelight to this topology discussion, we offer a solution using BatchPipes and DataStage.

BatchPipes is a little-known subsystem of z/OS. It was used to speed up batch processing. Normally when a job has to update a data set in z/OS, it has to allocate it nonshared. Nobody is able to access it, even to read it. If the next job in the batch process needs to read this data set, it has to wait for the first job to finish and release it to start reading the records. BatchPipes removed this limitation by “piping” the data into the next job. The second job was able to start just after the first record was treated, so both jobs could run at the same time.

DataStage, an ETL tool running in Linux on System z, has been developing a staging program that makes it possible to do the same in a DB2 load. In this DataStage concept, every step must be “pipe enabled”. This was the case for everything except loading the database, such as to perform a LOAD utility, where DSNUTIL must have exclusive access to a data set.

Prerequisites for using the DB2Z stage

Even if you have the BatchPipes version 2.01 subsystem (FMID = HACH301), you need these PTFs.

- PK37032 - This PTF adds the site subsys=pipe1 command to the FTP server. It makes it possible to redirect data to a subsystem, in particular, to BatchPipes.
- PK54242 - This one fixes a problem, that is, not setting the LRECL parameter properly when using the site subsys=pipe1 command.
- PK34251 - This is an enhancement to the DB2 TEMPLATE utility statement to allow references to BatchPipes data streams.

The whole concept of DataStage is to forward the flow of data to the next step through UNIX-like pipes. This means that there is no lending of the intermediate data to the disk. It is an in-memory transformation mechanism. Another concept of DataStage is to parallelize.

Up to now, to load the data in a DBMS you had two choices. You could either insert data in database tables or you could create a flat file with the data and then use a load utility.

What the DB2Z stage is doing is shown in Figure 3-9 on page 56. The DataStage sequence follows the usual pattern. When the first row comes into DB2Z, FTP connections (for as many partitions as you are loading) are opened and a call to the DSNUTILS stored procedure is issued. SYSIN is passed as an argument to the call that will create a new utility to be started and data is being read from the batch pipes.

Archived

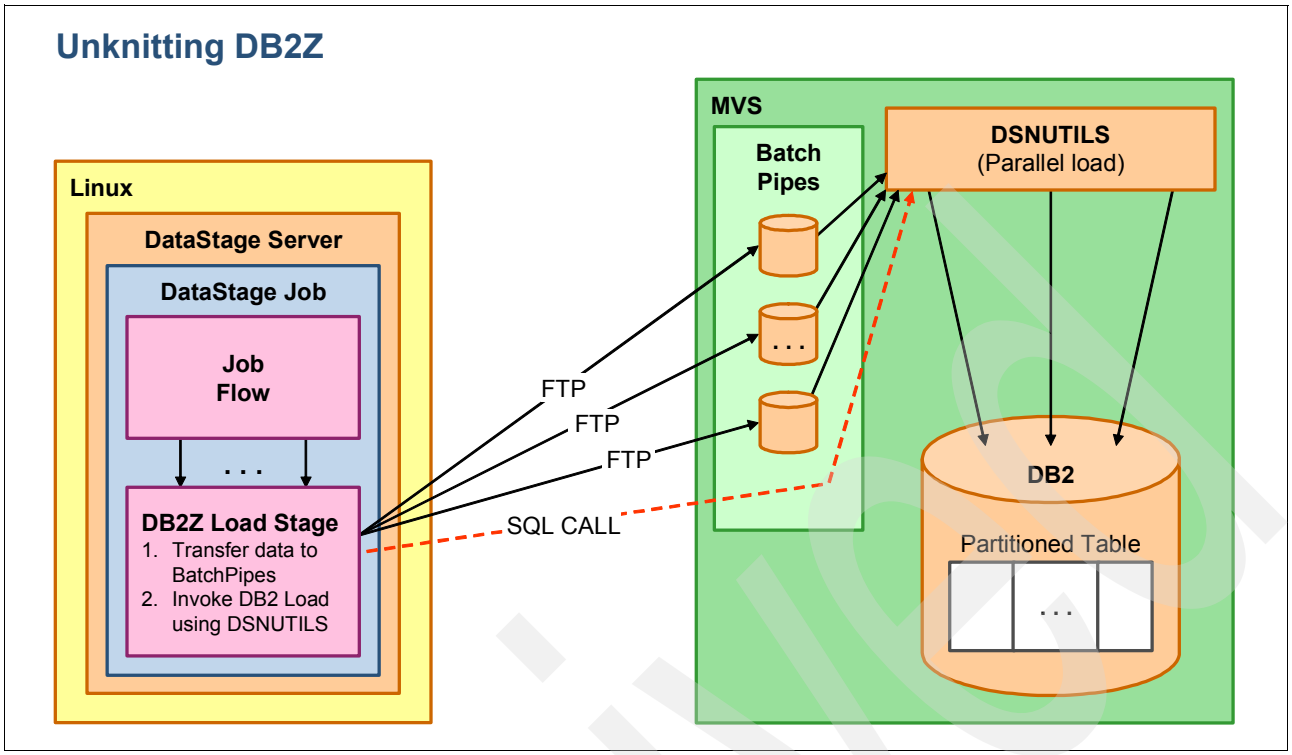


Figure 3-9 DataStage's DB2Z activities

This solution can help you reduce your ETL jobs' elapsed time. If implemented on a single CPC, with the use of HiperSockets, you will speed up the network transfer time without overloading the physical network. It will also increase the confidentiality of the data, avoiding exposure to network sniffing.

In addition, you will use less disk space. Otherwise you would have needed the same space three times: once on the machine running DataStage, once when transferring the data to z/OS as an input to the LOAD utility, and—the space you really care about—once in the final repository DB2 for z/OS.

Archive

Part 2

Project environment

In this part we describe our experiment environment and the workloads that we ran.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

4

Co-location data models

This chapter describes the logical and physical data models of the transactional database and data mart used in this book. These data models were developed based on the idea of operating a fictitious bookstore chain that has multiple physical stores geographically distributed across cities and states. The design of the data models is to capture a fairly simple yet representative set of business data elements in order to support a mixed workload of book sales and inventory tracking, as well as historical and operational business intelligence and reporting.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

4.1 Overview

Our fictitious book store chain manages multiple physical stores that stock books for purchase. The stores operate across many cities and states but have one centralized transactional application running on top of the z platform. A customer may place a book order containing one or more book titles, each of which is ordered with one or more copies. At the time of purchase, the inventory system gets updated to reflect the number of copies available for each title carried in a store.

The book store chain keeps a history of book orders placed by customers. The history includes when and what book titles were purchased at what prices. It also tracks the customers through their registered profiles to correlate their purchase history and preferences. Business analysts and store managers can then analyze sales trends with respect to stores, book categories, and customers.

This scenario reflects a miniature yet realistic business application that covers aspects of a transactional workload as well as a business intelligence workload. It is thus our intent to use the scenario to drive the co-location usage patterns.

In order to simulate real book order operations, several data distribution skews are introduced in our simulation environment, including customer orders, store volume, and seasonality. As we explain in the IBM Cognos BI report development tuning, data distribution skews are exploited to tune report queries that fall under CP-second ranges.

Before placing an order, a customer preferably has registered with a loyalty account that tracks the history of books bought. A customer may also elect to skip the account creation and use an anonymous customer ID. In our simulated data generation, a distribution skew was introduced to allow 5% of the overall book orders to have originated from anonymous customers (those who did not register).

There are two tiers of stores defined by the number of orders received by a store. The first tier stores, six out of a total of 26, take in 40% of all orders. The second tier, the remaining twenty stores, take in approximately 60% of all orders.

Seasonality reflected by an increase in order volume is introduced during holiday shopping. For the months of November and December, the total number of orders received doubled from the typical order volume in the non-holiday months, which are January through October.

Once an order is placed successfully, it is assumed to be accepted and fulfilled. Records associated with the new order are collected and loaded into the data mart tables. The data can then be analyzed through Business Intelligence tools such as the IBM Cognos BI.

4.2 The transactional data model

The transactional data model is designed to capture relationships among five main entities: customer, order, title, store, and inventory, which are simple yet representative of typical order capturing and inventory tracking systems.

In real life, the transactional system may operate continuously during business hours to accept order placement and update inventory. To simulate the system, we use a Java application to populate the tables.

At a high level, the transactional data model captures the relationships among five entities. The logical data model is depicted in Figure 4-1 on page 61.

4.2.1 Logical schema

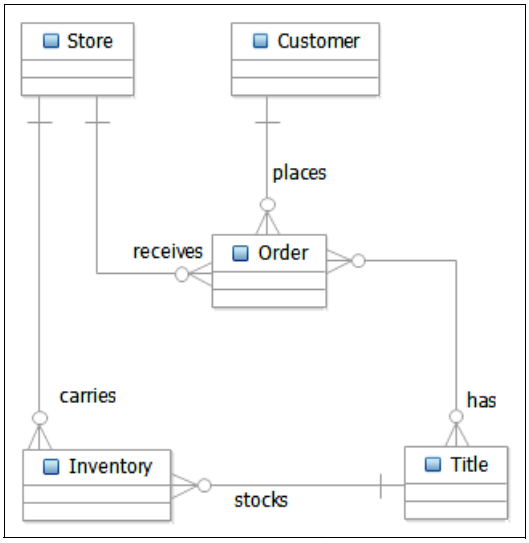


Figure 4-1 The logical data model of the order transaction system

Central in the relationship is the book order, which must be placed by a customer, received by a store, and consists of one or more books. Furthermore, a customer may place zero or more orders, while each order must be received by only one store. An order can have one or more book titles, each of which may be ordered with any number of copies. Each store carries a stock of book inventories to track copies available for each book title.

An important yet implicit element of the data model is *time* in these relationships. An order carries a timestamp to reflect when it was received by a store. As one will see in subsequent business intelligence reporting, time is often a key dimension to which various metrics are rolled up or drilled down.

4.2.2 Physical schema

The physical schema is depicted in Figure 4-2 on page 62, where entities in the logical data model are mapped to one or more physical tables in the DB2 database.

Archive

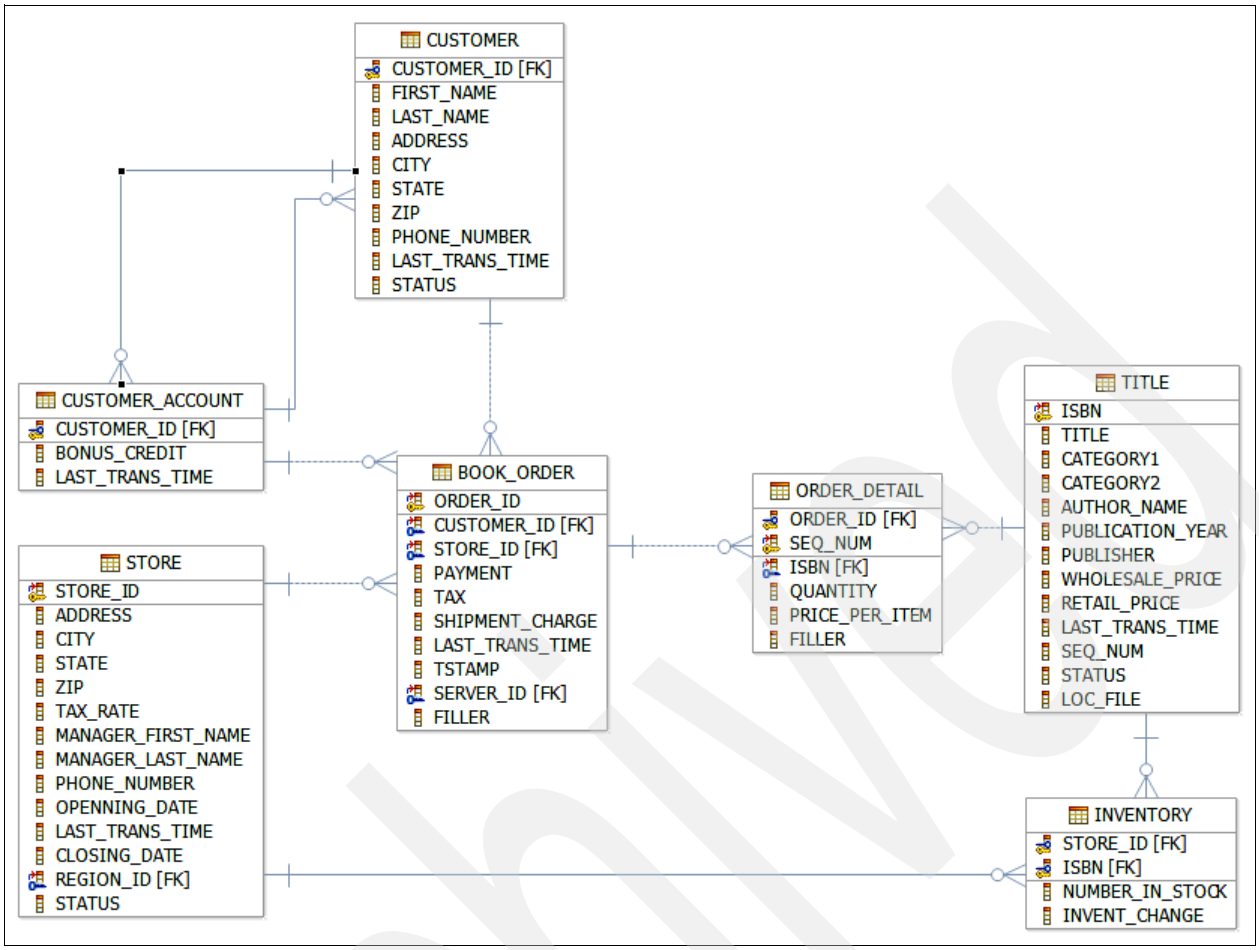


Figure 4-2 The physical data model of the order transaction system

In the physical schema, a single entity may be represented by a set of tables with multiple columns or attributes. For example, an order is translated into two tables: BOOK_ORDER and ORDER_DETAIL in its third normal form.

Table 4-1 lists the mapping from logical entities to their physical tables. In our simulated transactions, the most frequently inserted are the two order tables. Once an order is inserted along with its details, the confirmed order can no longer be updated. While the majority of the tables are append only, the INVENTORY table is updated to keep track of the number of books in stock for a store. The column INVENT_CHANGE in the INVENTORY table records the timestamp the stock was changed.

Table 4-1 The mappings from logical to physical data models

Logical entity	Physical tables
Store	STORE
Customer	CUSTOMER, CUSTOMER_ACCOUNT
Order	BOOK_ORDER, ORDER_DETAIL
Title	TITLE
Inventory	INVENTORY

4.3 The data mart data model

The purpose of the data mart model is to provide a consolidated view for the relevant business information. The physical schema is typically designed differently from the schema of a transactional data model to optimize for queries that access large numbers of records. In addition, database performance tuning such as index and Materialized Query Table (MQT) is often applied to meet the requirements for query response time.

4.3.1 Physical schema

The most notable difference in the schema of the book order data mart, as shown in Figure 4-3 on page 64, is the BOOKORDER_DETAIL_FACT table, which was originated from the BOOK_ORDER and ORDER_DETAIL tables in the transactional schema. The BOOKORDER_DETAIL_FACT table is denormalized from the two tables and leaves out several attributes associated with the orders, which can be computed from columns in the ORDER_DETAIL table. Such denormalization is not uncommon in data mart design in favor of fewer and perhaps bigger fact tables to keep the load and maintenance of the database simple. On the other hand, denormalized table records do add query processing overhead, which can be eased with database tuning techniques such as index and MQT.

A second notable difference is the addition of the TIME_DIM table, which reflects the importance of time range processing in most business intelligence reports. TIME_DIM is a prematerialized table with calendar date as its primary key and the corresponding year, quarter, month, week of the year, day of the month, and week day flag as attributes. While date conversion functions are often built in as database operators, having precomputed conversions in a table significantly saves time and processing. One can simply join the TIME_DIM table to look up these fields without incurring the cost of an SQL operator.

The rest of the data mart tables are mostly a straightforward mapping from the transaction tables. Table 4-2 lists the mappings.

Table 4-2 The mappings from transactional to data mart data models

Data Mart tables	Transaction tables
STORE_DIM	STORE
CUSTOMER_DIM	CUSTOMER
BOOKORDER_DETAIL_FACT	BOOK_ORDER, ORDER_DETAIL
TITLE_DIM	TITLE
TIME_DIM	

Archive

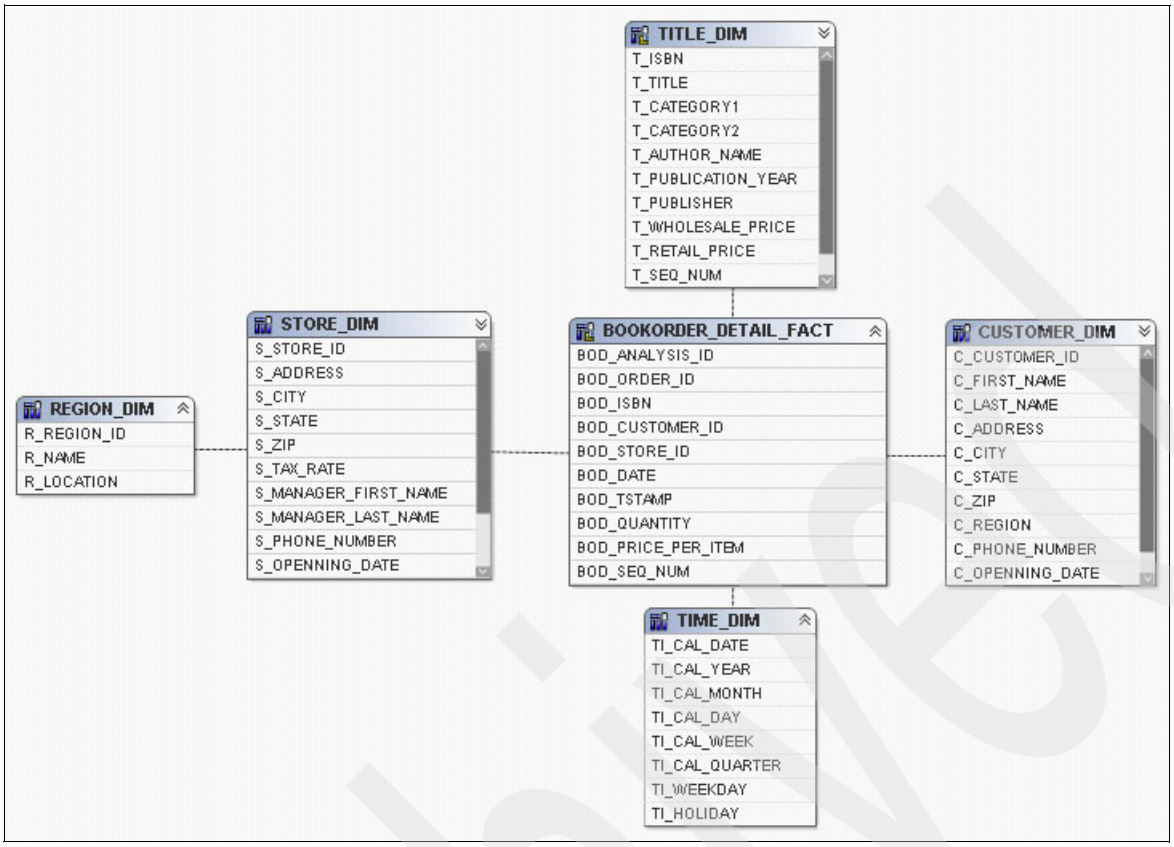


Figure 4-3 The Star schema of the book order data mart

4.3.2 Query speedup through Materialized Query Table (MQT)

A materialized query table (MQT) is a DB2 table that contains the results of a query, along with the query's definition. It may be thought of as a materialized view or automatic summary table that is based on an underlying table or set of tables.

MQT effectively eliminates overlapping work among queries by doing the computation once when the MQTs are built and refreshed, and then reusing their content for many queries. In our IBM Cognos BI workload (described in subsequent chapters), we created an MQT to answer a specific type of transactional report: for a book, say *Catcher in the Rye* in the shopping cart, what books did other customers buy along with *Catcher in the Rye*? This is commonly referred as the cross-sale opportunity identification by recommending two or more products that are often bought together.

It is not hard to see the rationale of building an MQT for the cross-sale query. In order to come up with the top recommendation of a pair of books, one would have to self-join the fact table BOOKORDER_DETAIL_FACT and count the number of occurrences of a pair of books bought by the same customer. During testing, based on the volume of data in our fact table, it was determined that the MQT should be built over one week of data to capture representative sampled distribution. The MQT was defined as followed:

```
CREATE TABLE AMAZON_MQT AS
(WITH FULL_LIST as

(SELECT CLIST.BOD_ISBN AS THIS_ISBN, FACT.BOD_ISBN AS OTHER_ISBN, COUNT(*)
TOTCOUNT
```

```
FROM PH3DW.BOOKORDER_DETAIL_FACT FACT, PH3DW.BOOKORDER_DETAIL_FACT CLIST
WHERE FACT.BOD_CUSTOMER_ID = CLIST.BOD_CUSTOMER_ID AND
FACT.BOD_ISBN <> CLIST.BOD_ISBN AND
FACT.DATE BETWEEN 2008-08-01 AND 2008-10-31
AND CLIST.DATE BETWEEN 2008-08-01 AND 2008-10-31
GROUP BY CLIST.BOD_ISBN, FACT.BOD_ISBN )
)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY USER
ENABLE QUERY OPTIMIZATION
IN ...
```

As the above clauses REFRESH DEFERRED and MAINTAINED BY USER suggested, the MQT will be refreshed by the DB administrator using a command.

This MQT was invoked through in-line SQL authored through IBM Cognos Report Studio. For details, refer to 6.3.4, “Operational BI reporting” on page 110.

4.4 Data refresh to build inventory history

Our project employed two types of commonly used data mart synchronization techniques. The first type is batch update, with data movement and transformation provided through InfoSphere Warehouse SQL Warehousing Tool (SQW). The second type is trickle update, with log-captured data changes reflected near real time through InfoSphere Change Data Capture (CDC) for z/OS.

While SQW may create temporary tables during the data movement and transformation flows, the tool manages the temporary tables and they are neither visible nor used in our experiments. The OLTP is schema shown in Figure 4-3 on page 64. It ends at the data mart incrementally populating two tables holding the history of price and inventory as shown in Figure 4-4 on page 66.

The PRICE_HISTORY table records the retail price of each copy of a book sold and the timestamp associated with the sale. The INVENTORY_HISTORY table records the store and the number of the books remaining in the inventory at the time of the sale. One can use these two tables to reconstruct the history of the book sales specific to an ISBN, with inventory information missing from the BOOKORDER_DETAIL_FACT table.

More details about the setup of InfoSphere CDC are described in Chapter 7, “BookStore workload 3 - data warehouse refreshing” on page 121.

Archived

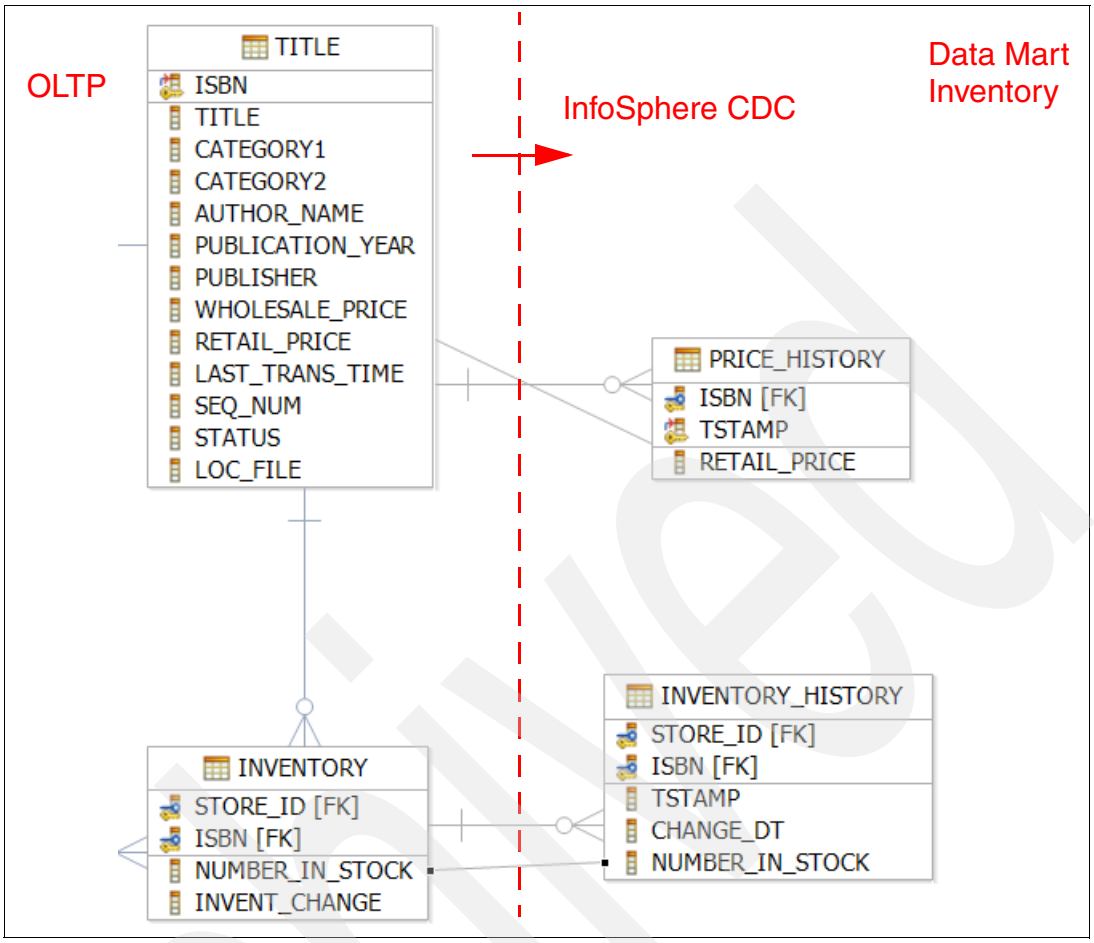


Figure 4-4 Inventory information captured through trickle update of InfoSphere CDC

4.5 Data modeling using IBM Industry Models

Our book store chain scenario, while simple and representative in the scope of this book, does not capture the full complexity of today's business applications. For many industries, IBM offers Industry Models based on deep expertise and best practice. The Models provide a good starting point for business and IT users to accelerate the development of business applications. For details, refer to the IBM Industry Models web page at:

http://www-01.ibm.com/software/data/industry-models/

Archived

5

BookStore workload 1 - transactional

In this chapter we discuss the Online Transaction Processing (OLTP) workload used in this project, including the following topics:

- OLTP workload
- BookStore transactions
- WebSphere WLM setup
- WebSphere setup topology introduction
- WebSphere Application Server environment variables

For general information on how to set up transactional applications and workloads such as ours, including the following topics, see Appendix E, “Setting up the transactional workload” on page 469.

- EJB Mapping using Rational® Application Developer (RAD)
- Application deployment on WebSphere Application Server
- JDBC Connections configuration
- WebSphere setup topology details
- WebSphere administration sample commands

Archived

© Copyright IBM Corp. 2010. All rights reserved.

5.1 Online transaction processing workload

The OLTP application used in this project is called “BookStore”, which is an IBM internally developed application. BookStore simulates an on-line bookstore and has the ability to search for books by title and author, add books to a shopping cart, purchase the contents of a shopping cart, get the status of a previously placed order, and other functions. The database supporting the BookStore application contains information about titles, authors, stores, customers, and orders.

For the purpose of this project, a book order transaction adds new orders to the database; no other database tables were modified when running this application. The database used for the transactional workload consisted of roughly 13 million customers, 6 million books and 1.8 billion orders. The overall size of the transactional database was approximately one half of a terabyte.The transactional database was used to create the data warehouse database and both were synchronized using Extract, Transform, and Load (ETL) processing with the transactional database as the source.

5.1.1 Components and architecture

The BookStore application is a three-tier Java2 Platform Enterprise Edition (J2EE) application. This architecture allows developers to focus on the business logic development while J2EE container code supports the underlying services that manage the transactions’ state, resource usage, and communication to the Enterprise Information System (DB2 in our case).

The BookStore application was developed using IBM Rational Application Developer V7. The application was built into an Enterprise Archive Repository (EAR) file and deployed on IBM WebSphere Application Server V7.

We used WebSphere Studio Workload Simulator (aka JIBE) as the client driver to send requests to the WebSphere Application Server. We chose JIBE instead of the more common tool, Rational Performance Tester, so that we could take advantage of previously developed JIBE scripts to run the workload. The JIBE scripts used for this project randomly created books to purchase and orders to query and then packaged these items into the appropriate HTTP request that was sent to WebSphere.

To simplify the test environment setup, we did not implement user authentication. We used container-managed authentication for the JDBC resource authorization. In general, security was disabled for this application when running experiments for this project.

WebSphere on z/OS architecture

From an application developer perspective, J2EE applications written for WebSphere Application Server utilize a common API with the other platforms that support WebSphere Application Server. However, WebSphere Application Server on z/OS has many additional benefits that are transparent to the application and that improve the quality of service of the application in production. WebSphere Application Server on z/OS actively exploits the following System z and z/OS attributes:

- SMP/E
- JES and common z/OS facilities
- zAAP specialty engines
- WLM
- RRS

- SAF and Crypto
- SMF
- z/OS exclusive Cross Memory Communications

While the detailed explanation of these advantages is the topic of another document, it is appropriate to understand WebSphere Application Server controller servant (CR/SR) architecture and exploitation of WLM, which is a foundational technology in a shared application and data environment.

WebSphere Application Server on z/OS has an application server architecture that is unique to the WebSphere Application Server family. Rather than a single JVM, it consists of multiple JVMs, each in its own address space: a controller region (CR) and one to many servant regions (SRs). This is configurable by you, the default behavior being one CR and one SR. One of the key advantages of this structure is that it separates pure “plumbing” code from user application code, which allows the plumbing code to run in a more protected address space. It also protects the plumbing code from JVM outages caused by misbehaving user applications. Figure 5-1 provides a description of what function is contained in which region. WLM sits in the middle of this and performs several key functions to ensure that resources are effectively managed.

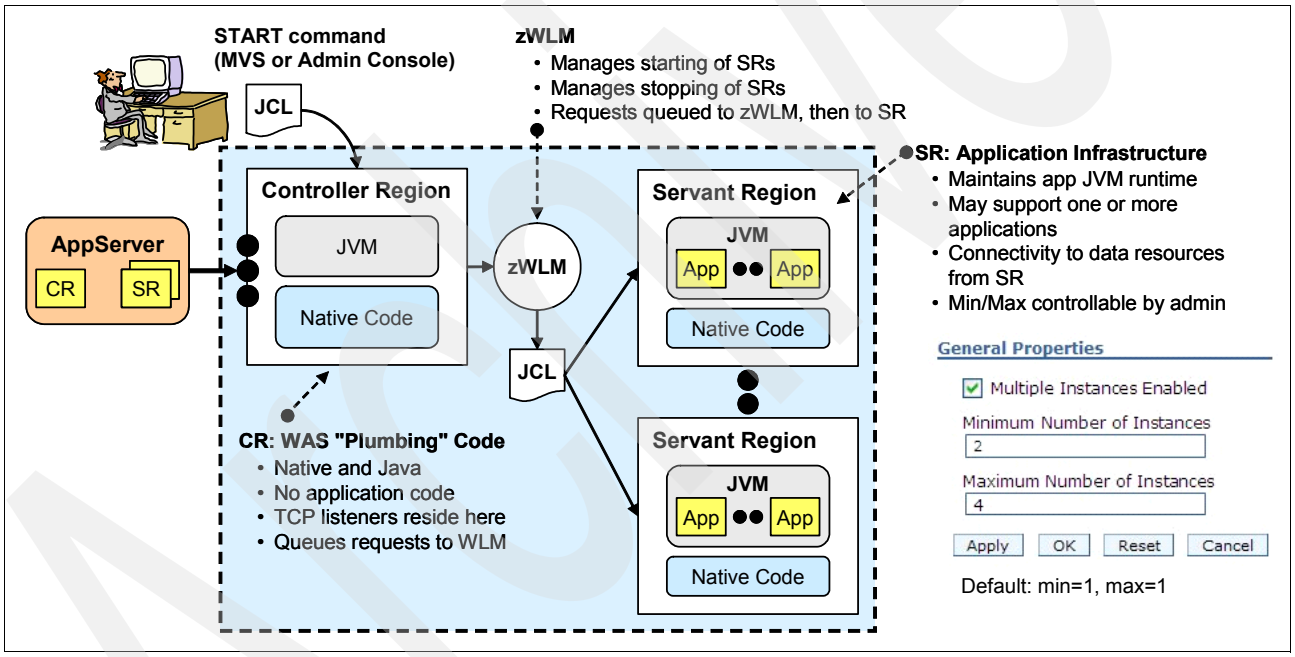


Figure 5-1 A description of what function is contained in which region

Dynamic capacity expansion

One of the things this CR/SR structure allows is the ability to dynamically expand the capacity for a given application server by WLM starting a second (or more) servants. This is based on WLM goals and WLM's monitoring of how well work is meeting the goals. If WLM sees that goals are not being met, WLM starts up a second servant region. When WLM starts another servant region, it creates an entirely new address space with a physically separate JVM from the first SR. The inbound work comes to the CR, which then queues the work to WLM. The SRs pull the work from the WLM queue based on its ability to process work. As WLM continues to monitor the goal attainment it may come to a point where it determines that the additional SR is no longer needed. It will then stop work from flowing to that servant region, allow the in-flight work to flush out, and then the SR is brought down.

Intelligent workload flow control

The model between the CR and the SR is a pull model, not a push. That means that inbound work that hits the CR goes first to a queueing mechanism, then gets pulled by the SR based on the SR's ability to do the work at that point in time. If a single SR is falling behind and WLM sees goals going unmet, it starts up a second servant region (provided it is configured to allow multiple servants). This queuing mechanism also provides the ability to route work based on priorities to specific servant regions, where WLM can manage system resources servant by servant.

Intelligent management of mixed work in the server/cluster

Work coming into a controller region may take many forms. Some of the work may be of higher priority than other work. The WebSphere Application Server z/OS CR/SR structure provides a way to segregate the work to individual servant regions so WLM can prioritize the allocation of system resources to achieve the goals. In this picture higher priority work goes to one SR and gets more system resources; lower priority work goes to the other SR and WLM grants it relatively less system resources based on the defined goals.

Intelligent workload routing advice

Finally, WLM offers WebSphere z/OS advice on where to route certain work flows. This is based on a real-time awareness on the part of WLM for the system resources across the Sysplex (not just within a given LPAR) and the goals defined.

J2EE architecture

J2EE provides a standard for developing multitier enterprise services. J2EE architecture supports component-based development of multitier enterprise applications. A J2EE application typically includes the following tiers:

- Presentation tier

  In the presentation tier, web components, such as servlets and JavaServer pages (JSPs), or standalone Java applications, provide a dynamic interface to the business logic tier.

- Business logic tier

  In the business logic tier, or middle tier, enterprise beans and Web Services encapsulate reusable, distributable business logic for the application. These server-tier components are contained on a J2EE Application Server, which provides the platform for these components to perform actions and store data.

- Enterprise data tier

  In the data tier, the enterprise's data is stored and made persistent, typically in a relational database.

J2EE applications are comprised of components, containers, and services. Components are application-level components. web components, such as servlets and JSPs, provide dynamic responses to requests from a web page. EJB components contain server-side business logic for enterprise applications. Web and EJB component containers host services that support web and EJB modules.

Container code provides underlying services for the components, such as security management, transaction management, Java Naming and Directory Interface (JNDI) lookup, and remote connectivity.

The container’s underlying services also provide the interfaces between the components and platform-specific functions. This design removes the burden of developing intricate code, for various standard services, from the application developers so that they can focus on the development of business logic code for the application.

J2EE has three container types:

- Client Container - Manages the execution of the J2EE client.
- Web container - Manages the execution of servlets and JSPs.
- EJB container - Manages the execution of Enterprise Java Beans (EJB).

The web container and EJB container run in the WebSphere application server. See Figure 5-2.

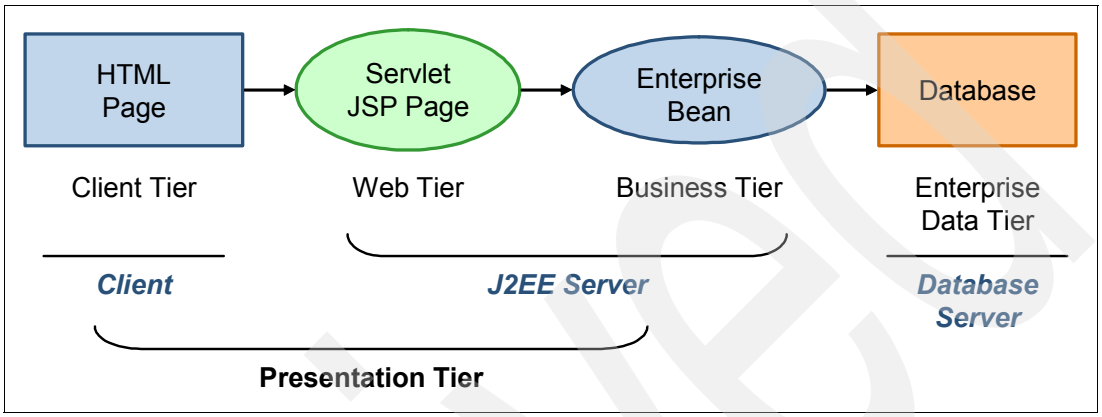


Figure 5-2 J2EE architecture

5.2 BookStore transactions

Our BookStore test environment emulates shopping transactions in an online book store. The functionality of the application is very broad and includes some of the following:

- Books can be searched for by title, author, ISBN, and so on.
- Details about the book are displayed, including a graphic of the cover.
- Books can be ordered from the search lists, with purchases added to a shopping cart, and the shopping cart can be edited to make changes.
- The status of previous orders can be queried.
- Inventories are updated based on sales.

Functionality outside the scope of the book includes an invoice generator from the order details, and the ability to add books based on information from the Library of Congress.

Depending on the function that is to be tested, when orders are placed, order processing is done in one of two ways:

- An MQ message is queued, the queued message causes code to be invoked that creates database records for the order and does inventory updates.
- A stored procedure is called that creates database records for the order and does inventory updates.

For this project, we did not implement MQ, but used the stored procedure to do order processing. We needed only two transactions: one that could be prioritized as more business-critical, and one that just consumed processor cycles and was considered less critical. A higher priority transaction was required so that it would force WLM to make resource decisions when the application was running at various loads. The two transactions

that were chosen were the book order purchase transaction and the administration order search transaction. The book order purchase transaction was the more business-critical transaction of the two transactions.

During the test, virtual clients sent transaction requests directly to the WebSphere Application Server, using HTTP, where the BookStore application was running. The two transactions were assigned to different WLM service classes, with the book order purchase transaction having a higher execution priority. (Refer to 6.4, “WebSphere Application Server setup topology” on page 65 for how to set up a WebSphere WLM transaction class.)

Details of the two transactions are:

- Book order purchase, which consists of the following steps:
  a. Randomly select the number of books to buy. Each BookOrder purchases a maximum of 3 books.
  b. Randomly select an author from a list.
  c. Randomly select 1 to 3 books (up to the BookOrder maximum of 3 books) from the book list returned from the previous step and put the selected books in the shopping cart.

  Repeat steps b and c until the number of books to buy has been met.

  d. Place the order, invoke the stored procedure BookBUY to update the DB2 backend.
  e. Log out to clean up any resources associated with this virtual client.
- Administration Order Search
  a. Randomly select an Order ID.
  b. Display the Order detail and status for the Order ID selected.
  c. Log out to clean up any resources associated with this virtual client.

5.3 WebSphere WLM setup

WebSphere for z/OS has the unique capability that it can classify incoming work based on communications protocol information and allocate resources used for execution of that work based on the work’s priority.

5.3.1 WLM classification of BookStore transactions

There are multiple communication protocol choices used to determine WLM transaction classification. These protocols include IIOP, HTTP, and MDB. For the co-location project, the incoming work that we wanted to classify arrived into the WebSphere server via the HTTP protocol.

The WebSphere Application Server maps incoming work based on protocol and information in that protocol to a WLM transaction class. WLM takes the transaction class as input and maps it to a service class from the WLM service policy. WLM uses information about how all service classes currently running in the system are performing to allocate system resources based on the overall priority of that work and how well that work is performing against its service goals.

Example 6-1 and 6-2 illustrate the definition of the two z/OS WLM classes used by the BookStore application in this project.

Example 5-1 WLM class WASTCHI

```
Service Class WASTCHI - Bookstore TC High
 Base goal:
 CPU Critical flag: NO

   #  Duration   Imp  Goal description
   -  ---------  -    ---------------------------------------
   1             2    90% complete within 00:00:00.250
```

Example 5-2 WLM class WASTCLOW

```
Service Class WASTCLOW - Bookstore TC Low
 Base goal:
 CPU Critical flag: NO
   #  Duration   Imp  Goal description
   -  ---------  -    ---------------------------------------
   1             3    90% complete within 00:00:00.900
```

This incoming work was made up of two different transactions, book order purchase and administration order search. These two transactions mapped to two different transaction classes, WASTCHI for a book order purchase and WASTCLOW for an administration order search. The transaction classes map to the corresponding and same-named WLM service classes, through the WLM service policy.

The WLM service policy for WebSphere work used for this project was really rather simple; there were only those two service classes for the WebSphere work. The book order purchase transaction was classified to the WASTCHI service class and all other work would be classified to the WASTCLOW service class. In the WLM service policy, WASTCHI was classified as the highest priority WebSphere work. This simple approach was taken to prove, in the project experiments, that WLM could adjust system resources based on the service class priority, service class goals, and incoming workload. For details about the WLM service policy and WebSphere service classes, refer to Appendix F, “WLM refresher” on page 491.

WLM will try to assign a service class to a particular instance of a WebSphere servant and only queue work for that service class to that servant. If there are more service classes than servant instances, work for multiple service classes may be queued to one servant instance. But based on the incoming transaction workload, that assignment of service class to servant instance may change to help WLM ensure that the WebSphere service classes are meeting their service goals. In this case, the servant instances are just another resource WLM must manage.

If a minimum and maximum number of servant instances are specified through the WebSphere Admin user interface, WLM will manage the number of servant instances and the assignment of service classes to servant instances to ensure that the service goals are achieved for the various service classes. This, coupled with the ability to dynamically change what service class work is being queued to a particular servant instance, gives WLM significant power to that ensure service goals are met for the WebSphere workload.

Example 6-3 illustrates the content of the WebSphere WLM classification file used for this project.

Example 5-3 wlm_classification_file used by the zDW project

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE Classification SYSTEM "Classification.dtd" >
<Classification schema_version="1.0">

<!--
     HTTP Classification Rules
-->
   <InboundClassification  type="http"
                           schema_version="1.0"
                           default_transaction_class=”WASTCLOW">
     <http_classification_info  transaction_class="WASTCLOW"
                                host="*"
                                description="Virtual Host xx.x.xxx.x">
       <http_classification_info transaction_class="WASTCLOW"
                                 port="908*"
                                 description="Def xx.x.xxx.x HTTP reqs">
         <http_classification_info transaction_class="WASTCLOW"
                                   uri="/zipseries/AdminAction"
                                   description = "order detail info" />
         <http_classification_info transaction_class="WASTCHI"
                                   uri="/zipseries/Action"
                                   description="book order" />
       </http_classification_info>
     </http_classification_info>
  </InboundClassification>

</Classification>
```

5.3.2 Verifying the WebSphere WLM classification implementation

WebSphere version 7 will cut an SMF Type 120, Subtype 9 record with the transaction class information along with the protocol information that was used to classify the incoming work to that transaction class. This SMF record can be used to verify that your WebSphere WLM classification XML file is set up as desired.

Using the SMF browser available from IBM, the contents of the classification data section, from the SMF type 120 subtype 9 record, for the first of our two transactions (administration order search), looked like this:

```
Triplet #: 7; Type: ClassificationDataSection;
  Version    : 1;
  Data Type  : 6 (URI);
  Data Length: 22;
  Data       : /zipseries/AdminAction (EBCDIC);

Triplet #: 7; Type: ClassificationDataSection;
  Version    : 1;
  Data Type  : 7 (Target Hostname);
  Data Length: 10;
  Data       : xx.x.xxx.x (EBCDIC);

Triplet #: 7; Type: ClassificationDataSection;
  Version    : 1;
  Data Type  : 8 (Target Port);
  Data Length: 4;
  Data       : 9081 (EBCDIC);
```

For this particular SMF record, the ZosRequestInfoSection contained the following:

```
Transaction Class              : WASTCLOW;
```

The contents of the SMF type 120 subtype 9 classification data section of the SMF record for the second workload transaction (book order purchase) looked like this:

```
Triplet #: 7; Type: ClassificationDataSection;
   Version    : 1;
   Data Type  : 6 (URI);
   Data Length: 17;
   Data       : /zipseries/Action (EBCDIC);

 Triplet #: 7; Type: ClassificationDataSection;
   Version    : 1;
   Data Type  : 7 (Target Hostname);
   Data Length: 10;
   Data       : xx.x.xxx.x (EBCDIC);

 Triplet #: 7; Type: ClassificationDataSection;
   Version    : 1;
   Data Type  : 8 (Target Port);
   Data Length: 4;
   Data       : 9081 (EBCDIC);
```

And for this particular SMF record, the ZosRequestInfoSection contained the following:

```
Transaction Class              : WASTCHI;
```

The SMF data from the SMF browser for the BookStore transactions confirms our WLM classification file was setup correctly. The SMF browser can also be used to generate a skeleton WLM classification XML file. This skeleton classification file can then be modified as appropriate and used to classify incoming WebSphere Application Server work.

The SMF type 120 subtype 9 data is new for WebSphere version 7. For information on the SMF type 120 subtype 9 data, refer to IBM TechDoc WP101342, “Understanding SMF Record, Type 120, Subtype 9".

5.3.3 Monitor the WebSphere WLM work queues

You can use the tool WLMQUE to view status on the WLM application environments, service classes, and supporting WLM work queues. It is an ISPF based tool that can monitor status of the WebSphere servant instances. For each servant instance’s work queue, it shows which WLM service class was assigned. It also shows how many requests are waiting to be processed in the WLM queues. This was useful for the co-location project so that we could get a quick glimpse of how well the BookStore workload and WebSphere were performing.

Figures 5-3 and 5-4 on page 76 show the contents of the WLMQUE monitor ISPF panels for the BBOC003 application environment (for server BBOS0003). There are two WebSphere servant address spaces for this server.

Initially, in Figure 5-3, the only work queued to those address spaces is from the WASTCLOW service class. This is because the WebSphere Application Server has been started but no incoming work has been received for the BookStore application. The only work classified to the WASTCHI service class is from the BookStore application book order purchase transaction. All other work is classified as WASTCLOW. So the only service class work that should be visible at this point is for the WASTCLOW service class.

Archived

Figure 5-4 shows the WLMQUE panel after the BookStore application has started receiving incoming work for the two BookStore transactions. Notice that the WASTCLOW and WASTCHI service classes are each assigned to a WebSphere servant address space. Work for each of those service classes is being executed on those servants.

Download the WLMQUE tool from this website:

http://www.ibm.com/servers/eserver/zseries/zos/wlm/tools/wlmque.html

To invoke the tool from TSO, type the following command:

==> ex '*data set.name*(WLMQUE)'

```
Command ===>                                          Scroll ===> PAGE
                  Application Environment Monitor
Selection:  >HELP< >SAVE< >OVW< >ALL<
System: P61       Sysplex: PATPLX61  Version: z/OS 011000  Time: 17:46:34

ApplEnv_ Type SubName_ WMAS Del Dyn NQ QLen Str Hav Unb Trm Min_ Max__ ICnt
BBOC003  CB   BBOS003  0058 No  Yes  1    0   0   2   0   0   30    30    0

WorkQue_ Del Wnt Hav ICnt QueIn_ QueOut QueLen QueTot__ Act_ Idl_
WASTCLOW No    2   2    0      0      0      0        1    0   30

SvAS Binding_ Ter Opr Btc Dem Have PEU_ ICnt WUQue____ Aff AffQue
0055 WASTCLOW No  No  No  No    15   15    0         1   0      0
005A WASTCLOW No  No  No  No    15   15    0         0   0      0
```

Figure 5-3 WLM queues before the Book_Order transaction was started

```
Command ===>                                          Scroll ===> PAGE
                  Application Environment Monitor
Selection:  >HELP< >SAVE< >OVW< >ALL<
System: P61       Sysplex: PATPLX61  Version: z/OS 011000  Time: 17:48:25

ApplEnv_ Type SubName_ WMAS Del Dyn NQ QLen Str Hav Unb Trm Min_ Max__ ICnt
BBOC003  CB   BBOS003  0058 No  Yes  2   21   0   2   0   0   30    30    0

WorkQue_ Del Wnt Hav ICnt QueIn_ QueOut QueLen QueTot__ Act_ Idl_
WASTCLOW No    1   1    0      0      0      0        1    0   30
WASTCHI  No    1   1    0     16      0     21       54    0    0

SvAS Binding_ Ter Opr Btc Dem Have PEU_ ICnt WUQue____ Aff AffQue
0055 WASTCHI  No  No  No  No    15   15    0        34   0    107
005A WASTCLOW No  No  No  No    15   15    0         0   0      0
```

Figure 5-4 WLM queues after the Book_Order transaction was started

5.4 WebSphere Application Server setup topology

There are two WebSphere configuration environments for this project, single LPAR and multiple LPAR environments. The first was using a *single DB2 instance* that contained the transactional and data warehouse data. The second environment was using a *data sharing group* with multiple DB2 instances able to access both the transactional and data warehouse data.

Multiple WebSphere setup topologies can be chosen. In this project, all of the WebSphere topologies used Network Deployment topologies, with a deployment manager, node agents and one or more WebSphere servers. The decision was made to run this way because of the flexibility it allowed when switching between environments. Nothing had to be reconfigured, re-installed or even changed when switching between environments. If runs were needed to be made in the single DB2 instance environment, the system was IPLed and the deployment manager, node agent, and an application server were started. If runs were needed to be made in the data sharing environment, the systems were IPLed and the deployment manager, two node agents, and in this case two application servers, one on each node, were started.

In the *single DB2 instance* environment, there was one WebSphere Application Server that was used for the two BookStore transactions. This WebSphere Application Server had two servant instances configured. The decision to run with two servants was made for a couple of reasons:

- To double the number of application threads.
- There were two BookStore transactions, each classified to its own WLM service class. WLM will try to queue work for a service class to its own servant if available. With the two transactions, the incoming work for each transaction would be queued to its own servant. This should alleviate any issues that might arise from having both service classes execute on a single servant because of the two service classes and different priorities.

In the *data sharing environment*, the WebSphere Deployment Manager managed the cell across two System z LPARs, with each LPAR containing a node agent and an application server. Each application server also had two servant instances running and for the same reasons listed for the single DB2 instance environment.

During the OLTP testing, the goal was to achieve a 90% processor utilization and maintain that utilization for the duration of the measurement. If running in the data sharing environment, each of the servers in the two LPARs should produce equal transaction rates and also achieve similar processor utilizations. In either the single DB2 instance environment or the data sharing environment, the decision was made to have the systems run at roughly the same transaction rates.

We used a standard WebSphere configuration, except for configuring the application servers with two servants each. We made no modifications to thread counts and server_region_workload_profile. We modified control region and servant region Java heap to be 512 m and 768 m, respectively. We also set special WebSphere environment variables. See 5.5, “WebSphere environment variables” for more details.

There were several ways to configure WebSphere servers to service the work to meet our requirements. Since these application clients used the HTTP protocol, these topologies needed a web server to handle the HTTP protocol. Three web server topologies are described for reference. Since the main topic discussed in this book is WLM resource management rather than the OLTP performance tuning, for ease of the configuration effort the decision was made to send requests directly to WebSphere Application Servers running with the HTTP transport. The configuration is described in Appendix E, “Setting up the transactional workload” on page 469.

5.4.1 Distributing incoming work to application servers

What really drove the WebSphere configuration for this project? In the data sharing environment there were two DB2 instances in a data sharing group that handled the incoming HTTP clients and the requests for transactional data. Each of the two DB2 instances resided on its own System z LPAR configured into a sysplex. The one requirement we had for the

WebSphere configuration was that it had to be able to drive those DB2 instances to a high processor utilization and with equal client distribution.

The decision was made to go with the simpler approach for the WebSphere configuration. Rather than drive the two DB2 instances from a single WebSphere Application Server, two WebSphere Application Servers were used, each connected to a DB2 instance. Since the DB2 instances were on separate systems, driving from a single WebSphere Application Server would have meant that one or possibly both DB2 instances would have been connected to WebSphere with a Type 4 JDBC driver. In most cases, Type 2 JDBC will perform better than Type 4 JDBC when the WebSphere Application Server and DB2 instance are in the same z/OS LPAR. Since this was our configuration and many clients run with Type 2, we chose to run with Type 2 JDBC. If one WebSphere Application Server was connected with Type 2 and the other connected with Type 4, the possibility of not driving equal transaction rates existed. Getting similar behavior out of both systems was important, and if the systems are configured differently, getting to the root cause of performance differences could be difficult. Taking this into account, each System z LPAR would have its own WebSphere Application Server connected to the DB2 instance on that same system with a Type 2 JDBC driver.

How to handle the incoming HTTP work and distribute that work to the WebSphere Application Servers was the next issue that needed to be resolved. Three possible alternatives were explored:

- Use the HTTP transport that is built into the WebSphere Application Server. Each application server would be a standalone entity that would handle the incoming HTTP protocol and execute the request.
- Use a standalone web server with a WebSphere plug-in to route the requests to the WebSphere Application Servers.
- Use the WebSphere Proxy server and route the requests to the application servers.

Both the standalone web server and the WebSphere Proxy server seemed more applicable to the IBM clients. Each has affinity capability, which is the capability to route incoming work based on affinity to an application server caused by previous associated work that was routed to that application server. Routing of new work to the application servers for the standalone web server was a round-robin approach that could be weighted. The routing of work for the Proxy Server is based on metrics from WLM, which will give better distribution of work based the current load on the application servers.

The decision was to go with the HTTP transport that is built into the WebSphere Application Server. Clients would be set up to send HTTP requests directly to each of the WebSphere Application Servers. Each of the web servers would have an equal distribution of the two BookStore transactions routed to it from these clients. This simplified the overall approach and would be more than adequate for what was needed.

Send requests to the HTTP server

Figure 5-5 depicts one of the possible topologies for the co-location book in a data sharing environment. The graphic depicts the topology used for the project.

Archive

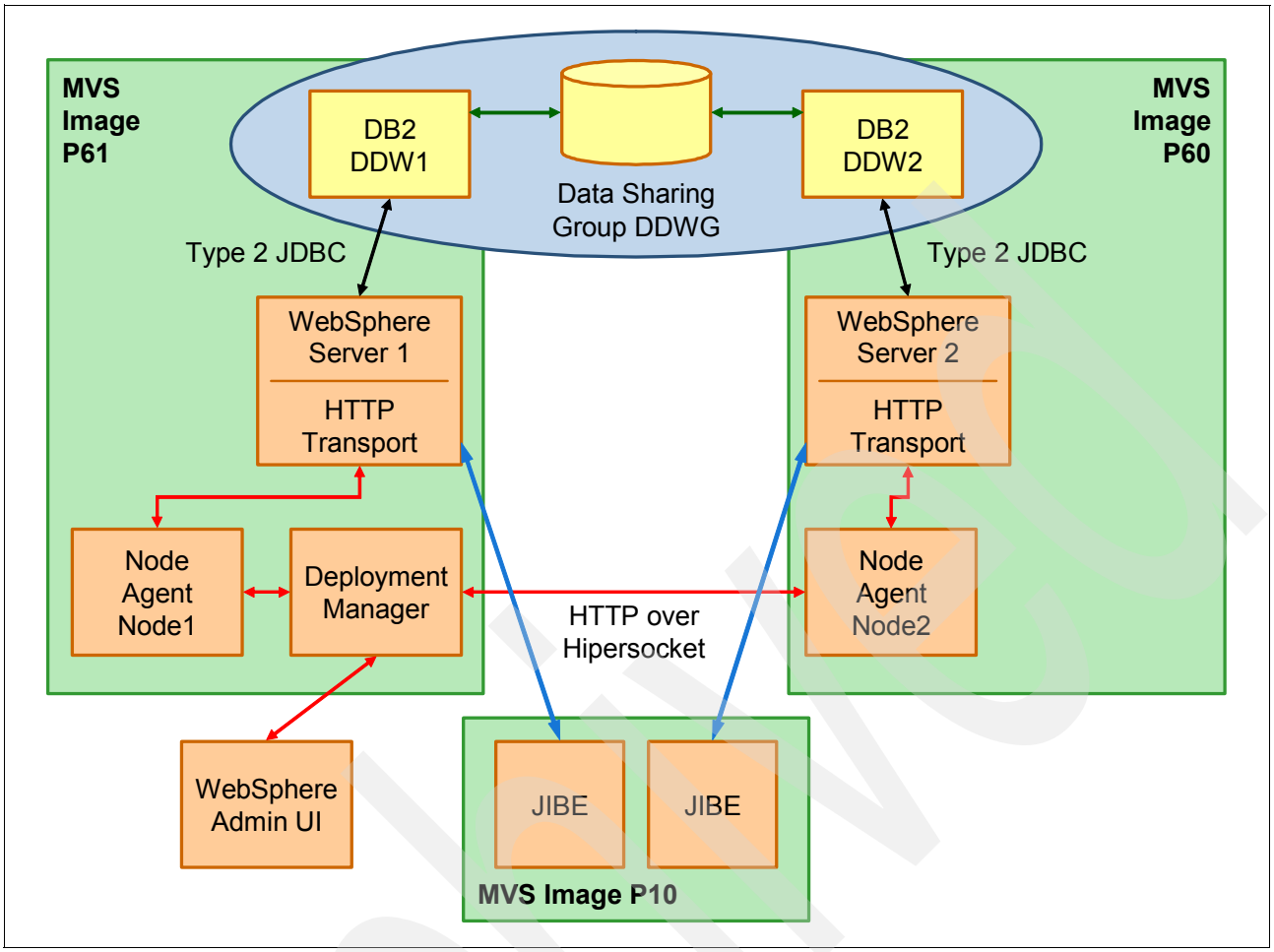


Figure 5-5 Transactional workload topology using the HTTP transport

Send requests to a WebSphere Application Server

Figure 5-5 shows clients running on system P10, sending HTTP requests to WebSphere Application Servers running on systems P60 and P61. There are two sets of clients, one set sending HTTP requests to the application server on system P60, and the other set sending HTTP requests to the application servers on system P61. The application servers are using the HTTP transport that is built into the WebSphere Application Servers. There are two DB2 instances running in a data sharing group. One of these instances is running on system P61, and the other is running on system P60. The data sharing group also includes DB2 instances that access the data warehouse data. The DB2 instances for the data warehouse data are not shown in this diagram.

5.5 WebSphere environment variables

In the test environment, WebSphere environment variables explicitly set for the BookStore application are listed here.

To set one of these variables, go to the WebSphere administrative console.

Click **Environment** → **WebSphere variables**, select the scope for either an application server, node, or the deployment manager, then click **New** and fill in the value. The values specified in the following section are examples.

- protocol_http_timeout_output_recovery= SESSION

  Specifies the recovery action that is taken when an HTTP request has timed out and does not complete. The default value for this property is SERVANT, which will terminate a servant when a time-out occurs. When it is set to SESSION, no attempt is made to disrupt the processing of a dispatched HTTP request within a servant. However, the HTTP request and socket are still cleaned up. There were occasional issues where clients would experience an HTTP time-out. If the recovery action was to terminate the servant, this could be disruptive to many more than just one client, which could be catastrophic to a particular test run. It was set to SESSION so that other client requests running in the servant could hopefully continue and complete successfully. Review the information about this environment variable in the WebSphere Infocenter and understand the ramifications to your environment before changing this environment variable.

- JDBC driver path

  ```
  DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH=/db2910/currentdb2g/db2910_jdbc/lib
  DB2UNIVERSAL_JDBC_DRIVER_PATH=/db2910/currentdb2g/db2910_jdbc/classes
  ```

- wlm_classification_file=/WebSphere/V7R0/bkstwlm/bkstwlm.xm

  Takes various attributes from incoming protocols and maps them to a WLM transaction class. The mapping is specified in the XML file.

Figure 5-6 displays part of the WebSphere environment variables at the server scope.

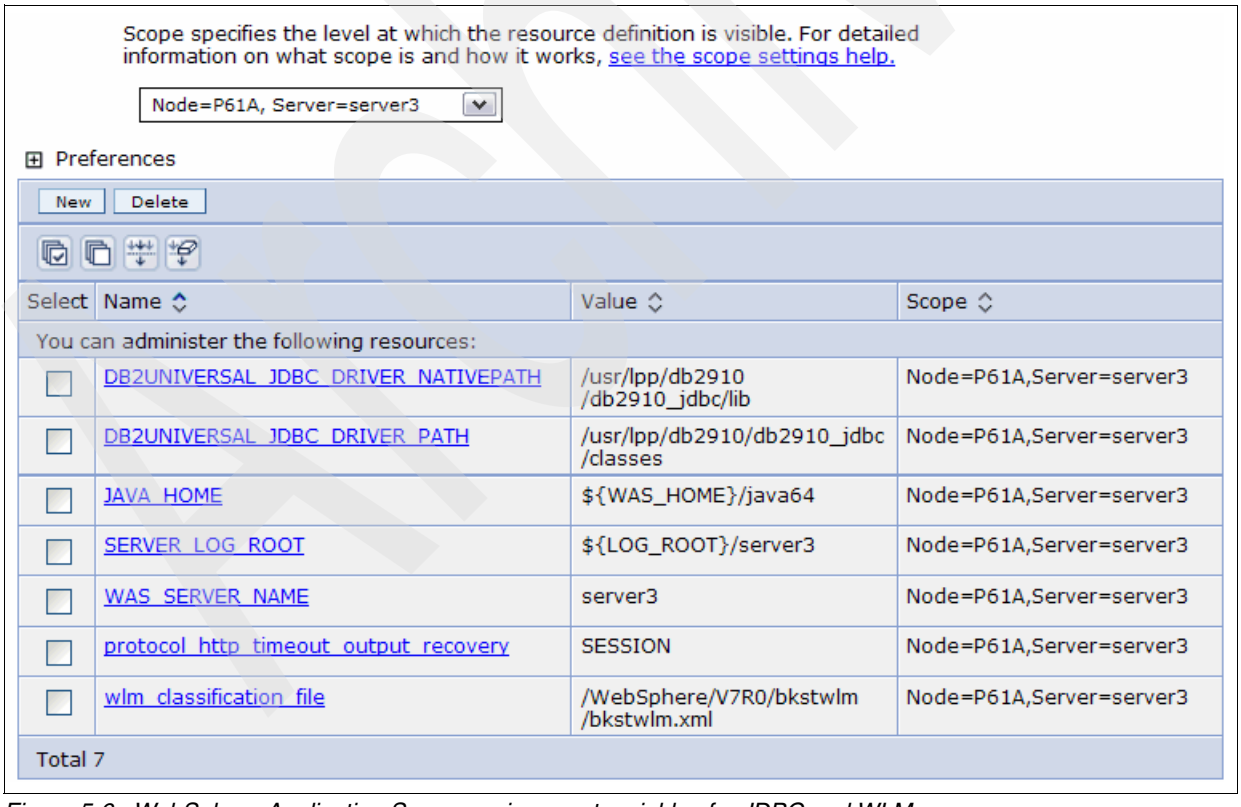


Figure 5-6 WebSphere Application Server environment variables for JDBC and WLM

6

BookStore workload 2 - IBM Cognos BI reporting

In this study, we focused on the reporting and analysis features incorporated in the IBM Cognos Business Intelligence (BI) product. Using these technologies, IBM Cognos BI reports were developed against data that resided in DB2 for z/OS 9. Reports were created and categorized by four levels of CP utilization for a given set of parameterized inputs. In other words, CP-intensive workloads could be run against reports using a particular set of defined inputs, while CP “trivial” workloads could be driven with the same reports using a different set of defined inputs. The environment used to generate WLM classified IBM Cognos BI workloads, along with tuning suggestions derived from test case analysis, is documented in this chapter.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

6.1 Business Intelligence with IBM Cognos BI

IBM Cognos Business Intelligence (BI) provides organizations with the ability to leverage trusted information to understand how their business is performing, to make faster, more informed, more aligned decisions, and to focus on optimizing performance across the enterprise. This is accomplished through business intelligence and performance management. IBM Cognos BI offers a complete set of BI capabilities, including reporting, analysis, dashboards, and scorecards, that allow users across an organization to get vital performance information where, when, and how they need it. Moreover, IBM Cognos BI represents a complete range of BI capabilities in an open, enterprise-class platform, built on a service-oriented architecture. The BI capabilities are exposed through a zero footprint web browser interface with centralized administration. These features make it easy for businesses to integrate, use, and maintain the IBM Cognos BI platform.

The broad range of BI capabilities ensures that all users receive the most relevant information how, when, and where it is needed. The open, enterprise-class IBM Cognos BI platform provides IT with cost-effective scaling to meet growing user demands. Decades of expertise in both mainframe and business intelligence have led to world-class technologies and services that drive client success. Running IBM Cognos BI on System z brings the well-known benefits of System z to the Business Intelligence front end.

The IBM Cognos BI architecture can be separated into three distinct tiers: the presentation tier, the application tier, and the data tier; see Figure 6-1.

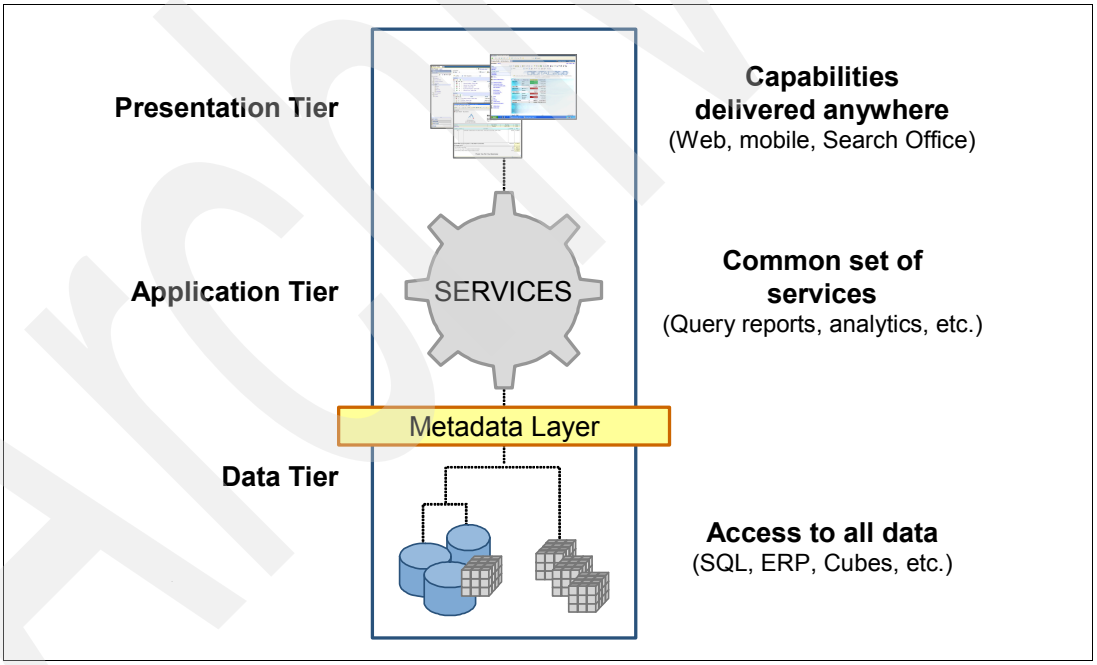


Figure 6-1 IBM Cognos BI high-level view

Presentation tier

The presentation tier of IBM Cognos BI conforms to these operating criteria:

- First, it provides all BI capabilities in a zero-footprint browser. This means users access the capabilities 100% through the web, without requiring any installation of software or downloads on their local PC. For the business this means employees can use the familiar environment of a browser to easily gain access to the information they need to drive decisions. The value for IT in this approach is no install, no compatibility concerns, no

costly upgrade, and freedom to support a user community not tied to specific PCs. This zero footprint enables broad cost-effective rollouts to thousands of users, and does not require lots of IT resources.

- Second, the platform is designed to fit into any web environment. It works with Microsoft IIS, IBM WebSphere, and Apache, to name a few. Many clients have existing web infrastructure in place that they want to keep using, such as firewalls, to ensure security or load balancing routers that spread requests across server farms. The IBM Cognos BI platform fits within and exploits an existing infrastructure, requiring few or no modifications.
- Third, IBM Cognos BI is built on an open web services architecture. Integration with portals, Microsoft Office, business process software and enterprise search are provided. IBM Cognos BI can easily be integrated with existing applications and customized through open interfaces.

Application tier

The IBM Cognos BI platform is based on an SOA architecture. Services in the IBM Cognos BI application tier operate on a peer-to-peer basis. That means that no service is more important—there is no “master” service. Any service of the same type, on any machine in a IBM Cognos BI configuration, is capable of servicing an incoming request. The benefit? Complete fault tolerance; any request can be routed to and handled by any server in the system. The dispatching of requests is done in an optimal way, with load balancing built into the system. As requests come in, they are automatically distributed in “round robin” fashion to servers within the system, based on server capacity.

The dispatcher receives requests from the gateways and forwards them to the appropriate service to handle them. When you install IBM Cognos BI on a server in a configuration, the dispatcher simply registers itself in the Content Store and lets the configuration know what services it can provide from that particular location. This greatly simplifies the installation and configuration of a BI system.

Each dispatcher can handle one or more services, and you have control to determine which services or group of services run on each machine. For example, you can choose to run only the query service that generates SQL or MDX on a set of servers. Or you can easily dedicate a server or two just to presentation services by disabling other services on that server. In this way you can configure the services in your environment for optimal use.

We already talked about peer-to-peer in the context of scalability and reliability. Another characteristic of these peer-to-peer services is that they are loosely coupled, with each performing a discrete function. This means there is independence between the services, for example the presentation layer rendering of results in HTML or PDF is separate from the data access layer. (However, both the query engine and the report server must run on the same system.) IBM Cognos BI can add new capabilities in one area that can leverage all the power of another area.

In addition, the architecture has a single open API. All communication between services takes place on a high performance IBM Cognos BI bus. All these services plug into the network using an Internet protocol of SOAP and XML, which gives the platform location transparency. The internal communication between services utilizes the same API that is exposed in the software development kit. You can use the development kit to exploit these APIs with the knowledge that they are used within the architecture.

Data tier

In order to be successful, all data must be accessible to the service, regardless of where the data resides (*data tier*).

The IBM Cognos BI platform accesses the data by using a combination of open standards such as SQL and MDX, as well as user-entered statements. This optimizes data retrieval for all these different data providers. The IBM Cognos BI platform also provides flexibility in terms of sourcing the data. When trying to address the wide variety of user needs, requirements will vary. Some data needs to be aggregated, calculated, or history captured (ETL). Some data needs to be sourced in real time (direct). Some data needs to be federated across multiple systems.

IBM Cognos BI supports four main BI capabilities: reporting, analysis, dashboards, and scorecards.

Reporting is often the best understood and the most broadly adopted capability. IBM Cognos BI supports all the types of reporting an enterprise requires including transactional reports, managed reports, high fidelity reports (like those you might provide to a customer), business ad hoc reporting, and analytical reporting, and so on. Features such as scheduling, briefing books, flexible object embedding, and interactive charting abilities, provide you the ability to customize BI output in virtually any desired style or quality. IBM Cognos BI is unique in its ability to let business users decide which information they want delivered to them. With self-service personal alerts, a manager can select information in a report and identify this as something they want to pay attention to.

The IBM Cognos BI analysis functionality enables users to explore information and perform multidimensional analysis against IBM Cognos BI data. This feature offers the ability to analyze and report against several leading OLAP data sources and dimensionally modeled relational data. IBM Cognos BI Analysis Studio offers an engaging and highly flexible interface; its features include the ability to explore large data sets using drag and drop technology, analyze and gain insight from relationships using drill down, ranking, sorting, graphing capabilities (and many others), convert large data sets from multiple disparate sources into Power Cubes, and so on.

Dashboards provide highly visual, at-a-glance views of information to ensure that executives and business managers can quickly focus on the areas of performance that need attention and action. They help monitor the business. For example, “Are sales trending up or down?”

Scorecarding provides the ability to perform comparisons of performance against a benchmark, a threshold, or any other target that the organization has set as an objective. It also helps align decisions and tactics to those strategic initiatives. Scorecards essentially help measure the business. For example, “Are actuals exceeding forecast?” or “Is my margin within an acceptable range of the target?”

With dashboards and scorecarding, the business users get a snapshot of how they are doing at any point in time. They need reports, analysis, and scorecards for additional context to understand what it means and what they might do about it. IBM Cognos BI lets you develop a complete view of the business for truly actionable business intelligence.

6.1.1 IBM Cognos BI and Linux on System z

IBM Cognos BI for Linux on System z offers the comprehensive business intelligence solution for the mainframe. With this product, customers can efficiently report and analyze transactional systems directly on the mainframe, so workers across an organization can use information to quickly identify and respond to critical business trends.

At the time of writing this book, IBM Cognos Business Intelligence Version 8.4 SP2[1] was available for Linux for System z. This provided built-in support for DB2 for z/OS as a data source and could run in the context of WebSphere Application Server for Linux for System z.

This book does not describe how to install and configure the IBM Cognos BI software components on Linux for System z. Refer to the IBM Cognos BI installation documentation and *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637 for a more detailed description of this.

6.1.2 IBM Cognos BI software components

IBM Cognos BI comes with several product components and interacts with existing components on both z/OS and Linux for System z.

Figure 6-2 shows an overview of the IBM Cognos BI architecture on Linux for System z that was used for this book. Each of the components is described in more detail.

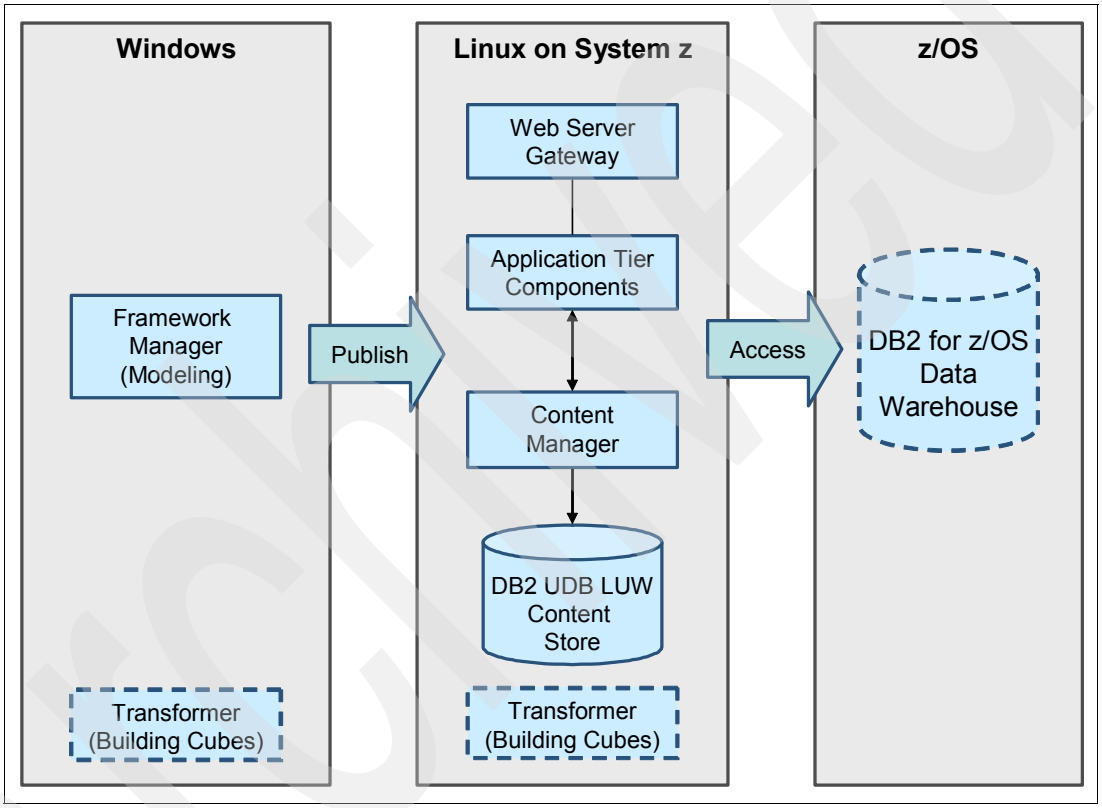


Figure 6-2 IBM Cognos BI software components

Modeling

IBM Cognos Framework Manager is a natural starting point in developing IBM Cognos BI end user information. IBM Cognos Framework Manager is the window-based tool used to create models of source data (in the form of metadata) that is exposed to IBM Cognos BI users. In developing the framework model, users define relationships between data entities, create new query subjects, create calculated query items and filters, and so on. The end result is a *package* that is given to users. This package allows users to work with the defined data objects in analysis and reporting, without needing any knowledge of the source of the data. The modeling used for this study is described in further detail in this chapter.[2]

[1] For more information on versions and support, see http://www-01.ibm.com/software/data/cognos/customercenter/
[2] Various OLAP sources require no additional modeling.

Web server (gateway)

IBM Cognos Connection, the web portal for IBM Cognos BI Server, is accessed by users through web browsers. IBM Cognos BI comes with extensions to a web server (such as modules for the IBM HTTP server) to act as a gateway, routing any requests to the dispatcher in the IBM Cognos BI application tier components. For scalability, multiple web servers may be installed.

Application tier components

The IBM Cognos BI application tier contains one or more IBM Cognos BI Servers. An IBM Cognos BI Server runs requests (such as reports, analysis, and queries) that are forwarded to it by a gateway.

Three of these services are particularly important in our environment: the *dispatcher*, the *content manager service,* and the *report service*.

The dispatcher starts all IBM Cognos BI services configured and enabled on a computer, then routes requests. The dispatcher is a multithreaded application that uses one or more threads per request; see Figure 6-3.

The content manager service performs object manipulation functions in the content store, such as add, query, update, delete, move, and copy. It also performs content store management functions, such as import and export.

The report service manages interactive requests to run reports and provides output for a user in IBM Cognos Viewer.

Content manager and content store

The content manager is an application tier component with special characteristics. While you can define multiple content manager services for fail-over purposes, only one instance can run at a time.

The content store is a relational database that contains the objects that IBM Cognos BI needs to operate. This includes report specifications, published packages (metadata), connection information for data sources, information about the external namespace, and the IBM Cognos BI namespace itself. The IBM Cognos BI content store database can be installed with other IBM Cognos BI components or installed on a separate computer. For this book, DB2 for Linux for System z was used as the content store.

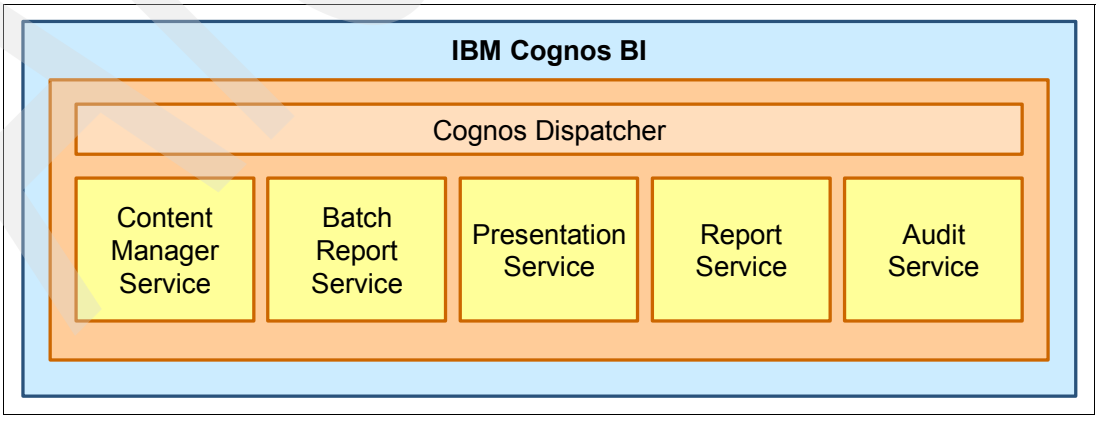


Figure 6-3 A set of IBM Cognos BI services

Data sources

Data sources, known as query databases, are relational databases, dimensional cubes, files, or other physical data stores that can be accessed through IBM Cognos BI. Application tier components use data source connections to access data sources.

For IBM Cognos BI for Linux on System z, the most common data source is a relational database residing in DB2.

6.2 System environment and configuration options

The following sections describe the environment and configuration that was used for this portion of the study, as well as any tuning and optimization practices that were applied.

Note: The environmental settings defined here describe a sample configuration that was used for these tests. It should not be considered a "best practice" configuration, nor should it be construed as a fully optimized and tuned environment for the class of tests that were run.

Consult the IBM Cognos BI Server documentation for recommended configuration decisions, as it provides a far more comprehensive discussion of the various options.

6.2.1 Product components

For the hardware configuration, two LPARs were defined: one for z/OS and the other for z/VM with Linux for System z running as a guest. Table 6-1 gives an overview of the product components that were installed and used for this portion of the study.

Table 6-1 Installed product versions

Component	Product and Version
z/VM	Version 5.3.0, driver 5 Linux for System z is the only guest system
Linux for System z	SUSE Linux Enterprise Server 10 (SP2) 8 GB physical memory 8 CPs (IFLs) assigned to Linux
HTTP Server (gateway)	IBM_HTTP_Server/6.1.0.25 Apache/2.0.47
Application Server	WebSphere Application Server 6.1.0.25, 31 bit Installed in `/opt/IBM/WebSphere/AppServer`. Two application server profiles defined: AppSrv01 and AppSrv02. JVM heap size is (256 M initial, 512 M max) for both servers. Thread pool for web sessions is set to 512 with the option to exceed the limit, if required. (Note that 64-bit IBM Cognos BI.4 is now available and can be run with 64-bit WebSphere Application Server.)

Archived

Component	Product and Version

Java Runtime Environment	We use the JDK that comes with the Application Server.
	`java -version` reports:
	`java version "1.5.0"`
	`Java 2 Runtime Environment, Standard Edition (build pxz31devifx-20090512c (SR9-SSU +PK76176+IZ44410+IZ44495+148247))`
	`IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux s390-31 j9vmxz3123ifx-20090225 (JIT enabled)`
	`J9VM - 20090224_30451_bHdSMr`
	`JIT - 20081112_1511ifx1_r8`
	`GC - 200811_07)`
	`JCL - 20090512c`

Database for IBM Cognos BI Content Store	DB2 for Linux for System z, version 9.5.0 with FP1
	`db2level` reports
	`DB21085I Instance "db2inst1" uses "64" bits and DB2 code release "SQL09054" with level identifier "06050107".`
	`Informational tokens are "DB2 v9.5.0.4", "s090429", "MI00305", and Fix Pack "4".`
	`Product is installed at "/opt/ibm/db2/V9.5"`

Data source for relational data	DB2 9 for z/OS
IBM Cognos BI Server	IBM Cognos BI Server Version 8.4 We installed the server components twice on the same machine in the following directories: ► `/opt/IBM Cognos` ► `/opt/IBM Cognos_rs2`

6.2.2 Topology and setup

The IBM Cognos BI Server components (such as the gateway, the application tier components and the content store database) may be distributed horizontally and vertically among multiple Linux for System z installations. Figure 6-4 on page 89 shows a sample distribution where multiple Linux systems are used for different tiers as well as duplicate systems within the same tier. All of the Linux for System z installations may be managed by z/VM or run natively in dedicated LPARs.

Note: At the time this book was started, DB2 for z/OS was not supported as a content store for IBM Cognos BI. DB2 for z/OS is now the preferred content store location for IBM Cognos BI deployments on Linux for System z.

Archived

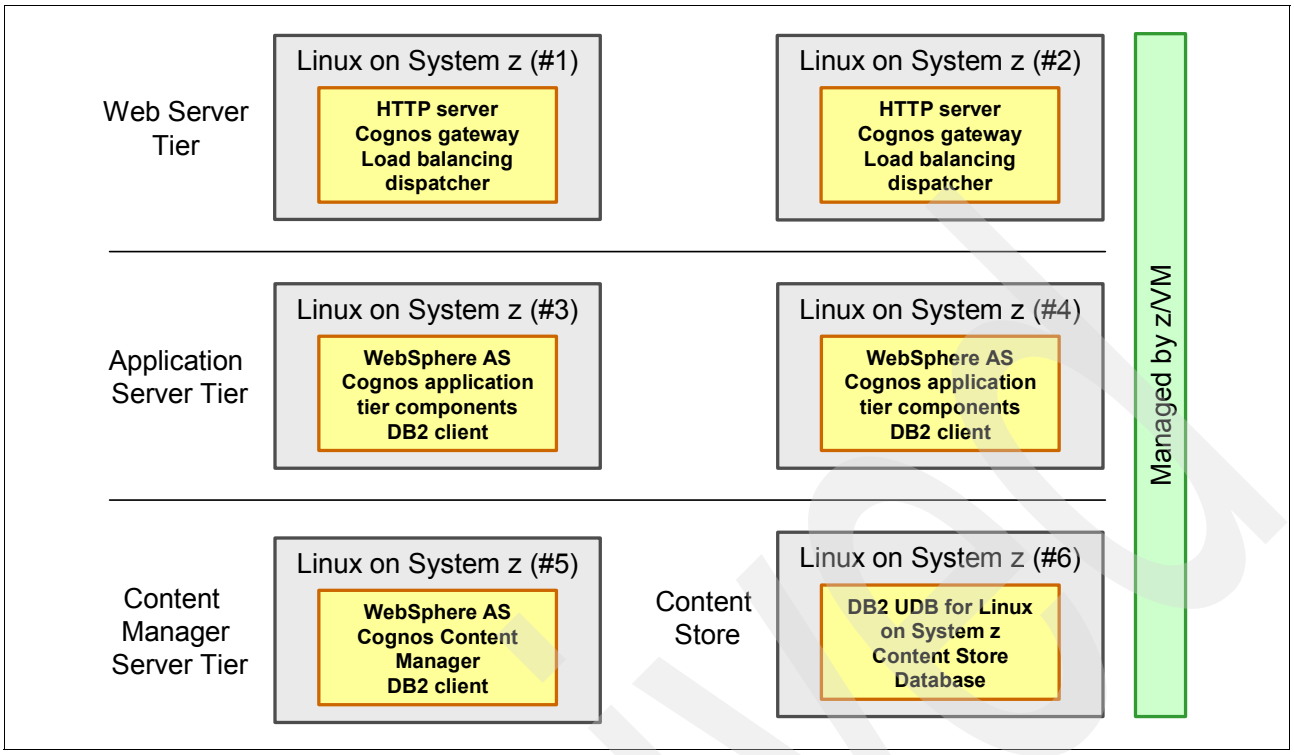


Figure 6-4 Options to distribute IBM Cognos BI components over multiple Linux systems

An advantage of distributing components throughout different tiers on different Linux installations is that it provides the ability to define resource assignments with more granularity. For example, a user could limit the CP and memory capacity assigned to system #6 (which is hosting DB2 with the IBM Cognos BI content store database), if this is desired. It also allows you to manage and tune the server components individually without having to consider the potential impacts to other components on the same Linux system.

For this book, the IBM Cognos BI Server was run in a 31-bit Application Server environment. Therefore, the virtual heap size that could be assigned to an instance of the Java virtual machine was limited to 2 GB (due to other limitations, it was actually slightly lower). The IBM Cognos BI system architecture allows you to scale horizontally and vertically by adding additional application servers as needed. In doing so, the aggregate virtual memory for the entire system can easily be increased over the 2 GB limit. Additionally, we created two JVM application servers in a single z/VM guest, with each one running an IBM Cognos BI Report Server. See Figure 6-5 on page 90.

Note: Note that there can only be one active instance of the content manager at any time. You may, however, have multiple content manager instances installed and configured in order to implement fail-over support.

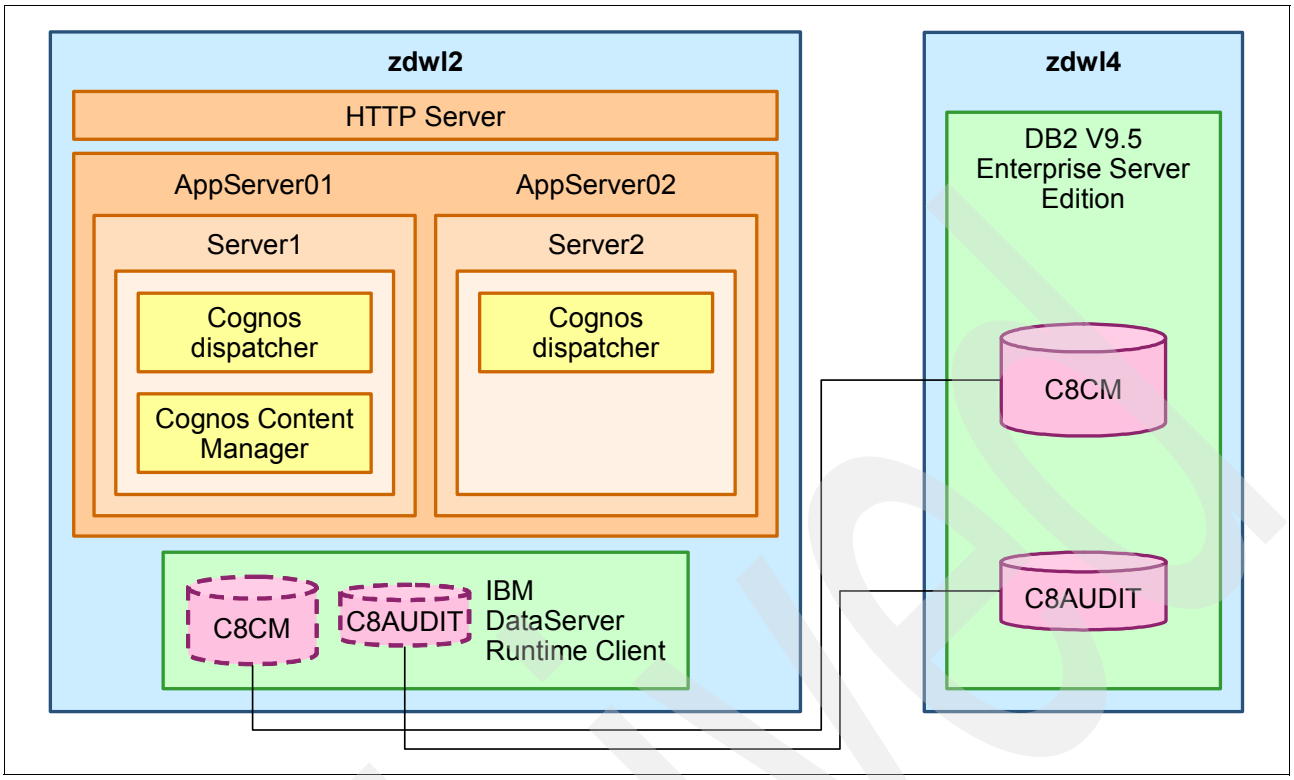


Figure 6-5 IBM Cognos BI installation

This decision was made based on the assumption that Linux is very capable of handling a large number of concurrent (operating system) processes and the requisite memory allocation requirements. Moreover, the total number of CPs and physical memory allocated to the system would remain the same, irrespective of the number of Linux installations that were created.

An advantage of this approach is reduced communication overhead between multiple servers and less resource consumption from duplicate components (such as the operating system itself). Depending on your workload, the installation topology may be different. For instance, if you are creating a number of PowerCubes (the cubing technology shipped with IBM Cognos BI), it would be recommended to run the IBM Cognos Transformer service on a separate Linux guest in z/VM as shown in Figure 6-6 on page 91. This would isolate the resources used for building the cube. Transformer modeling is done with a Windows client tool. Then you build the PowerCube using the Transformer engine on Linux. PowerPlay is a user tool to access a PowerCube, and IBM Cognos BI Analysis Studio is another.

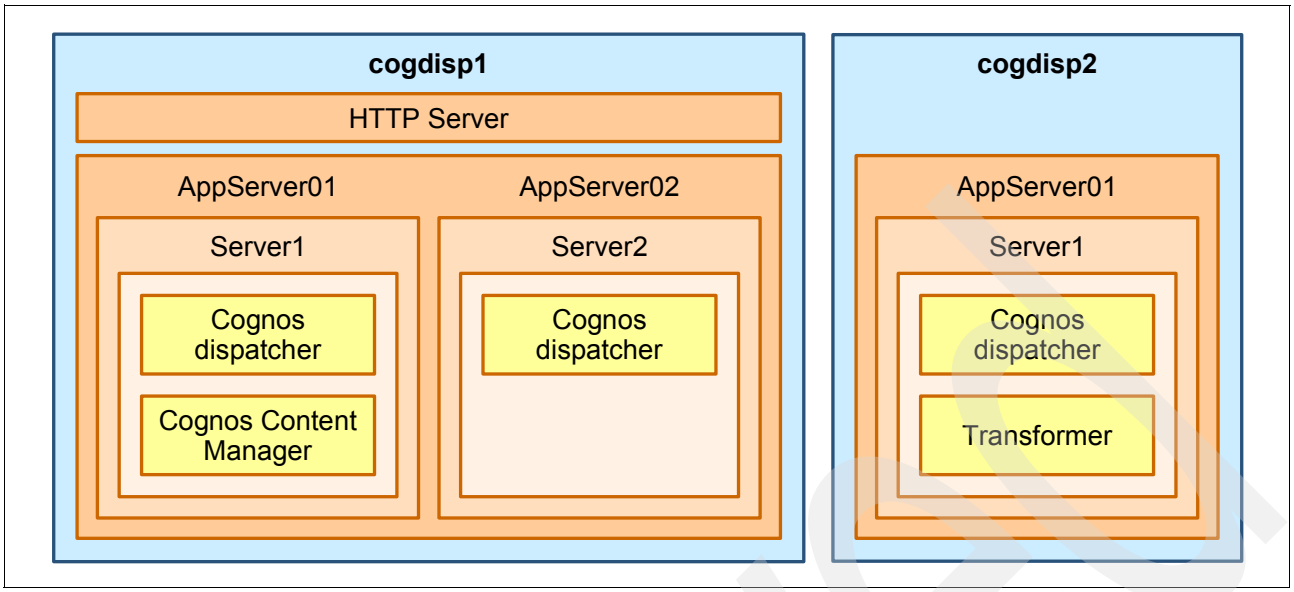


Figure 6-6 IBM Cognos BI set up with two Linux guests

In order to set up a system like the one outlined, *two* WebSphere Application Server profiles are needed, and the IBM Cognos BI Server product code must be installed *twice* on the same machine. This is due to the fact that each of the two instances running in the WebSphere Application Server profiles requires its own configuration and setup.

The WebSphere Application Server code was installed into a single location (`/opt/IBM/WebSphere/ApplicationServer` in this case) and the **`manageprofiles.sh`** command or the scripts provided by IBM Cognos BI (**`create_profile.sh`**) were used to create an additional profile in this installation directory. At this point, two profiles have been created (default location `/opt/IBM/WebSphere/ApplicationServer/profiles`), in this case AppSrv01 and AppSrv02. Note that the profiles need to have different ports assigned. From a topology point of view, these represent two standalone servers in different cells and nodes with no cluster or network deployment setup.

For the IBM Cognos BI components, two instances of the code were installed (that is, the installation procedure was run twice). For one installation (`/opt/IBM/IBM Cognos`), all product components were selected (content manager, application tier components), and the resulting EAR file was deployed to the AppSrv01 profile. For the second installation (`/opt/IBM/IBM Cognos2`), only the application tier components were selected. The resulting EAR file was deployed in the AppSrv02 profile.

The configuration of the first instance is shown in Figure 6-7 on page 92. Note that during these experiments, report and presentation services were disabled. As a result, all of the requests that required these services would run on the second server.

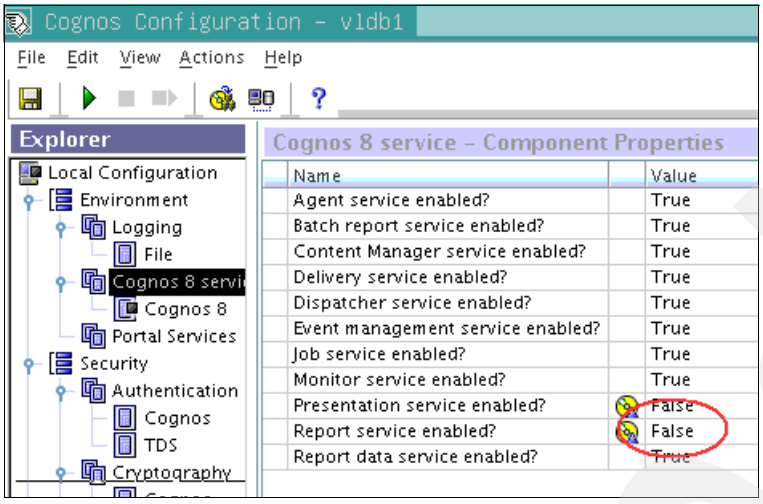


Figure 6-7 IBM Cognos BI configuration for first instance

Figure 6-8 shows the configuration of the second instance. Because the content manager service was not selected during installation, the option is not shown here. All of the available services for this installation are left at their default settings. In the environment settings of this installation, the content manager URI must be pointed to the installation of the first instance.

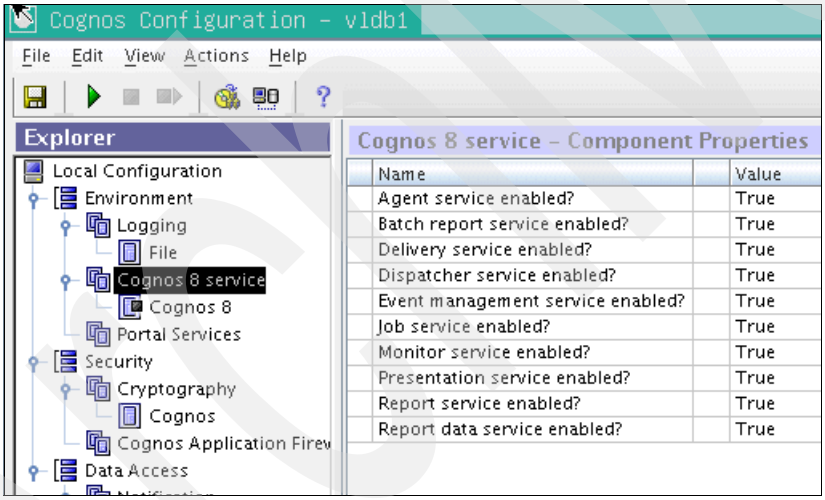


Figure 6-8 IBM Cognos BI configuration for second instance

Once the configuration of the two servers is complete, the two dispatchers appear in IBM Cognos BI Administration.

Note: The topology used here might not be considered as best practice. While it worked well in these tests, a production system may have different requirements where an installation with multiple systems is advisable.

6.2.3 Tuning IBM Cognos BI gateway

IBM Cognos BI offers several options for the gateway implementation in the HTTP Server, including a CGI gateway, a servlet gateway, and also an Apache module.

Because the IBM HTTP Server was used here (derived from Apache HTTP Server 2.0), the *mod2* connector gateway was chosen in order to achieve the best scalability and performance.

Example 6-1 shows the configuration settings for the IBM HTTP Server. This configuration loads the corresponding shared library with the implementation (`mod2_IBM Cognos.so`) and defines the location as `/IBM Cognos8/cgi-bin/IBM Cognos_module`.

Example 6-1 Configuration for mod2 gateway in file httpd.conf

```
LoadModule IBM Cognos_module /opt/IBM/IBM Cognos/c8/cgi-bin/mod2_IBM Cognos.so
...
<Location /IBM Cognos8/cgi-bin/IBM Cognos_module>
SetHandler IBM Cognos-handler
</Location>
```

In IBM Cognos BI Configuration, the corresponding gateway URI is then defined to point to the previously defined location setting.

To cope with the number of concurrent web sessions, `maxclients` was increased to 600. When the maximum is reached, the following message will appear in the HTTP log:

```
server reached maxclients setting, consider raising the maxclients setting
```

This setting controls the number of server processes that are allowed to start. When this condition exists and a new request comes in, the browser waits until one of these slots frees up. In a production environment a log monitoring tool, such as logwatch, could be used to alert you when this situation occurs.

ThreadLimit	25
ServerLimit	64
StartServers	4
MaxClients	600
MinSpareThreads	25
MaxSpareThreads	75
ThreadsPerChild	25
MaxRequestsPerChild	0

Figure 6-9 MaxClients setting in httpd.conf

6.2.4 Tuning IBM Cognos BI report service

Whenever a user requests a new report to be created, this request is ultimately processed by an operating system process that runs the report service. Therefore, one of the most important configuration options for scalability with multiple concurrent users is the number of processes that may be spawned for the report service.

The processes run multithreaded; that is, they can work on multiple (for example, four) reports at the same time. For this testing, the number of processes used for the report service was varied between 10 and 15. The ideal number depends upon available resources (such as number of CPs and memory) and also on the types of requests (short or long running reports, simple or complex reports).

In order to configure the number of report service processes (both for peak and off-peak hours), use IBM Cognos BI Administration. Select the server and the service (`report`

service). See Figure 6-10 for an example where 14 report service processes were configured.

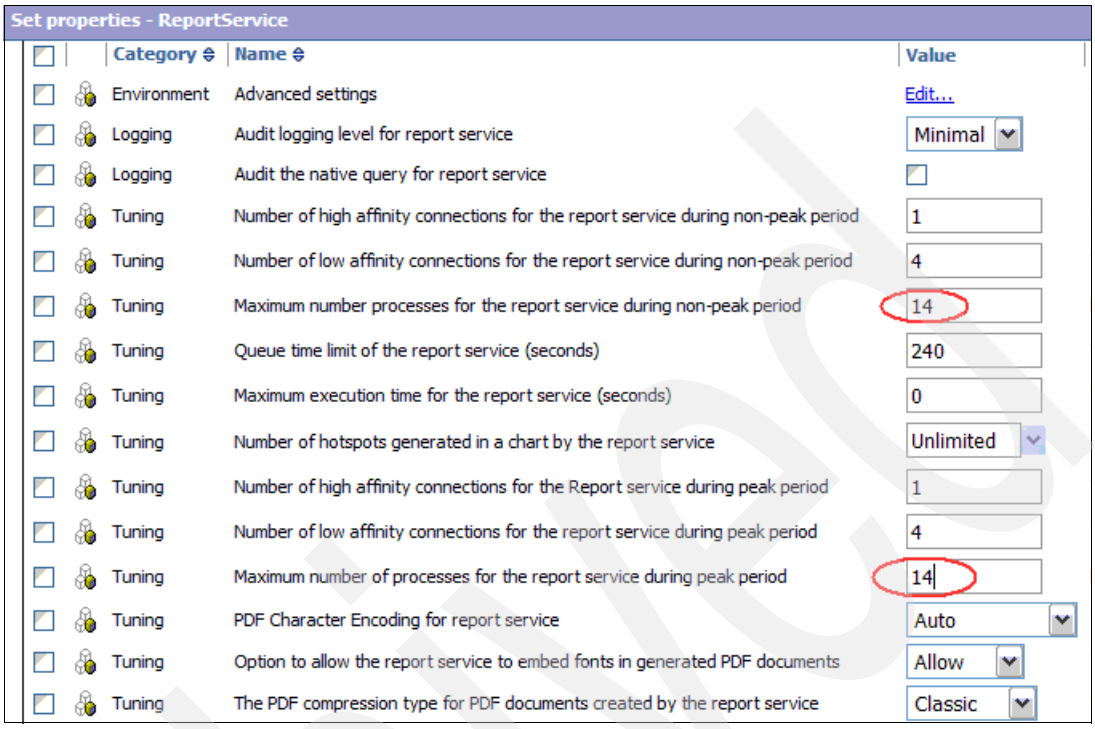


Figure 6-10 Configuration of report service processes in IBM Cognos BI Administration

6.2.5 Tuning WebSphere Application Server

For the two WebSphere Application Server profiles, the following configuration settings were changed

- JVM heap size
- Web container thread pool size

The JVM heap size determines how much memory is made available for Java classes in the virtual machine. The default setting is 256 MB. While IBM Cognos BI Server can run with this default setting, frequent Java garbage collection can be observed during periods of high load. This in turn impacts performance.

For this environment, JVM heap size was set to 768 MB for both application server profiles. To change this setting, use the WebSphere Application Server administration console and select **Application Server** → **server 1** → **Process Definition** → **Java Virtual Machine** (see Figure 6-11 on page 95).

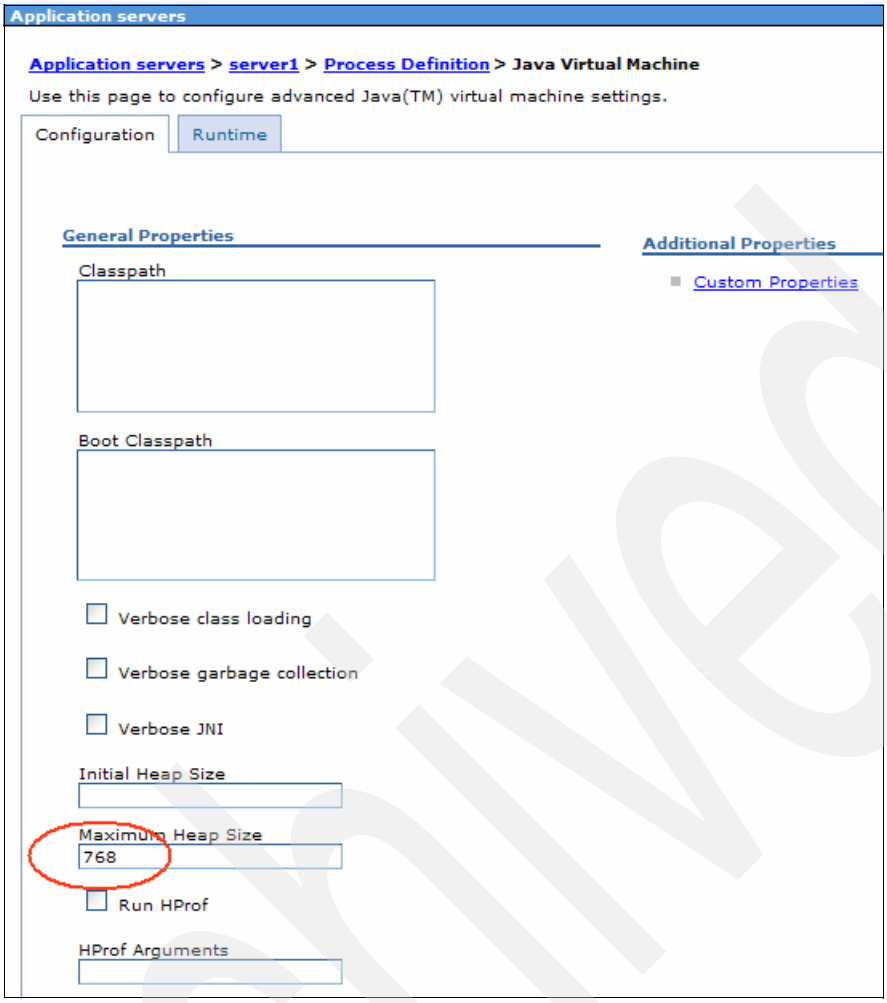


Figure 6-11 Setting JVM heap size for the Application Server

The web container in the WebSphere Application Server uses a dedicated thread pool for web sessions. Through a setting in the administration console, the size of this thread pool can be adjusted so that the web container can reuse threads instead of creating new threads at runtime. Creating new threads is typically a time and resource intensive operation.

For these tests, the number of threads was set to 512 with the option to allocate more threads if required. To set this value, go to **Application Servers** → **servers1** → **Thread Pool**.

The maximum size was changed to 512 (see Figure 6-12). The thread pool has an “Inactivity Timeout” setting that controls when an idle thread should be removed. The default setting for Inactivity Timeout is 3.5 seconds, which is probably applicable to a fairly busy system. On a less busy system this number should probably be increased.

Archived

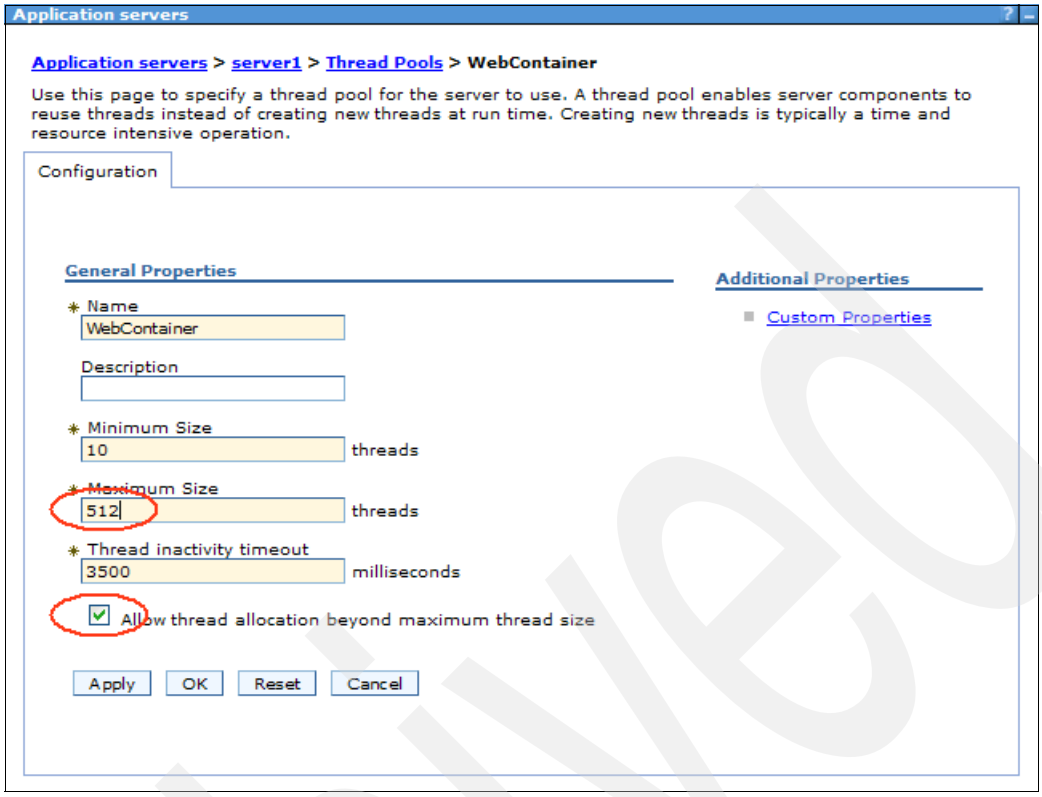


Figure 6-12 Setting for the maximum size of the thread pool in WebSphere Application Server

Improperly setting the initial Java heap size or the maximum Java heap size in relation to the maximum number of server threads can result in inefficient garbage collection and heap compaction. Refer to the appropriate HTTP and Application Server documentation for more information on JVM tuning.

The IBM Tivoli Performance Viewer can be used to monitor the JVM Memory Usage and Web Container pool size. The metric called JVM RU: UsedMemory shows the amount of memory in the JVM heap that is currently used. This value will grow until it hits the JVM Maximum Heap Size setting, at which point a garbage collection will occur and the value will drop down again. Over time, this graph will resemble a sawtooth pattern; see Figure 6-13 on page 97.

Tuning garbage collection in a JVM is a complex matter but the basic goal is to minimize the effect of garbage collection on the application. Since the entire JVM pauses for a short time when a major garbage collection cycle occurs, we want to optimize the frequency that garbage collection cycles occur while minimizing the duration of each. The maximum heap size in the JVM will affect the duration of each garbage collection, so we have to resist making this number too large.

The frequency of garbage collection is affected by the size of objects and the frequency of their creation. One goal is to keep the processor time spent in garbage collection less than 10% of total processor time. It is best to keep it around 3%. These processor times can be found in the verbose garbage collection log for the application server, and there are monitoring tools available to chart this data.

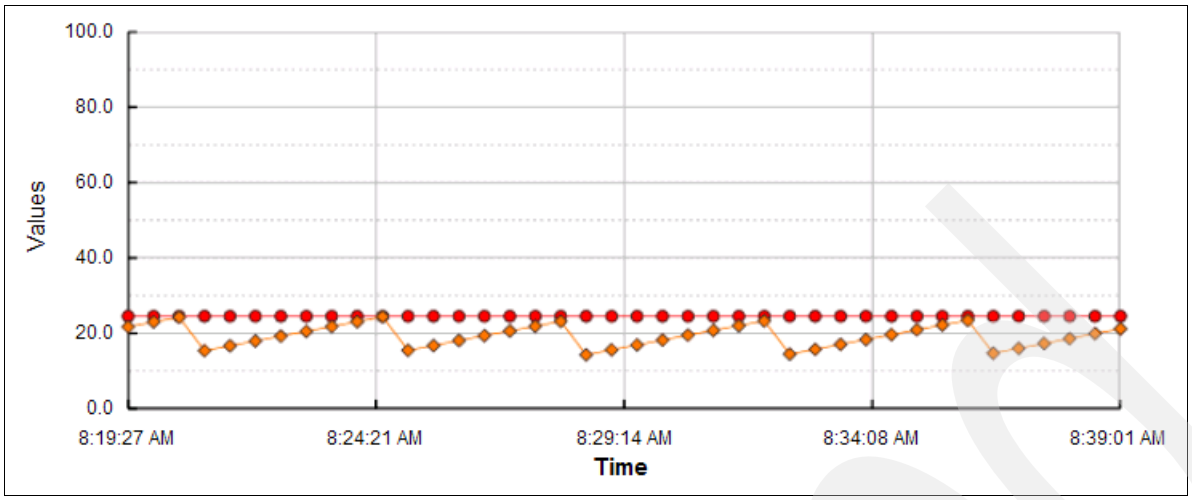


Figure 6-13 Garbage collection graph

The metric called WebCon: PoolSize shows the number of threads in the Web Container thread pool currently in use; see Figure 6-14.

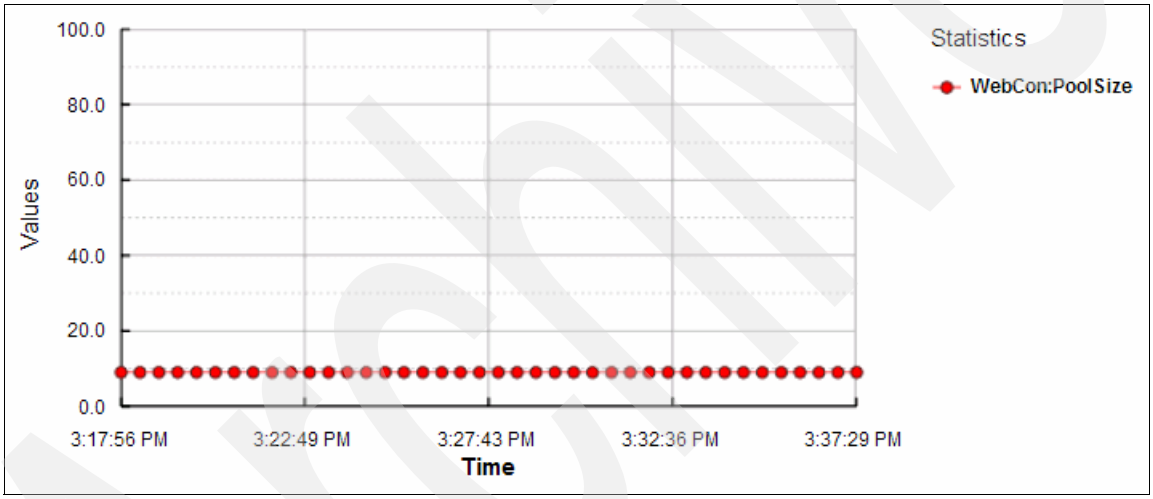


Figure 6-14 Web Container threads currently in use

6.2.6 Tuning DB2 for Linux on System z

The log `db2diag.log` was monitored during the initial test runs to determine whether the self-running memory option was working as expected and to monitor for deadlocks or other situations that need attention.

As shown in Example 6-2, most of the settings were left at `AUTOMATIC`. The utility heap size was reduced to 8000 pages because utilities were not expected to be used extensively in this setup. The content store database (including table spaces and buffer pools) was created using the values in the scripts provided in the IBM Cognos BI C8SE installation directory.

Example 6-2 DB2 database configuration for content store database

```
Self tuning memory                    (SELF_TUNING_MEM) = ON
 Size of database shared memory (4KB)  (DATABASE_MEMORY) = AUTOMATIC
 Database memory threshold               (DB_MEM_THRESH) = 10
```

```
Max storage for lock list (4KB)              (LOCKLIST) = AUTOMATIC
Percent. of lock lists per application       (MAXLOCKS) = AUTOMATIC
Package cache size (4KB)                   (PCKCACHESZ) = AUTOMATIC
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = AUTOMATIC
Sort list heap (4KB)                         (SORTHEAP) = AUTOMATIC

Database heap (4KB)                           (DBHEAP) = AUTOMATIC
Catalog cache size (4KB)               (CATALOGCACHE_SZ) = 260
Log buffer size (4KB)                         (LOGBUFSZ) = 98
Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 8000
Buffer pool size (pages)                      (BUFFPAGE) = 1000
SQL statement heap (4KB)                      (STMTHEAP) = AUTOMATIC
Default application heap (4KB)              (APPLHEAPSZ) = 2048
Application Memory Size (4KB)                (APPL_MEMORY) = AUTOMATIC
Statistics heap size (4KB)                  (STAT_HEAP_SZ) = AUTOMATIC
```

6.2.7 General Linux OS tuning

The amount of memory available to the Linux guest should be sufficient so that little or no swapping occurs. The Linux **vmstat** command can be used to show memory usage. **vmstat 5 5** says to display statistics 5 times and delay 5 seconds between each display; see Figure 6-15. The fields of interest are so (pages swapped out per second) and si (pages swapped in per second). These fields should be single digits or zero. The Linux **top** command could be used to show the processes using the most memory.

- **The virtual machine size in the z/VM directory entry for the Linux guest controls the amount of memory available to the guest.**

```
vmstat 5 5
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 863100  65504 1906132    0    0     7    18  110   36  1  0 98  0  0
 4  0      0 862860  65504 1906132    0    0     0    14  211  337  1  0 99  0  0
 0  0      0 862860  65504 1906132    0    0     0    47  226  345  1  0 99  0  0
 0  0      0 862868  65504 1906132    0    0     0    71  227  326  1  0 99  0  0
 0  0      0 862868  65504 1906132    0    0     0    44  210  305  0  0 99  0  0
```

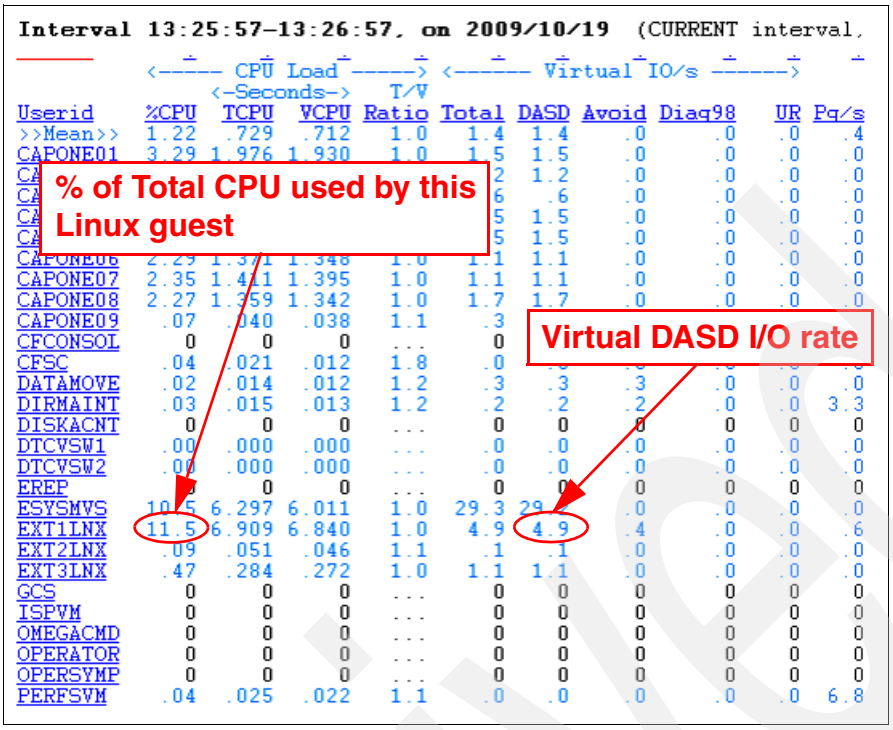
- **The Linux "top" command could be used to show the processes using the most memory.**

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2443 root      24   0 1241m 463m 7172 S    0  11.6  26:20.30 java
 3013 cognos    16   0  661m 364m  73m S    0   9.1  14:52.61 java
 2336 db2inst2  18   0 1035m 162m 139m S    0   4.0   4:51.82 db2sysc
 4065 root      16   0  687m  99m 8500 S    0   2.5   0:53.76 java
```

Figure 6-15 Linux memory commands

6.2.8 z/VM tuning

While outside the scope of this book, the z/VM performance toolkit can be used to monitor and report performance data for z/VM (see Figure 6-16 on page 99), as well as Linux for System z performance data; see Figure 6-17 on page 99.



Interval 13:25:57-13:26:57, on 2009/10/19 (CURRENT interval,

Userid	%CPU	TCPU	VCPU	Ratio	Total	DASD	Avoid	Diag98	UR	Pg/s
>>Mean>>	1.22	.729	.712	1.0	1.4	1.4	.0	.0	.0	.4
CAPONE01	3.29	1.976	1.930	1.0	1.5	1.5	.0	.0	.0	.0
CA					1.2	1.2	.0	.0	.0	.0
CA					.6	.6	.0	.0	.0	.0
CA					1.5	1.5	.0	.0	.0	.0
CA					1.5	1.5	.0	.0	.0	.0
CAPONE06	2.29	1.371	1.348	1.0	1.1	1.1	.0	.0	.0	.0
CAPONE07	2.35	1.411	1.395	1.0	1.1	1.1	.0	.0	.0	.0
CAPONE08	2.27	1.359	1.342	1.0	1.7	1.7	.0	.0	.0	.0
CAPONE09	.07	.040	.038	1.1	.3					
CFCONSOL	0	0	0	...	0					
CFSC	.04	.021	.012	1.8	.0					
DATAMOVE	.02	.014	.012	1.2	.3	.3	.3	.0	.0	.0
DIRMAINT	.03	.015	.013	1.2	.2	.2	.2	.0	.0	3.3
DISKACNT	0	0	0	...	0	0	0	0	0	0
DTCVSW1	.00	.000	.000	...	.0	.0	.0	.0	.0	.0
DTCVSW2	.00	.000	.000	...	.0	.0	.0	.0	.0	.0
EREP	0	0	0	...	0	0	0	0	0	0
ESYSMVS	10.5	6.297	6.011	1.0	29.3	29.	.0	.0	.0	.0
EXT1LNX	11.5	6.909	6.840	1.0	4.9	4.9	.4	.0	.0	.6
EXT2LNX	.09	.051	.046	1.1	.1	.1	.0	.0	.0	.0
EXT3LNX	.47	.284	.272	1.0	1.1	1.1	.0	.0	.0	.0
GCS	0	0	0	...	0	0	0	0	0	0
ISPVM	0	0	0	...	0	0	0	0	0	0
OMEGACMD	0	0	0	...	0	0	0	0	0	0
OPERATOR	0	0	0	...	0	0	0	0	0	0
OPERSYMP	0	0	0	...	0	0	0	0	0	0
PERFSVM	.04	.025	.022	1.1	.0	.0	.0	.0	.0	6.8

Figure 6-16 z/VM performance toolkit z/VM view

Memory Statistics

Interval 14:23:48-14:24:56, on 2009/10/19 (CURRENT interval, select interim or average data)

Linux Userid	M Total	%MUsed	H Total	%HUsed	Shared	/CaFree	Buffers Used	Cache S Total	%SUsed	In	Out	Allocates	Major	Minor	Read	Write
>System<	491.5	51.5	.0	.0	.0	44.6	149.4	253.9	5.1	.000	.000	9.337	.000	.707	.000	.353
EXT2LNX	491.5	51.5	.0	.0	.0	44.6	149.4	253.9	5.1	.000	.000	9.337	.000	.707	.000	.353

CPU Statistics

Interval 14:21:49-14:22:51, on 2009/10/19 (CURRENT interval, select interim or average data)

Linux Userid	Virt CPUs	TotCPU	User	Kernel	Nice	IRQ	SoftIRQ	IOWait	Idle	Stolen	Runabl	Waiting	Current Total	Average Running 1 Min	5 Min	15 Min	Nr of Users
>System<	1.0	.3	.1	.1	.0	.1	.0	.0	99.7	...	1.0	.0	54.0	.00	.00	.00	1
EXT2LNX	1	.3	.1	.1	.0	.1	.0	.0	99.7	...	1	0	54	.00	.00	.00	

Network Statistics

Interval 14:24:56-14:25:55, on 2009/10/19 (CURRENT interval, select inter

Linux Userid	Interfaces	Data Transfer/sec Packets P Recv	P Xmit	Bytes B Recv	B Xmit	Errors/sec BadPac	Packet Recvd	X Mit	No Space Buffer	Transmit Linux	Collisn
>System<	3.0	.205	.000	20	0	.000	.000	.000	.000	.000	
EXT2LNX	3.0	.205	.000	20	0	.000	.000	.000	.000	.000	

Figure 6-17 z/VM performance toolkit Linux view

6.2.9 IBM Cognos BI and DB2 client information

With IBM Cognos BI Version 8.1, there are connection command blocks to pass client information (also referred to as “client attributes” or “client strings”), with DB2 work requests (queries). This provides a means for more granular qualification of work, which is valuable for purposes such as resource management, monitoring, reporting, or problem determination. For details on how we configured IBM Cognos BI to exploit DB2 client information, see 13.3,

"WLM_SET_CLIENT_INFO in IBM Cognos BI" on page 326 and 13.4, "Leveraging WLM set client information in IBM Cognos BI" on page 329.

6.3 Reporting with IBM Cognos BI

We now describe the processes of building IBM Cognos BI data models, designing reports with Report Studio, tuning reports using IBM Optim Query Tuner, and administering operational BI reporting.

6.3.1 Building IBM Cognos BI data models

In this section, we provide an overview of the IBM Cognos BI client modeling tools used to design and develop our business intelligence reporting workload. For more details, interested readers should consult *IBM Cognos BI Installation and Configuration Guide* and *IBM Cognos BI Framework Manager User Guide*.

IBM Cognos BI Framework Manager (Figure 6-18) is a client tool installed on a Windows machine to enable data modeling and package publishing for different business user groups. Framework Manager is a metadata modeling tool that drives query generation for IBM Cognos BI. A model is a collection of metadata that includes physical information and business information for one or more data sources. With additional security and multilingual enablement, publishing one model can serve the needs of reporting, ad hoc querying, and analysis for multiple groups of users around the globe.

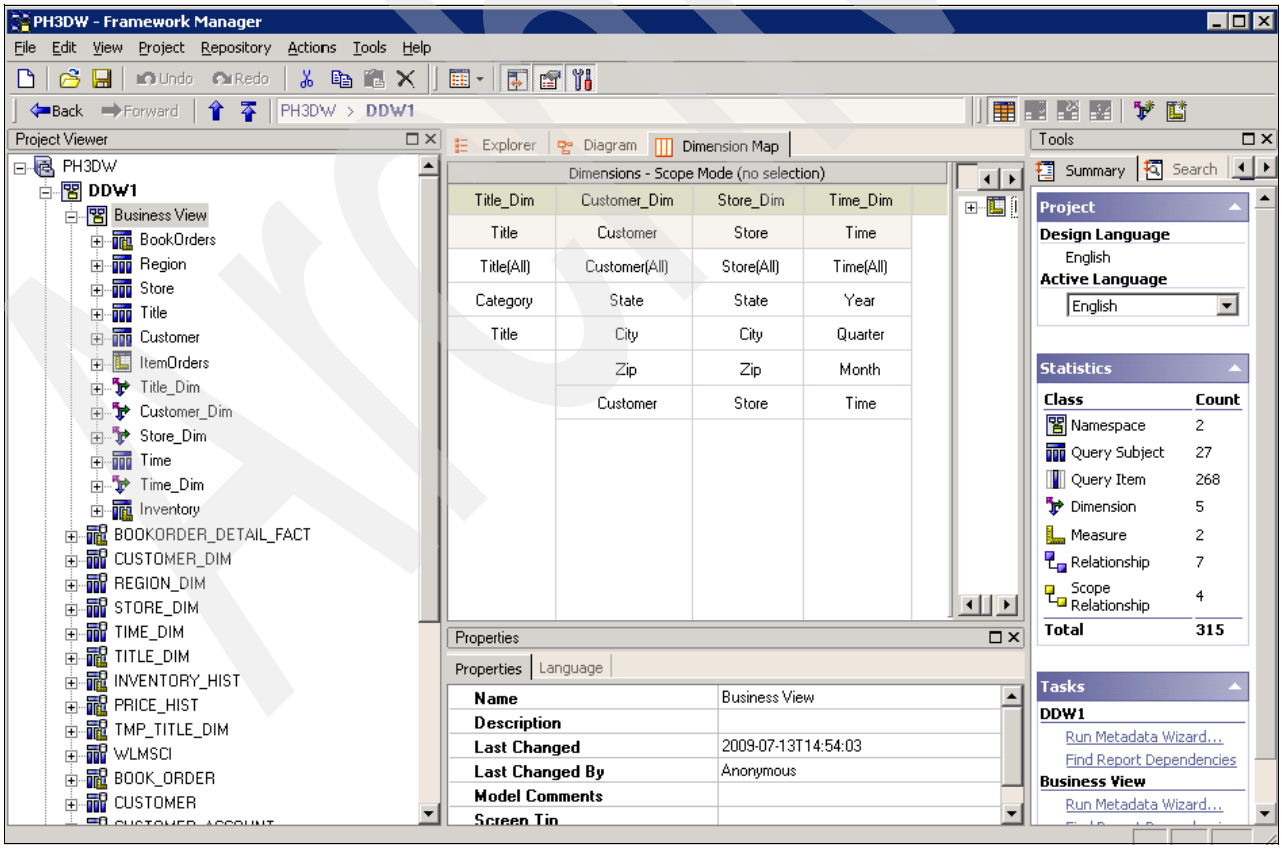


Figure 6-18 Defining query subjects and dimensional views in Framework Manager

Figure 6-18 shows a screen shot of the Framework Manager. The left pane contains the query subjects of interest defined in our scenario: BookOrders, Region, Store, Title, Customer, Time, and Inventory. In the center pane is the dimension map showing grouping hierarchies at different granules. In the right pane is a summary of metadata objects created by the user.

In addition to defining query subjects in business terms, a key part of the Framework Manager modeling is to create relationships between the query subjects. For example, Figure 6-19 illustrates the link definition panel between the tables STORE_DIM and BOOKORDER_DETAIL_FACT. Normally, when a new data source is established, the Metadata Wizard performs an analysis of the physical schema in the database to recognize foreign key relations between tables, such as BOD_STORE_ID referencing S_STORE_ID depicted in Figure 6-19. In our data mart design, however, there are no foreign key relationships defined. Such relations would be defined in the Framework Manager model in order for the IBM Cognos BI Server to establish join paths. Note that foreign key relationships may be defined in DB2 for z/OS as “not enforced” to reduce maintenance overhead.

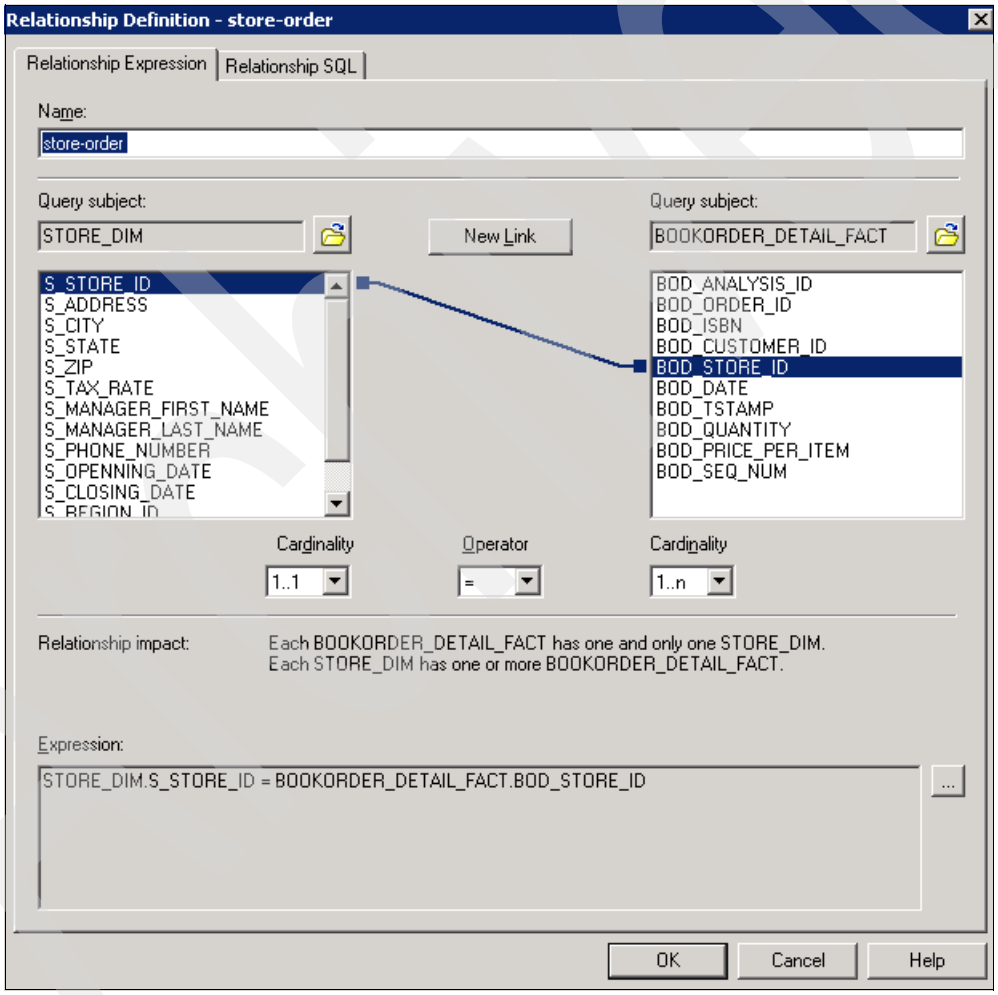


Figure 6-19 Defining relationships between the order fact table and the store dimension table

Similarly, Figure 6-20 on page 102 depicts a panel to establish relationships between the TITLE_DIM and BOOKORDER_DETAIL_FACT tables. As described in the verbiage of the link explanation, each BOOKORDER_DETAIL_FACT record references one and only one record in TITLE_DIM, while each TITLE record can have multiple BOOKORDER records.

Once the Framework Manager modeling is complete with the relations and dimensions, a package can be created to make the metadata model available to IBM Cognos BI users. In the *Framework Manager User Guide*, a checklist is provided to walk through the steps from verifying the model being correct to publishing the package to the IBM Cognos BI Server.

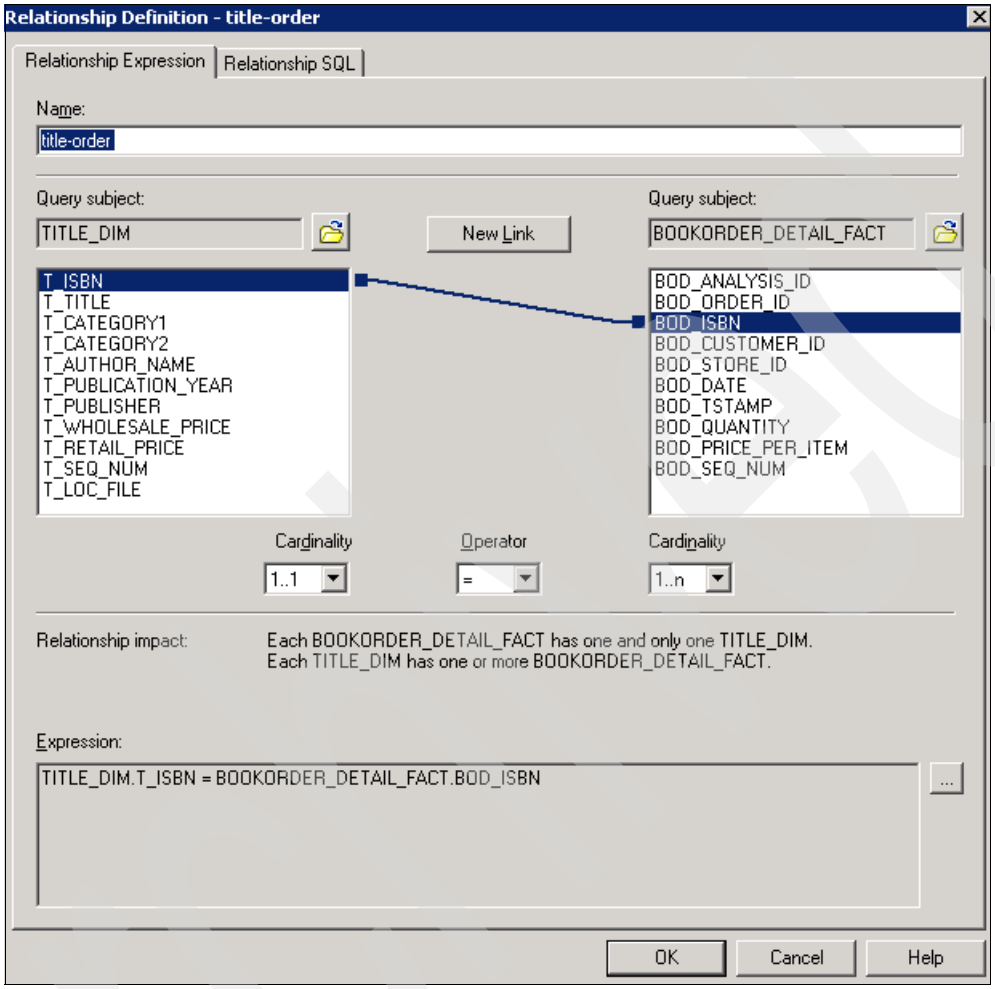


Figure 6-20 Defining relationships between the order fact table and the title dimension table

6.3.2 Designing reports with Report Studio

Report Studio is a web-based tool used to create reports. In our scenario, we designed reports that generate DB2 queries with various CP consumption to generate a workload on z/OS.

Consult the *Report Studio Express Authoring User Guide* and the *Report Studio Professional Authoring User Guide* for details.

Sample reports designed using Report Studio

In order to drive workloads with different CP consumption characteristics, we developed over a dozen reports with prompt values that can be used to control the data volume and processing complexity of the generated query.

For example, Figure 6-21 on page 103 depicts a report in pie chart showing total book sales organized by the city and zip code of the stores. The report query consists of a join of the book orders, store, and time tables with predicate on data ranges. The query aggregates the

copies of books placed in orders by the zip codes of stores. Users viewing the report can quickly get a glimpse of the number of books sold where. One can also move the mouse cursor over a pie slice to get the sales volume.

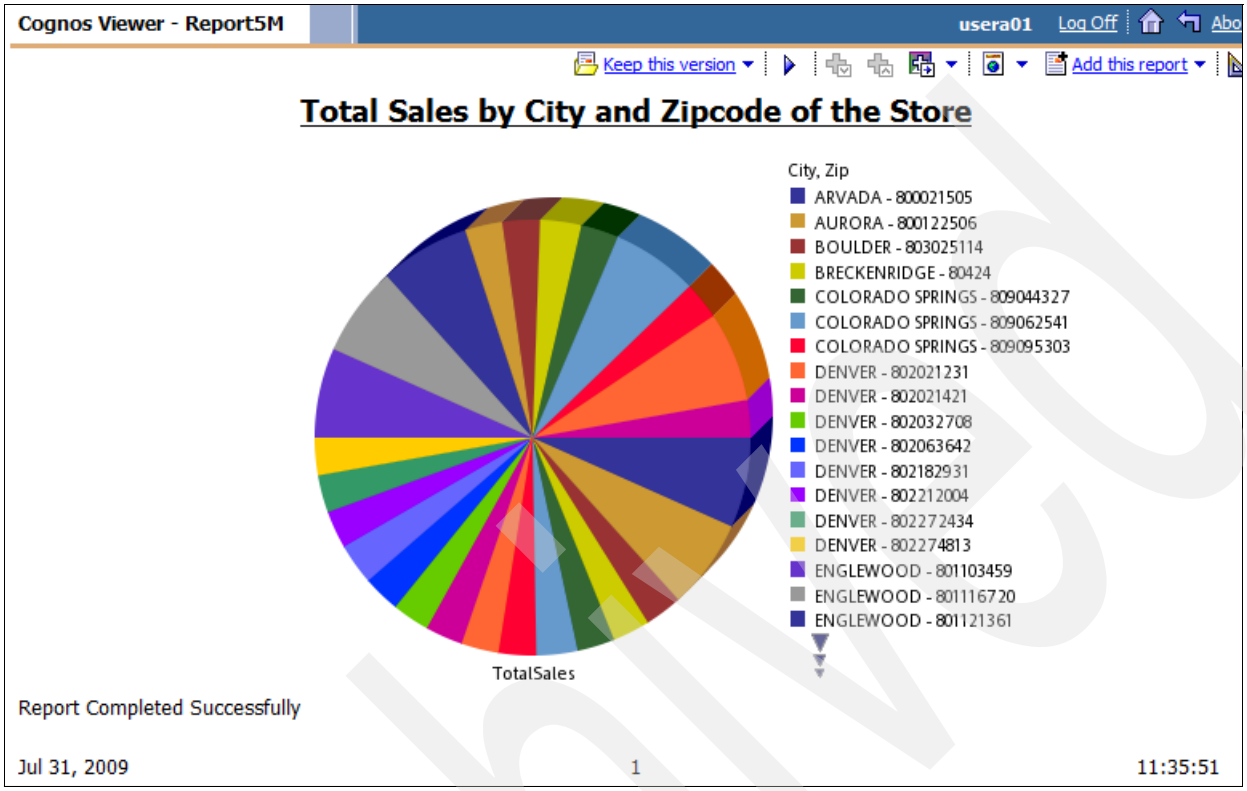


Figure 6-21 A sample report showing book sales by city and zip code

Another example of a report is to illustrate the sales trend over a period of time in a certain book category or title. For example, Figure 6-22 on page 104 shows daily sales numbers in the subject of physics in the month of January 2008. The report query consists of a join of the book orders, title, and time tables with a predicate on data ranges as well as a predicate on the book category. The query aggregates the copies of books placed in orders on a daily basis.

Archive

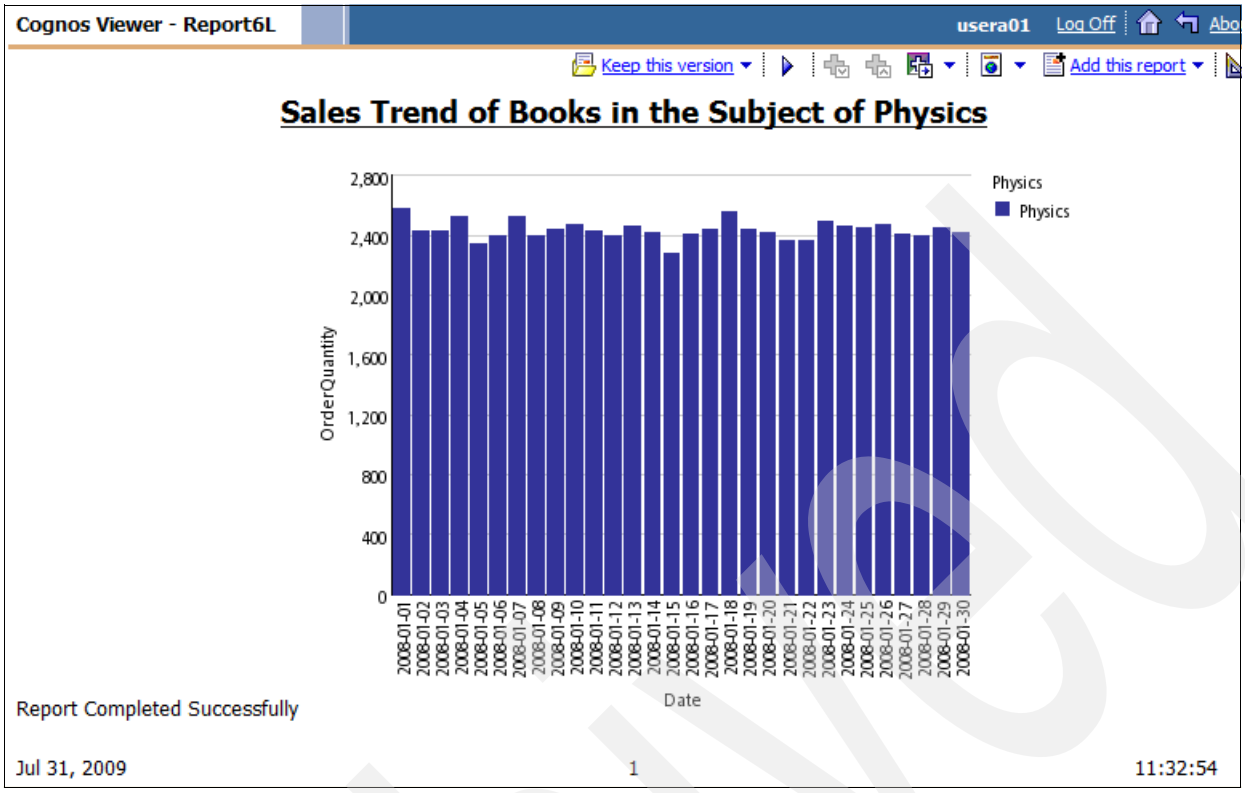


Figure 6-22 A sample report showing the sales trend of physics books over time

In addition to historical data analysis and reporting on accessing data in the data mart, we also authored reports to simulate operational BI activity by accessing OLTP transactional tables. For example, Figure 6-23 on page 105 shows a two-part report on the book title “Plane and spherical trigonometry”. The two-part report consists of a query to check the inventory of the book and another query to report the historical sales figures. In the workload design, such report is meant to be light in resource consumption and is prioritized high to gain speedy response time.

Archive

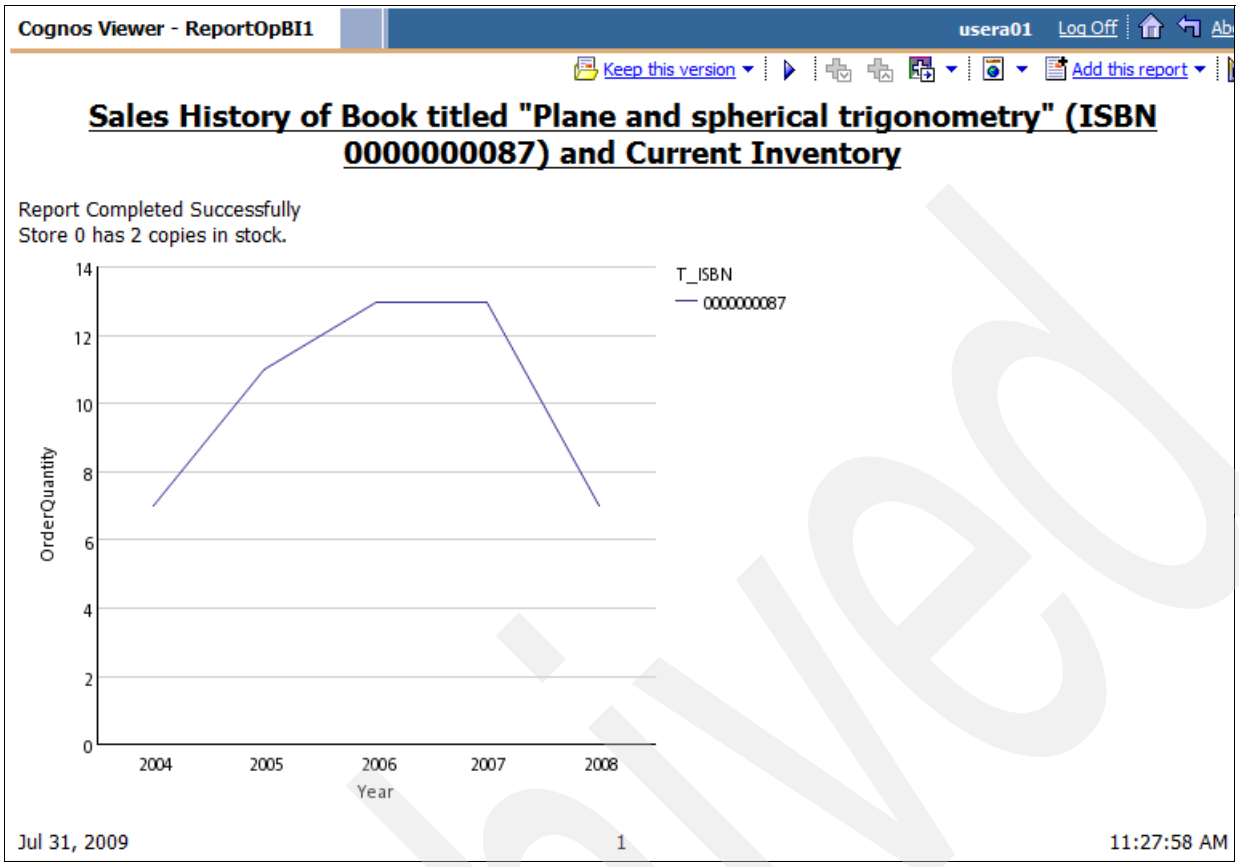


Figure 6-23 A sample report showing both sales trend and current inventory of a book in a store

Categorizing report workload by CP seconds

z/OS Workload Management (WLM) enables fine-grained control over service priorities and resource consumption on a per client, per report basis. To better demonstrate such capability, we designed a portfolio of reports and categorized them based on the use of CP seconds.

Table 6-2 lists the designation of report categories and their corresponding CP utilization measured in seconds. In order to meet the categorization and ensure a balanced mix of reports from each category, we had to adjust several aspects of parametric input in order to control the CP seconds a report query may consume.

Techniques we employed to control CP seconds include:

- Varying date and time ranges to include more data
- Using skews in data distributions in store and holiday season to differentiate data volume
- Introducing computations, such as gross revenue and profit, to increase processor consumption
- Using precomputed aggregates when a report query consumes excessive resources

Table 6-2 Report categories and their corresponding CP seconds

	z10 CP seconds
Trivial	0.01-0.5
Small	0.5-10
Medium	10-50

Archived

	z10 CP seconds
Large	50-500
XL	>500

While you may not be able to precategorize specific BI reports to processor resource consumption as granularly as we needed to for simulation of workloads in our measurement scenarios, the categories and accompanying WLM policies may be a useful reference for automating your own service management.

6.3.3 Tuning reports using IBM Optim Query Tuner

IBM Cognos BI will generate one or multiple queries to handle a given report object. IBM Cognos BI will delegate as much of the query to the database as possible. As applicable, the query engine will perform additional query processing on the results provided by the database. IBM Cognos BI reports run with a number of properties that can affect query performance. Consult the IBM Cognos BI product documentation for more information about these settings.

One significant piece of tuning that can drastically affect query performance is SQL tuning. To ensure that IBM Cognos BI continues to perform well, it is important to maintain the performance of your database. This means optimizing it for querying, reporting, and analyzing. If IBM Cognos BI and other applications demand more of a database than it can provide, or queries are too large for a database to handle efficiently, you may experience increased response times and degradation in IBM Cognos BI performance and scalability.

Users can view the SQL (generated or Cognos) for any query. The generated SQL is what is being passed to the database, whereas the Cognos SQL is the entire SQL used to perform the query. Users can also use products such as IBM Optim Query Tuner to view a filtered view of all of the IBM Cognos BI queries that have been run. By doing so, they can tune the worst running queries or the most commonly run queries using the various incorporated advisors.

To view the SQL for the entire report, from the Tools menu, set the validate options to “information”, then click **Validate**. This option shows the SQL that will be executed in the data source. The SQL is organized by query and by query result. If a query is used in more than one data container, a query result is generated for each data container.

To view the SQL for a specific query, pause the pointer over the query explorer button and click the query. In the Properties pane, double-click the Generated SQL/MDX property. The SQL for the query appears in the Generated SQL/MDX dialog box. You can choose to view generated SQL or Cognos SQL, which is a generic form of SQL that Report Studio uses. Cognos SQL is converted to generated SQL before the query is executed. The generated SQL may be a subset of the Cognos SQL, which could indicate that the application will do some local processing. See Figure 6-24 on page 107. You can tell whether the query requires some local processing by setting the processing option to `database only`. This will show an exception if it needs to perform additional relational operations on the data set returned by the database.

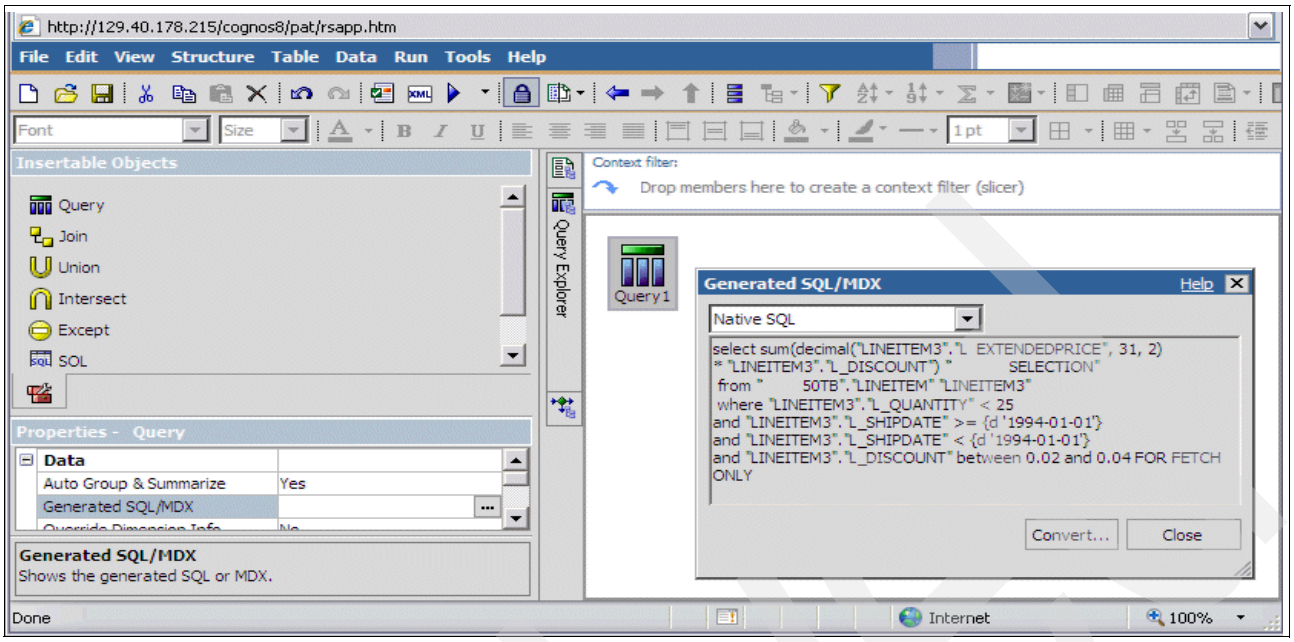


Figure 6-24 Generate SQL

IBM Optim Query Workload Tuner for DB2 for z/OS provides query workload monitoring and analysis tools. It enables you to view, capture and analyze query workloads that run on DB2 for z/OS data servers. When the license for IBM Optim Query Workload Tuner for DB2 for z/OS is activated on a data server, all advanced query tuning features are enabled in the query tuner client, in addition to the following workload tuning activities:

- Creating monitor profiles to capture information about the performance queries and query workloads that run on the data server
- Defining and capturing query workloads
- Getting expert advice for statistics, index design, and query design
- Capturing and sending detailed information about workload environments to different data servers or IBM software support
- Creating detailed reports about tables that are referenced by queries in the workloads

You can define and capture a query workload to investigate the performance of a group of queries that run on a data server; see Figure 6-25 on page 108. A query workload is any group of queries that run on a data server or data sharing group. A workload is created when you monitor queries, and when you capture a snapshot from a monitor, but you can also define and capture workloads separately. The source of the workload can be the dynamic statement cache, the DB2 catalog, QMF, a file, and so forth.

Filters enable you to specify display criteria and thereby refine the level of data detail seen regarding query activity. For example, if a monitoring profile specifies the collection of queries run by a specific ID, a user can define a filter to view only those jobs submitted by a specified user ID or job name. Then the data the user sees via DB2 Query Monitor is limited to only those jobs submitted by the user specified by the filter; see Figure 6-26 on page 108.

Note: Filters do not affect the data that is collected by DB2 Query Monitor; they only filter data for display purposes.

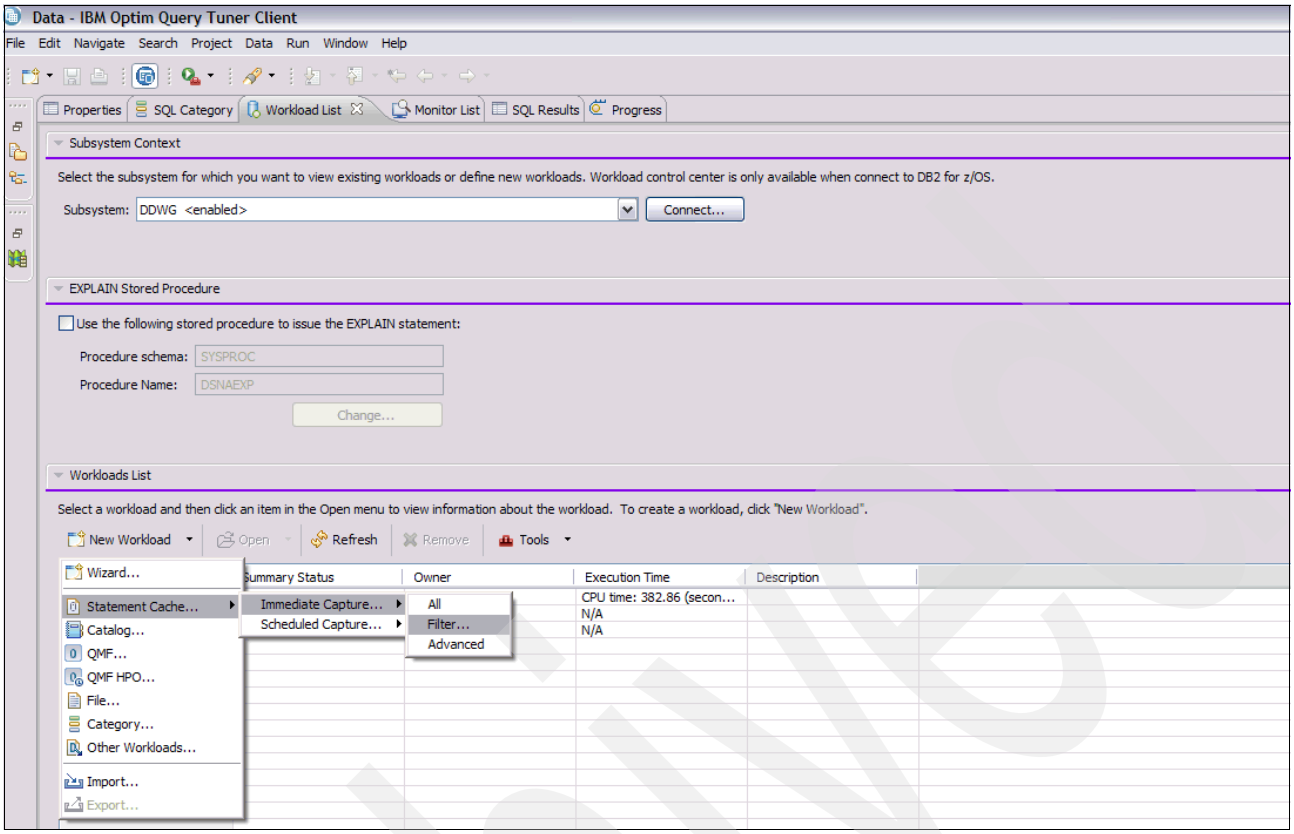


Figure 6-25 Capture and filter statement cache

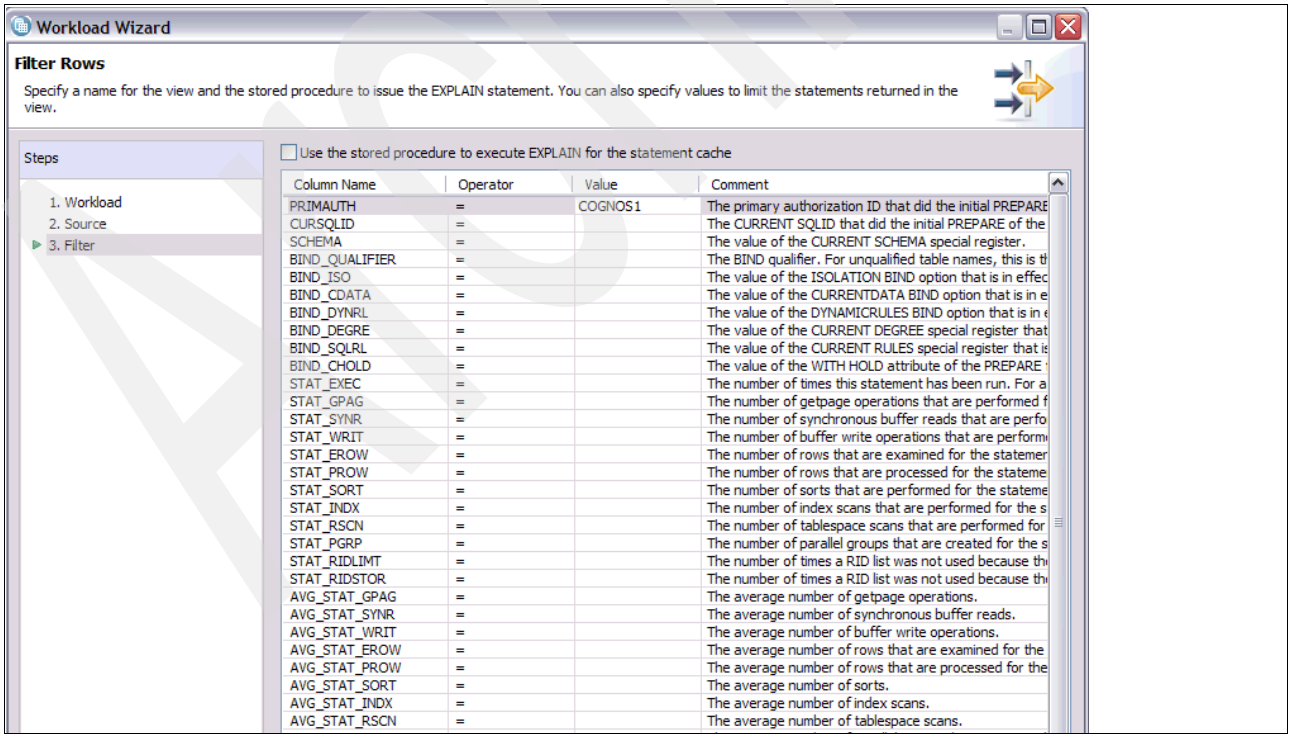


Figure 6-26 Filtering rows by authorization ID

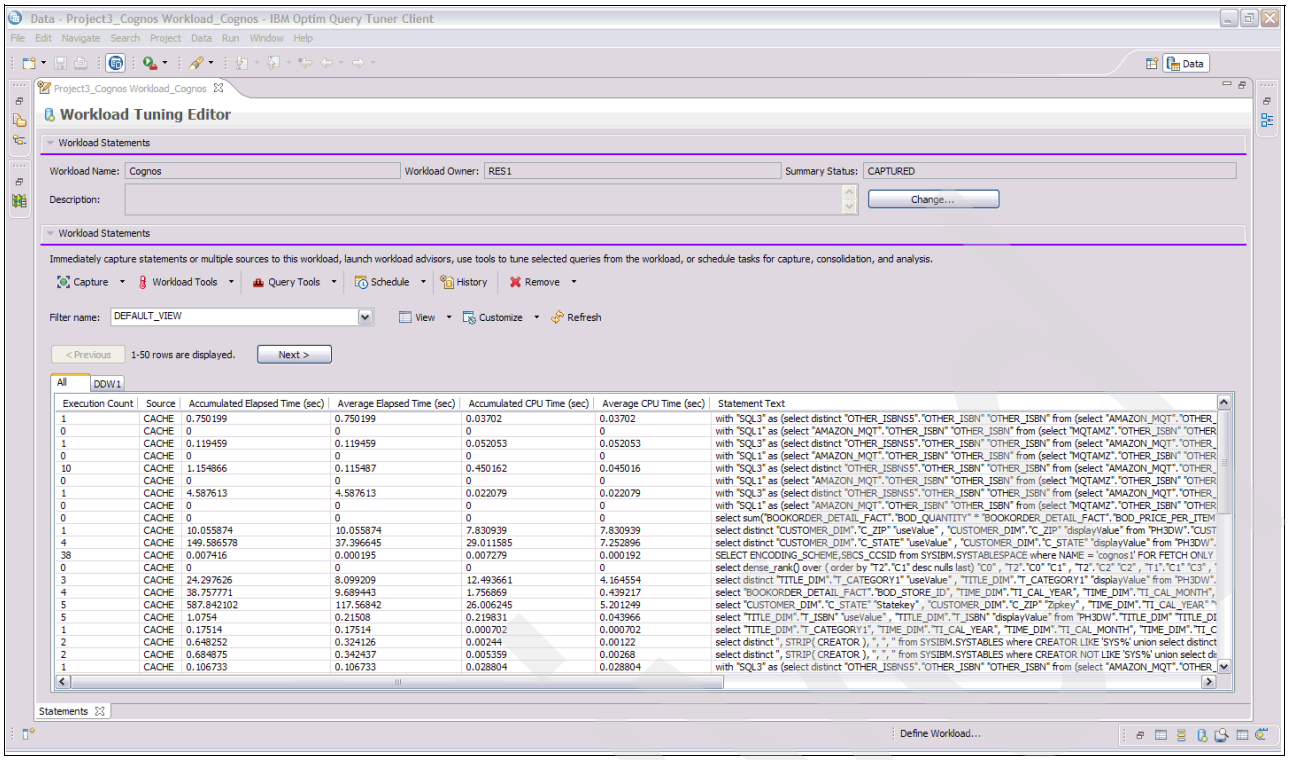


Figure 6-27 Viewing filtered statements

With the Query Workload Tuner, we can perform analysis of the generated workload and individual queries.

The *statistics advisor* considers a set of queries in a query workload and looks for missing, incomplete, obsolete, and conflicting statistics that might lead to suboptimal performance for the queries in the workload. The statistics advisor provides a consolidated set of statistics recommendations that apply to the entire workload, and provides RUNSTATS jobs that you can run to improve the performance of the workload as a whole.

The *workload index advisor* considers a set of queries in a query workload, and looks for ways to improve the performance of the workload, or reduce the amount of disk space used by indexes on the data server. The workload index advisor recommendations include DDL scripts for creating indexes to improve the performance of the workload as a whole, or for dropping indexes to recover disk space, depending on the scenario that you specify.

The *workload query advisor* uses a set of best practice rules to consider a set of queries in a query workload and recommends ways that you might rewrite queries in the workload to improve the performance of the workload as a whole.

Visually explaining an access plan graph enables you to see how the database processes a query in a visual diagram of the access plan. The diagram depicts the processing of the query as a series of nodes arranged in a flow chart that represents the relationship between the different operations. Each node in the diagram is annotated with statistics that can help you understand the performance of a query. See Figure 6-28 on page 110.

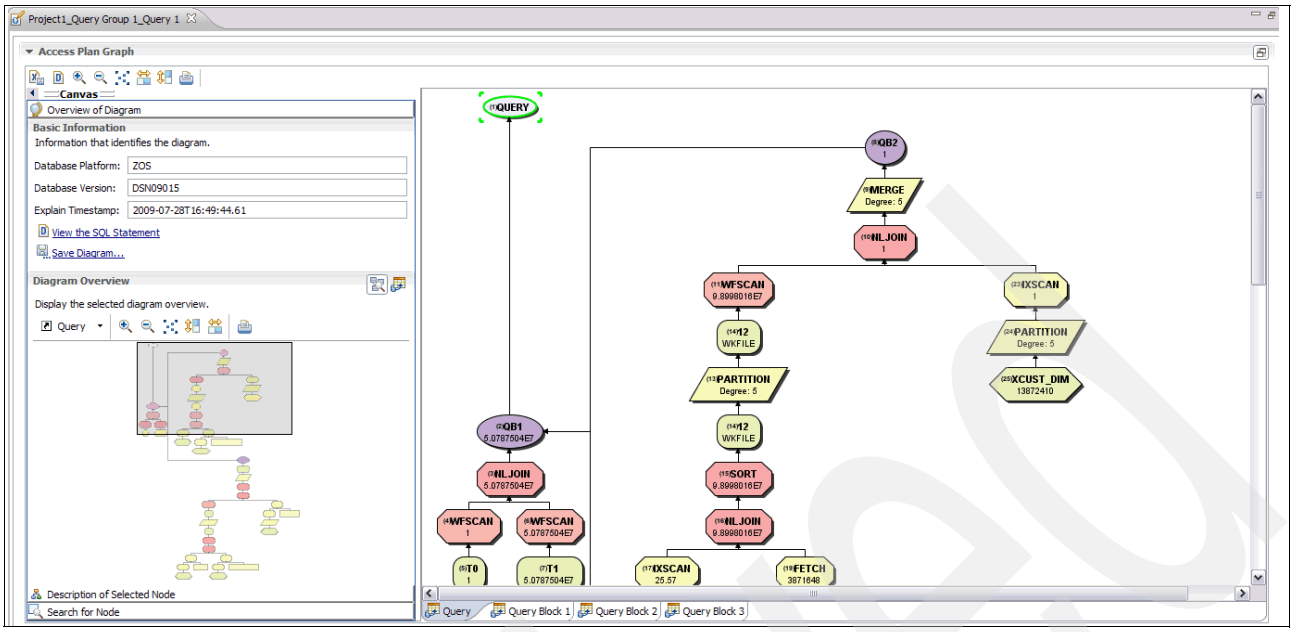


Figure 6-28 Visual explanation of access path

6.3.4 Operational BI reporting

The Operational BI report was intended to show how transactional input could be used to make timely business decisions. In this example, the ability to correlate real-time order information with materialized bookstore cross sell information enables you to make more informed sales decisions. A large Materialized Query Table was used to correlate the top selling books purchased in the same order as an input book. This MQT data was joined with operational inputs to provide you with bookstore cross-sell opportunities.

In the BookStore, for each individual book purchased between August and October of 2008, this MQT will generate the books and their associated counts purchased along with the aforementioned book. For instance, *Catcher in the Rye* was purchased 60 times during a purchase of *Moby Dick*. It does a self join to get all of the THIS_ISBN OTHER_ISBN combinations along with the counts. During testing, based on the volume of data in our fact table, it was determined that the MQT should be built over one week of data. The MQT with one week of data generated about 180 million rows. In 2005 approximately 3 billion books were sold in the U.S. As our BookStore fact table contains 2 billion rows, this BookStore would have sold two-thirds of the books in the U.S. in 2005. Thus, it seemed that 180 million rows was a proper sampling size for testing.

```
CREATE TABLE BOOKSTORE_MQT AS (
  WITH FULL_LIST as (
     SELECT CLIST.BOD_ISBN AS THIS_ISBN, FACT.BOD_ISBN AS OTHER_ISBN, COUNT(*)
TOTCOUNT, ROW_NUMBER OVER(PARTITION BY CLIST.BOD_ISBN ORDER BY COUNT(*) DESC)
ROWNUM
     FROM PH3DW.BOOKORDER_DETAIL_FACT FACT, PH3DW.BOOKORDER_DETAIL_FACT CLIST
     WHERE FACT.BOD_CUSTOMER_ID = CLIST.BOD_CUSTOMER_ID
     AND FACT.BOD_ISBN <> CLIST.BOD_ISBN
     AND FACT.DATE BETWEEN 2008-08-01 AND 2008-10-31
     AND CLIST.DATE BETWEEN 2008-08-01 AND 2008-10-31
     GROUP BY CLIST.BOD_ISBN, FACT.BOD_ISBN )
  SELECT THIS_ISBN, OTHER_ISBN, TOTCOUNT
  FROM FULL_LIST
```

```
   WHERE ROWNUM<=50)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY USER
ENABLE QUERY
```

Note that because there were less than 50 distinct “other” books purchased with any given book in this sample data set[3], the ROW_NUMBER OLAP function and Common Table Expression could be eliminated to:

```
CREATE TABLE BOOKSTORE_MQT AS (
      SELECT CLIST.BOD_ISBN AS THIS_ISBN, FACT.BOD_ISBN AS OTHER_ISBN, COUNT(*)
TOTCOUNT
      FROM PH3DW.BOOKORDER_DETAIL_FACT FACT, PH3DW.BOOKORDER_DETAIL_FACT CLIST
      WHERE FACT.BOD_CUSTOMER_ID = CLIST.BOD_CUSTOMER_ID
      AND FACT.BOD_ISBN <> CLIST.BOD_ISBN
      AND FACT.DATE BETWEEN 2008-08-01 AND 2008-10-31
      AND CLIST.DATE BETWEEN 2008-08-01 AND 2008-10-31
      GROUP BY CLIST.BOD_ISBN, FACT.BOD_ISBN )
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY USER
ENABLE QUERY OPTIMIZATION
IN ...
```

The IBM Cognos BI report was designed to take transactional information (for example, get one ISBN from an online order ID) to determine what other books to suggest to this user along with the book that they are purchasing. When an order was generated, the report would look up one book that was purchased as part of the order. It would then find the top books bought along with this book by other customers. It would do this by querying the MQT table. Note that this could be done with MQT matching, but it is shown querying the MQT for clarity. The individual pieces of the query are broken up and explained.

Query accessing OLTP for both the ISBN look-up and for customer look-up:

```
SELECT OTHER_ISBN FROM BOOKSTORE_MQT WHERE THIS_ISBN =
   (SELECT ISBN FROM ORDER_DETAIL WHERE ORDER_DETAIL.ORDER_ID = ? FETCH FIRST
ONE ROW ONLY)       //Get one ISBN in the order
   AND OTHER_ISBN NOT IN
   (SELECT DISTINCT ISBN FROM BOOK_ORDER, ORDER_DETAIL WHERE
BOOK_ORDER.CUSTOMER_ID=?     //Eliminate books I have already purchased
   AND BOOK_ORDER.ORDER_ID = ORDER_DETAIL.ORDER_ID)
```

IBM Cognos BI SQL

The part in blue is the query that gets one ISBN from the inputted order and joins it to THIS_ISBN in the MQT. The middle part in black selects the OTHER_ISBN from the join. The part in purple generates the list of other books bought by this customer. The part in red eliminates all of the “OTHER_ISBN” items that this customer has already purchased.

Example 6-3 Sample SQL for our IBM Cognos BI test

```
with SQL1 as
    (select
          BKS_MQT.OTHER_ISBN as OTHER_ISBN
     from
```

[3] Windowed aggregates in an MQT typically will not be used by query rewrite.

```
                ( select
                      BOOKSTORE_MQT.OTHER_ISBN as OTHER_ISBN,
                      BOOKSTORE_MQT.THIS_ISBN as THIS_ISBN
                from
                      DDW1..PH3DW.BOOKSTORE_MQT BOOKSTORE_MQT )  BKS_MQT,
                ( select
                      ORDER_DETAIL.ISBN as ISBN
                from
                      DDW1..PH3OLTP.ORDER_DETAIL ORDER_DETAIL
                where ORDER_DETAIL.ORDER_ID = cast(#prompt("Order_Id_Prompt")# as
integer )  filter rank (  ) over ( order by ORDER_DETAIL.ISBN desc nulls last )
<= 1 )  ORDERDETAIL_FNR1
     where ORDERDETAIL_FNR1.ISBN = BKS_MQT.THIS_ISBN),
OTHER_ISBNS5 as
    (select
           SQL1.OTHER_ISBN  as  OTHER_ISBN
     from
           SQL1
    ),
DUPE_ISBNLIST6 as
    (select
           BOOKORDER_DETAIL_FACT.BOD_ISBN  as  BOD_ISBN
     from
           DDW1..PH3DW.BOOKORDER_DETAIL_FACT BOOKORDER_DETAIL_FACT
     where
           (BOOKORDER_DETAIL_FACT.BOD_CUSTOMER_ID =
cast(#prompt("Customer_Id_Prompt")# as integer))
    )
select distinct
       OTHER_ISBNS5.OTHER_ISBN  as  OTHER_ISBN
 from
       OTHER_ISBNS5,
       DUPE_ISBNLIST6
 where
OTHER_ISBNS5.OTHER_ISBN <> DUPE_ISBNLIST6.BOD_ISBN)
```

6.4 Performance and load testing using Rational Performance Tester

This section outlines the high-level concepts in Rational Performance Tester, a description of the internal workings of an IBM Cognos BI report including the asynchronous communication used by IBM Cognos BI reports, and the methodology applied for load testing in this book, including the individual tests, schedules, data pools, and user group alignment defined as part of the testing conducted.

Archived

6.4.1 Rational Performance Tester - introduction

Rational Performance Tester (RPT) is a performance testing tool that provides testers with automated performance testing capabilities to validate the scalability of web and server-based applications.

RPT enables a tester to emulate multiple users to test the performance of an application (IBM Cognos BI) and in turn the underlying data source. For the purpose of this book we wanted to simulate multiple simultaneous users querying the transactional database as well as the data warehouse. Key features of RPT include:

- Code free test creation manageable via a rich tree-based test editor
- Automated test data variation
- Use of custom Java code for flexible test customization
- Collection and integration of server resource data with real-time application performance data

For further information about RPT, refer to RPT Information Center at:

http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.rational.test.lt.doc/icwelcome_product_rpt.htm

RPT scalability testing - high-level concepts

The high-level concepts for scalability testing using RPT include:

- Schedule
- Test
- Data pools
- User groups

Schedule

A schedule can be as simple as one virtual user running one test, or as complicated as hundreds of virtual users in different groups, each running different tests at different times.

A schedule is the “engine” that runs a test. However, schedules are much more than simple vehicles for running tests. For example, you can use a schedule to control tests in the following ways:

- Group tests under user groups, to emulate the actions of different types of users
- Set the order in which tests run: sequentially, randomly, or in a weighted order
- Set the number of times that each test runs
- Run tests at a certain rate
- Run tests for a certain time, and increase or decrease virtual users during the run

When you first create a schedule, it is displayed with one user group, as shown in Figure 6-29 on page 114. You add user groups, tests, and other items to the schedule to emulate a workload.

Archived

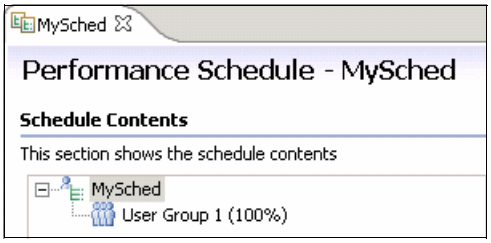


Figure 6-29 RPT - schedule

More user groups can be added to a schedule. Each user group can have different tests assigned and the data pool value selection can be done differently for each user group; see Figure 6-30.

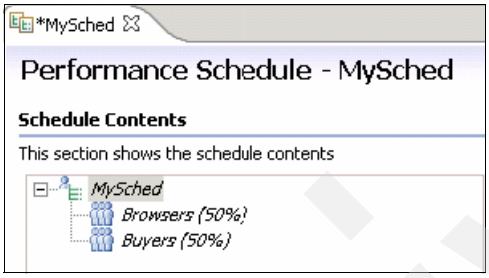


Figure 6-30 RPT Schedule - user groups for different workloads

Test

RPT is capable of performing different types of tests. As part of this project, we used only the HTTP tests. RPT allows for HTTP-based tests to be designed manually. Alternatively, there is an HTTP test recorder that can be used to get an initial version of representative page interactions in an IBM Cognos BI application. This recording can then be customized based on the testing requirements.

A test definition in RPT can contain one or more HTTP pages, and each HTTP page can in turn contain multiple HTTP requests. The tests can also be customized using Java coding. Key components in an RPT test include: -

- Custom code - This allows you to customize the RPT test using Java coding. Java-based RPT APIs can be invoked using custom coding in addition to standard Java APIs. An example of this is to exit a test based on certain conditions.
- Variable assignment - Variables can be defined and utilized at different stages in a RPT test. The value for the variables can be altered using the variable assignment operator.
- Delay - A time delay can be introduced using the Delay operator at any stage in an RPT test.
- Loop - This operator is used to repetitively execute parts of the test. Any of the other operators can be used inside of a loop. For example, as part of this project we had a need to execute multiple HTTP WAIT requests to obtain the status of an IBM Cognos BI report execution in accordance with the IBM Cognos BI ASYNC communication methodology.
- Condition (IF) - This operator is used to alter the test execution path based on values returned by Java custom code or variables.
- Random selector - This operator lets you define one or more weighted blocks and randomly alter the test execution path.
- Comment - This operator can be used to comment the RPT tests.

Archived

Figure 6-31 shows an RPT test demonstrating usage of HTTP Request, Java custom code, RPT test comments, RPT test transaction, Condition IF and RPT test LOOP.

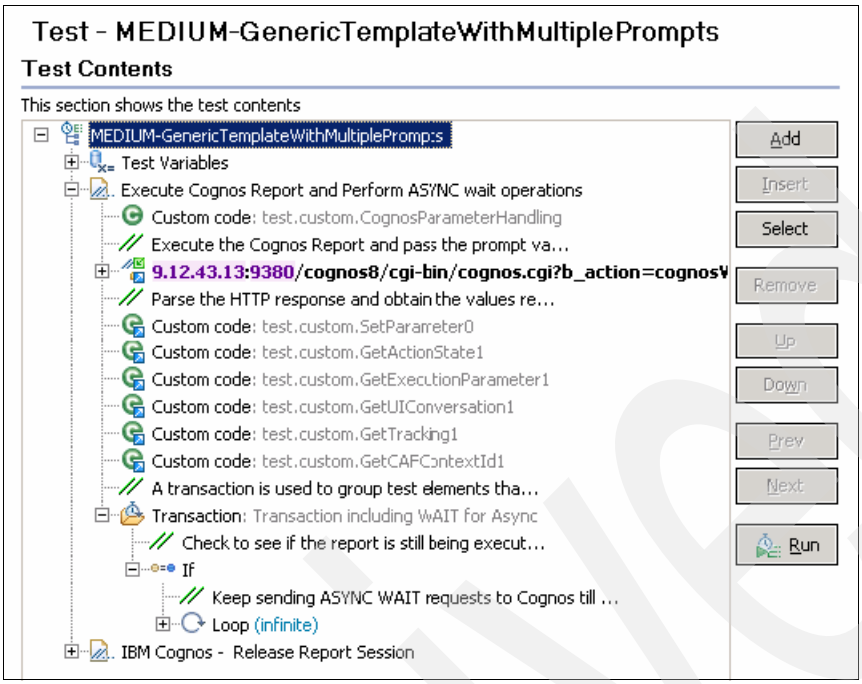


Figure 6-31 RPT test with multiple prompts

Data pools

Frequently, modifications are made to an RPT performance test to vary its interactions with the applications so that the same user responses are not repeated over and over again. RPT provides a useful mechanism, the datapool (DP), to provide any number of user responses based on application requirements, in the abstracted form of a number of rows, each of which is comprised of a number of fields.

Use of datapools and custom code can be combined. When this is done, the datapool is usually used for one of two purposes:

- To provide a variety of user response data (as previously described) for use within the custom code.
- To provide configuration or initialization information to the custom code.

RPT datapools are associated with performance tests. Similarly, custom code is also associated with a performance test. As a result, any datapool associated with a performance test can also be used by custom code associated with the same performance test.

Associating datapools with RPT tests

An RPT test can use multiple data pools. Figure 6-32 on page 116 demonstrates the usage of data pools in RPT. We have six variables defined in the data pool. One or more of these variables can be used in the RPT test.

Archive

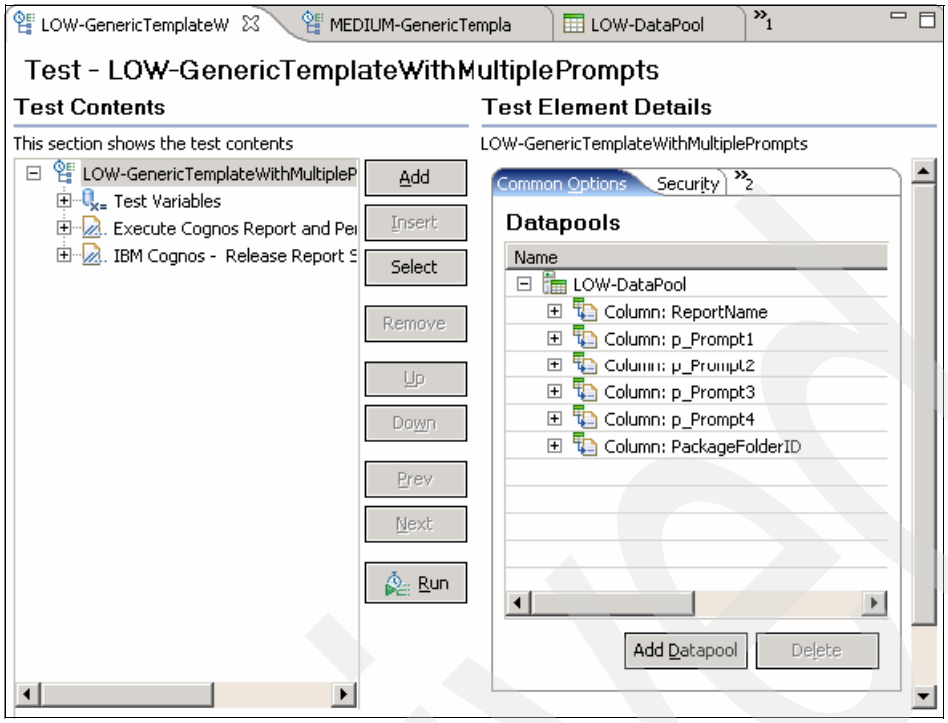


Figure 6-32 Datapool associated with an RPT test

Portions of the RPT test can be parameterized by highlighting and replacing it with values from the data pools; see Figure 6-33.

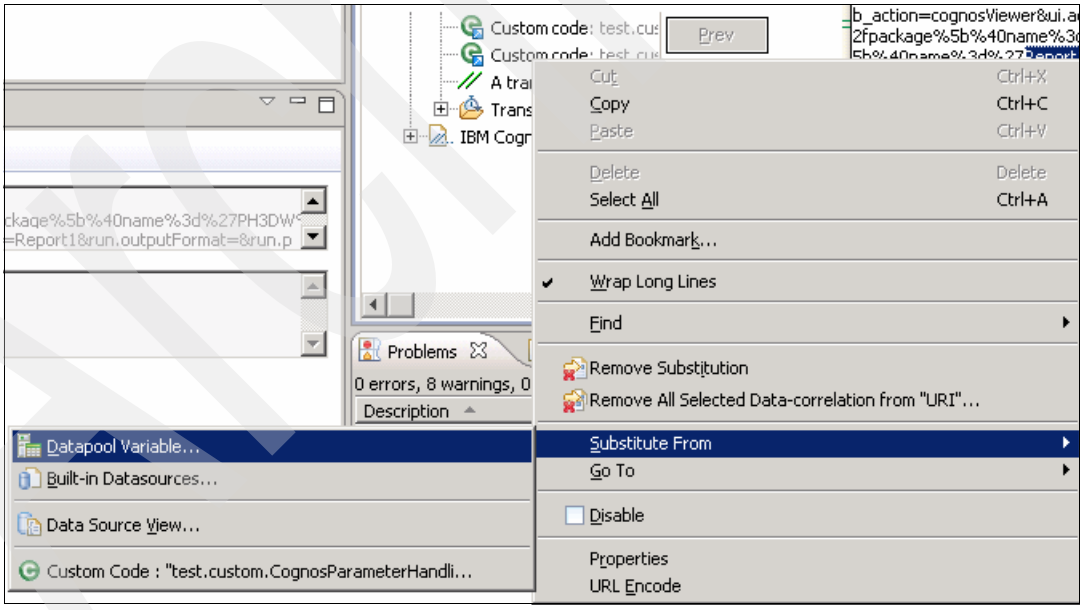


Figure 6-33 Replacing portions of the URL in an RPT HTTP request element

Figure 6-33 shows how we replaced portions of the URL in an RPT HTTP request element using substitution based on datapool column values.

When the performance test has an associated datapool, the custom code element that is associated with the test can access and use the information provided by the datapool. The

custom code does not have to use all the fields available in the datapool, but it can select which fields to use.

User groups

User groups enable the tester to group tests in a logical order to emulate the actions of different types of users. A schedule can contain one or more user groups. The users in the schedule can be distributed across the different user groups by specifying the percentage of users to be allocated to the user groups.

Each user group can further be segmented to run specific tests. This allows for a more realistic usage scenario. The example shown in Figure 6-34 demonstrates the use of user groups in test schedules. In this case we broke down the workload based on the CPU processing time. We have three categories of reports, LOW, MEDIUM, and HIGH. The setup will be distributing the users as 40% of the users running reports in the LOW category, 30% of the users running reports in the MEDIUM category, and 30% of the users running reports in the HIGH category.

We will have a total of 100 users in this case for this schedule, hence 40 user threads executing LOW category reports, 30 user threads executing MEDIUM category reports, and 30 user threads executing HIGH category reports.

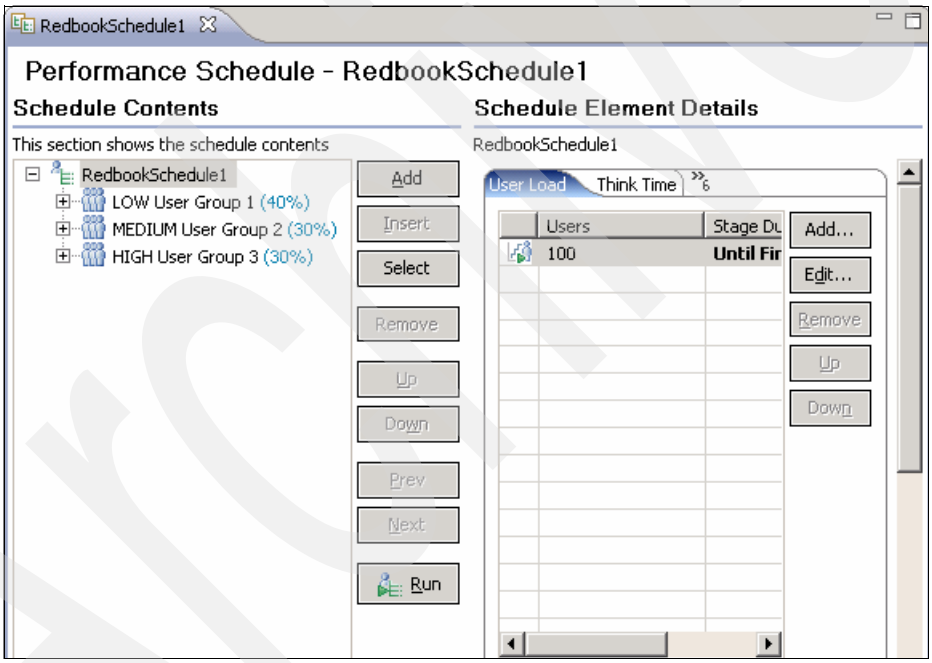


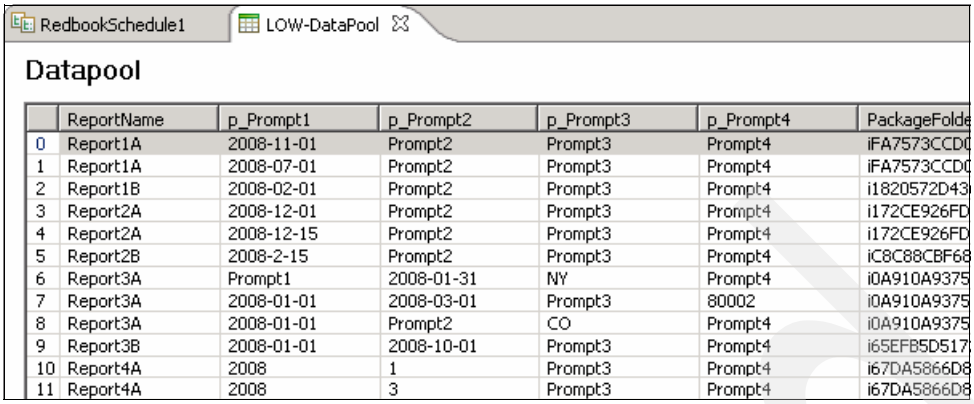
Figure 6-34 User group in RPT schedule

Load test definitions

For the purpose of this book we utilized a combination of datapools and RPT test schedules to define workloads. The following workload definitions were used for load testing.

Workload definitions at the datapool level

Workload 1 (Category=LOW) - This datapool will hold ten entries, each entry representing a combination of the User ID, Report name and all the prompt values required to run the report. The reports contained in this datapool are grouped based on the CP time (0.5 - 10 z10 CP seconds); see Figure 6-35 on page 118.

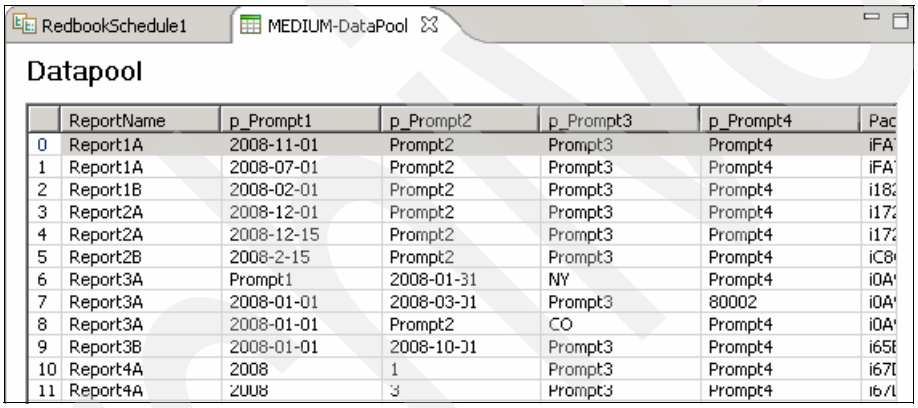

RedbookSchedule1 LOW-DataPool

Datapool

	ReportName	p_Prompt1	p_Prompt2	p_Prompt3	p_Prompt4	PackageFolde
0	Report1A	2008-11-01	Prompt2	Prompt3	Prompt4	iFA7573CCD(
1	Report1A	2008-07-01	Prompt2	Prompt3	Prompt4	iFA7573CCD(
2	Report1B	2008-02-01	Prompt2	Prompt3	Prompt4	i1820572D43(
3	Report2A	2008-12-01	Prompt2	Prompt3	Prompt4	i172CE926FD
4	Report2A	2008-12-15	Prompt2	Prompt3	Prompt4	i172CE926FD
5	Report2B	2008-2-15	Prompt2	Prompt3	Prompt4	iC8C88CBF68
6	Report3A	Prompt1	2008-01-31	NY	Prompt4	i0A910A9375
7	Report3A	2008-01-01	2008-03-01	Prompt3	80002	i0A910A9375
8	Report3A	2008-01-01	Prompt2	CO	Prompt4	i0A910A9375
9	Report3B	2008-01-01	2008-10-01	Prompt3	Prompt4	i65EFB5D517
10	Report4A	2008	1	Prompt3	Prompt4	i67DA5866D8
11	Report4A	2008	3	Prompt3	Prompt4	i67DA5866D8

Figure 6-35 Datapool to support the LOW category reports

Workload 2 (Category=MEDIUM) - This datapool will hold ten entries, each entry representing a combination of the User ID, Report name, and all the prompt values required to run the report. The reports contained in this datapool are grouped based on the CP time (10 - 50 z10 CP seconds); see Figure 6-36.

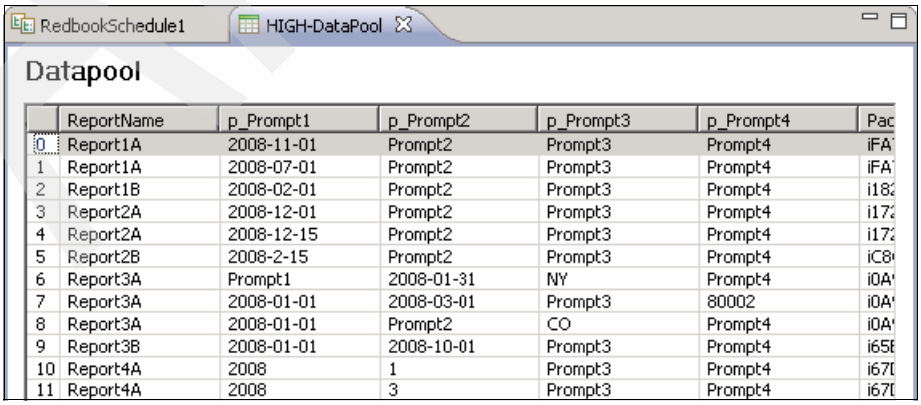

RedbookSchedule1 MEDIUM-DataPool

Datapool

	ReportName	p_Prompt1	p_Prompt2	p_Prompt3	p_Prompt4	Pac
0	Report1A	2008-11-01	Prompt2	Prompt3	Prompt4	iFA
1	Report1A	2008-07-01	Prompt2	Prompt3	Prompt4	iFA
2	Report1B	2008-02-01	Prompt2	Prompt3	Prompt4	i182
3	Report2A	2008-12-01	Prompt2	Prompt3	Prompt4	i172
4	Report2A	2008-12-15	Prompt2	Prompt3	Prompt4	i172
5	Report2B	2008-2-15	Prompt2	Prompt3	Prompt4	iC8
6	Report3A	Prompt1	2008-01-31	NY	Prompt4	i0A
7	Report3A	2008-01-01	2008-03-01	Prompt3	80002	i0A
8	Report3A	2008-01-01	Prompt2	CO	Prompt4	i0A
9	Report3B	2008-01-01	2008-10-01	Prompt3	Prompt4	i65
10	Report4A	2008	1	Prompt3	Prompt4	i67
11	Report4A	2008	3	Prompt3	Prompt4	i67

Figure 6-36 Datapool to support the MEDIUM category reports

Workload 3 (Category=HIGH) - This datapool will hold ten entries, each entry representing a combination of the User ID, Report name and all the prompt values required to run the report. The reports contained in this datapool are grouped based on the CP time (50 - 500 z10 CP seconds) see Figure 6-37.


RedbookSchedule1 HIGH-DataPool

Datapool

	ReportName	p_Prompt1	p_Prompt2	p_Prompt3	p_Prompt4	Pac
0	Report1A	2008-11-01	Prompt2	Prompt3	Prompt4	iFA
1	Report1A	2008-07-01	Prompt2	Prompt3	Prompt4	iFA
2	Report1B	2008-02-01	Prompt2	Prompt3	Prompt4	i182
3	Report2A	2008-12-01	Prompt2	Prompt3	Prompt4	i172
4	Report2A	2008-12-15	Prompt2	Prompt3	Prompt4	i172
5	Report2B	2008-2-15	Prompt2	Prompt3	Prompt4	iC8
6	Report3A	Prompt1	2008-01-31	NY	Prompt4	i0A
7	Report3A	2008-01-01	2008-03-01	Prompt3	80002	i0A
8	Report3A	2008-01-01	Prompt2	CO	Prompt4	i0A
9	Report3B	2008-01-01	2008-10-01	Prompt3	Prompt4	i65
10	Report4A	2008	1	Prompt3	Prompt4	i67
11	Report4A	2008	3	Prompt3	Prompt4	i67

Figure 6-37 Datapool to support the HIGH category reports

For the purpose of this book we have also used the RPT schedule definition to alter workloads. The RPT schedules we have defined essentially use three user groups and have various combinations of the workload distribution across these user groups.

The user groups we defined include:

- LOW User Group 1 - This user group executes only reports defined as part of the LOW datapool.
- MEDIUM User Group 1 - This user group executes only reports defined as part of the MEDIUM datapool.
- HIGH User Group 1 - This user group executes only reports defined as part of the HIGH datapool.

The user threads are distributed across the three user groups based on the distribution defined for the individual user groups.

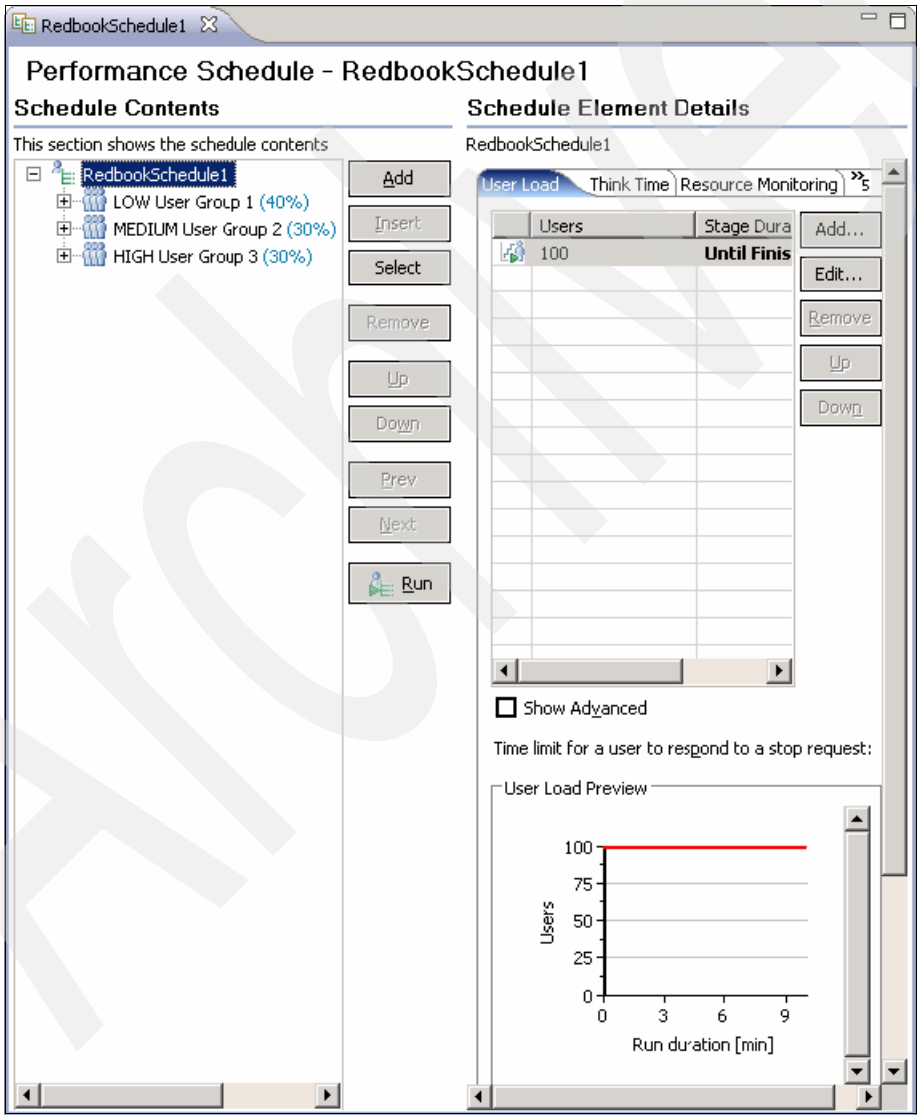


Figure 6-38 Schedule used for RPT test workload distribution

Archived

RPT test workflow

The workflow for the IBM Cognos BI test is as follows (Figure 6-39):

Step 1: For each user thread execute one report at a time; datapool values are used to define the reports to be executed.

Step 2: Read HTTP response and derive all the required session variables including the ASYNC status.

Step 3: If the ASYNC status=stillWorking then invoke an ASYNC WAIT request.

Step 4: Continue Step 3 till the ASYNC status=Complete.

Step 5: Release report session.

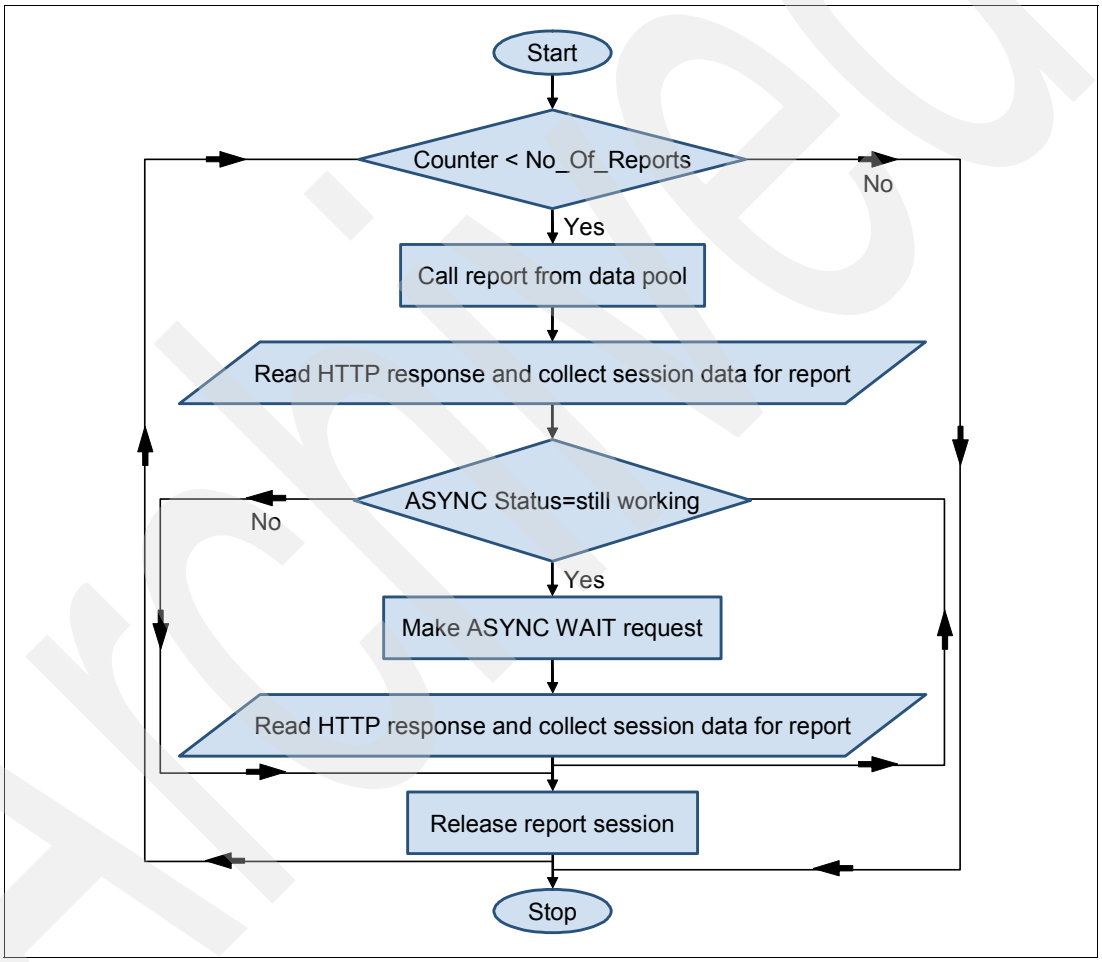


Figure 6-39 RPT Report execution flow

Archived

7

BookStore workload 3 - data warehouse refreshing

In this chapter we describe our implementation of traditional warehouse refresh with InfoSphere Warehouse SQL Warehousing Tool (SQW) and dynamic warehouse refresh with InfoSphere Change Data Capture. For each product, we describe the product and provide implementation details and how we used it for our experiments.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

7.1 Overview of data refreshing

Enterprise clients have traditionally used two approaches to refresh their data warehouses. The first approach is to use FTP plus unload and load utilities. Nightly, the data is unloaded, FTPed, and loaded into the warehouse. This is the case where the warehouse is basically a copy of the transactional data (an operational data store or ODS). The second approach adds some processing, such as having a COBOL program do some data transformation before FTPing and loading the warehouse. In this case, the warehouse has a different structure than the transactional data, such as a star schema warehouse.

The next level of modernization is to use an extract, transform, and load (ETL) program, or ELT tool. Many clients do this today, but there are still many that are using the FTP approaches above. The primary benefit of using an ETL or ELT tool is that the data transformations can be done without resorting to custom programming. Ideally, there are update timestamps on the transactional data that allow you to do an incremental update versus a full refresh. When this is the case, the ETL or ELT job can use SQL statements to identify changed data and apply the changes to the warehouse. In cases where these timestamps are not available, an ETL or ELT job can be used in conjunction with unload and load utilities to do full refreshes. The ETL or ELT update of the warehouse is still typically a nightly process.

The latest level of modernization is the idea of dynamic warehousing, where the goal is to always have current data in the warehouse. This is accomplished through the use of various change capture tools, such as InfoSphere Change Data Capture (as we used for this project) or InfoSphere Replication Server. If the data transformations are limited to transformations that the replication tool can perform, this approach works great. In other cases, the transformations are more complex and are better handled by an ETL or ELT tool. In these cases, we can use change capture tools to deliver change data to an ETL or ELT job, which does further processing and applies the changes.

Many clients are starting to look at doing dynamic warehousing, but frequently they are more interested in low-latency ODSs than in low-latency star schema warehouses. This might be because optimization techniques for star schema warehouses do not readily support low latency data. For example, it is very common for materialized query tables to be used to optimize star schema queries. However, MQTs are not updated in near real time, so it negates the benefit of updating the underlying data in near real time.

The second reason for not doing dynamic warehousing with star schemas is due to the fact that many clients believe that it is “not a good idea”, “not efficient”, or “too complex”. These might have been the case in the past, where change capture technologies and ETL and ELT technologies were not as mature.

There are other reasons for continuing to use a nightly ETL or ELT approach versus using a near real-time approach. One reason is that data in the warehouse is stored in a summarized or aggregated form. Here is an example: Let us assume that the data warehouse for a retail chain only stores the total daily sales and the average daily transaction amount, and that it does not store data concerning every transaction. Current change capture tools do not have the capability to recompute averages in near real-time as changes occur at the source. So, in these cases, it makes more sense to continue to use a nightly job to compute the daily summarization. However, some change capture tools have the ability to do basic types of summarization in near real-time. For example, InfoSphere Change Data Capture can do summarization via addition or subtraction in near real-time, but it still cannot do averages or other types of summarization.

The most common reason that clients do not do dynamic warehousing is that they do not see any additional value in providing intra-day data in the data warehouse. The value of doing dynamic warehousing seems to depend strongly on the industry involved. However, over time all industries seem to be moving in the direction of dynamic warehousing.

7.2 Batch refresh with InfoSphere Warehouse SQW

Here we discuss the specific batch refresh environment for our project.

7.2.1 InfoSphere Warehouse components

InfoSphere Warehouse on System z (ISWz) is part of a comprehensive IBM data warehousing and business intelligence solution for System z. A highly scalable, highly resilient, lower cost infrastructure to optimize a DB2 for z/OS warehouse.

ISWz simplifies transactional complexity by deploying both transactional and warehouse data on a single platform, reducing costs related to data movement while providing data compliance and security.

It dramatically improves query performance, saving on processor cost and elapsed time through the use of Cubing Services caching for Multidimensional (MDX) query support.

It exploits unique System z advantages including hardware-based data compression, world class workload management, and high availability through data sharing.

InfoSphere Warehouse for System z includes: -

- Design Studio

  Design Studio makes it easier to model, design, and implement analytic structures in the warehouse, and also helps populate and maintain them. Design Studio provides the plug-in for physical database design and provides an easy-to-use environment supporting the following capabilities:

  - Flexible, extensible (plug-ins), easy-to-use eclipse environment
  - Logical Entity Relationship design with Rational Data Architect plug-in (optionally available from Rational); Physical design
  - Extract, load, and transform
  - Data movement and transformation
  - Scripting
  - Cube and Materialized Query Table
  - Debugging and testing

- Browser-based administration console

  Administration Console provides a common browser interface to manage data warehouse resources, schedule and monitor data warehouse processing jobs and to configure and manage multidimensional cubes and cube servers.

  The Administration Console also provides troubleshooting, statistics, and logs and traces.

- Simplified SQL-based data movement and transformation with SQL Warehousing Tool (SQW)

  Customers can build and transform their physical data structures with the SQL Warehousing tool (SQW) provided in IBM InfoSphere Warehouse on System z. Designed

as an extension to the Rational Data Architect Physical Modeling environment, SQW provides data flow, data transformation and control flow operations required to load the data warehouse or data mart. SQW features include:

- SQL-based data movement and transformation features
- Load data sets from the server
- Staging table tuning: creating indexes, range partition, runstats
- Order by “get first n rows”
- Flow debugging and execution visual enhancements
- Online and offline modes of flow designing
- Virtual columns, expression sorting, filtering enhancements
- SQW runtime features
- Parallel execution in control flows
- Sub process modularizations
- Variable enhancements, reserved system variables
- Activity skip during execution
- Enhanced diagnostics (user control with file write operator), logging

SQW utilizes the concept of data flows and control flows to define and deploy intra-warehouse data movement. All the transformations against the various source and target operators are done using data flows.

One or more data flows can be executed in a predefined order using control flows. The output of the data flows can be routed to an email or just end the control flow.

- Access to multidimensional data with no-copy OLAP analytics

  InfoSphere Warehouse on System z clients can also build and deploy multidimensional analytics with Cubing Services. It empowers users to ask intuitive and complex ad hoc questions about their enterprise, such as “What is my product's profitability for this quarter across my territory?” This type of business query requires the grouping, aggregation and calculation of information across various business dimensions, such as time, product, customer, and supplier. Cubing Services is designed to resolve just this type of multidimensional query, known as Online Analytical Processing (OLAP).

  - Cubing Services supports the industry standard multidimensional (MDX) query language as well as industry standard OLAP client APIs that immediately enable the following industry leading BI tools:
    - OLE DB for OLAP (ODBO): Microsoft Excel, third party vendors
    - XML for Analysis (XMLA): IBM Cognos BI, third party vendors

7.2.2 Physical data modeling using Design Studio

A physical model is required for both the dimensional modeling as well as the design of data flows for the data warehouse. Design Studio, the Eclipse-based IDE, can be used to do the physical data modeling. A physical model can be created in two ways:

- Manually, based on a template
- By reverse engineering an existing schema

For the purpose of this project, we already have a physical data mart. So we will use the process of reverse engineering an existing database schema to obtain our physical model.

Upon creating the physical model, which is essentially a file in the local file system, all the data flows and OLAP cubes can be designed without being connected to the data warehouse.

Reverse engineering an existing schema

The steps involved in reverse engineering are:

Step 1: Create a data warehousing project

In Design Studio, select **File** → **New** → **Data Warehousing Project**. See Figure 7-1.

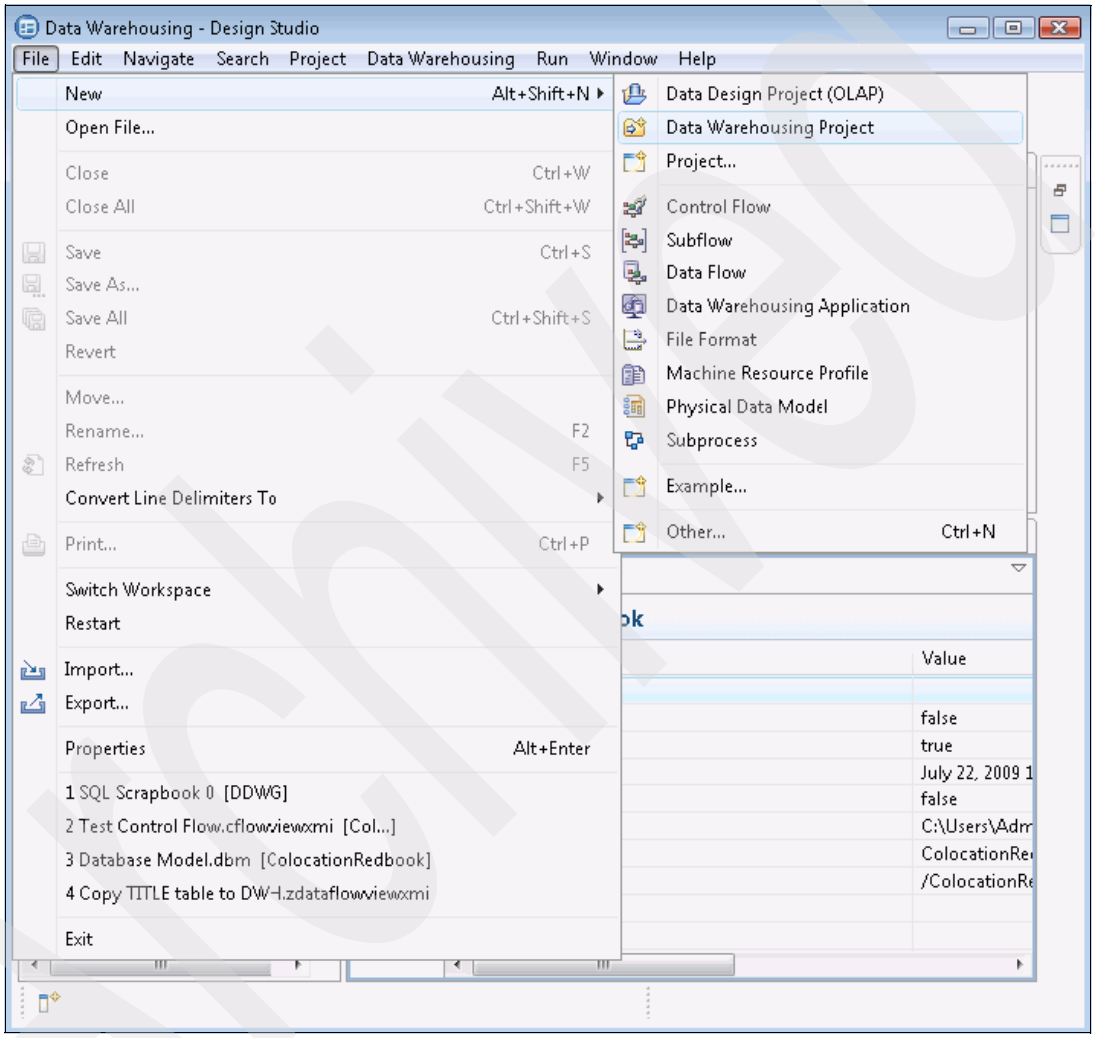


Figure 7-1 Creating a data warehousing project

Step 2: Enter the project name

<DATA_WAREHOUSE_PROJECT_NAME> and press **Finish**. See Figure 7-2 on page 126.

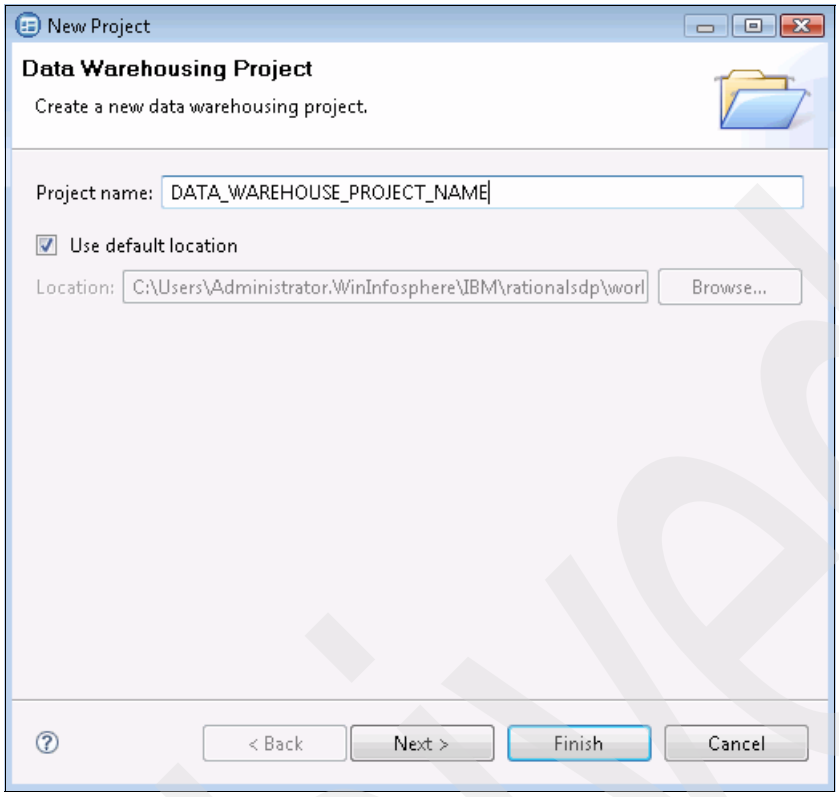


Figure 7-2 Nominate Design Studio project name for the data warehouse project

Step 3: Reverse engineer an existing database schema

From the Data Warehousing Design Studio, click the Open the Data Project Explorer. See Figure 7-3 on page 127.

Archived

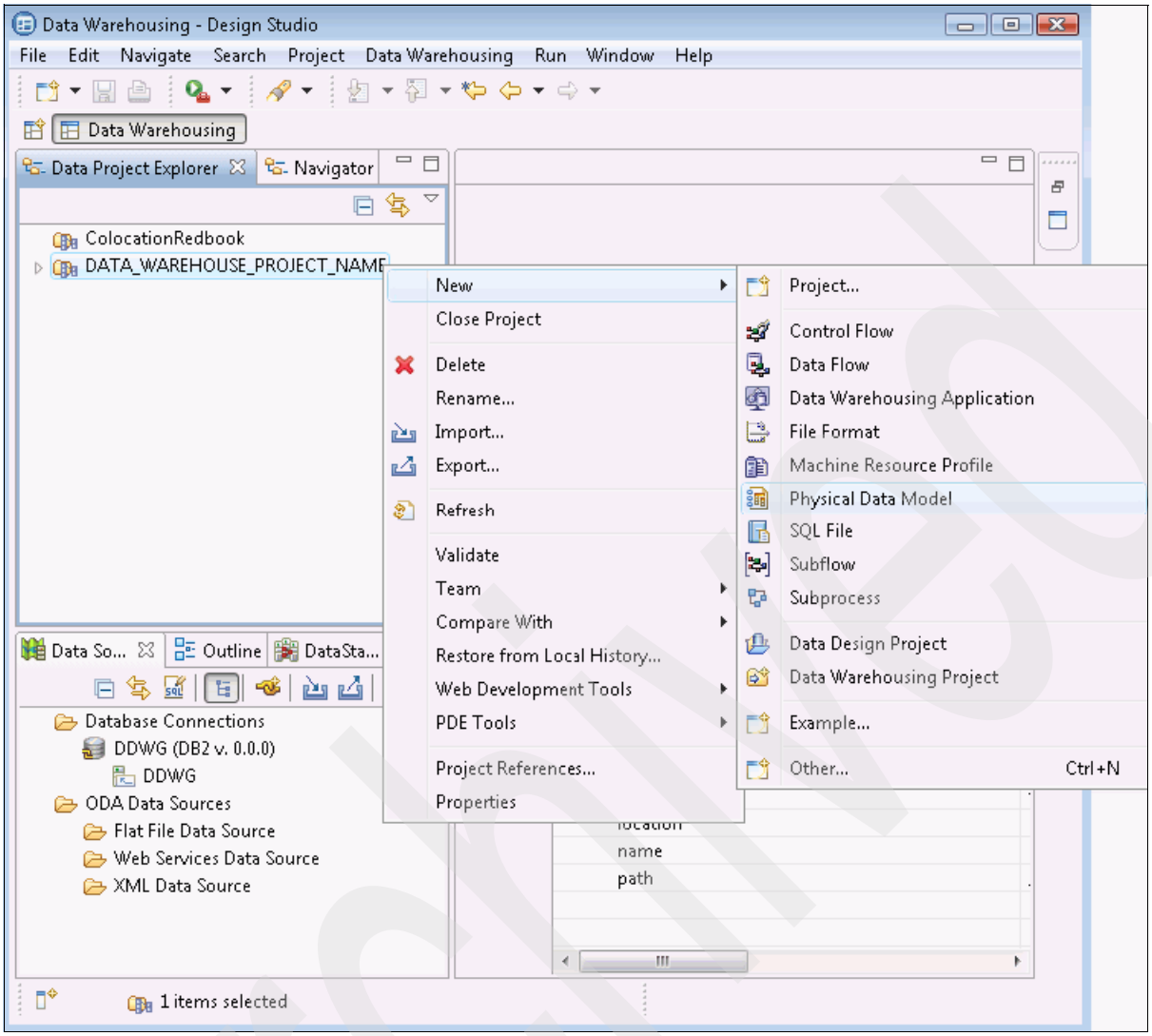


Figure 7-3 Creating a new physical model

Right-click the project name and select **New** → **Physical Data Model**.

Archive

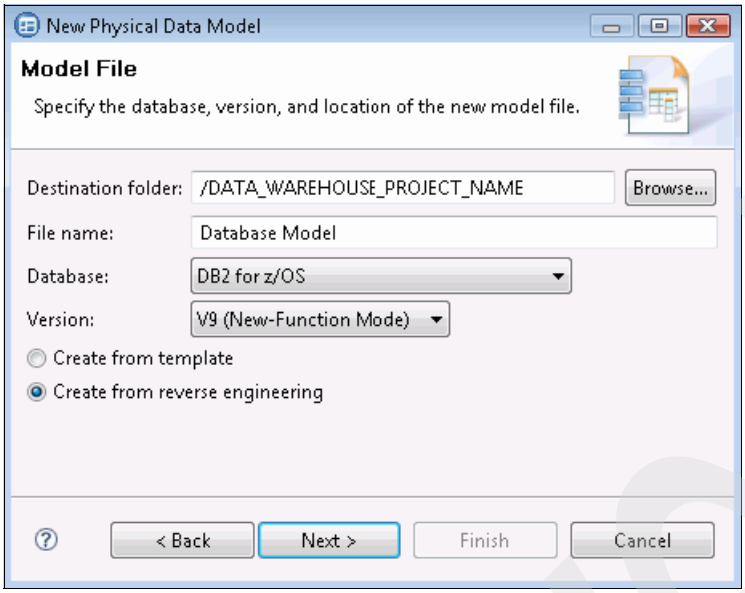


Figure 7-4 Choose physical model creation method Reverse Engineering

Step 4: Select source database for reverse engineering based on a defined connection

See Figure 7-5.

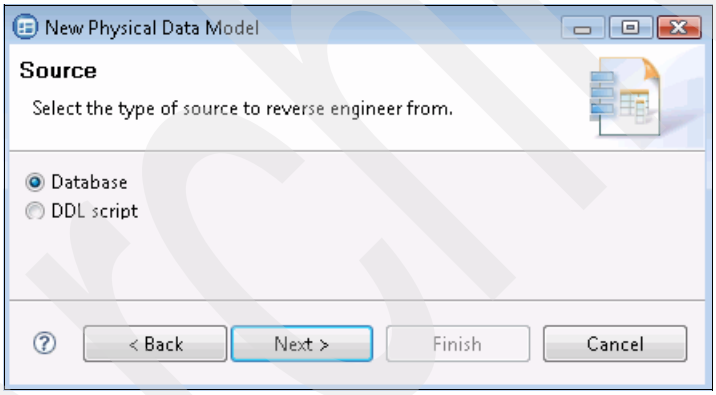


Figure 7-5 Select database as source for physical model

Archived

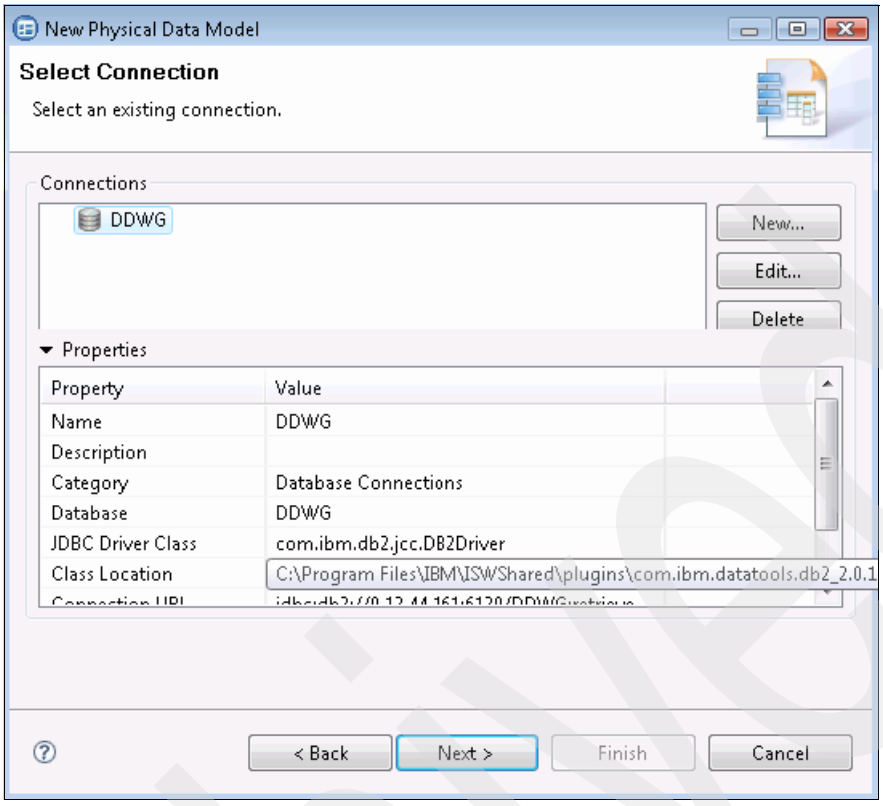


Figure 7-6 Select the appropriate source database connection

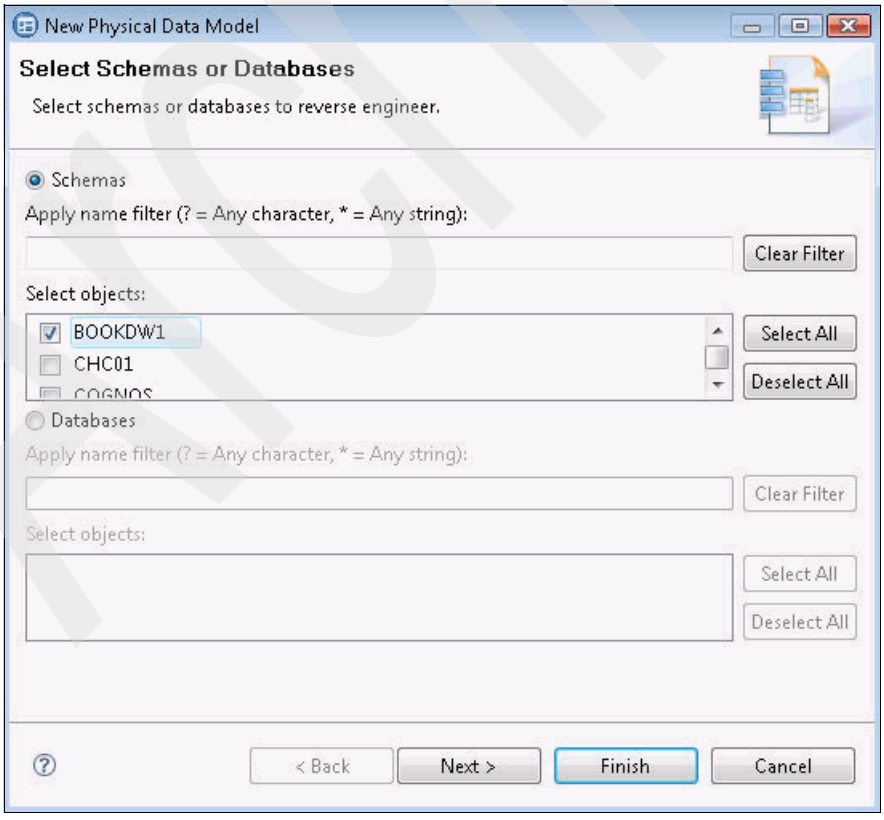


Figure 7-7 Apply filter to isolate the reverse engineering to required objects only

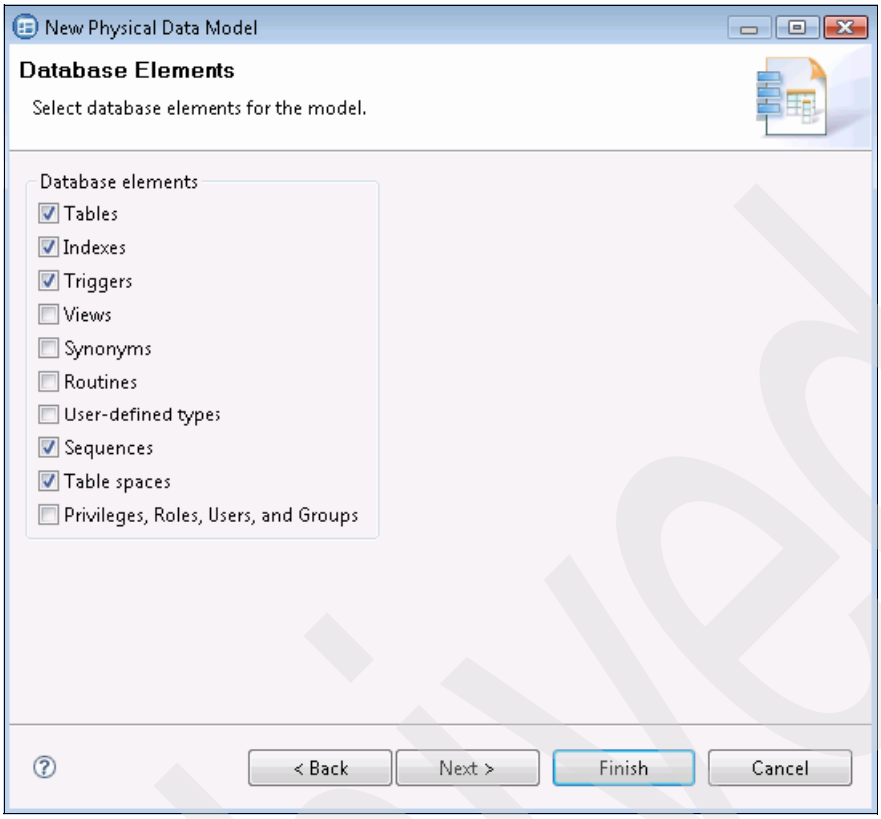


Figure 7-8 Filter out required database elements

These steps should give you a physical model to work with from the client workstation. This physical model is disconnected from the underlying data source and can be used for all the design and modeling of SQL Warehousing (SQW) and OLAP objects.

Creating and executing data flows using Design Studio

The high-level steps for defining a data flow using Design Studio include:

1. Open a new data flow

   Select **Data Project Explorer** → **Data Flow** (right click) → **New**.
2. Set the SQL execution database and schema to be used.
3. Drag and drop a source operator to the palette.
4. Define the source operator properties.
5. Drag and drop a transform operator.
6. Connect the two operators.
7. Define the transform operator's properties.
8. Drag and drop a target operator.
9. Connect the output ports of the transform operator to the input ports of the target operator.
10. Define the target operator's properties.
11. Save, validate, and test the data flow.

Source operators

- SQL query source

- Table source
- Data set import

Target operators

- Cross loader operator
- Custom SQL operator
- Table target
- Data set export

Transform operators

- Data station operator
- DB2 table function operator
- Distinct operator
- Fact key replace operator
- Group by operator
- Key lookup operator
- Order by operator
- Pivot operator
- Select list operator
- Sequence operator
- Splitter operator
- Table join operator
- Union operator
- Unpivot operator
- Where condition operator

The sample data flow in Figure 7-9 on page 132 demonstrates the movement of data from two source tables, BOOK_ORDER and ORDER_DETAIL, into the target table BOOK_ORDER_DETAIL. The records from the two input tables are JOINED based on their Primary Key<->Foreign Key relationship.

Archived

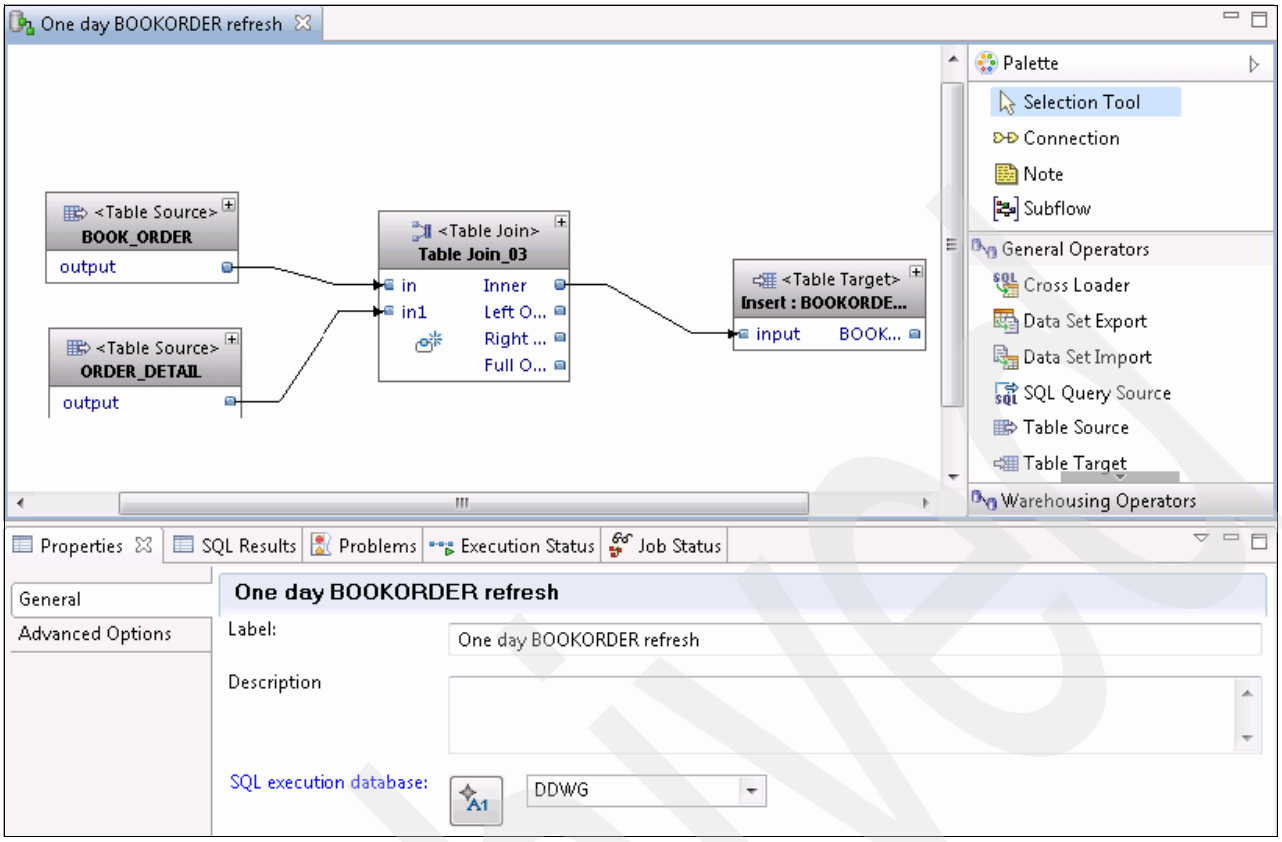


Figure 7-9 Simple control flow to JOIN two tables and copy data onto a third table

Data flow operators reference DB2 tables, views, and other database objects. You need to understand the limitations that are imposed on these references. DB2 objects that are referenced in data flows must exist in the SQL execution database, except for tables in table source and table target operators, which can be accessed via JDBC. The following objects must exist in the execution database:

- Tables that are referenced in SQL merge and bulk load target operators
- Target tables and views in data station operators
- DB2 table functions
- DB2 sequence objects

Table source operators can access remote tables via JDBC directly or by using nicknames for tables in federated data sources. Table target operators must access remote tables directly via JDBC. You cannot use a DB2 nickname (a reference to a remote, federated table) as the target table in a data flow.

The SQL Warehousing Tool might use DB2 declared temporary tables to stage data during execution.

In DB2 for z/OS Version 8, the DB2 database stores all declared temporary tables in the TEMP database (a database that is defined as TEMP). You cannot define a declared temporary table unless a segmented table space with at least an 8 KB page size exists in the TEMP database.

In DB2 9.1 for z/OS, the DB2 database stores all declared temporary tables in the work file database. You cannot define a declared temporary table unless a table space with at least a 32-KB page size exists in the work file database.

SQW data flows and control flows might create internal staging tables when you run flows in the Design Studio. The Design Studio drops these tables and views after the execution of the flow is complete. However, these internal staging tables and views might have the same names as the objects that already exist in the database. These database objects might currently be used or locked by some active SQL statements or database utilities. Owing to the conflict with the names of internal staging tables and views, SQW flows might fail to run. Therefore, before you run SQW flows in the Design Studio, ensure that there are no active SQL statements and database utilities that are being processed by the database.

The Design Studio interface allows you to also execute and debug data flows. Execution of data flows is mostly done using the Design Studio interface in the development phase, whereas the web-based Administration Console is used to schedule and maintain ongoing transform and load jobs using data flows.

Execution of data flows can be done using the DB2 for z/OS **Data Flow** → **Execute** menu item. Design Studio can also be used to debug data flows and step through each operator and inspect the values for the variables used.

Creating and executing control flows using Design Studio

A control flow is a container model that sequences one or more data flows and integrates other data processing rules and activities. Data warehouse applications deployed to the Administration console depend on control flows.

Create a new control flow by right-clicking the **Control Flow** folder in the data warehouse project, as shown in Figure 7-10.

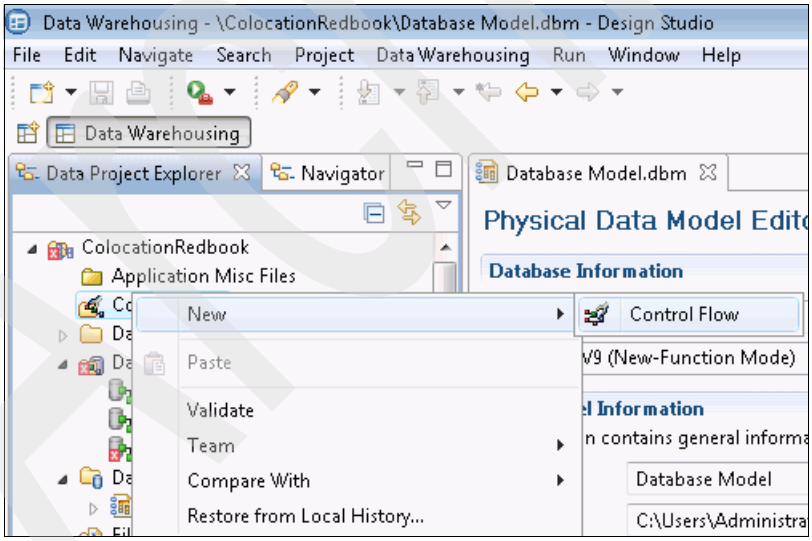


Figure 7-10 Create a new control flow

Here we demonstrate how to execute a data flow. Based on the outcome, we either end the control flow or email the status to the operator. In Figure 7-11 on page 134 we select the current project and give a name to the control flow: Executing_A_DataFlow.

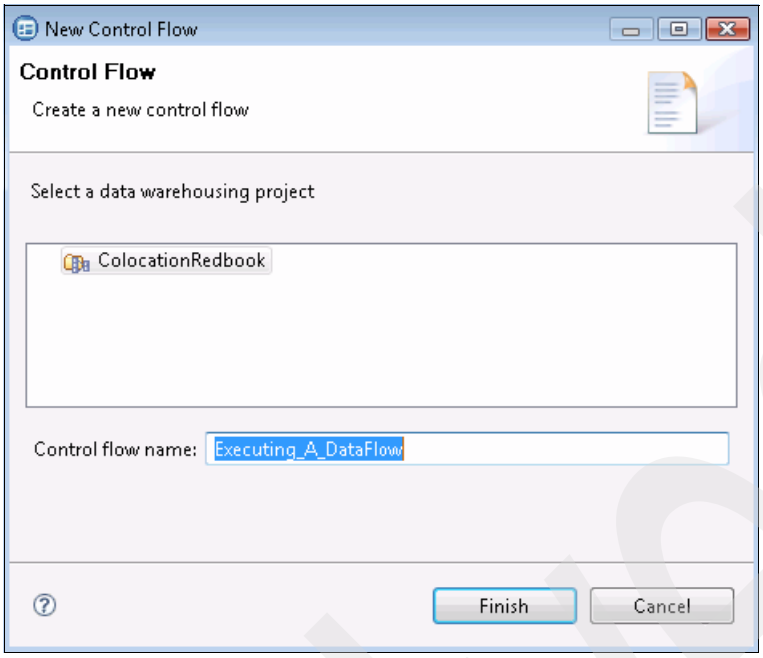


Figure 7-11 Create a new control flow - Step 2

Steps for creating the sample control flow include (see Figure 7-12 on page 135):

- Drag-drop the Data Flow operator to the pallet.
- Connect the output port of the Start operator to the input port of the Data Flow operator.
- Modify the properties of the Data Flow operator and select the data flow to be executed.
- Connect the On Success output port to the End operator.
- Connect the On Failure output port to the Email operator.
- Modify the properties of the Email Operator and include the sender, recipient, subject and message fields.
- Connect the output port of the Email operator to the input of the End operator.

Archive

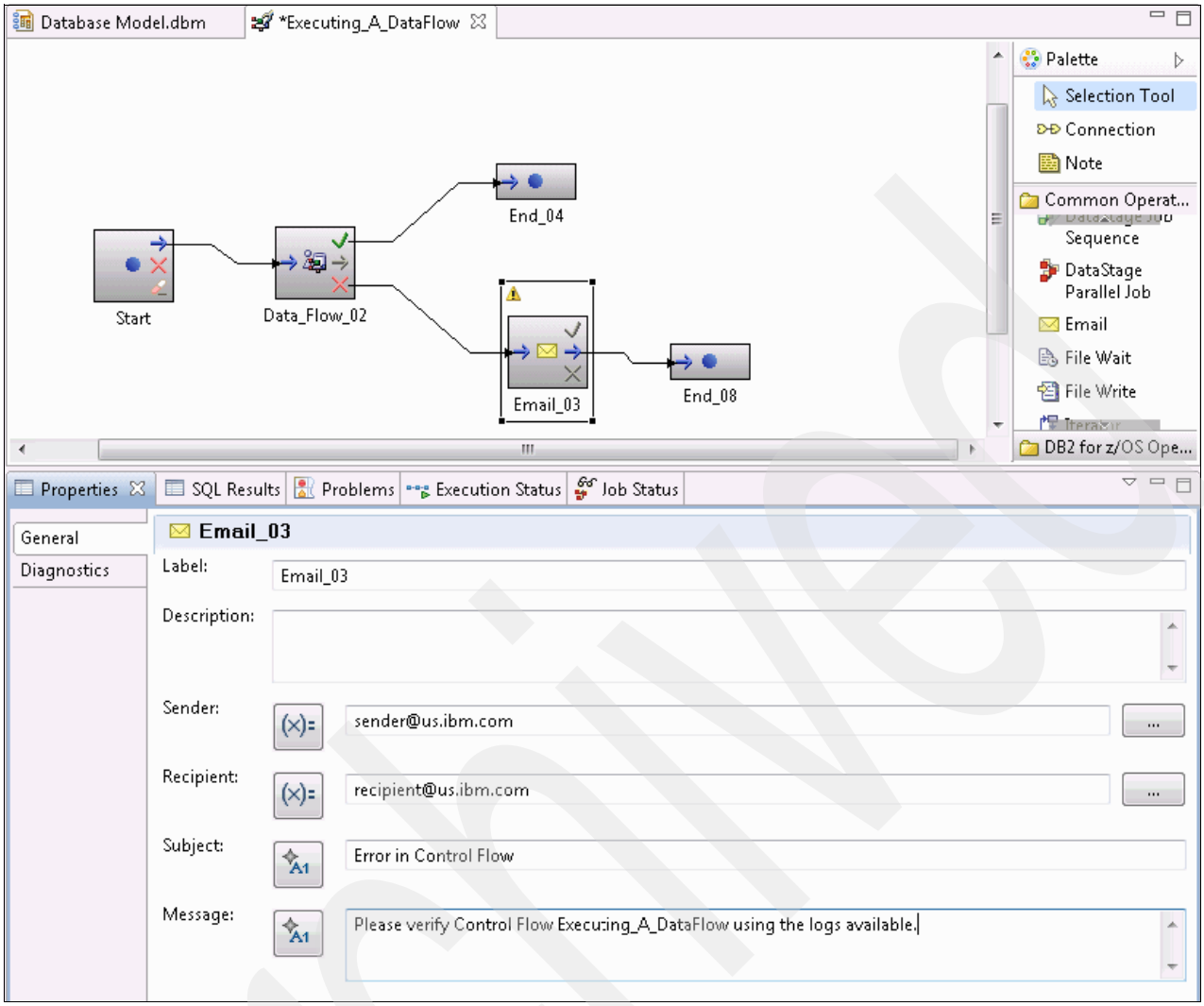


Figure 7-12 Sample control flow

Creating and deploying a data warehouse application

All data warehouse applications are created using the Design Studio. A data warehouse application can contain one or more control flows that use connections, variables and resources that can be parameterized.

Creation and deployment of a data warehouse application is a two-step process. The application is created using the Design Studio interface while the web-based Administration Console is used to deploy the data warehouse application.

Creating a data warehouse application

Steps for creating a data warehouse application include:

- From within the Design Studio interface, select **File** → **New** → **Data Warehousing Application**. See Figure 7-13 on page 136.

Archive

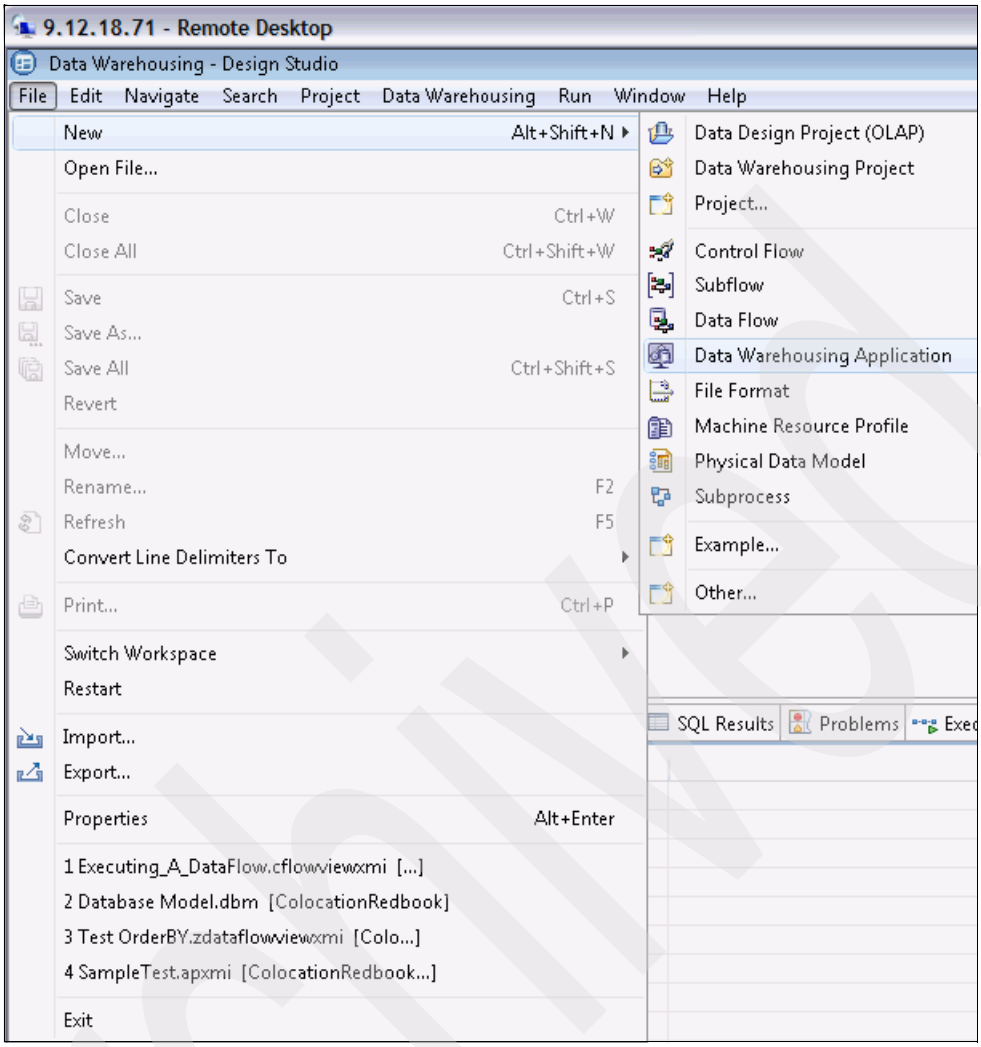


Figure 7-13 Creating a data warehouse application

- Input the profile name and description for the data warehouse application; see Figure 7-14 on page 137.

Archive

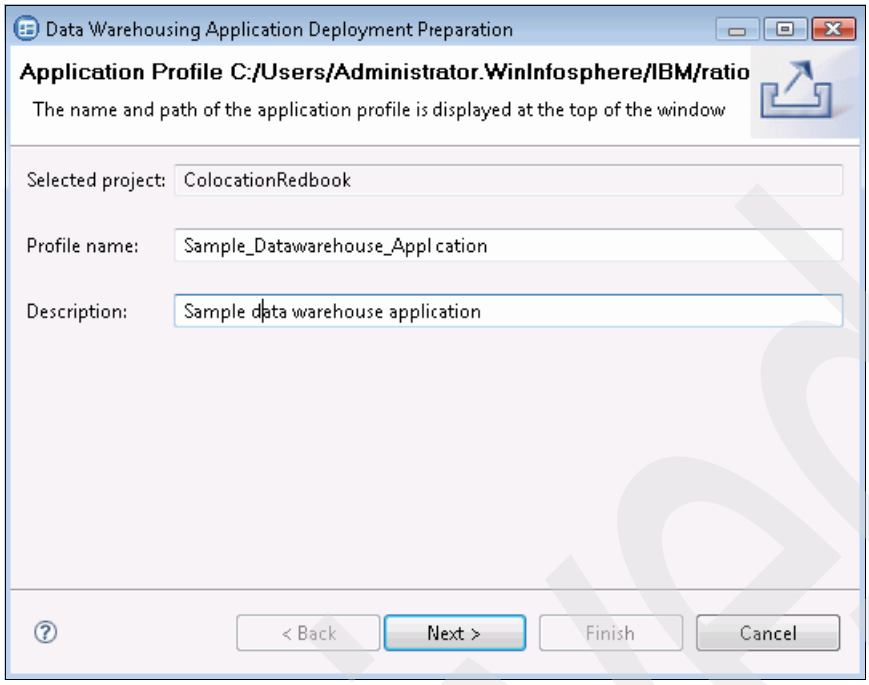


Figure 7-14 Profile name and description for the data warehouse application

- Select the control flows to include in the data warehouse application; see Figure 7-15.

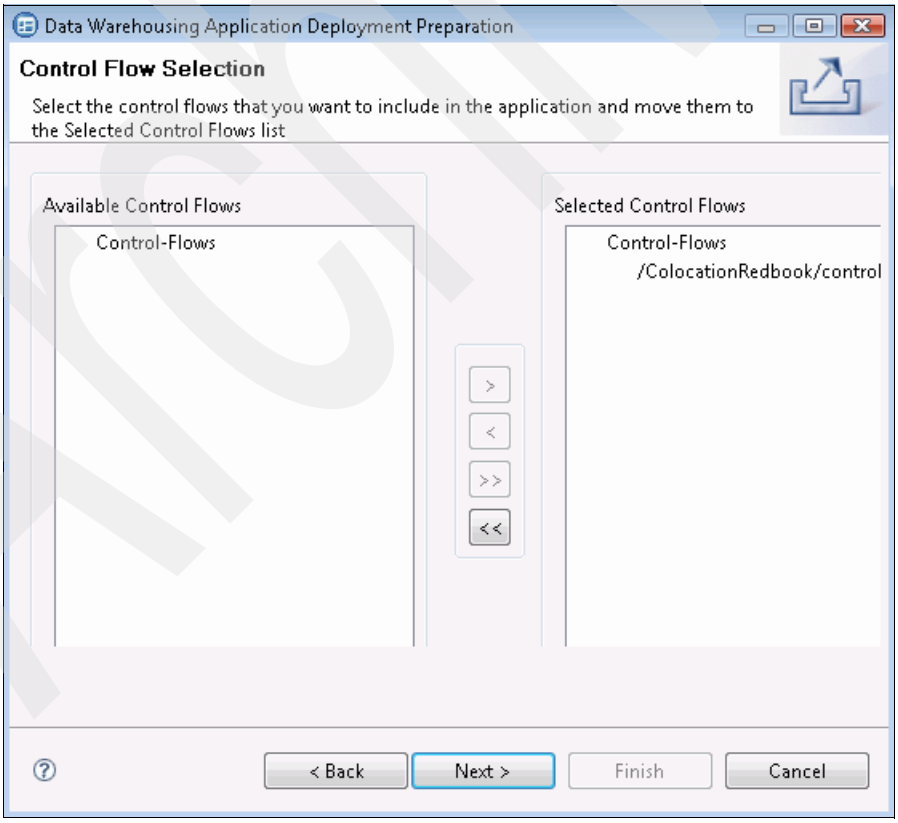


Figure 7-15 Select control flows to include in the data warehouse application

- Include resources used by the control flows in the data warehouse application; Figure 7-16

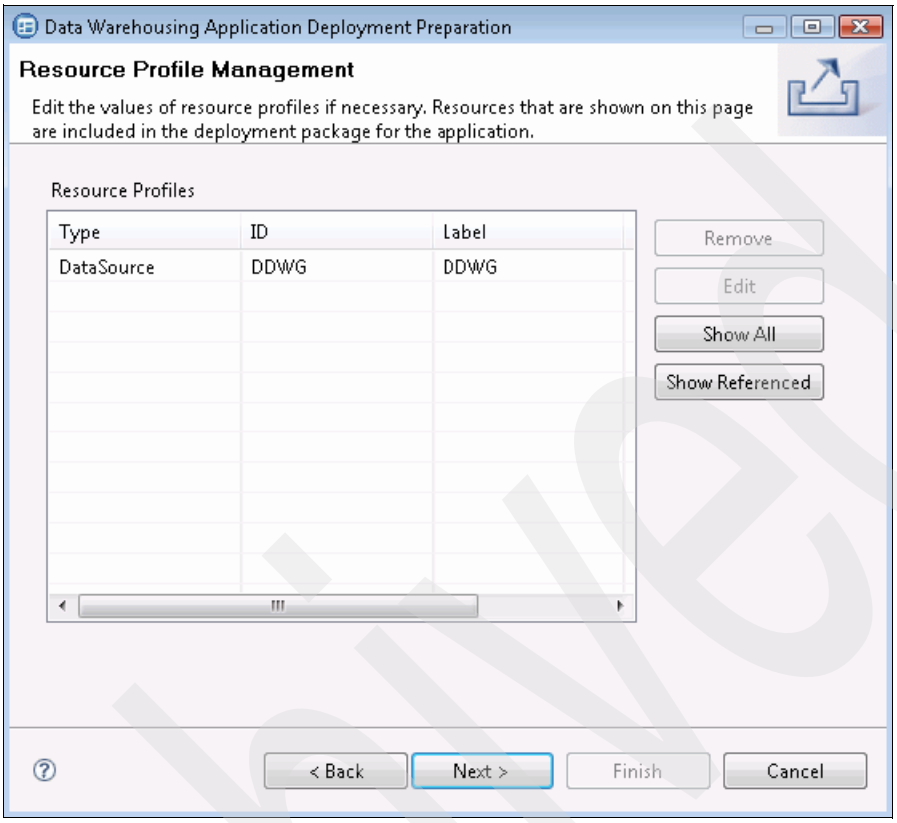


Figure 7-16 Select control flow resources

- Select variables used in the control flow (if applicable); Figure 7-17 on page 139.

Archive

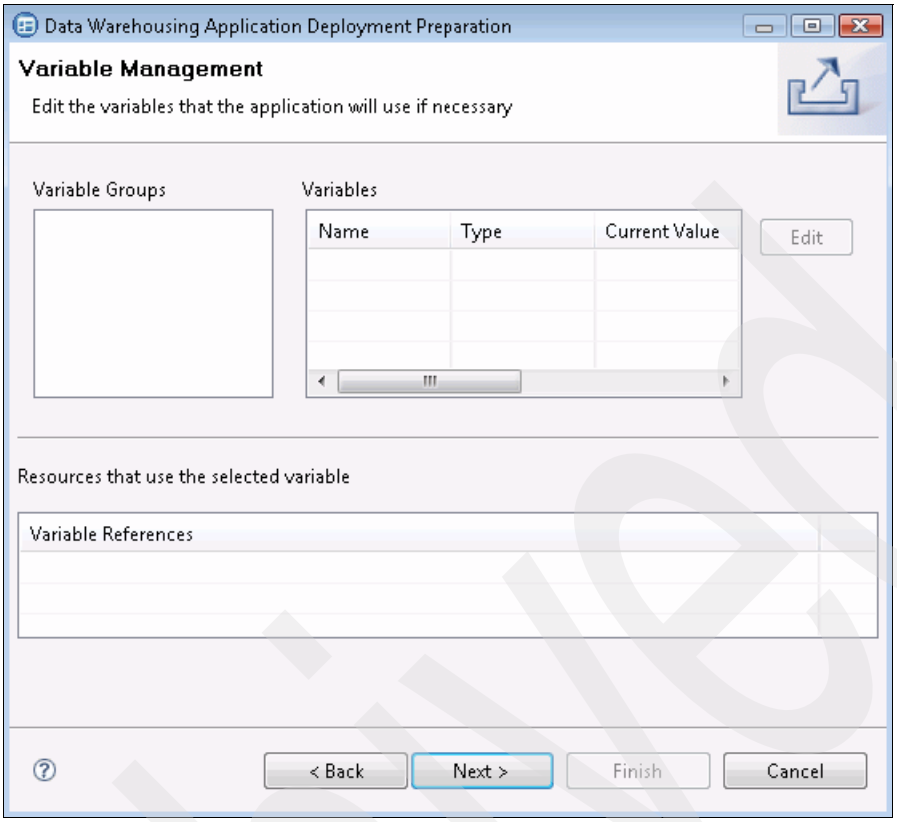


Figure 7-17 Select variables used in the control flow (if applicable)

- Complete the data warehouse application save process; Figure 7-18 on page 140.

Archive

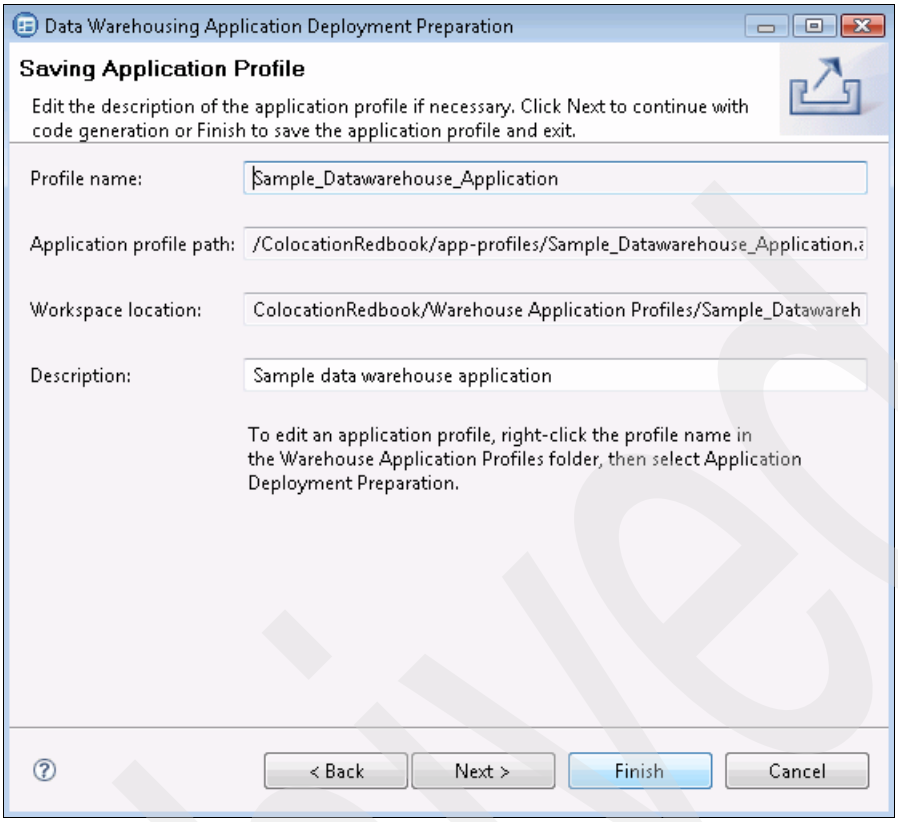


Figure 7-18 Complete data warehouse application save process

Upon clicking **Finish**, a zip file is created for the data warehouse application in the nominated directory.

7.2.3 InfoSphere Warehouse refresh scenario in a non-data sharing environment

For our non-data sharing-related data warehouse refresh runs, we use a data flow as it is depicted in Figure 7-19 on page 141 to move data from the OLTP source tables to the data warehouse target table.

Archived

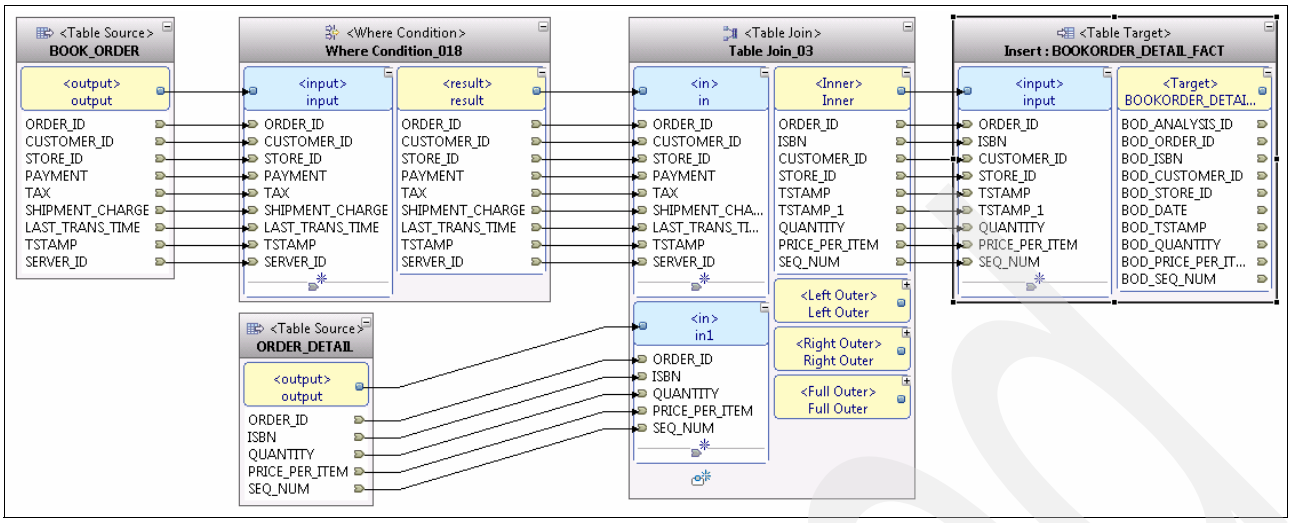


Figure 7-19 Data flow modeling for non-data sharing refresh

The data flow shown in Figure 7-19 joins tables BOOK_ORDER and ORDER_DETAIL using a remote connection to the remote OLTP DB2 for z/OS subsystem and inserts the result set into the BOOKORDER_DETAIL_FACT table in the local data warehouse DB2 for z/OS subsystem. The attributes used for the source tables specify the remote source DB2 for z/OS subsystem while the attributes for the target table specify the local target DB2 for z/OS subsystem and an appropriate commit frequency of 1,000 to be used during the insert processing.

You can find the generated code for this data flow, incorporating two different connections to read and write the data from and to different DB2 for z/OS subsystems in Example 7-1.

Example 7-1 Generated code for non-data sharing refresh scenario

```
-- <ScriptOptions statementTerminator="@" />

---- BEGIN NON-SQL CODE ----

-- JAVA

-- Java Runtime Unit Class:
com.ibm.datatools.etl.dataflow.baselib.runtimeunits.JDBCInsert

-- Source Database Name = DDWG
--
--Source SQL Query = SELECT Q1035.ORDER_ID AS ORDER_ID, Q1045.ISBN AS ISBN,
Q1035.CUSTOMER_ID AS CUSTOMER_ID,
--  Q1035.STORE_ID AS STORE_ID, DATE(Q1035.TSTAMP) AS TSTAMP, Q1035.TSTAMP AS
TSTAMP_1,
--  Q1045.QUANTITY AS QUANTITY, Q1045.PRICE_PER_ITEM AS PRICE_PER_ITEM,
--  Q1045.SEQ_NUM AS SEQ_NUM
--  FROM PH3OLTP.BOOK_ORDER AS Q1035, PH3OLTP.ORDER_DETAIL AS Q1045
--  WHERE (Q1035.ORDER_ID = Q1045.ORDER_ID)
--    AND (Q1035.TSTAMP >= TIMESTAMP('${Colocation/IntervalBegin}'))
--    AND (Q1035.TSTAMP <= TIMESTAMP('${Colocation/IntervalEnd}'))
--
--
--Target Database Name = DBDMDDF
```

```
--
--Target Insert Statement = INSERT INTO PH3DW.BOOKORDER_DETAIL_FACT(BOD_ORDER_ID,
BOD_ISBN, BOD_CUSTOMER_ID, BOD_STORE_ID, BOD_DATE, BOD_TSTAMP, BOD_QUANTITY,
BOD_PRICE_PER_ITEM, BOD_SEQ_NUM) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
--
--NumTargetColumns = 9
--
--CommitInterval = 1000
--
--BatchSize = 1000
--
--Duplicate Connection On = TARGET
--
--Number of Database Connection(s) = 2
--
--Database Connection 0 = DDWG
--
--Database Connection 1 = DBDMDDF
--
--OperatorIDs = com.ibm.datatools.etl.common.impl.StringListImpl@37333733
(content: [01, 02, 06, 018, 031])

---- END NON-SQL CODE ----
```

Inside InfoSphere Warehouse, you can see the generated code by clicking **DB2 for z/OS Data Flow** and selecting **Show Generated Code** or **Show Generated Execution Plan**.

By using two different connections to the OLTP and the data warehouse DB2 for z/OS subsystems, the generated code allows for intermediate COMMITs to release any acquired locks on the target objects. With respect to the relation of the amount of data the query inserts and the number of rows that already exist in the target table, we have chosen a commit frequency of 1,000 to minimize lock contention with concurrently executing queries on the DWH DB2 for z/OS subsystem.

7.2.4 InfoSphere Warehouse refresh scenario in a data sharing environment

The difference for a refresh scenario in a data sharing environment compared to the refresh scenario we have shown here for non-data sharing environments is that the source DB2 for z/OS subsystems is no longer classified as a remote database. Since all tables in a data sharing environment can be accessed from each member, we simply specify the same local DB2 for z/OS subsystem for all tables. The location option in InfoSphere Warehouse allows you to:

- Select the SQL execution database option when the database that contains the target table is local to the DB2 database where the data flow runs.
- Select the Remote database option if the database that contains the target table is not local to the SQL execution database and is accessible through JDBC.

The code generated for this data flow, where all tables are accessible in the same data sharing group, is an INSERT statement with a subselect clause as shown in Example 7-2.

Example 7-2 Generated SQL for data sharing refresh

```
INSERT INTO PH3DW.BOOKORDER_DETAIL_FACT (BOD_ORDER_ID, BOD_ISBN,
  BOD_CUSTOMER_ID, BOD_STORE_ID, BOD_DATE, BOD_TSTAMP, BOD_QUANTITY,
```

```
  BOD_PRICE_PER_ITEM, BOD_SEQ_NUM)
  SELECT Q1035.ORDER_ID AS ORDER_ID, Q1045.ISBN AS ISBN, Q1035.CUSTOMER_ID AS
CUSTOMER_ID,
  Q1035.STORE_ID AS STORE_ID, DATE(Q1035.TSTAMP) AS TSTAMP, Q1035.TSTAMP AS
TSTAMP_1,
  Q1045.QUANTITY AS QUANTITY, Q1045.PRICE_PER_ITEM AS PRICE_PER_ITEM,
  Q1045.SEQ_NUM AS SEQ_NUM
    FROM PH3OLTP.BOOK_ORDER AS Q1035, PH3OLTP.ORDER_DETAIL AS Q1045
    WHERE (Q1035.ORDER_ID = Q1045.ORDER_ID)
    AND (Q1035.TSTAMP >= TIMESTAMP('1999-1-1-00.00.00.000000'))
    AND (Q1035.TSTAMP <= TIMESTAMP('1999-1-1-00.59.59.999999'));
```

Note that the timestamps as shown in the WHERE clause have been provided using an input file during the execution of our test cases. Therefore, these values are not visible in the panel shots we have shown here.

However, since this is only a minor change in InfoSphere Warehouse for the refresh process, the result differs a lot from the behavior of the non-data sharing refresh process: Since the INSERT with subselect is now passed to DB2 for z/OS as one statement, it is not possible to honor any value being specified in the COMMIT frequency column. This can result in lock escalation on the target objects, thereby blocking non-uncommitted read access to the target table. because of this, this approach is only recommended if the amount of data going to be moved to your data warehouse DB2 for z/OS subsystem within one unit of recovery does not violate your Service Level Agreements in terms of availability of your data warehouse data.

Important: Make sure that your environment can tolerate INSERT with subselect statements without issuing intermediate COMMIT statements before implementing this solution.

In general: INSERT with subselect

INSERT INTO t1 (SELECT...) statements are powerful for populating a table in a very simple and quick manner. However, these statements do not allow for any control of COMMIT frequencies. Lack of frequent COMMITs could potentially cause lock escalations on the target table space, depending on your chosen locking parameters and the number of rows you transfer.

Other tables in the target table space are affected by the lock escalation as well, as are SQL statements that normally would use lock avoidance in the same DB2 for z/OS data sharing group (lock avoidance is allowed by using BIND options ISOLATION (CS) and CURRENTDATA (NO) for both qualifying and non-qualifying pages or rows; using ISOLATION (CS) for non-qualifying pages or rows). Since the log record sequence number (LRSN) for the oldest active unit of recovery (UOR) in a data sharing group is used to control whether SQL statements can make use of lock avoidance or not, having long-running UORs in the system can affect lock-avoidance capabilities of the entire data sharing group. You may observe additional Internal Resource Lock Manager (IRLM) requests for other SQL statements that made use of lock avoidance before. For more information about lock avoidance, refer to *Resource Serialization and Concurrency Control*, SG24-4725.

If this kind of in-database movement of data is essential for your data warehouse refresh processing without any additional tooling, you might want to consider implementing INSERT INTO (SELECT...) statements into an application that can control COMMIT frequency. See 7.2.5, “Other possible refresh scenarios specific to data sharing environments” on page 145 for more suggestions about implementing a potential application logic. If tooling is an option,

make sure that you can control a reasonable COMMIT frequency to reduce the overhead caused by additional IRLM lock requests.

One parameter to control lock escalation is the LOCKMAX parameter you can use to specify the maximum number of locks that a process can hold on a table space. This parameter is available at both CREATE and ALTER TABLESPACE statements.

When your data warehouse queries use uncommitted read to access data for reporting, a large number of locks or even lock escalations may probably not become a critical issue. In this specific case, adding data to your data warehouse or performing incremental updates in large numbers can be done by passing single units of recovery to the database engine for execution. To keep the locking overhead and the number of round trips to the Internal Resource Lock Manager (IRLM) to a minimum, consider issuing the LOCK TABLE statement using the EXCLUSIVE mode prior to passing a long-running INSERT or UPDATE statement to DB2 for z/OS. See Example 7-3 for the logical flow regarding the previous refresh example in our InfoSphere Warehouse refresh scenario for data sharing.

Example 7-3 LOCK TABLE and single unit of recovery

```
LOCK TABLE PH3DW.BOOKORDER_DETAIL_FACT IN EXCLUSIVE MODE;

INSERT INTO PH3DW.BOOKORDER_DETAIL_FACT (BOD_ORDER_ID, BOD_ISBN,
  BOD_CUSTOMER_ID, BOD_STORE_ID, BOD_DATE, BOD_TSTAMP, BOD_QUANTITY,
  BOD_PRICE_PER_ITEM, BOD_SEQ_NUM)
  SELECT Q1035.ORDER_ID AS ORDER_ID, Q1045.ISBN AS ISBN, Q1035.CUSTOMER_ID AS
CUSTOMER_ID,
Q1035.STORE_ID AS STORE_ID, DATE(Q1035.TSTAMP) AS TSTAMP, Q1035.TSTAMP AS
TSTAMP_1,
  Q1045.QUANTITY AS QUANTITY, Q1045.PRICE_PER_ITEM AS PRICE_PER_ITEM,
  Q1045.SEQ_NUM AS SEQ_NUM
    FROM PH3OLTP.BOOK_ORDER AS Q1035, PH3OLTP.ORDER_DETAIL AS Q1045
    WHERE (Q1035.ORDER_ID = Q1045.ORDER_ID)
    AND (Q1035.TSTAMP >= TIMESTAMP('1999-1-1-00.00.00.000000'))
    AND (Q1035.TSTAMP <= TIMESTAMP('1999-1-1-00.59.59.999999'));
```

Note that a partition clause is also available for the LOCK TABLE statement to lock only a single partition, which can dramatically reduce the amount of data locked.

Attention: A large number of rows being affected by a DML statement can result in monopolization of your group buffer pool when inter-DB2 read/write interest is detected for the target object. If it can be guaranteed that your data warehouse and BI queries are executed on the same member as the refresh process, this may not become an issue. On the other hand, if more than one member is dedicated to data warehousing workloads, the group buffer pool can become a point of concern.

To avoid the inter-DB2 read write interest, you should make sure that the unit of work populating your data warehouse tables is issued on the DB2 for z/OS members dedicated to data warehouse workloads. We do not recommend the usage of the NOSHARE attribute of a table space to restrict data access to a single member, although it would be one option to avoid group buffer pool dependencies in data sharing environments where only one member is dedicated to data warehousing.

See Table 7-1 on page 145 for a brief description when inter-DB2 read/write interest is detected.

Table 7-1 Determining group buffer pool dependency

One member's interest	Other member's interest	GBp dependent
Read-only	None, Read-only	No
Read-only	Read/Write	Yes
Read/Write	None	No (1)
Read/Write	Read-only	Yes
Read/Write	Read/Write	Yes

Exception: The page set remains GBP-dependent for some time before DB2 removes the dependency. DB2 might not be able to remove the GBP dependency if applications update the page set without issuing periodic commits.

Refreshing data warehouse data using SQL statements that access data in both OLTP and data warehouse environments can simplify the refresh processing, but it can also come with flavors of unwanted side effects. While using one unit of recovery for small amounts of data can still be attractive, we do not recommend using the same approach for large amounts of data.

7.2.5 Other possible refresh scenarios specific to data sharing environments

To overcome the situation we observed in 7.2.4, "InfoSphere Warehouse refresh scenario in a data sharing environment" on page 142, we provide some ideas in the following sections on how a refresh process can benefit from a data sharing implementation without the side effects we mentioned.

Application logic

One of the great benefits of having both OLTP and data warehouse members within one data sharing group is the ability of your application programs to access tables in both worlds inside the same application with only one database connection. Especially if you are running your applications natively on z/OS, for example applications written in COBOL or PL/I, this can be beneficial in terms of both application coding and runtime improvements.

Many installations exploit their own COBOL or PL/I applications to move and aggregate data into their data warehouse subsystem. This can either be part of an ETL process or coded inside the transactional logic itself. Regarding any ETL or ELT processing, combining the multirow operations that have been introduced in DB2 for z/OS Version 8, multiple rows can be read from the source system before inserting them again into the target table of your data warehouse, thereby reducing the number of round trips to DB2, leading to another reduction of elapsed time. The usage of multirow processing is highly recommended for applications that move data to your data warehouse tables. For more details on multirow processing, see *Application Design for High Performance and Availability,* SG24-7134.

Note: When scheduling your batch refresh processes, make sure that they are scheduled using your data warehouse DB2 for z/OS subsystem ID to direct any data warehouse workload to a designated member.

Triggers

Triggers are another mechanism to implement incremental updates in a data warehouse environment if more than one DB2 for z/OS subsystem operates in a data sharing group

where one or more members are dedicated to OLTP workloads while one or more members are dedicated to data warehousing at the same time.

This gives you the ability to define triggers on your transactional data to allow your OLTP transactions to update your data warehouse data incrementally whenever your transactional data changes. This method should only be used if data in your data warehouse can efficiently be accessed using index or unique index access without significantly increasing the response time for the transaction. By implementing triggers for this purpose, you can also achieve in-database transformation before data is applied to your data warehouse.

For more complex data transformations, calling a stored procedure from a trigger can also be an option. But due to the fact that each transaction increases in elapsed time if additional processing logic is added, either inside the application code or inside the DBMS, this approach is not recommended for time-critical applications. It might be worth considering if you rely on most current data in your data warehouse and when there is no other solution available. The recommended approach for running continuous updates driven by your OLTP data is the use of InfoSphere Change Data Capture (CDC). For more details about CDC, refer to 7.3, "Overview of InfoSphere Change Data Capture" on page 146.

Note: Using triggers to maintain data warehouse data needs to be evaluated carefully because triggers are fired synchronously, thereby increasing the elapsed time of your OLTP transaction. The invocation cost of the trigger is directly related to the type of trigger, row, or statement. It is recommend to apply this solution only if no solutions are available to perform these updates asynchronously, such as InfoSphere Change Data Capture.

7.3 Overview of InfoSphere Change Data Capture

In this section, we discuss InfoSphere Change Data Capture as a solution for enterprise data synchronization. We discuss the installation and configuration of InfoSphere Change Data Capture for z/OS. We also provide an explanation of key product parameters and their impact on performance.

This section deals with the following topics:

- "Introduction to InfoSphere Change Data Capture"
- "Configuration of InfoSphere Change Data Capture for z/OS"
- "Tuning InfoSphere Change Data Capture for z/OS"

7.3.1 Introduction to InfoSphere Change Data Capture

InfoSphere Change Data Capture is an enterprise data synchronization and change capture solution. InfoSphere Change Data Capture can be used to capture changes in near real-time as data is inserted, updated, and deleted in tables in a source database. These captured changes are then propagated to tables in target databases or used to drive other processes via messaging. The source and target databases could potentially be databases from different vendors running on different platforms.

The flow of data through Change Data Capture can be divided into three parts:

- Capture
- Transformations
- Apply

Capture

Changes are captured by asynchronously reading from the database recovery logs. Depending on the source database, InfoSphere Change Data Capture may read the log files directly or it may use an API to read data from the log. InfoSphere Change Data Capture has native log-based capture capabilities for the following databases:

- DB2 z/OS
- DB2 for IBM i
- DB2 Linux, Unix, Windows
- Oracle
- SQL Server
- Sybase

Upon reading data from the database recovery logs, InfoSphere Change Data Capture filters data based on the table where the change occurred. Only data pertaining to tables of interest is retained for further processing. InfoSphere Change Data Capture then stages the changes in a holding area until a commit of these changes occurs in the source database. If a rollback occurs instead, InfoSphere Change Data Capture discards the associated changes.

Transformations

In many cases, the data models of the source and target will not be the same. InfoSphere Change Data Capture can apply transformations to the data while it is in-flight between source and target. Some commonly used transformations are:

- Adding other information that can be obtained from the database recovery logs
- Concatenation and other string functions
- Data type conversion
- Arithmetic
- Joining to look up data in secondary tables
- If/then/else logic

For more complex transformations, a product such as InfoSphere DataStage can be used in conjunction with InfoSphere Change Data Capture.

Apply

After transformations, the changes are applied to the target database. Changes are applied by executing SQL statements against the appropriate target tables. InfoSphere Change Data Capture has native apply capabilities for the following databases:

- DB2 z/OS
- DB2 on IBM i
- DB2 Linux, Unix, Windows
- Oracle
- SQL Server
- Sybase
- Teradata

Archived

InfoSphere Change Data Capture also provides some level of customization around the SQL statements that are executed as part of apply processing. For example, it is possible to:

- Ignore deletes that occur at the source
- Convert source updates and deletes into target inserts (an audit or change history table)
- Convert source inserts into target updates (multiple source records merge to one target record)

InfoSphere Change Data Capture also supports some non-database targets, such as:

- Flat files
- WebSphere MQ
- JMS messaging

Access Server

Given that InfoSphere Change Data Capture is an enterprise-wide data synchronization solution, it may be in use on many physical machines across the enterprise. Some databases may be a source only, some databases may be a target only, and some databases may act as both a source and a target.

The administration server component of InfoSphere Change Data Capture is known as Access Server. Access Server maintains metadata describing how InfoSphere Change Data Capture is deployed across the enterprise. Only one Access Server needs to be installed in the enterprise, although multiple Access Servers may be used to provide a higher level of availability. There are three types of information that Access Server maintains:

- Information about every CDC agent
- Information about every CDC user ID
- Information about the mappings between CDC agents and CDC user IDs

A CDC agent is an instance of InfoSphere Change Data Capture that is installed and configured against a specific database instance. Every database that will act as either a source or a target for InfoSphere Change Data Capture needs an associated CDC agent. A single CDC agent is capable of acting as both a source and a target. During data synchronization, the source agent and the target agent communicate directly via TCP/IP.

A CDC user ID is a login name and password that is defined for the purpose of interacting with CDC across the enterprise. CDC user IDs map to operating system user IDs on the machines with databases that will be used as sources and targets for InfoSphere Change Data Capture. Various levels of CDC privileges are associated with each CDC user ID.

For each CDC agent that a CDC user ID needs access to, an agent or user ID mapping must be created. The mapping specifies that a specific CDC user ID is allowed to interact with the particular CDC agent. Additionally, it allows you to specify which user ID and password should be used to communicate with the database that the agent is installed against.

Management Console

Management Console is the user client interface for InfoSphere Change Data Capture. Management Console is used for several tasks:

- Create, modify, or delete CDC agent definitions
- Create, modify, or delete CDC user ID definitions
- Create, modify, or delete agent/user ID mappings
- Map source tables to target tables

- Start and stop data synchronization
- Monitor of InfoSphere Change Data Capture operations

To use Management Console, a user connects to an Access Server with a CDC user ID. The relationships between Management Console, Access Server, and CDC agents are summarized in Figure 7-20.

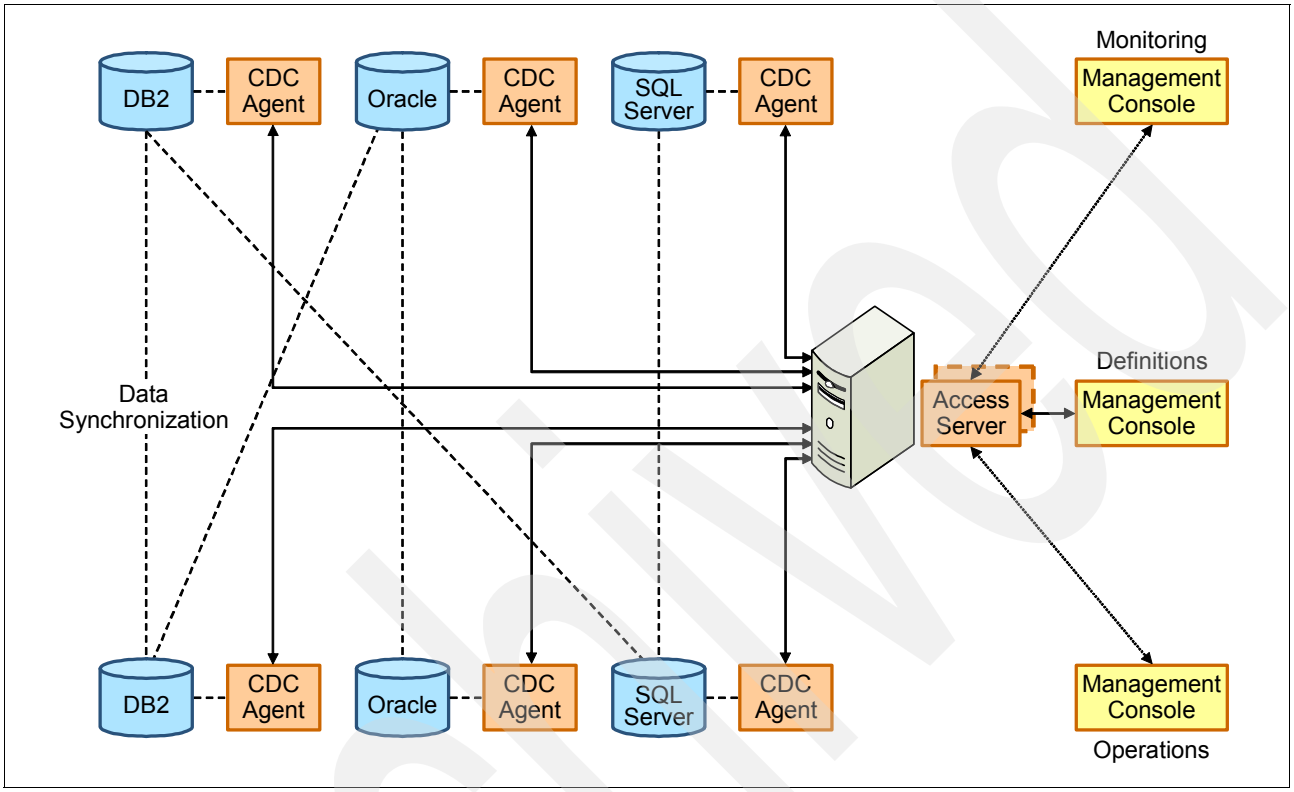


Figure 7-20 InfoSphere Change Data Capture enterprise architecture

7.3.2 Configuration of InfoSphere Change Data Capture for z/OS

In this section, we explain how we configured the CDC agent for DB2 z/OS. This section assumes that the product has already been SMP/E installed. Refer to the program directory, GI11-7880, for more information about the SMP/E installation process. Our initial configuration was created for product verification. In the last section of this chapter, we discuss performance tuning of product parameters.

InfoSphere Change Data Capture target libraries

In our installation, the target libraries were all under a high-level qualifier of ICDC.ZDW. The SMP/E installation gave us the following target libraries:

ICDC.ZDW.SCHCASM	Contains a sample assembler user exit.
ICDC.ZDW.SCHCC	Contains several sample C user exits.
ICDC.ZDW.SCHCCNTL	Contains sample JCL.
ICDC.ZDW.SCHCCOB	Contains several sample COBOL user exits.
ICDC.ZDW.SCHCDATA	Contains CDC agent configuration settings.
ICDC.ZDW.SCHCDBRM	Contains DBRMs used by the CDC agent for DB2 z/OS.

ICDC.ZDW.SCHCH	Contains C header files that are used when building C user exits.
ICDC.ZDW.SCHCLOAD	Contains the load modules for the CDC agent for DB2 z/OS. This library needs to be APF authorized.
ICDC.ZDW.SCHCMAC	Contains assembler macros that are used when building assembler user exits.
ICDC.ZDW.SCHCNOTC	Contains license information.
ICDC.ZDW.SCHCTTL	Contains data that can be loaded into sample tables to test the product installation.

The first step is to make a copy of SCHCCNTL and SCHCDATA. A copy of SCHCCNTL should be made for each CDC agent that will be configured. One CDC agent is needed for each DB2 for z/OS subsystem that will act as either a source or target for data synchronization. A single agent is capable of acting as both a source and a target. In the case of a DB2 for z/OS data sharing group, only one CDC agent is needed for the whole group. The SCHCDATA library can hold multiple configurations, so it is possible to use one copy to hold configurations for multiple CDC agents. In our environment, we created a separate copy of SCHCDATA for each CDC agent. Additionally, we allocated a new load library to hold CDC user exits. This gave us three new libraries:

- ICDC.ZDW.DDW1.SCHCCNTL
- ICDC.ZDW.DDW1.SCHCDATA
- ICDC.ZDW.SCHCEXIT

Define the product administration log

The product administration log (PAL) is used to hold CDC agent log data. This log data is viewable through the Management Console event log viewer. To define the PAL VSAM cluster, edit and run member CHCDFPAL in the SCHCCNTL sample library, as shown in Example 7-4. You can customize the DSN and VOLSER for the PAL cluster.

Example 7-4 Defining the product administration log

```
//CHCDFPAL JOB MSGLEVEL=(1,1),CLASS=A,MSGCLASS=X,REGION=0M,
//            TIME=300,NOTIFY=&SYSUID
//*
//VSAMDATA EXEC PGM=IEBDG
//SYSIN    DD *
         DSD      OUTPUT=(VSAMINIT)
         FD       NAME=KEYS,STARTLOC=1,LENGTH=32,FILL=X'FF'
         CREATE   NAME=(KEYS),QUANTITY=1
/*
//SYSPRINT DD SYSOUT=*
//VSAMINIT DD DSNAME=&&INITDATA,DISP=(,PASS),
//            UNIT=SYSALLDA,
//            SPACE=(TRK,(1,1)),
//            DCB=(RECFM=F,LRECL=36)
//*
//DEFLDPAL EXEC PGM=IDCAMS,
//            COND=(0,NE)
//SYSIN    DD *
  DELETE /* SO THAT THIS JOB CAN BE RERUN WITHOUT CHANGES */ -
    ICDC.ZDW.DDW1.PAL -
    CLUSTER -
    PURGE
  SET MAXCC = 0
```

```
  SET LASTCC = 0

  DEFINE -
    CLUSTER -
     (NAME(ICDC.ZDW.DDW1.PAL) -
      VOL(DWCDC1) -
      SHAREOPTIONS(2 3) -
      TRK(4 4)  -
      RECSZ(36 1024) -
      INDEXED -
      NOREUSE -
      KEYS(32 0)) -
    DATA -
     (NAME(ICDC.ZDW.DDW1.PAL.DATA)) -
    INDEX -
     (NAME(ICDC.ZDW.DDW1.PAL.INDEX))

  REPRO -
    INFILE(CHCPALDT) -
    OUTDATASET(ICDC.ZDW.DDW1.PAL)
/*
//SYSPRINT DD SYSOUT=*
//CHCPALDT DD DSNAME=&&INITDATA,DISP=(OLD,DELETE)
//*
//
```

Define the VSAM metadata cluster

Metadata describing the data synchronization definitions, the mappings from source tables to target tables, and alerts and alarms settings is stored in a metadata repository at the CDC agent level. Most metadata is stored in DB2 tables, but the alerts and alarms settings are stored in a VSAM cluster. Member CHCDFMTD in the SCHCCNTL library is used to define this VSAM metadata cluster; see Example 7-5. You can customize the DSN and VOLSER for the metadata cluster.

Example 7-5 Defining the metadata cluster

```
//CHCDFMTD JOB MSGLEVEL=(1,1),CLASS=A,MSGCLASS=X,REGION=0M,
//            TIME=300,NOTIFY=&SYSUID
//*
//FRMTDATA EXEC PGM=IEBDG
//SYSIN    DD *
         DSD      INPUT=(NLDMTDTA),OUTPUT=(VSAMINIT)
         FD       NAME=KEYS,INPUT=NLDMTDTA,STARTLOC=17,LENGTH=64,      X
               FROMLOC=1
         FD       NAME=DATA,INPUT=NLDMTDTA,STARTLOC=145,LENGTH=8,      X
               FROMLOC=65
         FD       NAME=FLTRFL,STARTLOC=65,LENGTH=1332
         FD       NAME=CHNLFL,STARTLOC=153,LENGTH=505
         FD       NAME=CTRLFL,STARTLOC=1,LENGTH=16
         FD       NAME=HIKYHI,STARTLOC=17,LENGTH=127,FILL=X'FF'
         FD       NAME=HIKYL1,STARTLOC=144,LENGTH=1,FILL=X'FE'
         FD       NAME=HIKYL2,STARTLOC=144,LENGTH=1,FILL=X'FF'
         FD       NAME=DATAFL,STARTLOC=145,LENGTH=1900
         CREATE   INPUT=NLDMTDTA,NAME=(KEYS,FLTRFL),QUANTITY=1
         CREATE   INPUT=NLDMTDTA,NAME=(KEYS,DATA,CHNLFL),QUANTITY=105
```

Archived

```
         CREATE   INPUT=NLDMTDTA,NAME=(KEYS,FLTRFL),QUANTITY=1
         CREATE   INPUT=NLDMTDTA,NAME=(KEYS,DATA,CHNLFL),QUANTITY=105
         CREATE   NAME=(CTRLFL,HIKYHI,HIKYL1,DATAFL),QUANTITY=1
         CREATE   NAME=(CTRLFL,HIKYHI,HIKYL2,DATAFL),QUANTITY=1
         END
/*
//SYSPRINT DD SYSOUT=*
//NLDMTDTA DD DSNAME=ICDC.ZDW.SCHCDATA(CHCMTDIN),DISP=SHR
//VSAMINIT DD DSNAME=&&INITDATA,DISP=(,PASS),
//            UNIT=SYSALLDA,
//            SPACE=(TRK,(1,1)),
//            DCB=(RECFM=VB,BLKSIZE=4096,LRECL=2048)
//*
//BLDMTDTA EXEC PGM=IDCAMS
//SYSIN    DD  *
  DELETE /* SO THAT THIS JOB CAN BE RERUN WITHOUT CHANGES */ -
    ICDC.ZDW.DDW1.META -
    CLUSTER -
    PURGE
  SET MAXCC = 0
  SET LASTCC = 0

  DEFINE -
    CLUSTER -
     (NAME(ICDC.ZDW.DDW1.META) -
      VOL(DWCDC1) -
      SHAREOPTIONS(2 3) -
      CYL(20 20) -
      RECSZ(512 2048) -
      CISZ(4096) -
      INDEXED -
      NOREUSE -
      KEYS(128 16)) -
    DATA -
     (NAME(ICDC.ZDW.DDW1.META.DATA)) -
    INDEX -
     (NAME(ICDC.ZDW.DDW1.META.INDEX))

 REPRO -
   INFILE(VSAMINIT) -
   OUTDATASET(ICDC.ZDW.DDW1.META)
/*
//SYSPRINT DD SYSOUT=*
//VSAMINIT DD DSNAME=&&INITDATA,DISP=(OLD,DELETE)
//*
//
```

DB2 metadata tables

Prior to creating the DB2 metadata tables, you need to create a z/OS user ID that the CDC agent started task will run under. It is necessary to pick a name for this user ID at this point due to the fact that the DB2 metadata tables must have a schema (owner) name that matches the user ID that the started task will run under. When defining this user ID, make sure to define an OMVS segment.

Archived

The user ID that the CDC agent started task runs under needs to have SYSCTRL privilege in DB2 z/OS. Member CHCGRNTA in the sample library can be used to grant this privilege.

Sample job CHCCRMTD in SCHCCNTL is used to create the DB2 metadata tables. Follow the instructions that are contained within the job. There are some of important considerations for this job:

- For DB2 z/OS V8 New Function Mode and later releases of DB2, the symbol V7CATLG must be set to FALSE. The actual DDL that executes is dependent upon the value of this symbol. In our installation, we set V7CATLG=FALSE, because we were using a DB2 z/OS V9 subsystem.
- The instructions in the job list three table space name parameters which need to be set. In reality, only two table spaces will be created. If V7CATLG=FLASE, CHCTBSP2 and CHCTBSP3 will be used. If V7CATLG=TRUE, CHCTBSP1 and CHCTBSP3 will be used.
- If you already have a predefined STOGROUP that you wish to use, delete the CREATE STOGROUP statement from the job.
- The last job step, BLDINDEX, only needs to be run once per DB2 z/OS subsystem or data sharing group. If you later configure a second CDC agent against the same DB2 subsystem or group, you do not need to rerun this job step.

The DDL that we used to create the DB2 metadata tables is listed in Appendix A, “InfoSphere Change Data Capture DDL and binds” on page 387.

Bind DB2 plans

Sample job CHCBNDPL can be used to bind the plans needed by the DB2 z/OS CDC agent. When editing this job, keep the following items in mind:

- The parameter <PlanSuffix> needs to be replaced with a 2-character suffix, which becomes a setting in the SCHCDATA configuration.
- When doing an initial configuration, you will not have any user exit DBRMs that need to be included in any of the plans.
- The parameter <CHCuser ID> needs to be replaced with the z/OS user ID that was created in the previous step.

The job that was used to bind the DB2 plans is listed in Appendix A, “InfoSphere Change Data Capture DDL and binds” on page 387.

SCHCDATA

The SCHCDATA library contains configuration settings for the CDC agents. A single SCHCDATA library can contain multiple sets of configurations. Each configuration member is suffixed with a 2-character value. The JCL for the CDC agent started task contains a parameter that specifies the 2-character suffix that should be used when reading SCHCDATA members. For the initial configuration, we made a copy of all of the members, using a suffix of F1, indicating a functional test configuration. Our initial configuration was as follows:

CHCCFGF1

The CHCCFGxx configuration members specify general configuration settings. These settings must start with the CONFIG keyword. Our initial CHCCFGxx configuration is shown in Example 7-6.

Example 7-6 CHCCFG initial configuration

```
* Comments may be coded "c" style by entering an asterisk in the first
* column, or by bracketing the comment with matching pairs of
* slash-asterisk and asterisk-slash (e.g.  /* comment  */).
```

```
*
* Any data entered past column 71 will be ignored.
* Do not type anything to the right of this point ------------------->
*
* Continuation is indicated by a coding a comma as the last non-blank
* character on a line (not including comments).
*
CONFIG  REPSTATSINTERVAL=5
```

The default value for REPSTATSINTERVAL is 0. By setting REPSTATSINTERVAL to 5, messages will be generated in the JES job log every 5 minutes, indicating the position of the DB2 log reader process. This parameter can be helpful during initial installation verification for CDC agents that will act as sources because it allows you to track the progress of the CDC agent as it reads through the DB2 log. Example 7-7 shows an example of a message generated by enabling REPSTATSINTERVAL.

Example 7-7 REPSTATSINTERVAL log reader position message

```
Latest scraped data was written at 2009-07-09-17.15.52, sent 1 changes, current
Log Position is X'0000001485813133'
```

CHCCMMF1

The CHCCMMxx configuration members specify configuration settings related to TCP/IP communications. These settings must start with the TCP/IP keyword. Our initial CHCCMMxx configuration is shown in Example 7-8.

Example 7-8 CHCCMM initial configuration

```
TCP/IP SERVICENAME=11001
```

SERVICENAME is a required parameter that specifies either a symbolic service name that has been defined to the communications server or a numeric port number. This defines the TCP/IP port that the CDC agent will listen on.

CHCDBMF1

The CHCDBMxx configuration members specify configuration settings related to DB2. The CHCDBMxx settings must start with the DB2 keyword. Our initial CHCDBMxx configuration is shown in Example 7-9.

Example 7-9 CHCDBM initial configuration

```
DB2 SSID=DDW1,
    PLANSUFFIX=01,
    SQLWHEREROWSEL=YES,
    LOGPOLLINTERVAL=(5,ALWAYS)
```

SSID is a required parameter that specifies the name of the DB2 subsystem or data sharing group that the CDC agent is to attach to. If InfoSphere Change Data Capture is being installed against a DB2 data sharing group, this should be set to the group name, so that the CDC agent is capable of attaching to any of the group members.

PLANSUFFIX is an optional parameter. It should be set to the plan suffix that was used when running CHCBNDPL. If not specified, PLANSUFFIX defaults to the xx suffix of the CHCDBMxx member in use.

SQLWHEREROWSEL is an optional parameter. If set to YES, InfoSphere Change Data Capture can use WHERE clauses to limit the data that is SELECTed from source tables during initial loads or refreshes of target tables. The default value for this parameter is NO.

LOGPOLLINTERVAL is a very significant parameter, which we explain in detail in the section about tuning InfoSphere Change Data Capture. The recommendation for initial configuration is to set LOGPOLLINTERVAL to (5,ALWAYS) in non-data sharing, or 5 in data sharing for agents that will act as sources.

CHCLDRF1

The CHCLDRxx configuration members specify configuration settings related to using the DB2 LOAD utility for refresh operation. By default, this feature is disabled, and it is not necessary to edit the contents of this configuration member during initial configuration. We will discuss the contents of this member as part of the discussion of tuning InfoSphere Change Data Capture.

CHCUCSF1

The CHCUCSxx configuration members specify configuration settings related to Unicode Conversion Services (UCS). If the settings in this member are not configured, the CDC agent will revert to using Language Environment (LE) to do codepage conversions. In certain cases, LE cannot do the necessary conversions and a failure occurs. Because of this, and because UCS provides a more efficient means of doing codepage conversion, we configured CHCUCSxx as part of our initial configuration.

To configure CHCUCSxx, you will need to add the appropriate CONVERSION statements. The first line of CHCUCSXX shows a dummy CONVERSION statement, shown in Example 7-10.

Example 7-10 Dummy CHCUCS CONVERSION statement

```
CONVERSION SRCCCSID=<SRCCCSID>,TGTCCSID=<TGTCCSID>,METHOD=<METHOD>
```

Three pieces of information are needed for every CONVERSION statement: the source CCSID, the target CCSID, and the conversion method. You must add a CONVERSION statement to CHCUCSxx for every codepage conversion that you want the CDC agent to be able to perform using Unicode Conversion Services. If the CDC agent needs to perform a codepage conversion and the necessary CHCUCSxx CONVERSION statement is not in place, it will attempt to use LE to do the conversion. The recommendation is to add CONVERSION statements for codepage conversions that you expect to take place. From there, monitor the JES job log or the event log viewer in Management Console for messages indicating that a codepage conversion used LE, as shown in Example 7-11.

Example 7-11 Missing CHCUCSxx CONVERSION statement

```
CHC1514W Language Environment's Code Page Conversion Services will be used.
```

The most common codepages required for codepage conversion are:

- 37 (EBCDIC)
- 1047 (EBCDIC)
- 1208 (UTF-8)
- 1200 (UTF-16)
- 1252(ASCII)

You should not add CONVERSION statements to CHCUCSxx unless the corresponding conversions are configured in Unicode Conversion Services. To determine which codepage conversions are available on your system, execute the z/OS console command D UNI,ALL, shown in Example 7-12.

Example 7-12 Output of D UNI,ALL

```
RESPONSE=P61
 CUN3000I 11.46.49 UNI DISPLAY 216
  ENVIRONMENT: CREATED       07/24/2009 AT 09.44.36
               MODIFIED      07/24/2009 AT 11.05.53
               IMAGE CREATED --/--/---- AT --.--.--
      SERVICE: CHARACTER      CASE           NORMALIZATION  COLLATION
               STRINGPREP     BIDI           CONVERSION INF
      STORAGE: ACTIVE        413 PAGES
               FIXED           0 PAGES
               LIMIT       12800 PAGES
     CASECONV: ENABLED
     CASE VER: UNI300  NORMAL
    NORMALIZE: DISABLED
     NORM VER: NONE
      COLLATE: DISABLED
   COLL RULES: NONE
  STRPROFILES: NONE
   CONVERSION: 00932-01200(13488)-RECL     00300-01200(13488)-RECL
               01200(13488)-00037-E        01200(13488)-00300-RECL
               01047-01200(13488)-RECL     01208-00037-E
               01200(13488)-00932-RECL     01200(13488)-00939-RECL
               01200(13488)-01047-RECL     01200(13488)-01208-RECL
               01200(13488)-01383-RECL     00037-00367-E
               00037-01208-R               01252-01208-R
               00367-00037-E               01208-01200(13488)-RECL
               00500-00850-RECL            01383-01200(13488)-RECL
               00037-01200(13488)-R        00819-01047-L
               00819-01047-R               00850-01047-
               00939-01200(13488)-RECL     01047-00819-L
               01047-00819-R               01047-00850-
               01047-00850-L               01047-00850-C
```

The letters at the end of each conversion listed indicate the conversion methods that are available. Each available method is represented by a single character. A concatenation of multiple characters indicates that multiple conversion methods are available. For each CONVERSION statement in CHCUCSxx, you must specify a METHOD, and that METHOD must be listed on the corresponding conversion in the output of D UNI,ALL. For an explanation of the various conversion methods, refer to *z/OS Support for Unicode: Unicode Services*, SA22-7649.

We used the CONVERSION statements in CHCUCSF1 shown in Example 7-13. We used the default CCSIDMAP and IANAMAP parameters that come in CHCUCSXX, the generic CHCUCSxx configuration member.

Example 7-13 CHCUCS CONVERSION statements

```
CONVERSION SRCCCSID=1047,TGTCCSID=1200,METHOD=L
CONVERSION SRCCCSID=1200,TGTCCSID=1047,METHOD=E
CONVERSION SRCCCSID=1208,TGTCCSID=1200,METHOD=R
```

Archived

```
CONVERSION SRCCCSID=1200,TGTCCSID=1208,METHOD=R
CONVERSION SRCCCSID=1047,TGTCCSID=1208,METHOD=R
CONVERSION SRCCCSID=1208,TGTCCSID=1047,METHOD=E
CONVERSION SRCCCSID=37,TGTCCSID=1047,METHOD=R
CONVERSION SRCCCSID=1047,TGTCCSID=37,METHOD=R
CONVERSION SRCCCSID=37,TGTCCSID=1200,METHOD=R
CONVERSION SRCCCSID=1200,TGTCCSID=37,METHOD=E
CONVERSION SRCCCSID=37,TGTCCSID=1208,METHOD=R
CONVERSION SRCCCSID=1208,TGTCCSID=37,METHOD=E
CONVERSION SRCCCSID=1252,TGTCCSID=1208,METHOD=R
```

By default, InfoSphere Change Data Capture will attempt to use the source CDC agent to do any necessary codepage conversions. If the source CDC agent is unable to do the requested conversion, the codepage conversion will be attempted on the target CDC agent. The parameter CODEPAGECONVERSION=TARGET can be added to CHCCFGxx to force InfoSphere Change Data Capture to first check whether the target CDC agent can do the codepage conversion. In this case, the source CDC agent will do the codepage conversion only if the target CDC agent is incapable of performing the required conversion.

The started task

A sample started task is provided in member CHCPROC in the SCHCCNTL library. Some things should be kept in mind when editing CHCPROC:

- Replace <ConfigSuffix> with the xx suffix that matches the SCHCDATA configuration that you wish the started task to use.
- The IEFRDER DD should point to the SCHCDATA library containing the appropriate configuration members.
- The data set referred to by the SYSMDUMP DD in the DELETE and IEFPROC job steps cannot be shared between multiple InfoSphere Change Data Capture started tasks.
- The CHCMTDTA DD must point to the VSAM metadata cluster that was defined.
- The CHCPALOG DD must point to the PAL cluster that was defined.
- For the initial configuration, the CHCCACHE and CHCCHCTL DDs can be set to DUMMY. Just comment out the CHCCACHE and CHCCHCTL DDs provided in case the caching feature needs to be turned on at a later point in time.
- The CHCUXLIB DD should point to the user exit library that was allocated.
- Verify that the SYSTCPD DD is correct.

Our initial started task is shown in Example 7-14.

Example 7-14 Initial InfoSphere Change Data Capture started task

```
//CHCPROC1 PROC CONFIG=P1,
//            COMM=COMM,
//            DBMS=DBMS
//*
//DELETE   EXEC PGM=IEFBR14
//SYSMDUMP DD DSNAME=ICDC.ZDW.DDW1.SYSMDUMP,DISP=(MOD,DELETE),
//            UNIT=SYSALLDA,
//            SPACE=(CYL,(1))
//*
//IEFPROC  EXEC PGM=CHCMIT,
//            PARM=('/ENVAR(_ICONV_UCS2_PREFIX=CEE)',
//            '/&COMM,&DBMS,CONFIG=&CONFIG'),
```

```
//            REGION=256M,MEMLIMIT=3G
//STEPLIB  DD DSNAME=ICDC.ZDW.SCHCLOAD,DISP=SHR
//         DD DSNAME=DB2910.ZDW.SDSNEXIT,DISP=SHR
//         DD DSNAME=DB2910.ZDW.SDSNLOAD,DISP=SHR
//CHCCNTRL DD DDNAME=IEFRDER
//IEFRDER  DD DSNAME=ICDC.ZDW.DDW1.SCHCDATA,DISP=SHR
//CHCPRINT DD SYSOUT=*
//CHCAUDIT DD SYSOUT=*
//CHCREPRT DD SYSOUT=*
//CHCMTDTA DD DSNAME=ICDC.ZDW.DDW1.META,DISP=SHR
//CHCPALOG DD DSNAME=ICDC.ZDW.DDW1.PAL,DISP=SHR
//*CHCCACHE DD DSNAME=<CACHE.QUALIFIER>.CHCCACHE,DISP=SHR,
//*          AMP=('BUFND=256,ACCBIAS=SW')
//CHCCACHE DD DUMMY
//*CHCCHCTL DD DSNAME=<CACHE.QUALIFIER>.CHCCHCTL,DISP=SHR
//CHCCHCTL DD DUMMY
//CHCUXLIB DD DSNAME=ICDC.ZDW.SCHCEXIT,DISP=SHR
//SYSTCPD  DD DSNAME=TCPIP.TCPDATA(TCPDATA),DISP=SHR
//UTPRINT  DD DUMMY
//* SEE THE NOTE ABOVE ABOUT THE DISPOSITION CODED FOR THIS DATA SET  *
//SYSMDUMP DD DSNAME=ICDC.ZDW.DDW1.SYSMDUMP,DISP=(MOD,CATLG), *NOTE*
//            UNIT=SYSALLDA,
//            SPACE=(CYL,(150,50))
//ABNLIGNR DD DUMMY
//*
```

Once the necessary definitions are in place to ensure that the CDC started task will run under the appropriate user ID, it can be started.

Access Server and Management Console

At this point, the Access Server needs to be installed. Access Server supports the following 32-bit Windows operating systems:

- Windows 2000
- Windows Server 2003
- Windows XP Professional Software

Access Server supports the following UNIX and Linux operating systems:

- HP-UX Version 11.00 or higher (PA-RISC)
- AIX® 5L™ Version 5.1 or higher (POWER®)
- Sun Solaris Version 2.8 or higher (SPARC)
- Linux Kernel Version 2.4.x (x86 only) or 2.6.x (x86 or x64)

The Access Server requires two consecutive ports. By default, Access Server uses ports 10101 and 10102. When connecting to the Access Server with Management Console, the user always specifies the lower numbered port in the pair.

During the Access Server installation, also take note of the password that you give to the Admin user ID. When first connecting to Access Server with the Management Console client, it is required that you log on with this user ID and password.

Archived

Before running installation verification tests, the Management Console must be installed. Management Console supports the following 32-bit Windows operating systems:

- Windows 2000
- Windows Server 2003
- Windows XP Professional

For more information about installing Access Server and Management Console, refer to the Access Server and Management Console Installation Guide.

Access Server definitions

Next, we defined the CDC agents, CDC user IDs, and agent/user ID mappings to Access Server. This is done via the Management Console client. When connecting to the Access Server for the first time, use the Admin user ID and password that were specified during the Access Server installation. The server name refers to the machine on which Access Server is running. The port refers to the lower numbered of the two consecutive ports that were assigned to Access Server. The default value for this port is 10101.

CDC agents

The Access Manager tab is where all Access Server definitions are created and managed. To define a CDC agent, right-click in the datastore management area, and select **New Datastore**. For our initial tests we only had one CDC agent, because the same DB2 z/OS subsystem was being used as both the source and the target. Our CDC agent was defined as shown in Figure 7-21.

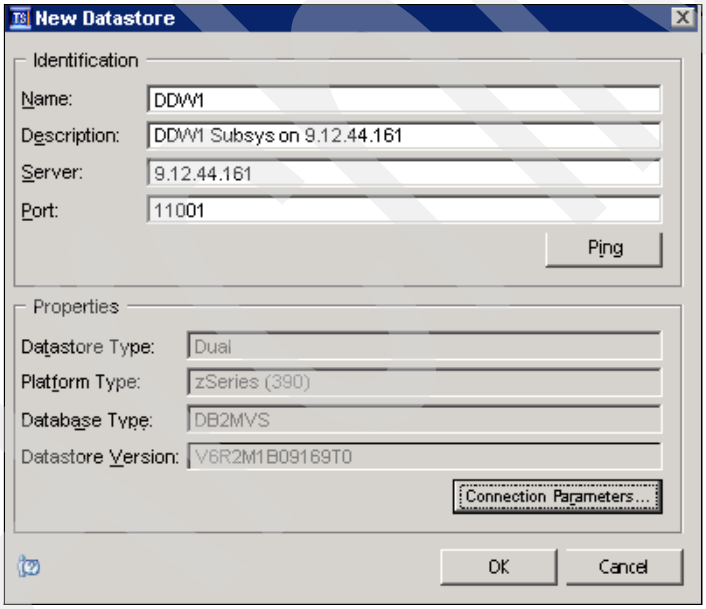


Figure 7-21 Defining a CDC agent

The port number should match the port specified in the CHCCMMxx configuration member. After filling in the name, description, server, and port fields, click **Ping** to test the connection. If the connection is successful, the information in the Properties fields will automatically populate.

Next, we defined a CDC user ID. To do this, right-click in the user management area, and select **New User**. Provide a user name and password. This password is used only for this

CDC user to connect to the Access Server via the Management Console. Figure 7-22 shows an example of creating a CDC user ID.

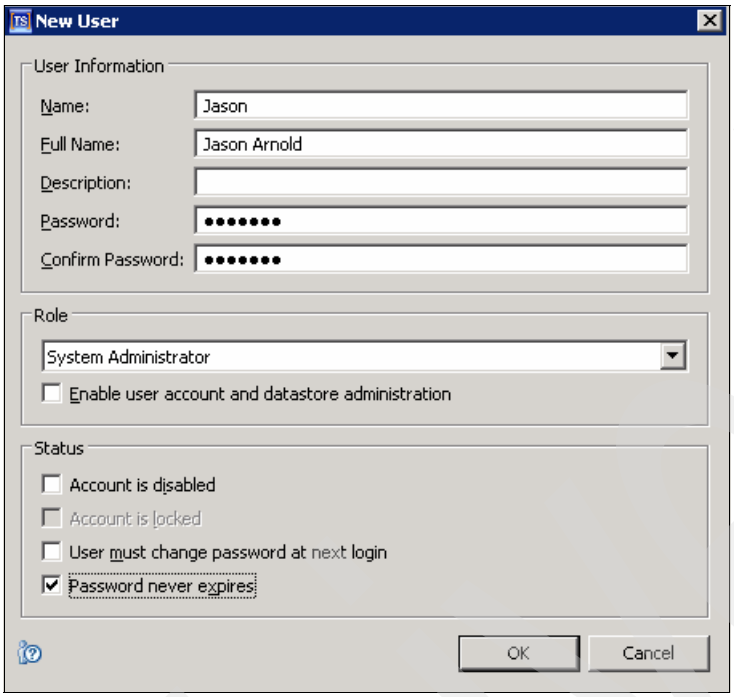


Figure 7-22 Creating a CDC user ID

CDC user IDs

There are several roles that can be assigned to the user ID. The role controls the authorities associated with the user ID.

System Administrator

Users assigned to this role can perform all available operations in the Management Console. Only super users should be assigned to this role since they will have full access to the monitoring and configuration perspectives in the Management Console, including the system parameters.

Administrator

Users assigned to this role can perform all available operations in the Management Console but cannot modify InfoSphere CDC system parameters. The ability to change system parameters through Management Console is applicable only to non-DB2 z/OS CDC agents.

Operator

Users assigned to this role only have access to the monitoring perspective in Management Console and have the following permissions:

- You can start/stop replication and monitor replication. You cannot configure replication.
- You can view the tables selected for refresh and start a refresh. You cannot select or remove tables for refresh.
- You have access to the topology view.
- You can view notifications for subscriptions and datastores.

Monitor

Users assigned to this role only have access to the monitoring perspective in Management Console and have the following permissions:

- You can view the event log, statistics, and table mappings.
- You can view replication state and status and latency threshold information.
- You have access to the topology view.
- You do not have the ability to start/stop replication, configure replication, refresh tables, or view notifications for subscriptions and datastores.

In addition to these roles, the “Enable user account and datastore administration” checkbox is used to give a CDC user ID the authority to create, edit, and delete Access Server definitions including CDC agents, CDC user IDs, and agent/user ID mappings.

Agent/user ID mappings

Before the new CDC user ID can work with the CDC agent, an agent/user ID mapping needs to be created. The agent/user ID mapping gives the CDC user ID access to the CDC agent, and also specifies the local OS user ID and password that should be used when the CDC user ID in question is communicating with the CDC agent.

First, select the CDC user ID in the user management area by clicking it. The mapping can then be created by right-clicking in the connection management area, and selecting **Assign Datastore**. Next, select the appropriate CDC agent. We created the mapping shown in Figure 7-23.

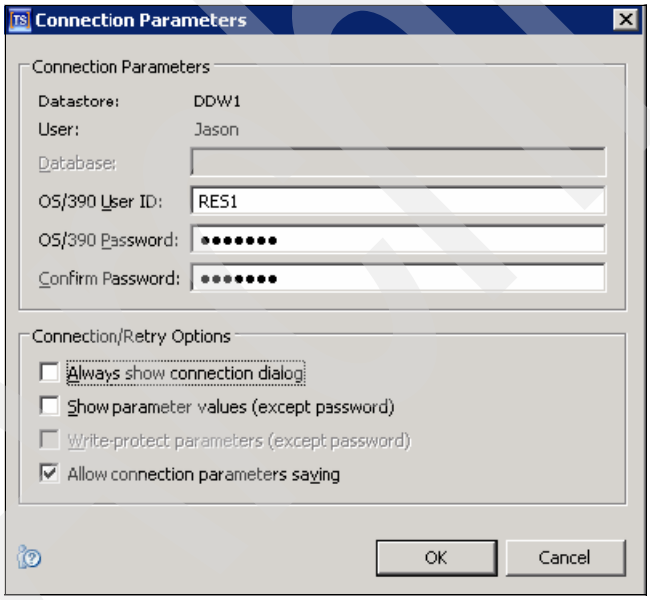


Figure 7-23 Creating an agent/user ID mapping

Verifying the InfoSphere Change Data Capture installation

To verify that all of the steps up to this point had been done successfully, we did a simple test using the DB2 sample tables. This type of test can be done with any table in your environment. The recommendation is that you do this initial test with a table with a very small number of rows so that data synchronization can be visually verified.

Creating the subscription

The Admin user ID was not given access to the DDW1 CDC agent, so the first step was to log out and log back in with the new CDC user ID. This can be done by selecting **File** → **Access Server** → **Disconnect** from the menu bar. After disconnecting, select the **Connect** option to reconnect as a different CDC user. The next step was to create a new subscription. All data synchronization definitions must belong to a subscription. A subscription corresponds to a thread at the CDC agent level. A subscription is also associated with a specific source agent and a specific target agent. Additionally, a single table can only be used as a source one time within any subscription.

To test functionality of the DDW1 CDC agent, we created a new subscription using DDW1 as both the source and target agent; see Figure 7-24. To create a new subscription, click on the configuration tab and the subscriptions subtab. Right-click in the subscriptions area and select **New Subscription**,

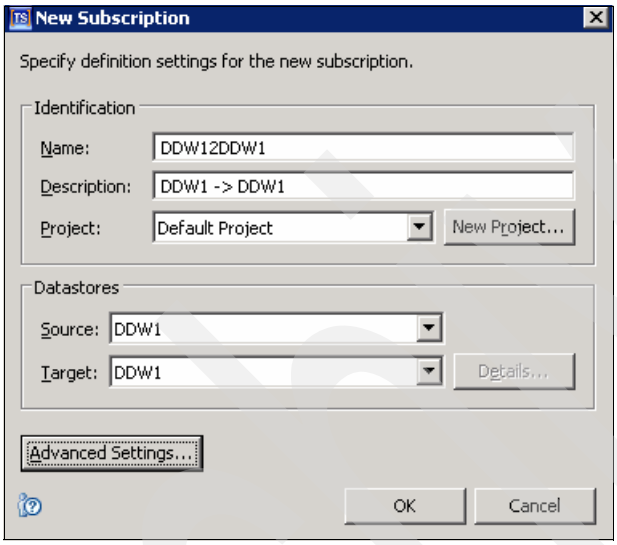


Figure 7-24 Creating a new subscription

Creating the table mapping

The next step is to create the table mapping. While Management Console can create the target table for you, it requires that the target table space already exists. For our purposes, we manually created the target table before creating the table mapping in Management Console. Initially, we did not have any data in the target table.

With the new subscription highlighted, right-click in the table mappings area and select **Map Tables**. On the next panel (Figure 7-25 on page 163), select **One table mapping of any type** and select a type of standard. A standard table mapping means that an insert at the source generates a corresponding insert at the target. An update at the source generates an update at the target. A delete at the source generates a delete at the target. The automatic mapping types (the first two radio buttons) are used for creating a large number of table mappings, whereas the custom mapping type option is used to create a single table mapping.

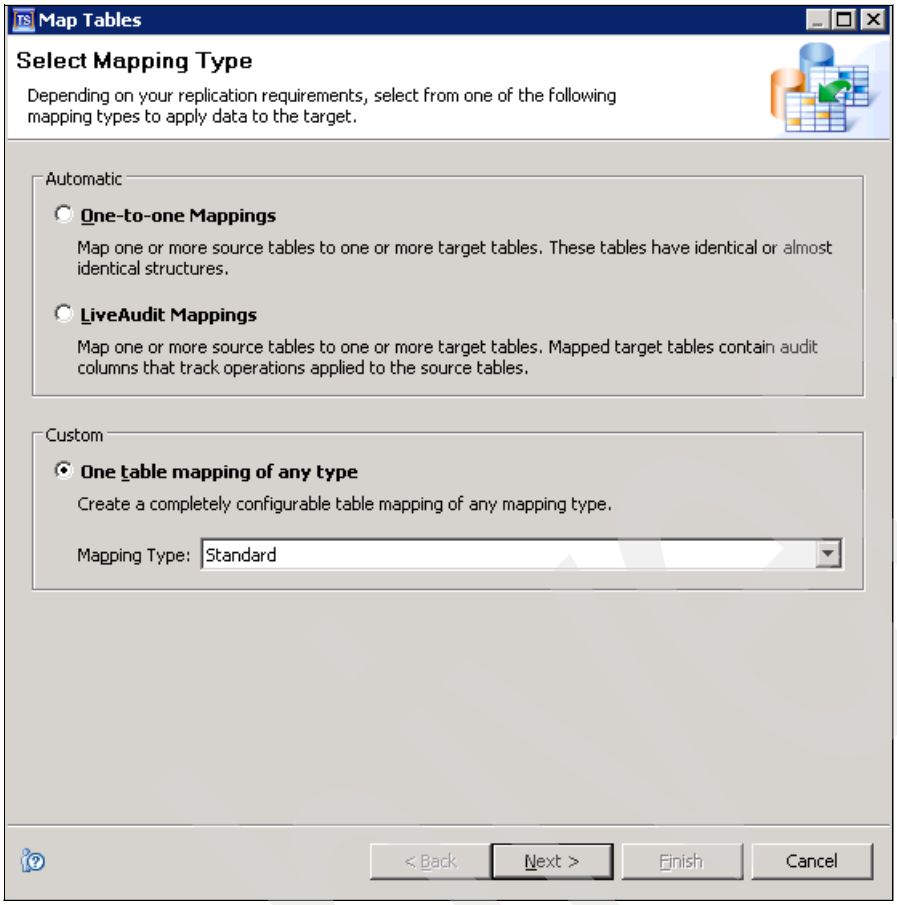


Figure 7-25 Creating a new table mapping

The next panel (Figure 7-26 on page 164) asks you to select the source table for data synchronization. The first level in the list represents database names. Expanding a database provides a list of schemas. Expanding a schema provides a list of tables.

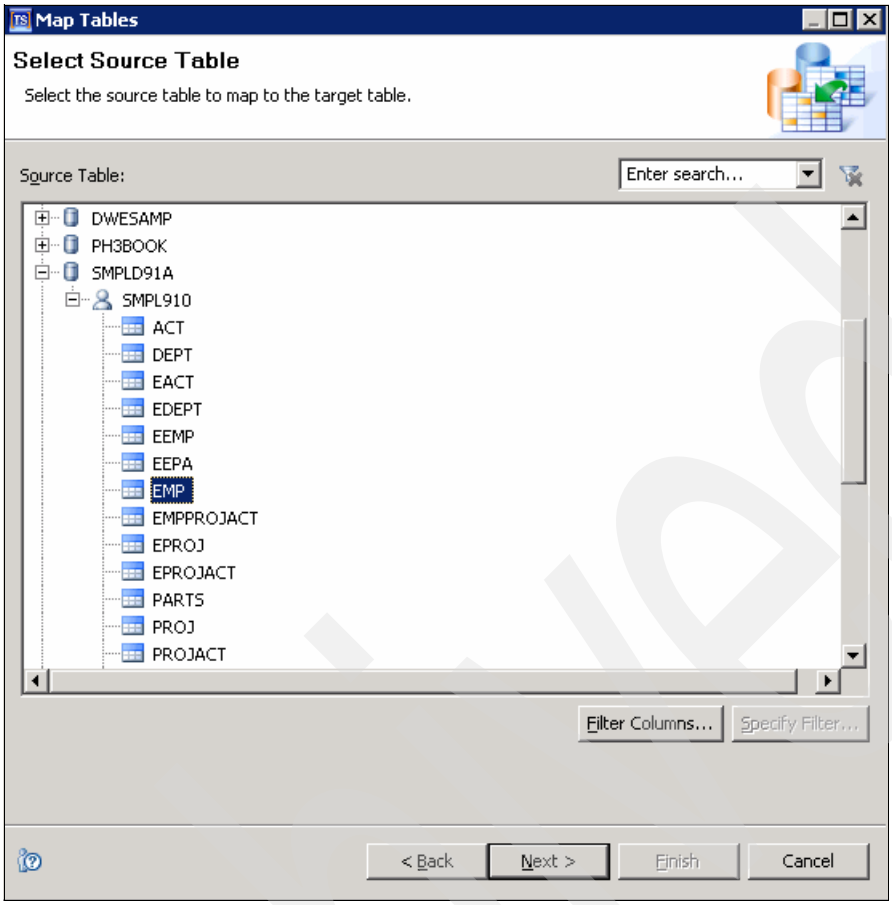


Figure 7-26 Selecting the source table

The next panel (Figure 7-27) asks you to select the target table in the same manner. If the target table does not already exist, it is necessary to click **Create Table** to create it.

Lastly, it is necessary to set the replication method. Refresh means that this table mapping can only be used to do initial loads and full refreshes. It cannot be used to do log-based change capture or near real-time synchronization. Mirror means that the table mapping can be used to do both initial loads and near real-time synchronization.

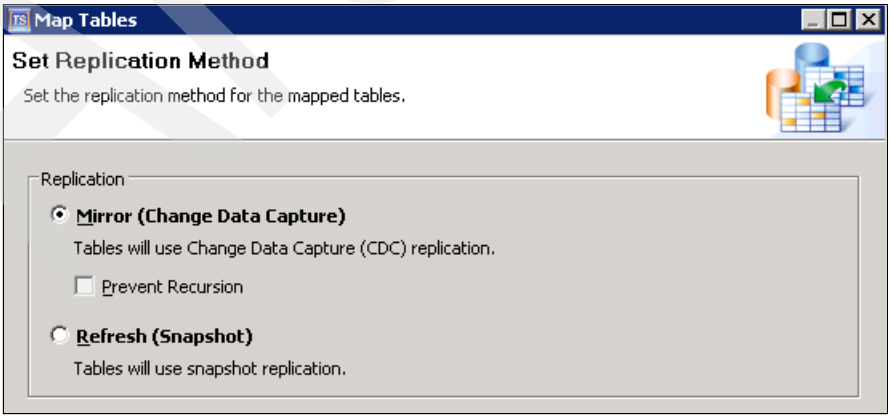


Figure 7-27 Set the replication method

Clicking **Finish** on the next panel causes the column mapping details to display (Figure 7-28). In the case of identical source and target tables, InfoSphere Change Data Capture automatically determines the appropriate column mappings.

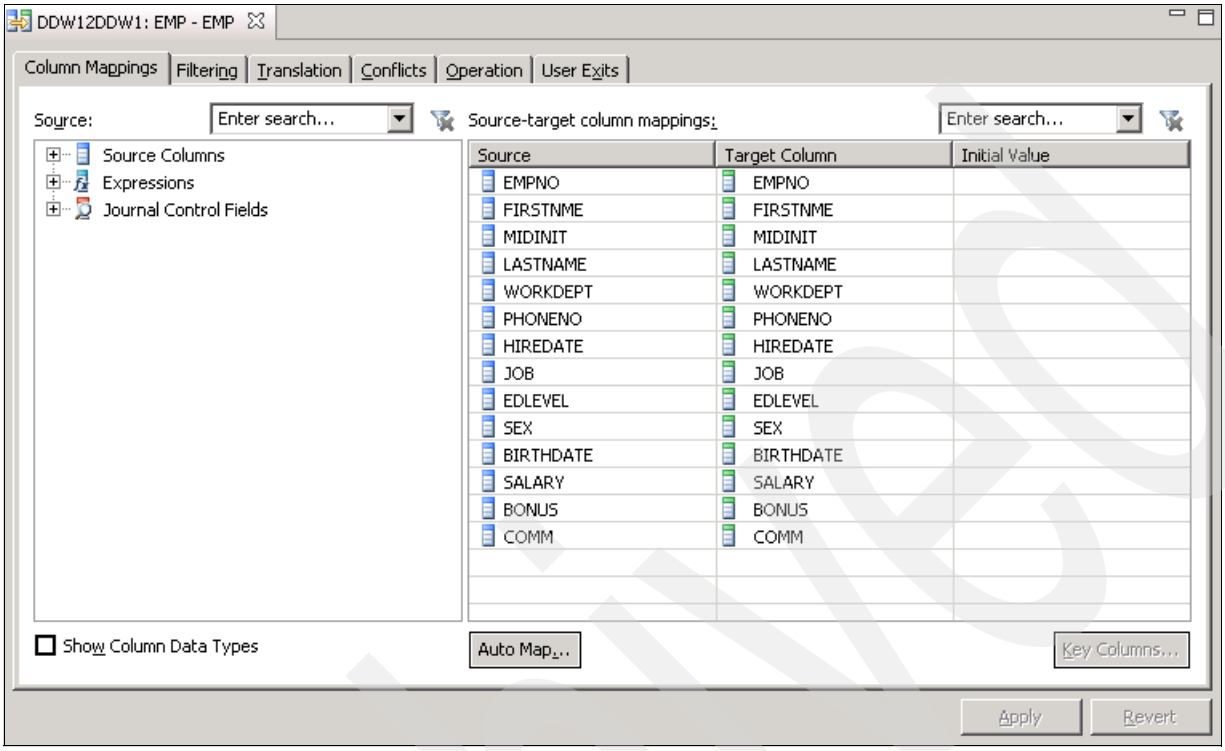


Figure 7-28 Source to target column mappings

Testing refresh

Under the table mappings tab, the new table mapping will show a status of refresh. The first test is to attempt an initial load of the target table. By default, InfoSphere Change Data Capture performs a refresh by reading all of the data from the source table and inserting it into the target table using SQL statements. As part of tuning, we will show how the DB2 LOAD utility can be invoked as an alternative to SQL statements to load data into the target table.

To test refresh of the subscription, right-click on the subscription under the subscriptions tab, and select **Start Mirroring (Continuous)**. Next, click the monitoring tab, and watch the status of the subscription. During refresh, the state will show as “Refresh before mirror”. When the refresh is complete, the state will change to “Mirror continuous”. If the subscription instead changes to an error status, right-click the subscription and select **Show Event Log** to view related error messages.

After the subscription has changed state to mirror continuous, the target table should look exactly the same as the source table. This can be verified with SPUFI or other query tools.

Testing continuous mirroring

After the refresh completes, InfoSphere Change Data Capture immediately puts the subscription into continuous mirroring. At this point, committed changes to the data in the source table are propagated to the target table in near real-time. This can be verified with SPUFI or other query tools.

Configuring notification settings

InfoSphere Change Data Capture divides all messages into 45 categories based on severity of the message, whether the message is related to source or target, and the aspect of the product that the message is related to. By default, the messages in all of these categories are written to the CHCPRINT DD in the CDC agent started task. InfoSphere Change Data Capture provides the ability for messages to be sent to the syslog or to user exits. It also provides the ability to stop messages from being sent to CHCPRINT. All of this is configured at the message category level.

To change notification settings for a subscription, return to the configuration tab. Next, right-click the subscription and select **Notifications**. Selecting a message category and unchecking the default checkbox will allow you to configure destinations for messages in that category.

For our purposes, we sent all fatal and error messages to the syslog. See Figure 7-29.

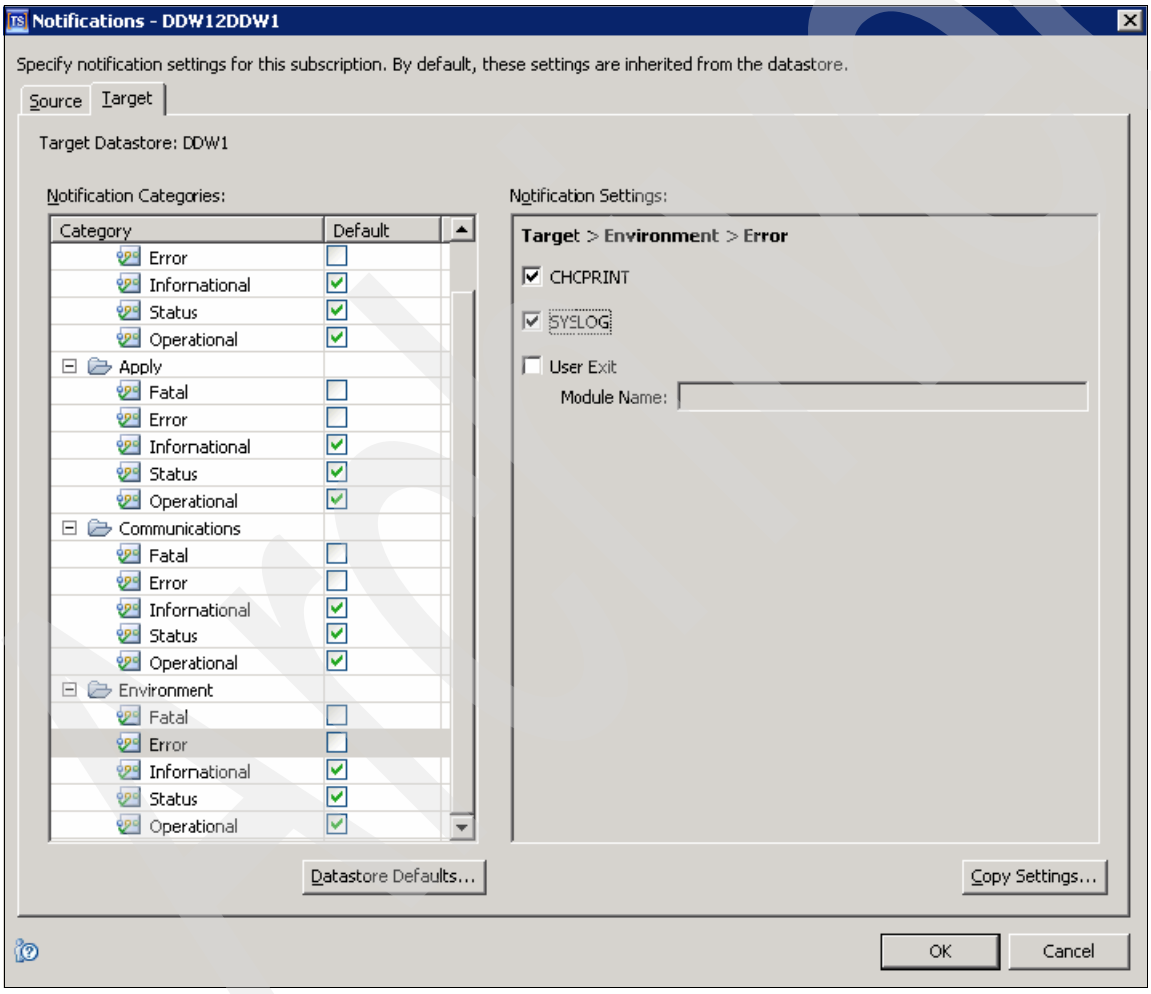


Figure 7-29 Configuring notification settings

7.3.3 Tuning InfoSphere Change Data Capture for z/OS

In the previous section, we explained how to create an initial CDC agent configuration. We explained the various SCHCDATA members that control product functionality, and we showed the initial functional test configuration that we used. In this section, we take a closer look at key SCHCDATA parameters. We discuss the impact of these parameters on data latency,

throughput, and processor usage. We show the tuned SCHCDATA configuration that we used for our testing. Finally, we discuss performance monitoring for InfoSphere Change Data Capture. Parameters that have already been set in the initial configuration and do not impact performance or availability are not discussed in this section.

For more information about all of the configuration parameters that are allowed in SCHCDATA members, refer to the IBM InfoSphere Change Data Capture Version 6.2 documentation.

CHCCFGxx

InfoSphere Change Data Capture reads general product configuration control statements when it is being initialized. These statements are read from the CHCCFGxx configuration control data set member. The CHCCFGxx member is the first member of the SCHCDATA library that InfoSphere Change Data Capture reads during initialization.

AUTORESTARTINTERVAL

AUTORESTARTINTERVAL works in conjunction with persistent subscriptions. By default, subscriptions do not automatically recover from temporary error conditions, such as a communications failure. Marking a subscription as persistent allows it to automatically recover from these types of conditions. AUTORESTARTINTERVAL controls the interval, in minutes, after which persistent subscriptions will attempt to recover.

The default value is zero, which means that even if a subscription is marked as persistent, no automatic recovery will be attempted. Intervals in the range of 1-60 are valid. If you plan to use persistent subscriptions, the recommendation is to set AUTORESTARTINTERVAL to a value that is low enough to re-establish the connection in a timely manner, but not flood the system with failed attempts.

HEARTBEATTIMEOUT

HEARTBEATTIMEOUT controls the timeout interval, in minutes, for heartbeat messages between source and target CDC agents. If either CDC agent involved in an active subscription does not receive a heartbeat message within the HEARTBEATTIMEOUT interval, it is considered a communications failure and the subscription is stopped. If the subscription is defined as persistent, it attempts to automatically recover from the communications failure at an interval controlled by AUTORESTARTINTERVAL.

The default value is 15, indicating a timeout interval of 15 minutes. Intervals in the range of 3-999 minutes are valid. A value of zero disables the heartbeat feature. The recommendation is to use a common value for HEARTBEATTIMEOUT across all CDC agents.

PALCLEANUPTIME

PALCLEANUPTIME specifies the time of day that product administration log (PAL) data is cleaned up. The PAL VSAM cluster holds the data that is viewed through the Management Console event log viewer. This cleanup occurs on a daily basis. This parameter works in conjunction with PALRETPD.

If this parameter is not explicitly specified, no daily PAL cleanup is performed. Values in the range 00:00-23:59 are valid. The recommendation is to provide a value for this parameter.

PALRETPD

PALRETPD is a number that controls what PAL data should be deleted when a daily PAL cleanup occurs. Messages older than the number of days specified on the PALRETPD parameter are deleted.

This parameter defaults to a value of 14, indicating that data older than 14 days should be deleted. Values in the range 1-9999 are valid.

REPSTATSINTERVAL

In the initial configuration, we recommended that REPSTATSINTERVAL be set to a value of 5 for CDC agents that will act as sources. Setting REPSTATSINTERVAL to 5 caused the CDC agent to generate messages every five minutes indicating the log scraper task's position in the DB2 log. Values in the range 5-1440 are valid.

Although minimal, REPSTATSINTERVAL does cause some additional overhead. Setting REPSTATSINTERVAL to zero disables messages reporting the position of the log scraper task in the DB2 log. REPSTATSINTERVAL has no effect on data synchronization. It strictly controls the frequency of the messages indicating the log scraper's position.

CHCCFGP1

We created a new set of SCHCDATA configuration members for our tuned configuration. We used a suffix of P1. To activate this configuration, the CDC agent started task was stopped and started using CONFIG=P1.

Our CHCCFGP1 configuration member was as shown in Example 7-15.

Example 7-15 CHCCFG final configuration

```
CONFIG  REPSTATSINTERVAL=0,
AUTORESTARTINTERVAL=1,
HEARTBEATTIMEOUT=3,
PALCLEANUPTIME=00:00
```

CHCCMMxx

InfoSphere Change Data Capture reads communications configuration control statements when communications are being initialized. These statements are read from the CHCCMMxx configuration control data set member.

CTRLQUEUESIZE

CTRLQUEUESIZE controls the size of the queue for the control communications link. The control communications link is used for message transfers that coordinate the replication of data between CDC agents.

The default value for CTRLQUEUESIZE is 64 K. Valid values are in the range 64 K-1 M. The default value is sufficient except in cases where there are a large number of subscriptions between the same pair of CDC agents.

DATAQUEUESIZE

DATAQUEUESIZE controls the size of the queue for the data communications link, which is used for the transmission of replicated data between CDC agents.

The default value for DATAQUEUESIZE is 1 M. Valid values are in the range 64 K-1024 M. The appropriate settings for CTRLQUEUESIZE and DATAQUEUESIZE depend upon the data being replicated. DATAQUEUESIZE can have a significant impact on virtual storage allocations because each subscription will allocate two data queues. The recommendation is to start with the default value and adjust upwards until latency goals are met or further adjustments no longer improve latency.

CHCCMMP1

Our final CHCCMMxx configuration was as shown in Example 7-16.

Example 7-16 CHCCMM final configuration

```
TCP/IP SERVICENAME=11001,
DATAQUEUESIZE=8M,
CTRLQUEUESIZE=64K
```

CHCDBMxx

InfoSphere Change Data Capture reads DBMS configuration control statements when DBMS repositories are being initialized. These statements are read from the CHCDBMxx configuration control data set member.

Many of the CHCDMBxx parameters are related to tuning of the log read process. For the parameters to make sense, it is necessary to first provide more detail about how InfoSphere Change Data Capture reads information from the DB2 log.

During initialization, InfoSphere Change Data Capture issues an IFI command to activate a DB2 MONITOR trace (IFCID 126) using an available OPx buffer as the target. As new data is written to the DB2 log, more trace data gets written to the OPx buffer. When a certain amount of data is available in the OPx buffer, DB2 notifies the InfoSphere Change Data Capture started task via the POST service. The amount of data required to be written to the OPx buffer before InfoSphere Change Data Capture is notified is controlled by the BUFSIZE and BUFTHRESHOLD parameters.

When InfoSphere Change Data Capture receives such a notification, it then proceeds to read the new DB2 log records one at a time via IFI. When it reaches the current end of the DB2 log, it then waits for the next notification from DB2 that the OPx buffer threshold has been reached.

By default, one instance of the above log read process occurs for every CDC subscription that specifies the DB2 z/OS subsystem or data sharing group as the source. The recommendation is that if you are going to create multiple subscriptions that have the same DB2 z/OS subsystem or data sharing group as the source, that you enable caching of the data that comes from the log read process. With caching turned on, there is one log read process that fills a cache. From there, each subscription locates the data it needs from the cache instead of issuing new IFI commands. Caching is controlled by the CACHEBLOCKISZE, CACHELEVEL1RESERVED, and CACHELEVEL1SIZE parameters.

Once the subscriptions have the data they need from the DB2 log, they communicate with the appropriate target CDC agent. The target CDC agent, in turn, applies the changes to the target tables by serially issuing the appropriate SQL statements. When the target CDC agent issues a commit, it stores the last source DBMS log position (RBA or LRSN) associated with work that has been committed at the target. This log position is known as a bookmark, and is used as a restart point for the source log read process should InfoSphere Change Data Capture be stopped or an error occur.

BUFSIZE

BUFSIZE controls the size of the DB2 OPx buffer that is used to hold data from the MONITOR trace that InfoSphere Change Data Capture starts.

This parameter works in conjunction with BUFTHRESHOLD to control asynchronous notification that new DB2 log data is available. The default setting is 132, which indicates a buffer size of 132 kilobytes. Valid values are in the range 8-1024.

BUFTHRESHOLD

BUFTHRESHOLD controls the percentage of the OPx buffer that must be full before InfoSphere Change Data Capture is notified that there is new DB2 log data to read.

The default is 1, which indicates 1%. When used in conjunction with the default BUFSIZE value, this indicates that 1.32 kilobytes of MONITOR trace data must be written to the OPx buffer before InfoSphere Change Data Capture wakes up and starts issuing new IFI reads. Valid values are in the range 1-99.

It is important to note that when there is continuous DB2 activity, BUFSIZE and BUFTHRESHOLD have no impact. Once InfoSphere Change Data Capture starts issuing IFI reads, it continues to do so until it hits the current end of log. Once hitting end of log, BUFISZE and BUFTHRESHOLD control the amount of new log data that must exist before InfoSphere Change Data Capture will start issuing new IFI reads.

In an environment without constant DB2 activity, BUFSIZE and BUFTHRESHOLD have a significant impact on processor usage. When InfoSphere Change Data Capture is woken up more frequently to issue new IFI reads, more processor usage occurs. So, while higher settings for BUFSIZE and BUFTHRESHOLD generally lead to lower processor usage in such situations, they result in higher latency because InfoSphere Change Data Capture will not start issuing new IFI reads until more MONITOR trace data has been written.

When adjusting BUFSIZE and BUFTHRESHOLD, the system log should be monitored for the message shown in Example 7-17.

Example 7-17 Incorrect OPx buffer settings

```
DSNW133I + DSNWVOPX - TRACE DATA LOST, <OPx> NOT ACCESSIBLE RC=08
```

It is important to understand that this message does not indicate that DB2 log data has been lost. Even when this occurs, InfoSphere Change Data Capture successfully replicates all changed data. However, this message does indicate that InfoSphere Change Data Capture is unable to clear data out of the OPx buffer at the rate that new data is coming in. If this occurs, there are three potential solutions:

- Increase BUFSIZE
- Decrease BUFTHRESHOLD
- Increase the priority of the InfoSphere Change Data Capture started task

CACHEBLOCKSIZE

CACHEBLOCKSIZE, CACHELEVEL1RESERVED, and CACHELEVEL1SIZE must all be set before caching is enabled. Additionally, the level 2 cache must be created.

CACHEBLOCKSIZE controls the size of the blocks of data that are moved between the level 1 and level 2 caches. The level 1 cache is an in-memory cache. The level 2 cache is a VSAM data set. The size of the level 1 cache must be a multiple of the block size. The size of the level 1 and level 2 caches is significant. If the data that a subscription is looking for is found in the level 1 cache, no disk I/O is required. If the data is instead found in the level 2 cache, the data will be retrieved from the level 2 VSAM cluster. If the data is not in either the level 1 or level 2 caches, a new synchronous log read process is started to issue IFI reads against DB2. This synchronous log read process continues until the data that the subscription needs is located in one of the caches.

CACHEBLOCKSIZE is specified as an integer value and indicates the block size in megabytes. A block size of one or two megabytes is sufficient in most cases.

CACHELEVEL1RESERVED

CACHELEVEL1RESERVED specifies the size, in megabytes, of the portion of the level 1 cache that is reserved for holding recent DB2 log data. Typically, 80% of the level 1 cache should be reserved. Additionally, the following restrictions must be met:

- The reserved portion of the level 1 cache must be at least 5 megabytes.
- The reserved portion of the level 1 cache must be an integer multiple of the cache block size (CACHEBLOCKSIZE).
- There must be at least one block in the level 1 cache that is not reserved. See CACHELEVEL1SIZE.

CACHELEVEL1SIZE

CACHELEVEL1SIZE controls the size, in megabytes, of the level 1 cache. CACHELEVEL1SIZE must be a multiple of CACHEBLOCKSIZE. Additionally, it must be at least one block larger than CACHELEVEL1RESERVED. The following restrictions apply:

- The level 1 cache must be at least 10 megabytes.
- The level 1 cache must consist of at least two blocks. See CACHEBLOCKSIZE.
- The level 1 cache must be an exact multiple of the block size.
- The level 1 cache is an in-memory cache. This storage will be allocated by the InfoSphere Change Data Capture started task. It should be sized to minimize paging.

Once CACHEBLOCKSIZE, CACHELEVEL1RESERVED, and CACHELEVEL1SIZE are set, the level 2 cache must be defined and allocated to the InfoSphere Change Data Capture started task.

To define the level 2 cache, edit and submit job CHCCRCCH in the SCHCCNTL library. Note that the level 2 cache must be at least one block larger than the level 1 cache. Our level 2 cache was defined as shown in Example 7-18.

Example 7-18 Defining the level 2 cache

```
//CHCCRCCH JOB MSGLEVEL=(1,1),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//PURGE    EXEC PGM=IEFBR14
//CHCCHCTL DD DSN=ICDC.ZDW.DDW1.L2CACHE.CHCCHCTL,
//            DISP=(MOD,DELETE),SPACE=(1024,1),
//            DCB=(LRECL=1024,RECFM=FB)
//*
//CREATE   EXEC PGM=IEFBR14,COND=(12,LT)
//CHCCHCTL DD DSN=ICDC.ZDW.DDW1.L2CACHE.CHCCHCTL,DISP=(NEW,CATLG),
//            SPACE=(1024,1),DCB=(LRECL=1024,RECFM=FB)
//*
//DEFMTDTA EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  DELETE -
    ICDC.ZDW.DDW1.L2CACHE.CHCCACHE -
    CLUSTER -
    PURGE

  SET MAXCC = 0
  SET LASTCC = 0

  DEFINE -
    CLUSTER -
```

```
     (NAME(ICDC.ZDW.DDW1.L2CACHE.CHCCACHE) -
      VOL(DWDBS1) -
      RECORDSIZE(32760 32760) -
      NUMBERED -
      SHAREOPTIONS(2))-
    DATA -
     (NAME(ICDC.ZDW.DDW1.L2CACHE.CHCCACHE.DA) -
      BUFFERSPACE(8388608) -
      CYLINDERS(250))
/*
//
```

To allocate the level 2 cache to the InfoSphere Change Data Capture started task, we added the DDs shown in Example 7-19.

Example 7-19 DDs required for caching

```
//CHCCACHE DD DSNAME=ICDC.ZDW.DDW1.L2CACHE.CHCCACHE,DISP=SHR,
//           AMP=('BUFND=256,ACCBIAS=SW')
//CHCCHCTL DD DSNAME=ICDC.ZDW.DDW1.L2CACHE.CHCCHCTL,DISP=SHR
```

COMMITFREQ

COMMITFREQ controls the frequency at which the target CDC agent commits changes to the target tables. This parameter is specified as a triplet of values. The first value is the number of changes that can be applied to the target tables before a commit is generated. The second value is the number of seconds that are allowed since the previous commit before another commit is generated. The third parameter is the number of commits that are allowed at the source before a new target commit is generated.

The target CDC agent issues a commit upon receipt of the next notification of a source commit after any one of the above three conditions is satisfied. Transactional integrity is always maintained. The three conditions are tracked at the subscription level. In other words, the number of changes applied, the number of seconds since the previous commit, and the number of source commits are all tracked separately for each subscription. A commit is executed for the changes for a single subscription when one of the conditions is met for that subscription.

Decreasing the frequency at which the target CDC agent issues commits can significantly increase throughput and decrease processor usage, at the risk of increasing the likelihood of locking conflicts and increasing replication latency. It is important to understand that InfoSphere Change Data Capture will never apply changes that have not been committed at the source. The target CDC agent only sees changes that have already been committed at the source.

The third parameter is of particular importance when the DB2 changes being captured consist of small units of work, such as online activity. Likewise, large DB2 units of work where only a small subset of the data is being replicated become small units of work when applied by the target CDC agent. The third parameter can be used to group these small units of work into a larger unit of work, thus greatly increasing throughput.

The default value for the COMMITFREQ parameter is (200,30,1).

LOGPOLLINTERVAL

LOGPOLLINTERVAL is used to force InfoSphere Change Data Capture to wake up and issue a new IFI read even if the OPx buffer threshold, as controlled by BUFSIZE and

BUFTHRESHOLD, has not yet been reached. If new DB2 log data is not yet available, the IFI read will be blocked until new data becomes available.

There are two distinct uses for LOGPOLLINTERVAL. The first usage is to account for non-local DB2 activity in a data sharing group. In a data sharing group, only activity on the local DB2 log causes data to be written to the MONITOR trace, which in turn causes InfoSphere Change Data Capture to wake up and issue new IFI reads. If the local DB2 member becomes dormant, LOGPOLLINTERVAL can be used as a way to assure that InfoSphere Change Data Capture continues to process changes occurring on the other members.

The other usage of LOGPOLLINTERVAL is applicable in both data sharing and non-data sharing. Consider the situation where new trace data has been written to the OPx buffer, but the threshold has not yet been reached. If, at this time, DB2 activity ceases for a period of time, the last DB2 changes will not replicate until more DB2 activity occurs. LOGPOLLINTERVAL can be used to force a new IFI read even though the OPx buffer threshold has not been met. This is generally used to ensure that SLAs are met, even in the event of an extended period of low DB2 activity.

The default value for LOGPOLLINTERVAL is 5, indicating that if the source DB2 subsystem is part of a data sharing group, InfoSphere Change Data Capture should issue an IFI read if it has not issued one for the last 5 seconds. If the source DB2 subsystem is standalone, IFI reads will never be issued unless the OPx buffer threshold is reached. To force the LOGPOLLINTERVAL parameter to take effect in non-data sharing, the ALWAYS statement must be used. For example, to force InfoSphere Change Data Capture to issue an IFI read after 5 seconds in non-data sharing, you would specify LOGPOLLINTERVAL=(5,ALWAYS).

Setting LOGPOLLINTERVAL to small values can cause increased processor usage due to the fact that InfoSphere Change Data Capture will wake up to process small amounts of new DB2 log data. Just as with BUFSIZE and BUFTHRESHOLD, LOGPOLLINTERVAL has no effect when there is constant DB2 activity. Typically, LOGPOLLINTERVAL should be set to a value just less than the desired SLA. For example, if the SLA is that data latency will be less than 60 seconds, LOGPOLLINTERVAL should be set to a value just less than 60. This way, the SLA should be met even in the event that DB2 activity stops on the subsystem or local member.

RECOVERYRETRYLIMIT

RECOVERYRETRYLIMIT controls the number of times InfoSphere Change Data Capture will attempt to re-apply changes to the target tables when a deadlock or timeout occurs. If the retry limit is exceeded, the subscription is stopped and put into an error state, unless this behavior is overridden with the ENDONERROR CHCDBMxx parameter. The recommendation is that you set RECOVERYRETRYLIMIT to an appropriate value and do not override the ENDONERROR default.

The default value for RECOVERYRETRYLIMIT is 0. Valid values are in the range 0-65535.

RETRYCACHESIZE

RETRYCACHESIZE controls the amount of storage used by each subscription to hold changes that InfoSphere Change Data Capture will re-attempt to apply to the target database due to a deadlock or timeout condition. This parameter only takes effect if RECOVERYRETRYLIMIT is non-zero. This storage is acquired above the bar.

The default value for RETRYCACHESIZE is 1 G. Valid values are in the range 1 M-9999 G.

USELOADER

USELOADER controls whether or not InfoSphere Change Data Capture will ever use the DB2 LOAD utility to handle a table refresh. By default, USELOADER is set to NO, indicating that all table refreshes will be processed by issuing SQL statements against the target table. To enable usage of the DB2 LOAD utility, set USELOADER=YES, and configure the CHCLDRxx configuration member.

CHCDBMP1

For our final CHCDBMxx configuration we enabled caching, set LOGPOLLINTERVAL to meet our target SLAs, enabled usage of the DB2 LOAD utility, and decreased the COMMIT frequency via the COMMITFREQ parameter. Our final configuration was as shown in Example 7-20.

Example 7-20 CHCDBM final configuration

```
DB2 SSID=DDW1,
PLANSUFFIX=01,
SQLWHEREROWSEL=YES,
LOGPOLLINTERVAL=(60,ALWAYS),
BUFSIZE=1024,
RECOVERYRETRYLIMIT=3,
RETRYCACHESIZE=64M,
USELOADER=YES,
CACHELEVEL1RESERVED=30,
CACHEBLOCKSIZE=2,
CACHELEVEL1SIZE=38,
COMMITFREQ=(5000,30,5000)
```

CHCLDRxx

InfoSphere Change Data Capture reads DB2 LOAD utility configuration control statements before starting to refresh tables. These statements are read from the CHCLDRxx configuration control data set member. The CHCLDRxx configuration member is only used if USELOADER=YES in the CHCDBMxx configuration member, or if DB2 LOAD utility support has been enabled for a particular subscription via the CHCMTAUT utility.

Setting USELOADER=YES in CHCDBMxx allows table refreshes to use the DB2 LOAD utility instead of refreshing the target table via SQL statements. However, the refresh of the target table reverts to using SQL statements if any of the following conditions are true:

- There is more than one table in the target table space.
- InfoSphere Change Data Capture is being used to capture changes on the target table.
- The total length of all the columns in the target table exceeds 32,760 bytes minus the number of columns in the table.
- A user exit has been enabled on this table mapping and the clear table operation has been disabled.
- The target table has a ROWID column.
- The table mapping is a LiveAudit table mapping.

The parameters in the CHCLDRxx configuration member correspond directly to parameters on the LOAD DATA REPLACE utility control statement that InfoSphere Change Data Capture will build to refresh the target table. For more information, refer to the *DB2 for z/OS Utility Guide and Reference*, GC19-1145.

Archived

DISCARDS

The DISCARDS parameter corresponds to the DISCARDS keyword in the LOAD DATA control statement. This parameter is optional. If not specified, a value of 0 is used, indicating that the LOAD should continue regardless of the number of input records that fail to LOAD. Specifying any other integer value will set a maximum as to the number of input records that can be discarded before the LOAD abnormally terminates.

DSNDEVT

The DSNDEVT parameter specifies the device type to use when allocating DB2 LOAD utility work data sets. Generic or actual device names can be specified. This parameter is optional. If not specified, SYSDA is used.

DSNHLQ

The DSNHLQ parameter specifies the high-level qualifier used when allocating the DB2 LOAD utility work data sets. The qualifier cannot exceed 11 characters. InfoSphere Change Data Capture builds data set names by concatenating the keyword setting, the name of the replication agent, the current date and time, and the default DD name of the data set specified in the DB2 LOAD utility documentation. InfoSphere Change Data Capture requires sufficient authority to allocate, write to, read from, and delete these data sets. This parameter is optional. If not specified, the authorization ID of the InfoSphere Change Data Capture started task is used.

ENFORCE

The ENFORCE parameter corresponds to the ENFORCE keyword in the LOAD DATA control statement. This parameter is optional. If not specified, CONSTRAINTS is used, indicating that DB2 will perform constraint checking when doing the LOAD. If ENFORCE=NO, no constraint checking will occur during the LOAD, and the target table is placed in check pending status following the LOAD if any referential or check constraints exist on the table.

KEEPDICTIONARY

The KEEPDICTIONARY parameter corresponds to the KEEPDICTIONARY keyword in the LOAD DATA control statement. This parameter is optional. If not specified, NO is used, indicating that DB2 is to build a new compression dictionary if the target table space is compressed. Using KEEPDICTIONARY=YES forces DB2 to reuse the existing compression dictionary.

LOG

The LOG parameter corresponds to the LOG keyword in the LOAD DATA control statement. This keyword is optional. If not specified, YES is used, indicating that DB2 is to log records that are inserted into the target table as part of the LOAD. Using LOG=NO causes DB2 to skip logging during the LOAD process. If LOG=NO, upon completion of the LOAD, the target table is put into copy pending status unless the table is defined as NOT LOGGED (keyword introduced in DB2 9).

PREFORMAT

The PREFORMAT parameter corresponds to the PREFORMAT keyword in the LOAD DATA control statement. This keyword is optional. If not specified, NO is used, indicating that DB2 is not to preformat remaining pages upon completion of the LOAD. Set PREFORMAT=YES, if you wish DB2 to do this preformatting.

REUSE

The REUSE parameter corresponds to the REUSE keyword in the LOAD DATA control statement. This keyword is optional. If not specified, NO is used, indicating that DB2 should

Archived

delete and reallocate the underlying VSAM data sets. Using REUSE=YES, forces DB2 to reuse the existing VSAM data sets.

SORTDEVT

The SORTDEVT parameter corresponds to the SORTDEVT keyword in the LOAD DATA control statement. This keyword is optional. If not specified, no SORTDEVT keyword is included in the LOAD DATA statement. Set SORTDEVT to the name of the device that the DB2 LOAD utility should use when dynamically allocating temporary data sets for DFSORT.

SORTKEYS

The SORTKEYS parameter corresponds to the SORTKEYS keyword in the LOAD DATA control statement. This parameter is optional. If not specified, a value of 0 is used. SORTKEYS should be set to an integer value that estimates the number of index keys that will need to be sorted during the LOAD. Refer to the DB2 z/OS Utility Guide and Reference for more information on setting SORTKEYS.

SORTNUM

The SORTNUM parameter corresponds to the SORTNUM keyword in the LOAD DATA control statement. This parameter is optional. If not specified, a value of 0 is used. SORTNUM should be set to an integer value that specifies how many data sets DFSORT should dynamically allocate. Refer to the DB2 z/OS Utility Guide and Reference for more information about setting SORTNUM.

SORTOUTSIZE

The SORTOUTSIZE parameter is required and should be set to a pair of values indicating the primary and secondary allocations, in cylinders, for the temporary work file for sort output. For example, SORTOUTSIZE=(100,10).

SYSDISCSIZE

The SYSDISCSIZE parameter is required and should be set to a pair of values indicating the primary and secondary allocations, in cylinders, for the discard data set. The discard data set holds records that failed to LOAD properly.

SYSERRSIZE

The SYSERRSIZE parameter is required and should be set to a pair of values indicating the primary and secondary allocations, in cylinders, for the error data set. The error data set holds information about errors that were encountered during the LOAD.

SYSMAPSIZE

The SYSMAPSIZE parameter is required and should be set to a pair of values indicating the primary and secondary allocations, in cylinders, for the SYSMAP data set. The SYSMAP data set is used as a work data set during error processing.

SYSUT1SIZE

The SYSUT1SIZE parameter is required and should be set to a pair of values indicating the primary and secondary allocations, in cylinders, for the sort input work file.

CHCLDRP1

Our final CHCLDRxx configuration is shown in Example 7-21.

Example 7-21 CHCLDR final configuration

```
DB2LOADER LOG=YES,
          PREFORMAT=NO,
```

```
KEEPDICTIONARY=NO,
REUSE=NO,
DSNDEVT=3390,
DSNHLQ=ICDC.DDW1,
SYSUT1SIZE=(700,70),
SORTKEYS=6000000,
SORTOUTSIZE=(700,70),
SYSERRSIZE=(10,1),
SYSMAPSIZE=(100,10),
SYSDISCSIZE=(10,1)
```

Latency and throughput monitoring

Tuning InfoSphere Change Data Capture is an iterative process. The appropriate settings for the various configuration parameters really depend on the data, the volume of changes, the transformations, and the SLAs that must be met. Fortunately, Management Console provides latency and throughput monitoring, which can help you determine how much more tuning is required and assess the impact of parameter changes.

Before you can view latency and throughput information for a subscription, you must enable collection of statistics for that subscription. To enable collection of statistics, first click the monitoring tab in Management Console. Next, click the subscriptions subtab. Finally, right-click the subscription of interest, and select **Statistics** → **Collect Statistics** (Figure 7-30).

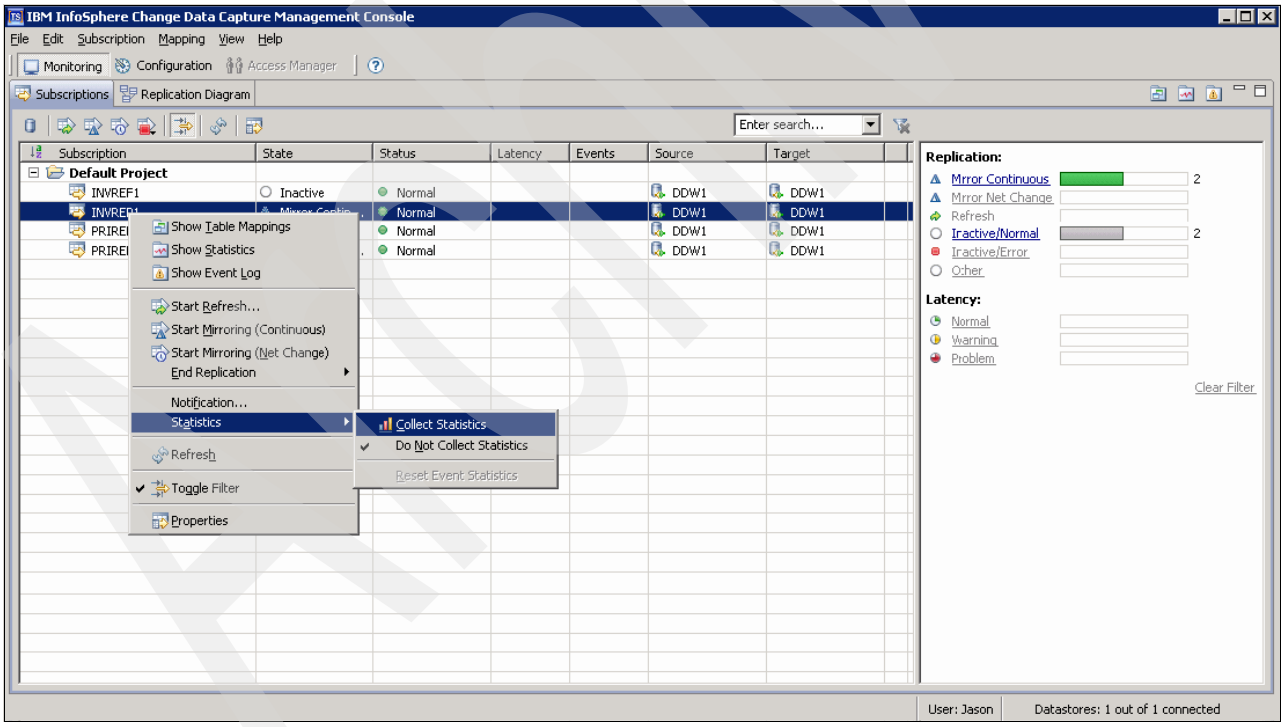


Figure 7-30 Enabling statistics collection

Now that statistics collection is enabled, right-click the subscription and select **Show Statistics**, to bring up the statistics monitoring window (Figure 7-31 on page 178).

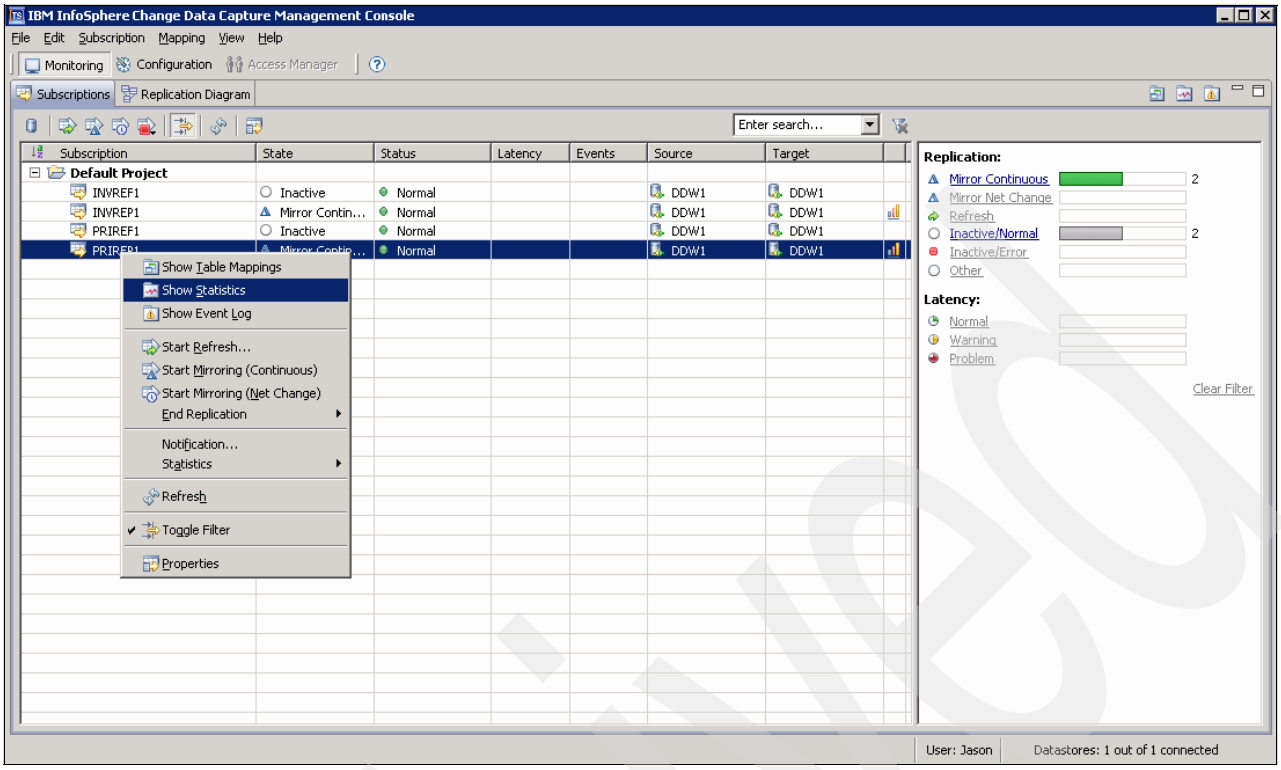


Figure 7-31 Opening the statistics monitoring window

In the statistics window, you are shown a tabular view on the left-hand side, which also tracks maximum, minimum, and average values. On the right-hand side is a graph that periodically refreshes. There is a graph pull-down list, which allows you to select from several different graphs.

With statistics collection enabled and the statistics window open, we ran test workloads against the DB2 z/OS source tables and watched InfoSphere Change Data Capture performance via Management Console.

Prior to tuning, InfoSphere Change Data Capture was unable to keep up with the DB2 z/OS workload. This was evident due to the fact that the latency, which measures the time between the source commit and the target commit, continued to grow; see Figure 7-32 on page 179.

Archive

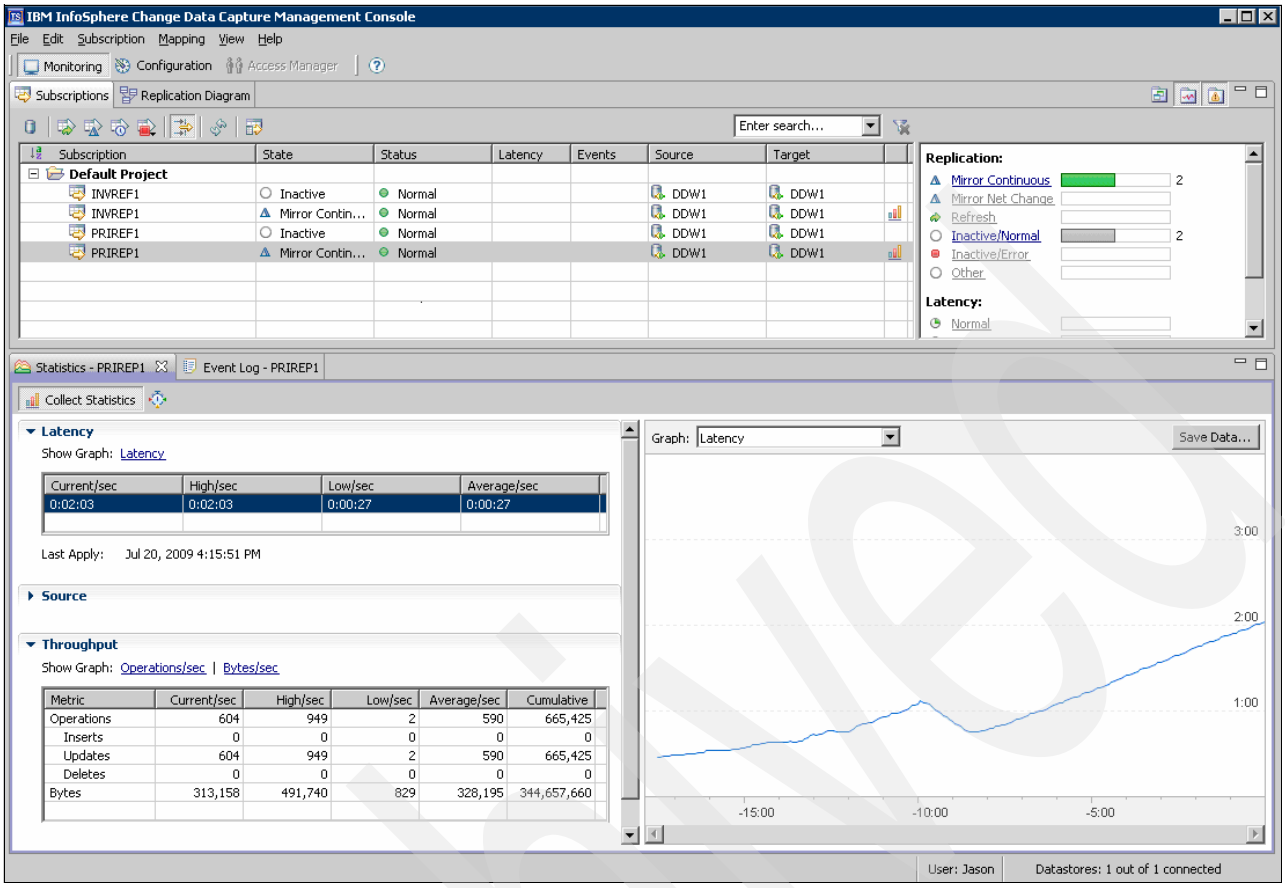


Figure 7-32 Growing latency indicates that tuning is required

Upon switching to the throughput graph, we saw that our maximum throughput was less than 500 kilobytes per second. This was an indication that we had a lot of room to grow via tuning; see Figure 7-33 on page 180. Throughputs in the range of 18 megabytes per second per subscription have been achieved during InfoSphere Change Data Capture for z/OS benchmarks. For more information, see the *IBM InfoSphere Change Data Capture Version 6.2 (DB2 for z/OS) Performance Evaluation and Analysis* white paper.

Archive

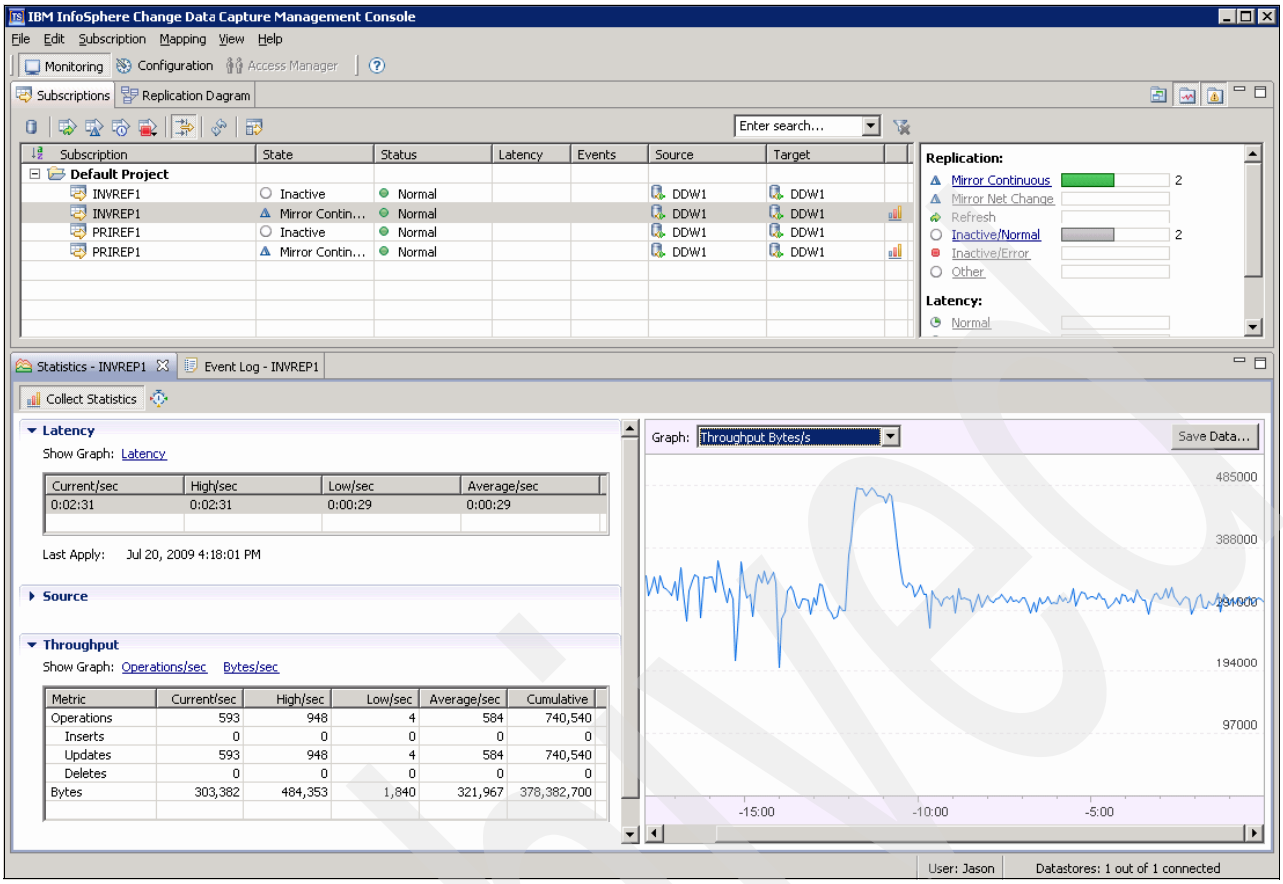


Figure 7-33 Initial throughput before tuning

After tuning, we were able to sustain subsecond latency with a throughput of 4 megabytes per second per subscription.

Monitoring of InfoSphere Change Data Capture processor usage

In addition to tuning InfoSphere Change Data Capture to meet latency and throughput requirements, it is important to understand how various configuration parameters impact processor utilization. In this section, we provide an explanation of the key tasks that run as part of the InfoSphere Change Data Capture address space. We will also take a look at how several configuration parameters can impact processor utilization.

For our purposes, we monitored processor usage of the various InfoSphere Change Data Capture tasks with IBM Tivoli Omegamon XE for DB2 Performance Expert on z/OS V4.2.

InfoSphere Change Data Capture tasks

In most cases, it is sufficient to monitor InfoSphere Change Data Capture at the plan level as most of the plans directly correspond to tasks that the CDC agent is performing. In certain cases, it becomes necessary to look at individual DBRMs. In general, we will refer to the various InfoSphere Change Data Capture tasks by plan name and by an associated 3-letter acronym. Keep in mind that all plan names are suffixed with two characters that were specified when running the bind job, CHCBNDPL. When we are referring to an individual DBRM, we will explicitly state this.

CHCDALxx (DAL)

DAL stands for DBMS Asynchronous Log. The DAL task is responsible for setting up and taking down the MONITOR trace, as well as clearing data from the OPx buffer upon notification from DB2. Every InfoSphere Change Data Capture started task will have a DAL task, even if the associated CDC agent is never used as a source.

DCW

The DCW task is not associated with any DB2 plans or DBRMs. The DCW task's sole responsibility is to move data between the level 1 and level 2 caches when caching is enabled.

CHCDLPxx (DLP)

DLP stands for DBMS Log Profile. The DLP task gathers information from the DB2 log that is not directly related to changes in the data of the source tables. For example, the DLP task looks for REORGs on the source tables.

CHCDLRxx (DLR)

DLR stands for DBMS Log data cache Reader. The DLR task is responsible for reading data from the DB2 log and populating the level 1 cache. One DLR task can be shared over multiple subscriptions.

CHCDLSDB (DLS)

CHCDLSDB is a DBRM associated with the DLS task, which is responsible for processing DB2 log data for a single subscription. The DLS task receives data from the corresponding DSL task. It will group change by units of work and stage them in hiperspace waiting for a COMMIT or a ROLLBACK. If a ROLLBACK is detected, the changes associated with that unit of work are discarded. If a COMMIT is detected, the changes associated with that unit of work are released to be sent to the target CDC agent. By default, the hiperspace staging area for a single DLS task, and therefore for a single subscription, is allowed to grow up to 2 gigabytes in size. The staging area can be restricted to a smaller maximum size via the MAXSUBSCRSTAGESIZE parameter in CHCDBMxx. The DLS DBRM is part of the SDT plan.

CHCDSCxx (DSC)

DSC stands for DBMS Supervision and Control. The DSC task is responsible for startup and shutdown of subscriptions. It is also responsible for transition of the subscription state, for example, from refresh to mirroring.

CHCDSLxx (DSL)

DSL stands for DBMS synchronous log. The DSL task is responsible for reading DB2 log data for a single subscription. If caching is enabled, the DSL task will attempt to find the necessary data in the level 1 or level 2 cache. If caching is not enabled, or the necessary data is not in either cache, the DSL task will obtain the necessary data by issuing an IFI read. The DSL task continues to read log data from the cache or via IFI reads until it hits the current end of log. Upon hitting end of log, it suspends until awoken by the DAL task.

CHCDTCxx (DTC)

DTC stands for DBMS Table Change. The DTC task is responsible for applying changes to the target tables for a single subscription. Data transformations that take place at the target occur as part of the DTC task.

Archived

CHCDTRDB and CHCDTRD8 (DTR)

These two DBRMs are associated with the DTR task. DTR stands for DBMS Table Refresh. The DTR task is responsible for reading data from a source table as part of a refresh. The DTR DBRMs are part of the SDT plan.

CHCPAAxx (PAA)

PAA stands for Product Administration Agent. The PAA task is responsible for supporting Management Console. For example, the PAA task is used to provide Event Log information to the end user via Management Console.

CHCSCTxx (SCT)

SCT stands for Source Control. The SCT task coordinates communication of the control information for a single subscription at the source side.

CHCSDTxx (SDT)

SDT stands for Source Data. The SDT task coordinates communication of the data information for a single subscription at the source side. Data transformations that take place at the source occur as part of the SDT task.

CHCTCTxx (TCT)

TCT stands for Target Control. The TCT task coordinates communication of the control information for a single subscription at the target side.

Figure 7-34 shows the relationship between the DAL, DSL, DLS, and DTC tasks. Note that DI (Data In) and DO (Data Out) are TCP/IP tasks that are not associated with a DB2 plan. This diagram assumes that caching is not enabled.

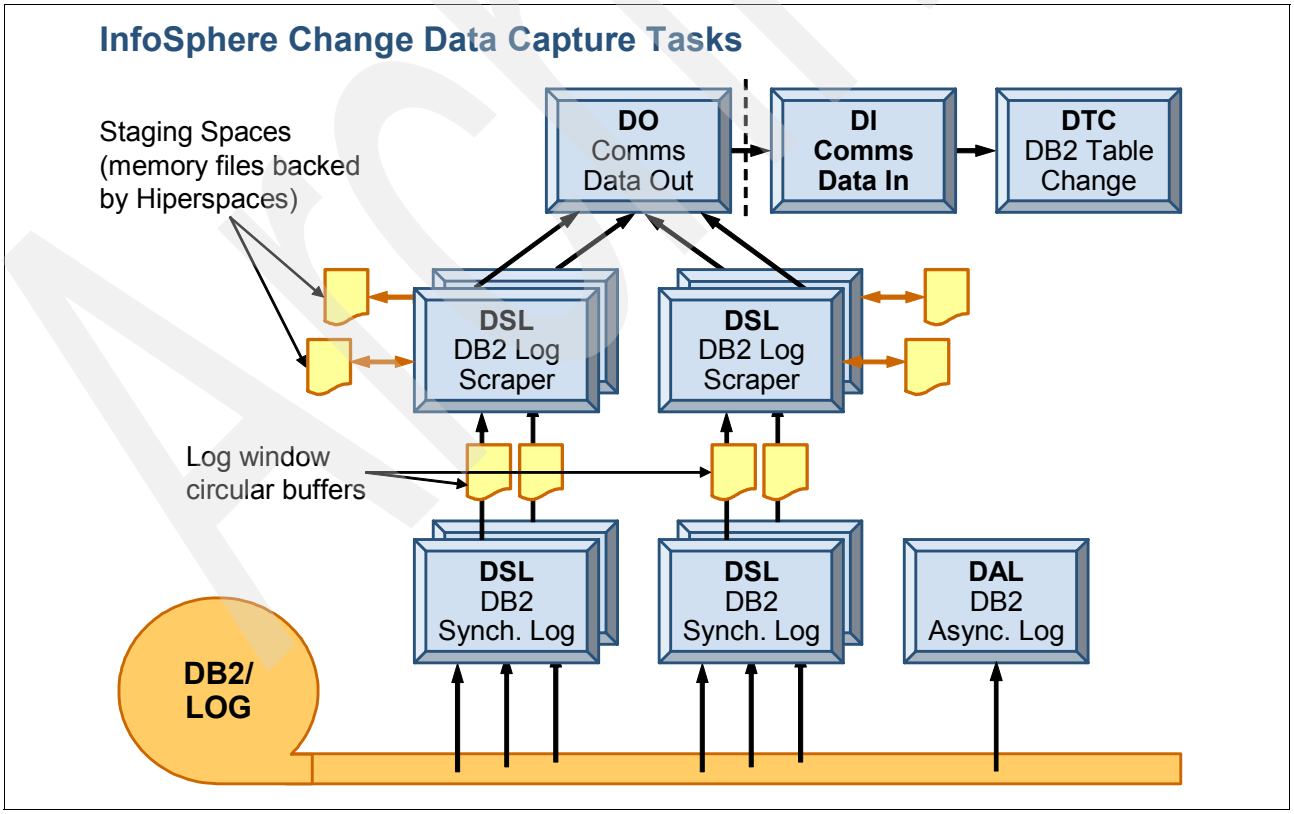


Figure 7-34 Task relationships with caching disabled

If caching is enabled, the DSL tasks instead attempt to read from the level 1 and level 2 caches. Only if the data is not found in the caches would the DSL task issue an IFI read. Figure 7-35 shows the relationship between the DLR, DCW, and DSL tasks when caching is enabled.

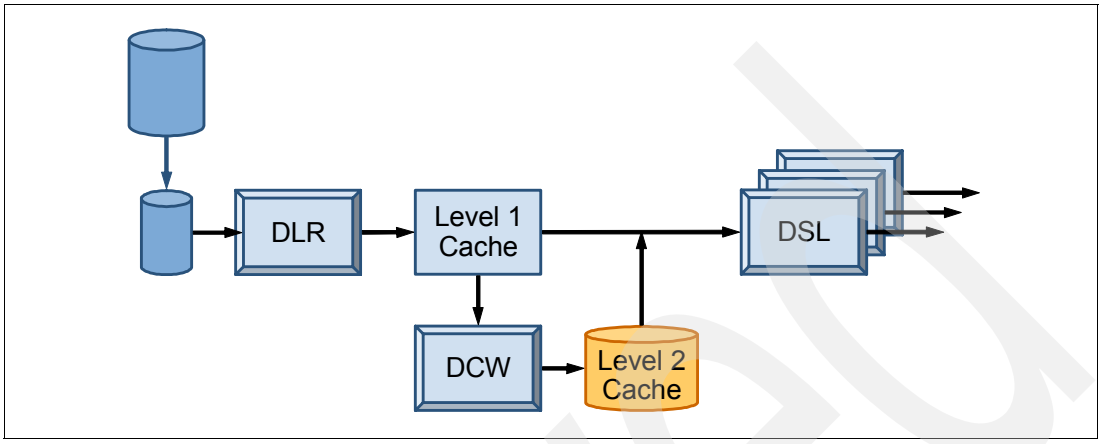


Figure 7-35 Task relationships with caching enabled

Using the DB2 LOAD utility for refresh

Typically, a refresh or initial load of the target tables is performed before continuous synchronization of the data begins. Ideally, this is the only time that a full refresh takes place. However, it is important to have a tuned refresh in the event that a full refresh is needed for recovery purposes. Any time InfoSphere Change Data Capture is not running and changes are occurring in the DB2 data, it will be falling behind the end of the DB2 log. When InfoSphere Change Data Capture comes back up, it picks up where it left off in the DB2 log, and tries to get caught up to the end of the log. If the DB2 workload is heavy enough and the current latency of the data is high enough, it may be decided that a full refresh of the target tables is the more appropriate recovery method. In these cases, the elapsed time of the refresh is the largest factor in the overall recovery time.

Table 7-2 shows a comparison of USELOADER=NO, where the refresh is done with SQL statements, and USELOADER=YES, where the refresh is done with the DB2 LOAD utility. In this case, we were refreshing a little over six million rows. While there was not a significant processor difference between the two methods, the elapsed time difference is substantial. Typically, in a recovery scenario, the elapsed time required to complete recovery and bring data latency back within the SLAs is critical.

Additionally, some of the work done by the DB2 LOAD utility is eligible for zIIP offload. In this case, the target table only had one index. Higher percentages of zIIP offload should be possible for tables with multiple indexes.

The total processor times in Table 7-2 were calculated by summing the processor times from the DTR, SDT, and DTC tasks, as well as processor time used by the LOAD utility, in the case of USELOADER=YES.

Table 7-2 Refresh of 6,344,819 rows

	USELOADER=NO	USELOADER=YES
Total processor time	3:57.196194	3:51.83886
zIIP eligible processor time	0.0	1.113035
Total elapsed time	8:09.810448	4:39.977580

The effect of the COMMITFREQ parameter

The COMMITFREQ parameter is very important, and very challenging to tune correctly. It has an impact on many different aspects of InfoSphere Change Data Capture performance. As we discussed in the section on CHCDBMxx, COMMITFREQ can greatly increase throughput. However, there is a trade-off that takes place, as COMMITFREQ can also cause data latency to increase, and can increase the likelihood of deadlocks or time-outs. Additionally, the COMMITFREQ parameter can have a large impact on the amount of processor that is used by the apply process, namely the DTC task.

Table 7-3 shows the amount of processor time, per change, that was used by the DTC task under two different COMMITFREQ settings. Data is given for two different subscriptions that were running concurrently.

Table 7-3 Effect of COMMITFREQ on CPU usage

Processor microseconds per change used by the DTC task	COMMITFREQ=(200,30,1)	COMMITFREQ=(5000,30,5000)
Subscription 1	234.768992	71.469031
Subscription 2	239.482282	72.047571

7.4 Using InfoSphere Change Data Capture for z/OS

In this section, we look at how we used InfoSphere Change Data Capture for z/OS, normally referred to as InfoSphere CDC, to capture changes in the OLTP environment and apply them in near real-time to the data warehouse environment. We called this setup “continuous update.” We discuss how to create replication definitions using the Management Console client. We also discuss operations and monitoring via Management Console. Lastly, we discuss implications of using InfoSphere CDC from a DB2 workload perspective. For information on installation, configuration, and tuning, see 7.3, “Overview of InfoSphere Change Data Capture” on page 146.

7.4.1 Building replication definitions

Our goal for using InfoSphere CDC was to capture changes as they occurred to data in the OLTP INVENTORY table. Upon capturing these changes, we would do various transformations and apply the changes to two history tables in the data warehouse. Our design is shown in Figure 7-36 on page 185.

Archive

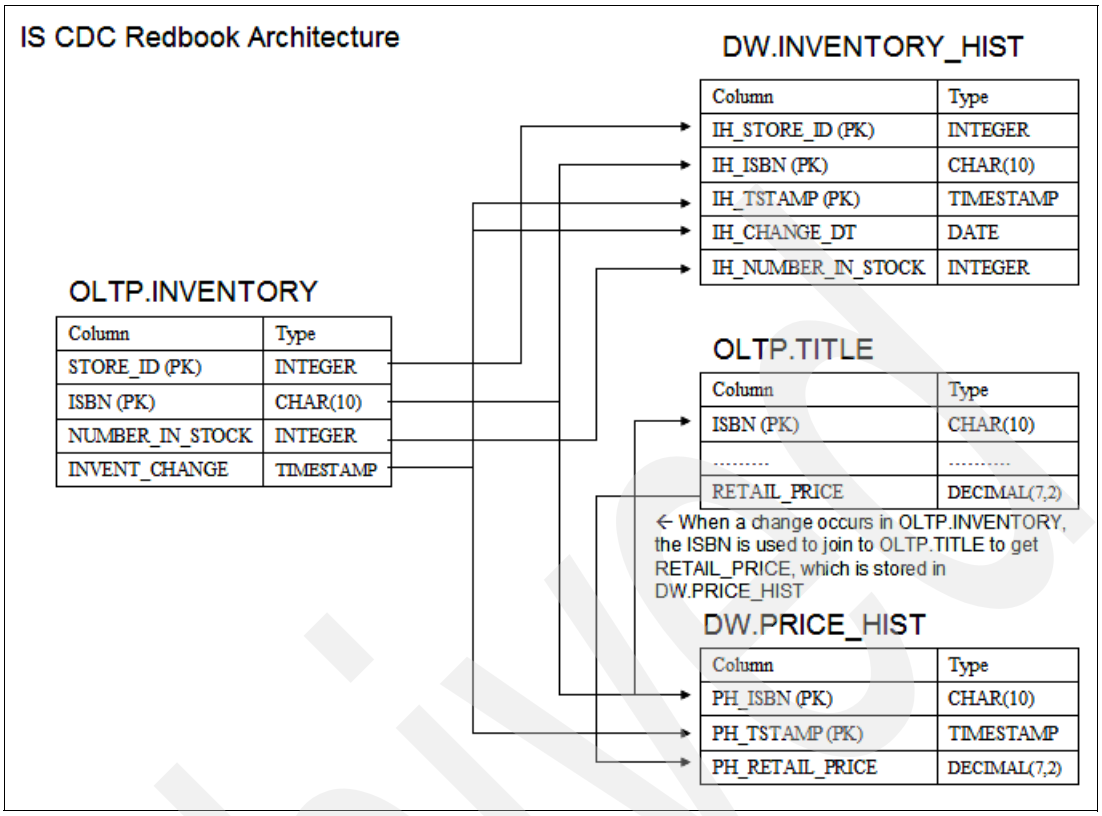


Figure 7-36 InfoSphere CDC workload design

The INVENTORY_HIST table provides a way to answer questions about how inventory levels have changed over time. We accomplish this by converting every insert or update to OLTP.INVENTORY into an insert into DW.INVENTORY_HIST. Notice that the primary key on INVENTORY_HIST has the additional column IH_TSTAMP to support this.

Likewise, the PRICE_HIST table tells us what the selling price of a book was at the time that its inventory level changed. This provides a way for us to answer questions about how discounting the sales price of a book affects sales of that book, and thus inventory levels. To accomplish this, we again convert all inserts and updates to OLTP.INVENTORY into inserts into DW.PRICE_HIST. Additionally, every time data in the INVENTORY table changes, we need to look up the current selling price of that particular book from OLTP.TITLE. This data is used to populate the PH_RETAIL_PRICE column in PRICE_HIST.

Lastly, it is worth pointing out that the primary keys used on both of these history tables can be used to join to the star schema data warehouse model described in 4.3.1, “Physical schema” on page 63. This can be used to generate even more complex queries.

To create these replication definitions, you first need to log in to Management Console with a CDC user ID that has been mapped to both the source and target CDC agents. For more information on CDC user IDs, agents, and user ID/agent mappings, see 7.3.2, “Configuration of InfoSphere Change Data Capture for z/OS” on page 149.

Creating subscriptions

If not already done, the first step is to create the necessary subscriptions. Each subscription is tied to a specific source and target CDC agent. Additionally, a DB2 table can only be used as a source once per subscription. Given that the INVENTORY table is the source for both INVENTORY_HIST and PRICE_HIST, this meant that we needed a minimum of two

subscriptions. We also made the decision to create separate subscriptions for doing full refreshes of the target tables, resulting in a total of four subscriptions required. For more information on creating subscriptions see 7.3.2, “Configuration of InfoSphere Change Data Capture for z/OS”. The subscriptions we created are shown in Figure 7-37.

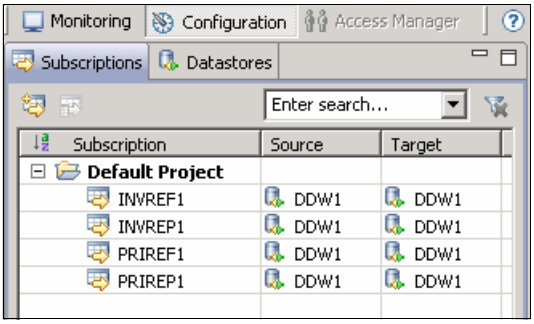


Figure 7-37 CDC workload subscriptions

- INVREF1 - Full refresh subscription for INVENTORY_HIST
- INVREP1 - Continuous replication subscription for INVENTORY_HIST
- PRIREF1 - Full refresh subscription for PRICE_HIST
- PRIREP1 - Continuous replication subscription for PRICE_HIST

Creating table mappings

Once the subscription is created, you can right-click it and select **Map Tables** to create a table mapping. A table mapping is a mapping between a source table and a target table as well as a definition of the data transformations that need to take place. When you create a new table mapping, the first panel will look as shown in Figure 7-38.

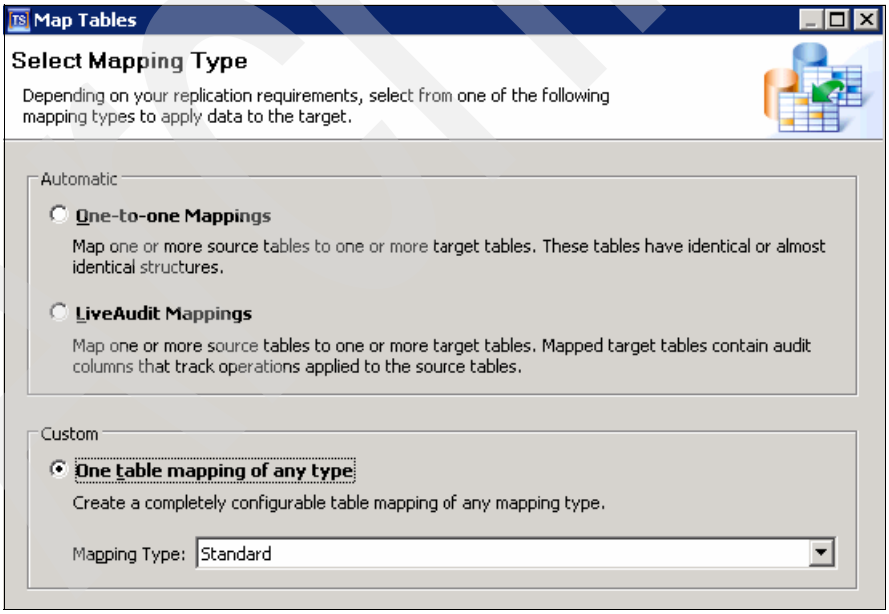


Figure 7-38 Table mapping types

The automatic mapping types are used to create a large number of table mappings. When creating individual table mappings, you will typically use “One table mapping of any type” and select the appropriate type from the list.

Available mapping types are:

- Standard - Under Standard replication, InfoSphere CDC applies the same operation that occurred on the source table to the target table. A row insert operation on the source table causes a row insert operation on the target table, and so on.
- LiveAudit - When you replicate data using LiveAudit, your target tables contain rows that track insert, update, delete, and clear (truncate) operations applied to the mapped source table. In other words, regardless of the action at the source, InfoSphere CDC switches the action to an insert at the target.
- Adaptive Apply - Adaptive Apply table mappings convert source inserts and updates to target updates or inserts (upserts). For example, if there is an insert on the source table, but that row already exists in your target table, InfoSphere CDC switches the insert to an update operation. Also, if there is an update on your source table, and this row does not exist on your target table, then InfoSphere CDC switches the update into an insert.
- Summarization - If you want to accumulate or deduct numeric values on the target, then map your source and target tables using the Summarization mapping type. There are two types of summarization: Accumulation and Deduction. Summarization table mappings ensure that numeric changes applied to the target columns are directly proportional to changes applied to the corresponding source columns.
- Consolidation one-to-one - If your business environment contains information that is scattered across different source tables, you may want to consolidate it to facilitate report generation, data management, or data security. To merge different information about a common entity into a single row, such as a person, a client, or a product part, map your source table to your target table using the Consolidation one-to-one mapping type.
- Consolidation one-to-many - Consolidation one-to-many table mappings are used where there are multiple source tables being consolidated to one target table, but where a change in one of the source tables may affect multiple rows in the target table. As an example, consider the case where data is being replicated from source to target. The source data has fields that contain code values, but the target table instead has descriptions. The mappings from code values to descriptions is done via a lookup table. A consolidation one-to-many table mapping could be used to not only replicate data from source to target, but also to update descriptions in the target table should descriptions change in the lookup table.

In general you would have multiple table mappings per subscription, with each table mapping having a different source table. In our scenario, we only had one source table for InfoSphere CDC, so each of our subscriptions only has one table mapping. Given this, we will refer to the subscription and the table mapping by the same name. For example, the INVREF1 table mapping is really referring to the table mapping (INVENTORY to INVENTORY_HIST) that belongs to the INVREF1 subscription.

For the INVREF1 table mapping, we used the Standard mapping type. This means that doing a full refresh of the INVENTORY_HIST table causes it to be truncated and loaded based on the current contents of the INVENTORY table. For the PRIREF1 table mapping, we used the Adaptive Apply mapping type. This is because of the fact that the primary key on the PRICE_HIST target table is (ISBN, TSTAMP) while the primary key on INVENTORY is (STORE_ID, ISBN). When we do a full refresh of PRICE_HIST, we want all of the rows in INVENTORY with the same ISBN to collapse to a single row in PRICE_HIST. Adaptive Apply allows us to do this with its update or insert (upsert) apply logic. If we had used a Standard mapping type for the PRICE_HIST refresh, we would get -803 SQL codes during refresh. For both of the replication table mappings (INVREP1 and PRIREP1), we used LiveAudit mapping types. This is because we wanted InfoSphere CDC to convert source inserts and updates to target inserts. To summarize:

- INVREF1 - Standard

- INVREP1 - LiveAudit
- PRIREF1 - Adaptive Apply
- PRIREP1 - LiveAudit

Let us continue with the table mapping creation. In the case of LiveAudit table mappings, you will next be asked to define the audit columns. Unless you will be using Management Console to create the target table, it is easier to define the auditing columns as part of the column mapping process. In this case, just delete all of the audit columns from the panel with Delete.

The next step in the table mapping process regardless of mapping type is to select the source table. The source table selection panel first lists database names, which can be expanded to show schemas, which can be expanded to show tables; see Figure 7-39.

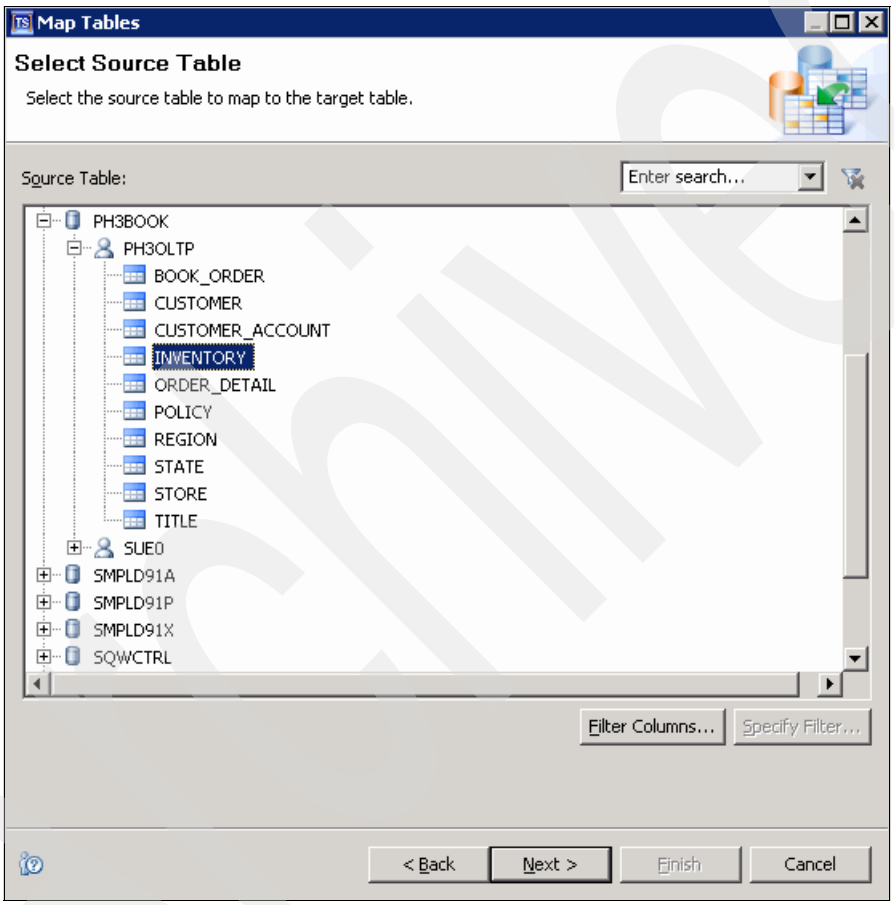


Figure 7-39 Selecting the source table

After selecting the source table, the next step is to select the target table. The target table selection panel works the same as the source table selection panel. If the target table does not yet exist, you can click **Create Table** to create the table using Management Console. Management Console can only create tables in existing table spaces. After selecting the target table, you are asked to select mirror or refresh as the replication method; see Figure 7-40 on page 189. Mirror means that the subscription can be used to do both full refreshes and continuous replication. Refresh means that the subscription can only be used to do full refreshes. The Prevent Recursion checkbox is used to prevent InfoSphere CDC from capturing and replicating changes that it has applied itself. This is required if you are trying to do bidirectional replication.

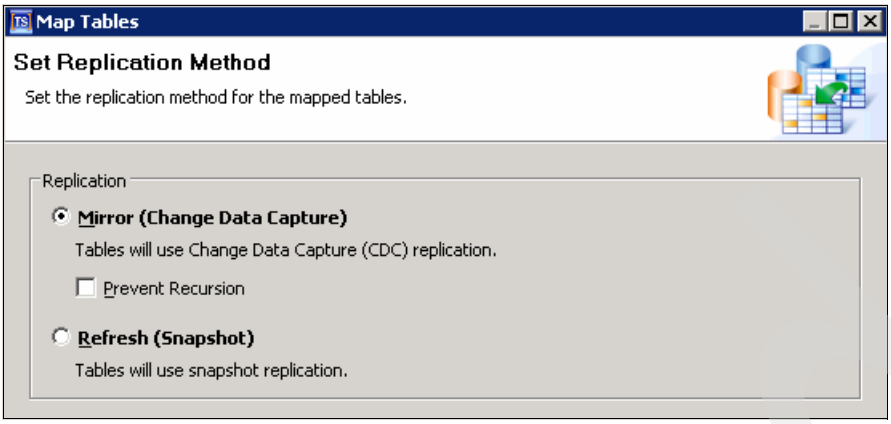


Figure 7-40 Selecting the replication method

At this point you are provided a summary of what you have defined so far, and you are asked what you would like to do next. Typically, you want to do the column mapping next, as that has not yet been completed; see Figure 7-41.

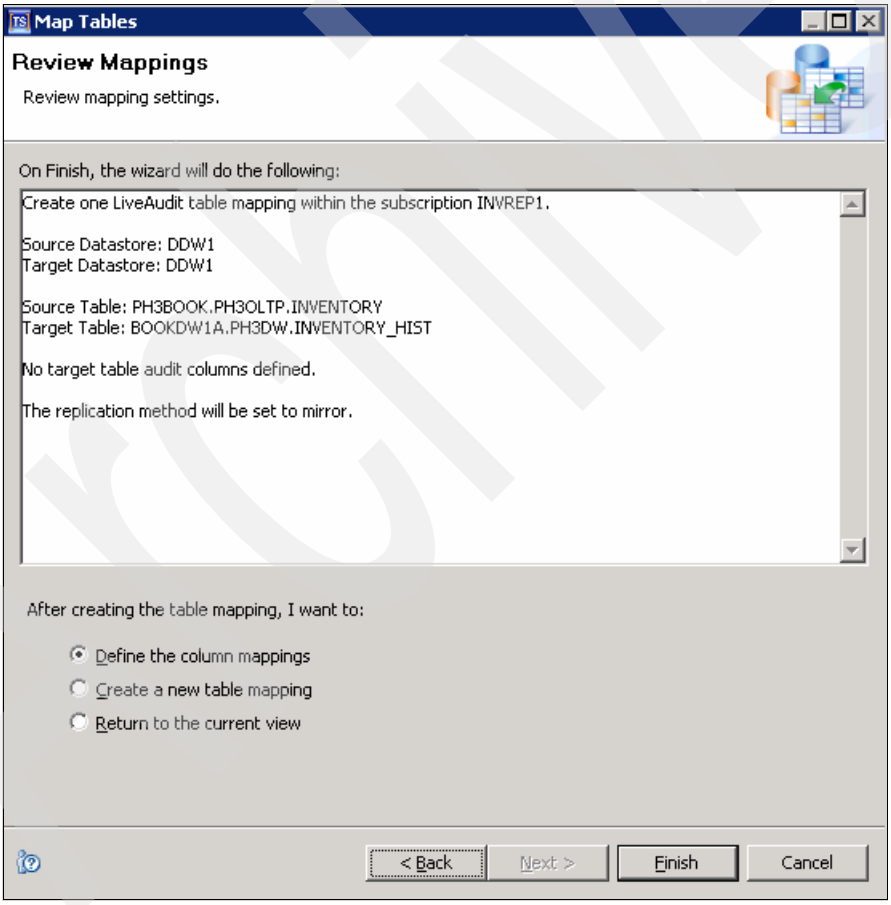


Figure 7-41 Review the table mapping definition

Creating the column mappings

The next step is to do the column mapping, which defines the data transformations that take place between source and target. By default, InfoSphere CDC automatically maps columns from source to target if they have the same name and data type.

Let us take a look at how we created the column mapping for the INVREF1 and INVREP1 table mappings. None of the column names in the target table matched the column names in the source table because of the IH_ prefix. So, initially the column mapping panel looked as shown in Figure 7-42.

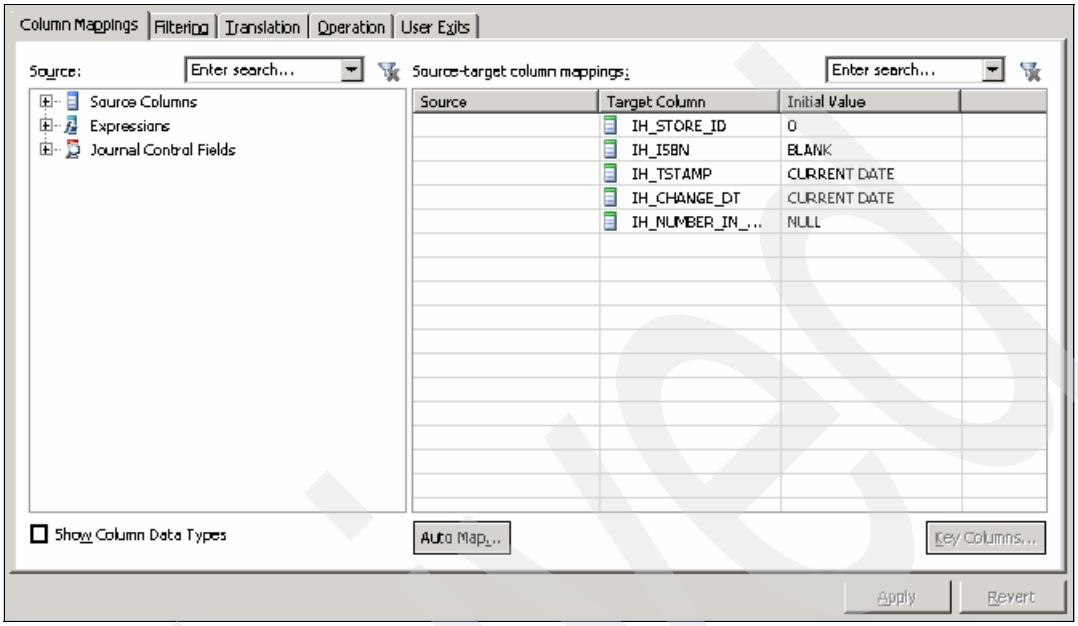


Figure 7-42 Column mapping pane

The simplest type of column mapping is to just use the initial value, which is a constant value that will be supplied for that column for every row applied to the target. Double-click the initial value to change it.

The next simplest type of column mapping is to directly map a column from the source table to a column in the target table. To do this, expand the source columns folder on the left side. Now, simply drag and drop the column into the appropriate row on the right side. Note that simple data type conversions, such as TIMESTAMP to DATE, are automatically inferred and handled by InfoSphere CDC.

The column mappings were the same for both INVREF1 and INVREP1 and consisted of nothing more complicated than dragging the appropriate source columns onto the appropriate target columns. Our final column mapping is shown in Figure 7-43 on page 191.

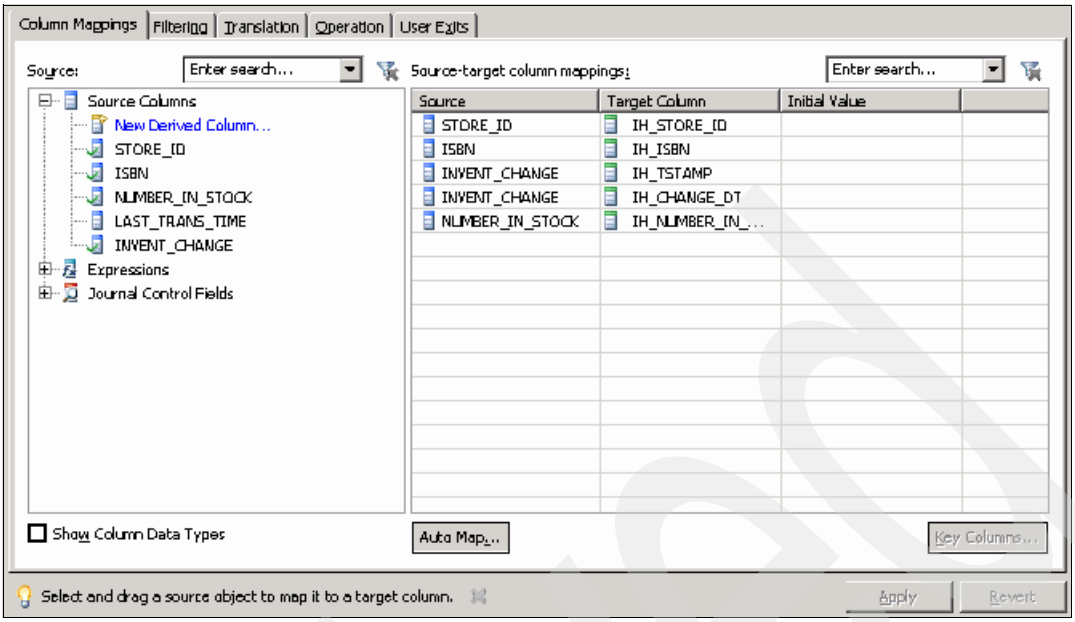


Figure 7-43 Completed column mappings for INVREF1 and INVREP1

Likewise, the column mappings were the same for both PRIREF1 and PRIREP1. However, for these we had to create a more complex transformation to look up the book price from the TITLE table.

Defining transformations

There are two ways to create more complex transformations. One method is to create a new derived column, which can be done under the source columns folder. The other method is to create a new expression under the expressions folder. Regardless of which method you choose, the transformation syntax is the same. The difference is that derived column transformations are executed by the source CDC agent, whereas expressions are executed by the target CDC agent. This is important when doing transformations such as lookups, where the lookup has to execute on the CDC agent that is attached to the database that has the lookup table.

For PRIREF1 and PRIREP1, we created a new derived column. The first thing that you need to do is to define a name and data type for the new derived column; Figure 7-44 on page 192.

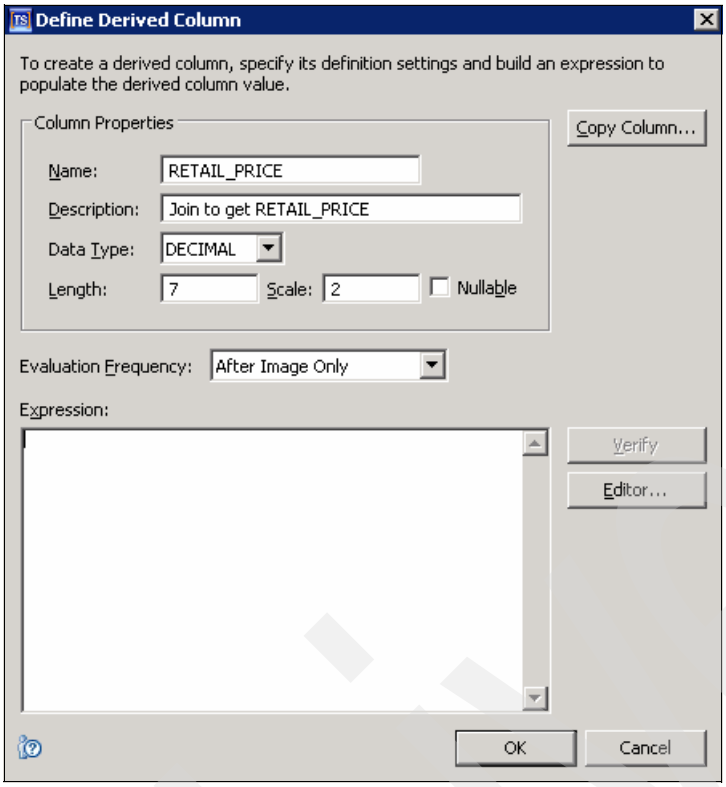


Figure 7-44 Creating a new derived column

An Evaluation Frequency of “After Image Only” is appropriate in almost all cases. An Evaluation Frequency of before and after images is only required if conflict detection will be required or if the target table is an audit table and you are auditing before images.

If you know the syntax for the transformation expression, you can just type it in the box on this panel. Otherwise, click **Editor** to bring up a wizard that can help you create transformations.

On the left side of the editor window, you have three folders. One folder contains all of the transformation functions that can be called. Another folder contains the source columns, which can be used as arguments to the transformation functions. The last folder contains journal control fields, which we will cover shortly. Arithmetic and boolean logic can be added by using the buttons on the bottom of the panel. When you finish building your transformation, you can verify your syntax with **Verify**. If this is a transformation you expect to reuse frequently, you should consider saving it by selecting **Save Expression**.

The transformation to look up the retail price from TITLE based on the ISBN of the row that changed in INVENTORY, was defined as shown in Figure 7-45 on page 193.

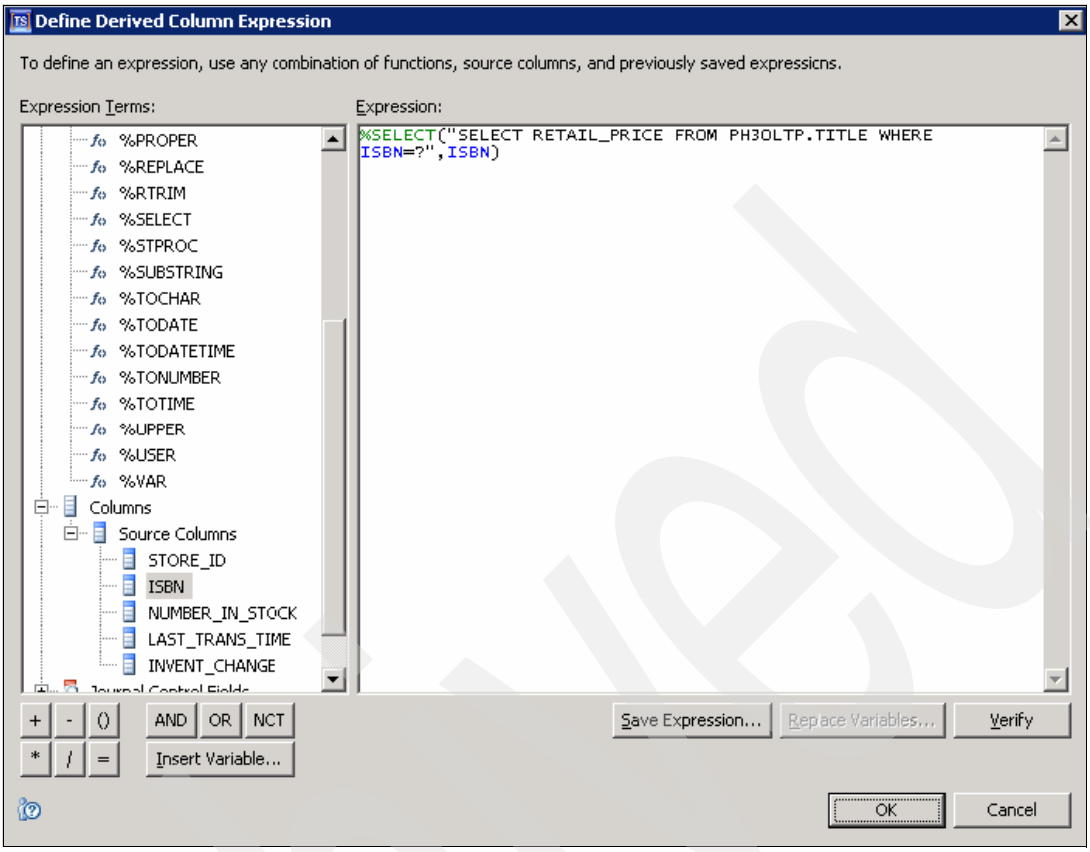


Figure 7-45 Using the %SELECT transformation to perform a lookup

Once the derived column or expression is created, it can be dragged and dropped on the appropriate row on the right side of the column mapping panel. Our final column mappings for PRIREF1 and PRIREP1 are shown in Figure 7-46.

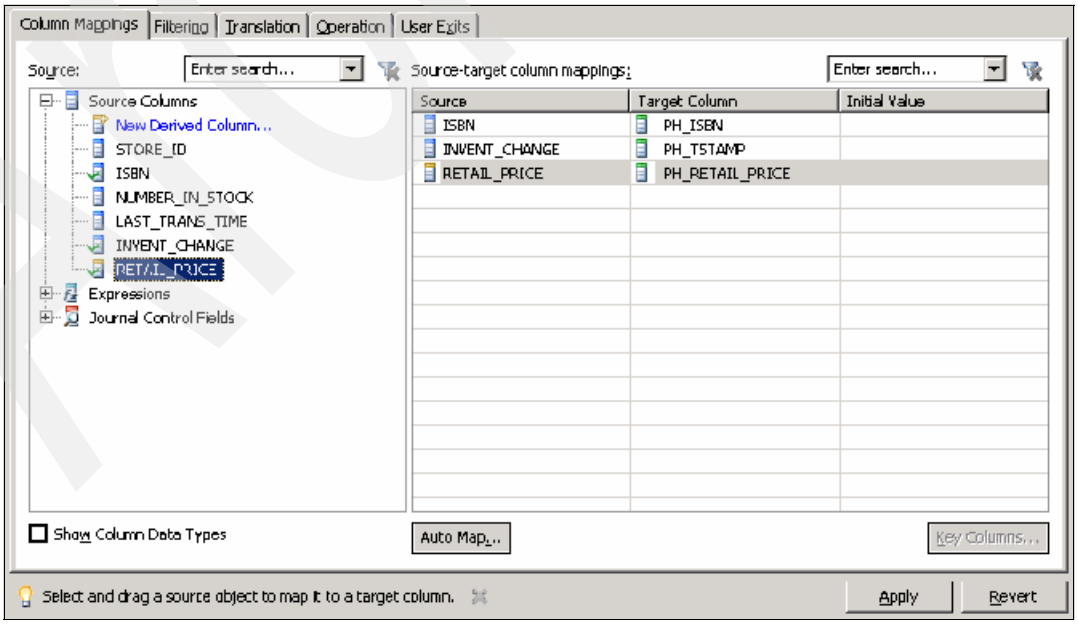


Figure 7-46 Completed column mappings for PRIREF1 and PRIREP1

Some other noteworthy transformation functions are:

- %BEFORE - the before value in case of an update
- %CONCAT - string concatenation
- %CURDATE - the current date
- %CURTIME - the current time
- %CURTMSTP - the current timestamp
- %IF - used to create conditional if/then/else logic
- %LOWER - convert to lower case
- %LTRIM - trim all leading blanks
- %PROPER - convert the first letter of each word to upper case, all other letters to lower case
- %REPLACE - string search and replace
- %RTRIM - trim all trailing blanks
- %SUBSTRING - create a new string from a substring of an existing string value
- %TOCHAR - convert to a string
- %TONUMBER - convert a string to a numeric type
- %UPPER - convert to upper case
- Numerous functions for creating and converting dates, times, and timestamps

Journal control fields

Notice that there is also a journal control fields folder on the column mapping panel. These journal control fields can be mapped directly to target columns or used as part of transformations. Journal control fields represent additional information that InfoSphere CDC can obtain from reading the database recovery log. Only certain journal control fields are available for certain source database types.

For DB2 z/OS, the following journal control fields are available:

- &CCID - An integer transaction identifier, only available during continuous mirroring.
- &CODE - Set to U when doing refresh and R when doing continuous mirroring.
- &ENTTYPE - Indicates the type of change that occurred at the source, that is, insert, update, or delete.
- &JOB - Contains the Logical Unit of Work's Correlation ID. The Correlation ID is an internal DB2 identifier, only available during continuous mirroring. It is usually the name of the job that created the Logical Unit of Work.
- &JOBUSER - Contains the user ID of the user who made the change at the source. If used during refresh, &JOBUSER is the user ID of the InfoSphere CDC address space.
- &JOURNAL - Contains the name of the source subsystem or data sharing group, only available during continuous mirroring.
- &LIBRARY - Contains the database name in which the table was created and the owner ID of the table in the format <dbname.owner>, only available during continuous mirroring.
- &OBJECT - Contains the name of the source table.
- &PROGRAM - Contains the associated DB2 plan name.
- &SEQNO - Contains the RBA or LRSN of the change during continuous mirroring. During refresh &SEQNO is set to zero.

Archived

- &SYSTEM - Contains the name of the source system. On z/OS, this is the name extracted from the CVT where InfoSphere CDC is running.
- &TIMSTAMP - Contains the timestamp of when the change was committed at the source during continuous mirroring. During refresh, &TIMSTAMP is set to the current date and time.
- &USER - The same as &JOBUSER, but only available during continuous mirroring.

The Filtering tab

The next tab after column mappings is the Filtering tab. On the left side is a text box where you can write WHERE clauses to do row filtering of the replicated data. Under the text box are buttons that control whether rows that match the WHERE clause are replicated or not replicated. As with creating new derived columns or expressions, there is an editor available to help you create filtering clauses.

On the right side of the Filtering window is a list of all the source columns followed by the checkboxes Replicate and Critical. The Replicate checkbox is used to specify which source columns should be replicated to the target. As we already described, you can manage this through the Column Mappings window instead. The Critical checkboxes control what constitutes a change. InfoSphere CDC will only process source updates where a Critical column has changed value. Otherwise, they will be ignored. Critical columns do not affect source inserts or deletes. By default, all columns are considered critical; see Figure 7-47.

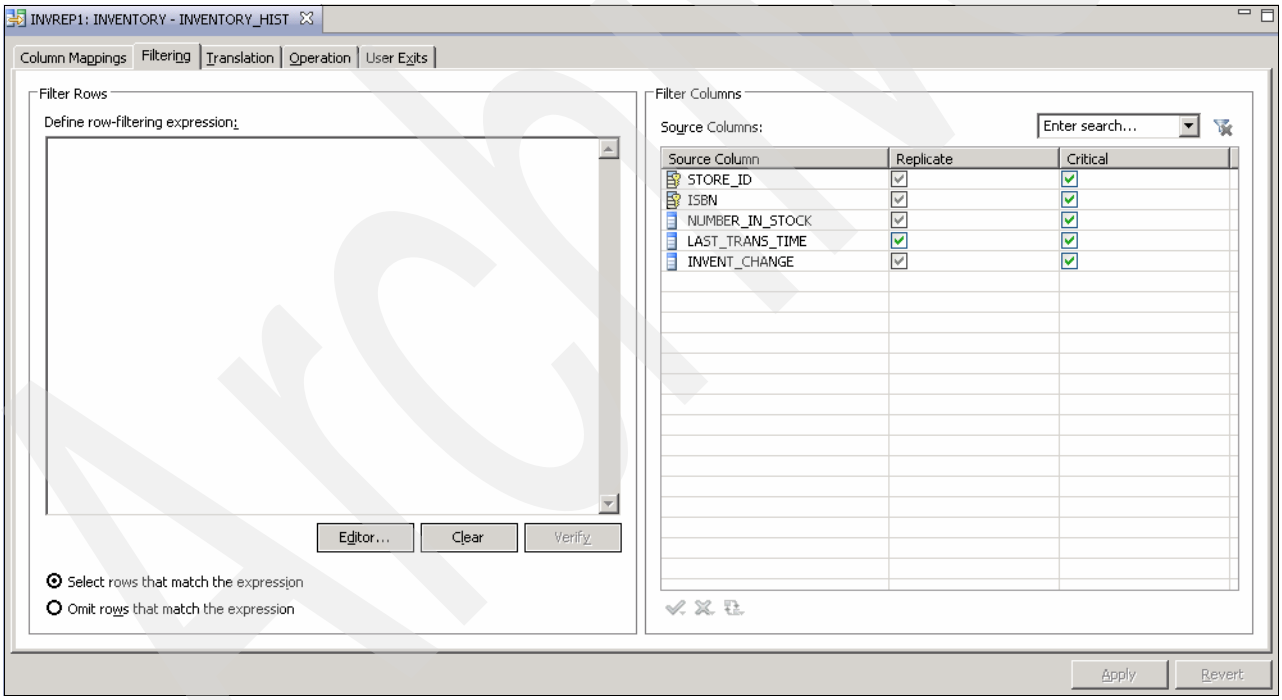


Figure 7-47 The Filtering window

The Translation tab

The next tab is the Translation tab; see Figure 7-48 on page 196. Here you can override the default codepage conversions that InfoSphere CDC will do. This is usually not necessary, because InfoSphere CDC determines the source and target codepages and automatically determines what codepage conversions need to be done. You can also use the Translation tab to create column level lookup tables. To do this, highlight the appropriate row on the left side, and then use **Add** on the right side to create mappings from incoming (source) field values to outgoing (target) field values. This essentially accomplishes the same task as using

%SELECT to do lookups from a code/value lookup table. However, this lookup table becomes part of the subscription table mapping definition. You can also import and export these CDC lookup table definitions in a CSV format using the Import and Export buttons.

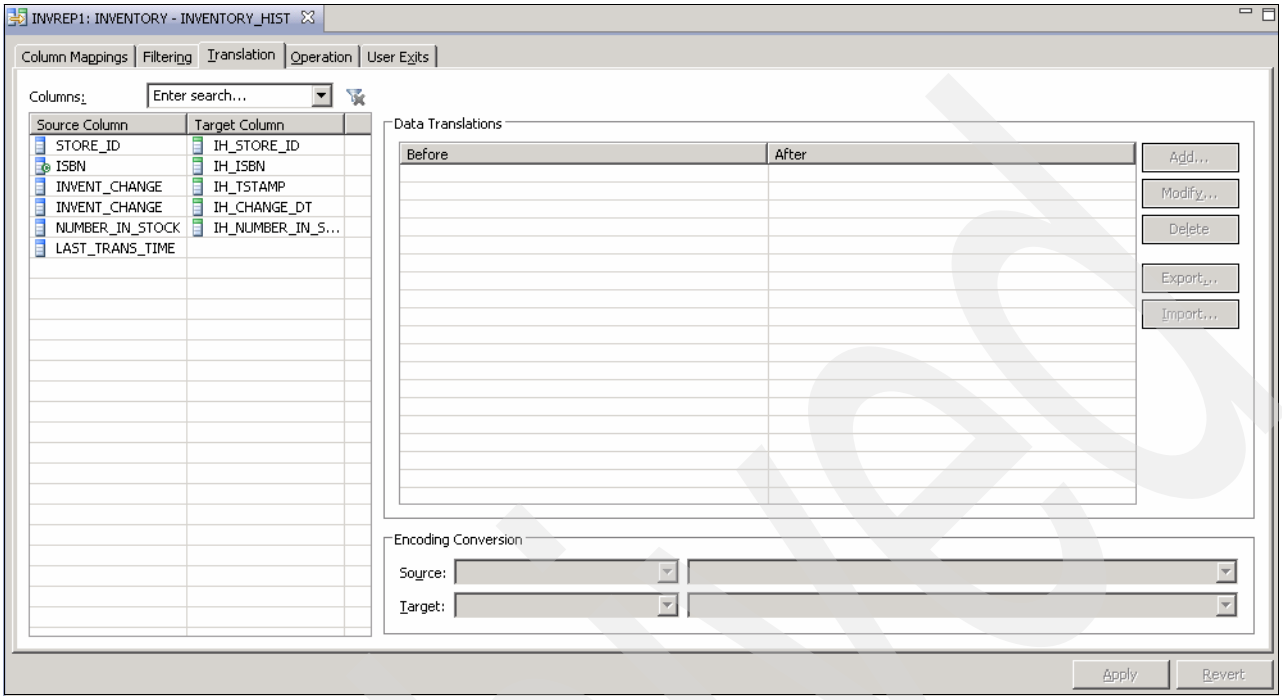


Figure 7-48 The Translation tab

The Conflicts tab

If the mapping is a standard mapping, the next tab will be the Conflicts tab; see Figure 7-49 on page 197. This tab is used to configure conflict detection and resolution rules. By default, conflict detection is off. Conflict detection and resolution is generally associated with bidirectional replication. However, it can also be used in unidirectional replication. For example, if an update occurs at the source, both the before image and after image are captured from the database log. If conflict detection is turned on for a column, the before image value of that column is compared to the current value of that column in the target database. If they do not match, this is a conflict and conflict resolution rules come into play. If they do match, the after image gets applied to the target via an UPDATE SQL statement.

There are several options for conflict resolution rules, including the option to call a user exit for more complex custom resolution scenarios. Resolution options are shown in Figure 7-49 on page 197.

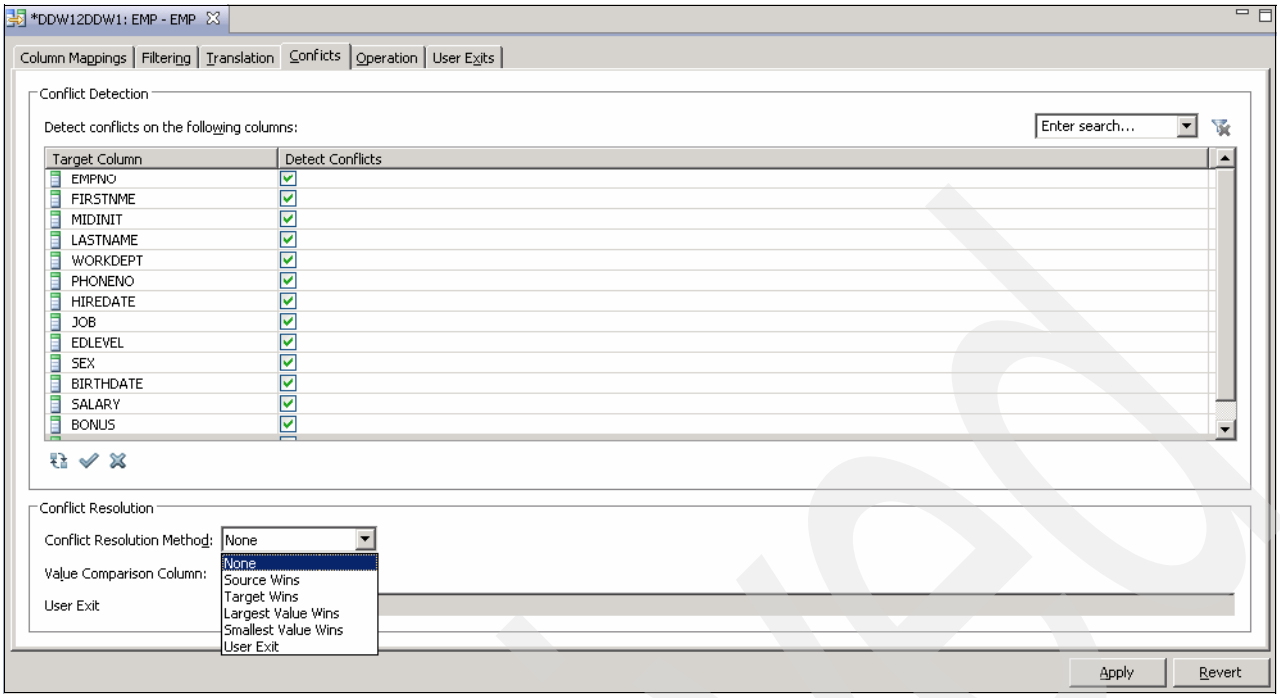


Figure 7-49 The Conflicts tab

The Operation tab

The next tab is the Operation tab. This tab allows for some fine tuning of the behavior of the table mapping. For example, the Operation tab settings for our INVREP1 and PRIREP1 table mappings where as shown in Figure 7-50.

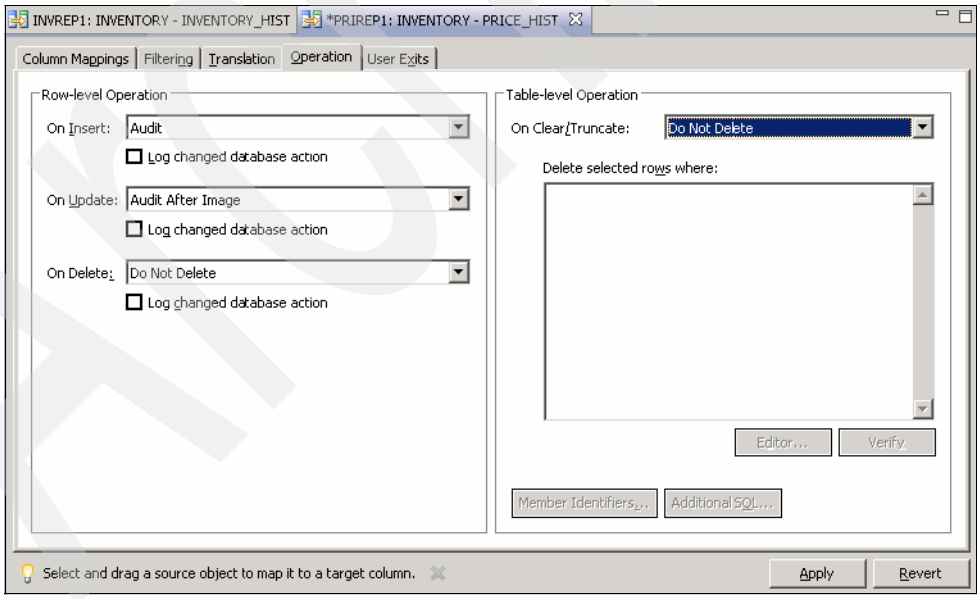


Figure 7-50 The Operation tab

To reiterate, INVREP1 and PRIREP1 were both LiveAudit table mappings. On the Operations tab, we overrode some of the default behavior. We specified that on source insert we wanted to insert a new row into the target table. This is the default behavior. However, on update, we only wanted to insert a new row into the target table based on the after image. The default is

to insert two rows into the target, one for the before image and one for the after image. In the case of a source delete or truncate, we specified that we wanted InfoSphere CDC to take no action. Again, this overrides the default of inserting rows into the target for every source row that was deleted or truncated.

The same thing could be done with a standard table mapping using the operation tab. For example, we could specify default behavior for source inserts and updates, namely that they propagate to the target. But we could override the default behavior for deletes and specify that on source delete, no action should be taken at the target. With standard table mapping types, the clear/truncate list gives you control over whether all, a subset, or none of the target table rows should be deleted before doing a full refresh of the target table.

The User Exit tab

The last tab is the User Exits tab; see Figure 7-51. This tab is used to specify user exit modules that should be called when certain operations occur on this table mapping. The actions available take place at the target CDC agent. For example, a Before Update user exit gets called before the target CDC agent processes a row update. There are “before” user exits and “after” user exits. You cause a user exit to execute in place of the normal target agent response by providing a user exit and disabling the normal target agent action in the operation tab.

Depending on the target database, user exits can be written in various languages including COBOL, Visual Basic, C/C++, and Java. If the target CDC agent is a DB2 z/OS agent, the user exit name corresponds to the name of a load module in the STEPLIB of the started task.

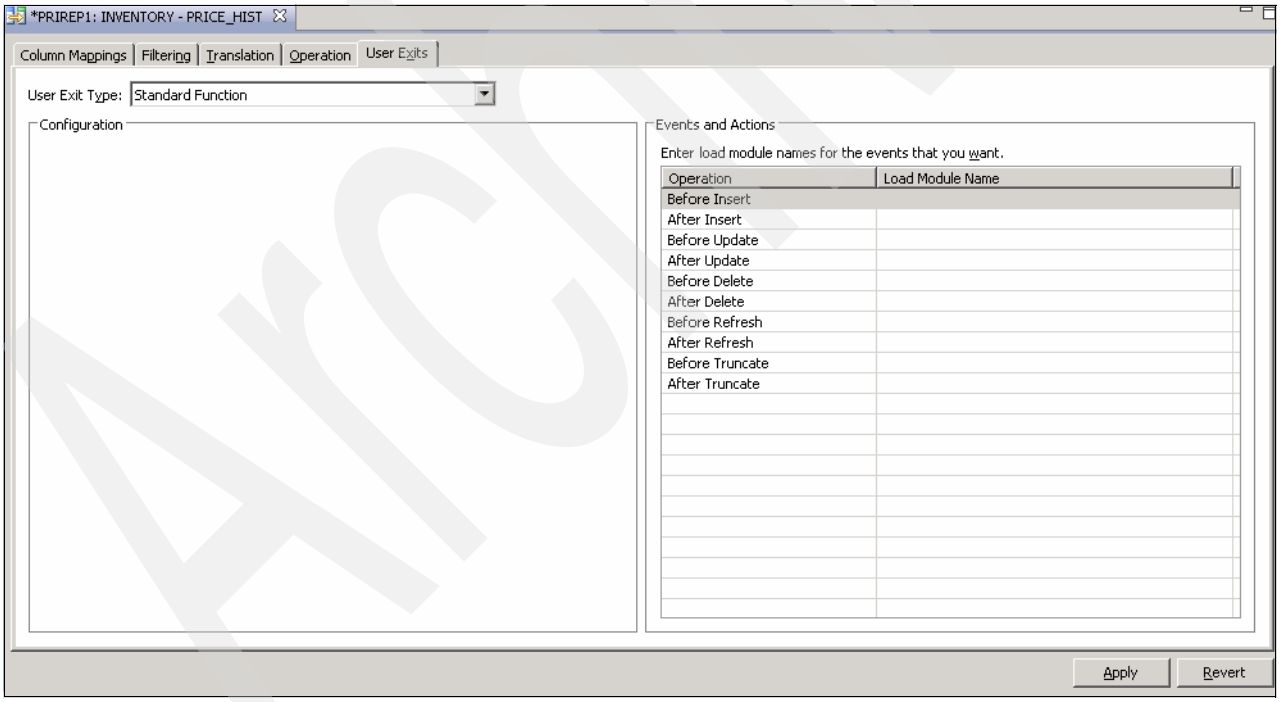


Figure 7-51 The User Exits tab

7.4.2 Operations and monitoring

Starting and stopping subscriptions

Now that the subscriptions and table mappings have been created, we are ready to put them to work. The simplest operation against a subscription is to start it; see Figure 7-52. To start a

subscription, right-click it and select **Start Mirroring (Continuous)**. If the subscription only contains refresh table mappings, you would instead select **Start Refresh**.

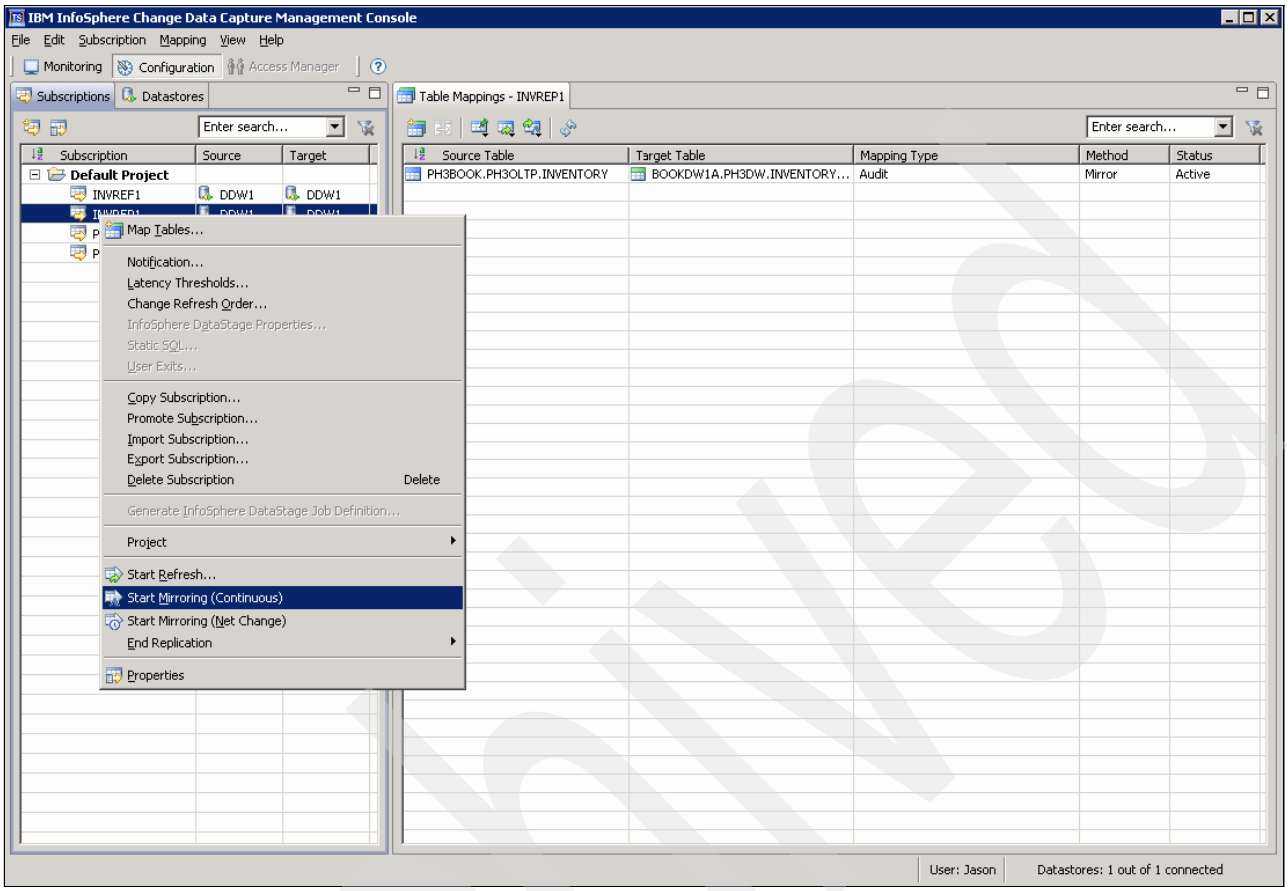


Figure 7-52 Starting a subscription

To stop a subscription, right-click it and select **End Replication → End Replication (Controlled)**. It may take a little while for the subscription to stop. If a controlled stop of the subscription fails to work, the subscription can be abruptly terminated by selecting **End Replication → End Replication (Immediate)**.

When starting subscriptions, pay close attention to the information in the table mappings window. There are various states that each table mapping can be in. These show up under the status column.

- Active - When the subscription starts, this table mapping picks up where it left off, controlled by the bookmark, replicating data from source to target.
- Parked - When the subscription starts, changes for this table mapping are not replicated.
- Refresh - When the subscription starts, the table will undergo a full refresh. When the full refresh completes, the table mapping begins active continuous mirroring.

If a table mapping does not have the desired status, you can set it to the desired status prior to starting the subscription.

- To set a status of Active - Right-click the table mapping and select **Park (Do Not Replicate)**. Right-click the table mapping again and select **Mark Table Capture Point for Mirroring**; see Figure 7-53. Note that this causes the bookmark to be destroyed. Replication begins from the current log position. This should only be done during a time when it can be guaranteed that no changes are occurring to the source table.

- To set a status of Parked - Right-click the table mapping and select **Park (Do Not Replicate)**. When restarting the table mapping, it can be set to a status of Active or a status of Refresh. If the table mapping is restarted by setting it to Active, all changes that occurred at the source while the table mapping was Parked will be lost.
- To set a status of Refresh - Right-click the table mapping and select **Flag for Refresh**.

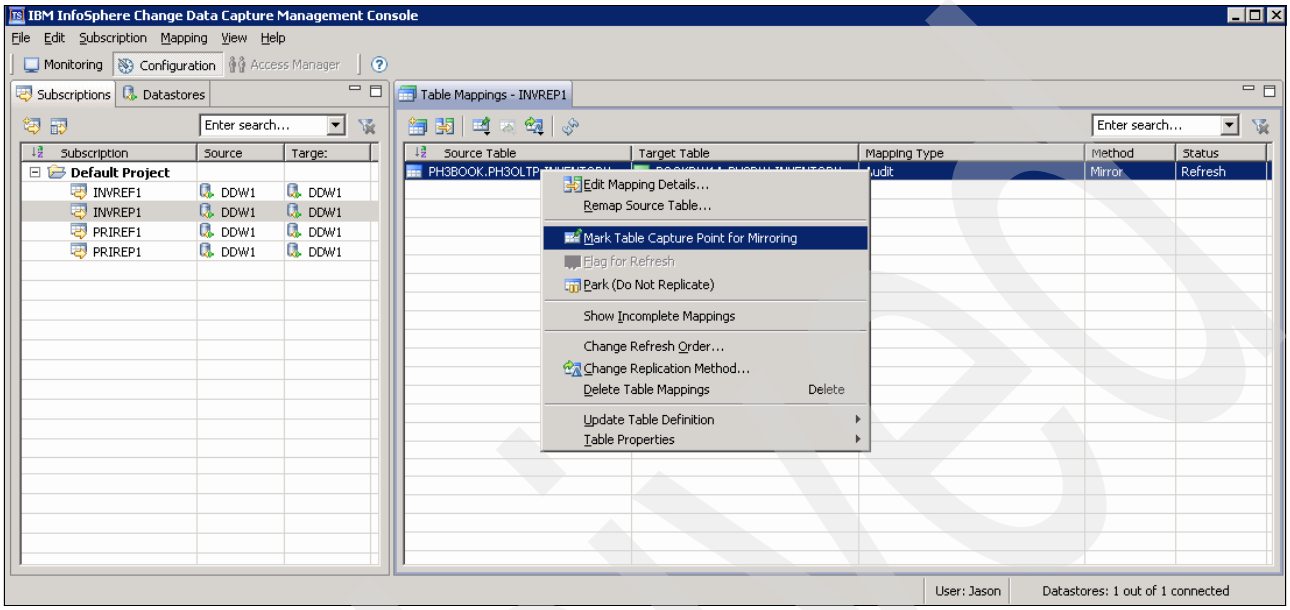


Figure 7-53 Setting a table mapping to Active

The Event log

Management Console also provides some problem determination and monitoring capabilities. Most of these are found by selecting the Monitoring tab. On this tab, you see all of your subscriptions as well as their current state and status. The State column tells you whether the subscription is currently active (started) or inactive (stopped). The Status column tells you whether the subscription is normal or in error.

If a subscription has a status of error, you can investigate the cause of the error by right-clicking the subscription and selecting **Show Event Log**. The Event log (Figure 7-54 on page 201) shows you informational, warning, and error messages from both the source and target CDC agents. Use the drop-down list to choose to look at messages from the source agent or from the target agent. Also use the Refresh icon in the Event log viewer to get the latest messages.

For a DB2 z/OS CDC agent, these same messages may also be available in the system log or in the CHCPRINT DD of the started task. This is configurable at the subscription level. For more information about configuring notification settings, see 7.3.2, “Configuration of InfoSphere Change Data Capture for z/OS” on page 149.

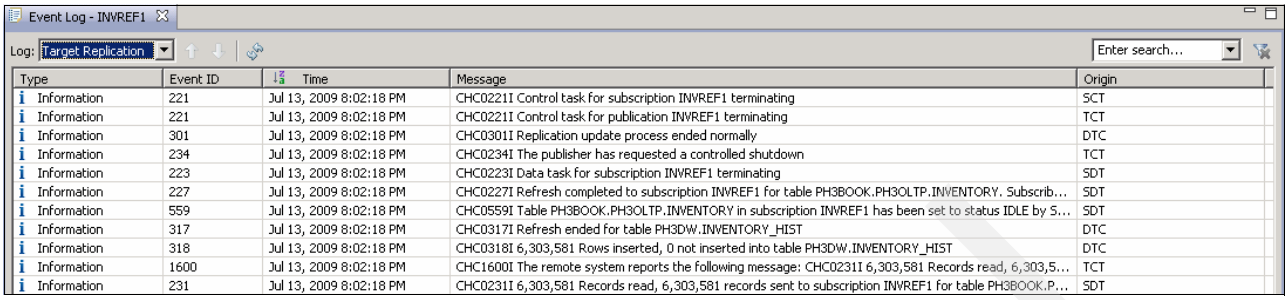


Figure 7-54 The Event log

The Monitoring tab also contains a real-time monitor that provides latency and throughput information at the subscription level. For more information about the real-time monitor, see 7.3.3, “Tuning InfoSphere Change Data Capture for z/OS” on page 166.

Setting latency thresholds

InfoSphere CDC allows you to configure two latency thresholds per subscription. Messages are generated whenever subscription latency crosses the latency threshold, both in the increasing and decreasing directions.

Subscription latency messages are always written by the target CDC agent. If the target agent is a DB2 z/OS agent, you have the option to have these messages written to the system log and/or the CHCPRINT DD in the started task.

To configure latency thresholds, click the **Subscriptions** tab and the **Configuration** subtab. Next, right-click the appropriate subscription and select **Latency Thresholds**. Fill in the two latency threshold values as appropriate; see Figure 7-55.

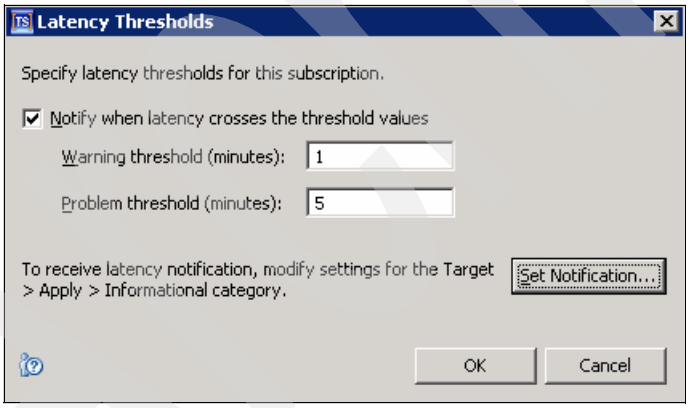


Figure 7-55 Specifying latency thresholds

Clicking **Set Notification** takes you to a panel where you can set the targets for informational target apply messages. Because our target CDC agent was a DB2 z/OS agent, we chose to send latency threshold messages to both CHCPRINT and the system log, as shown in Figure 7-56 on page 202.

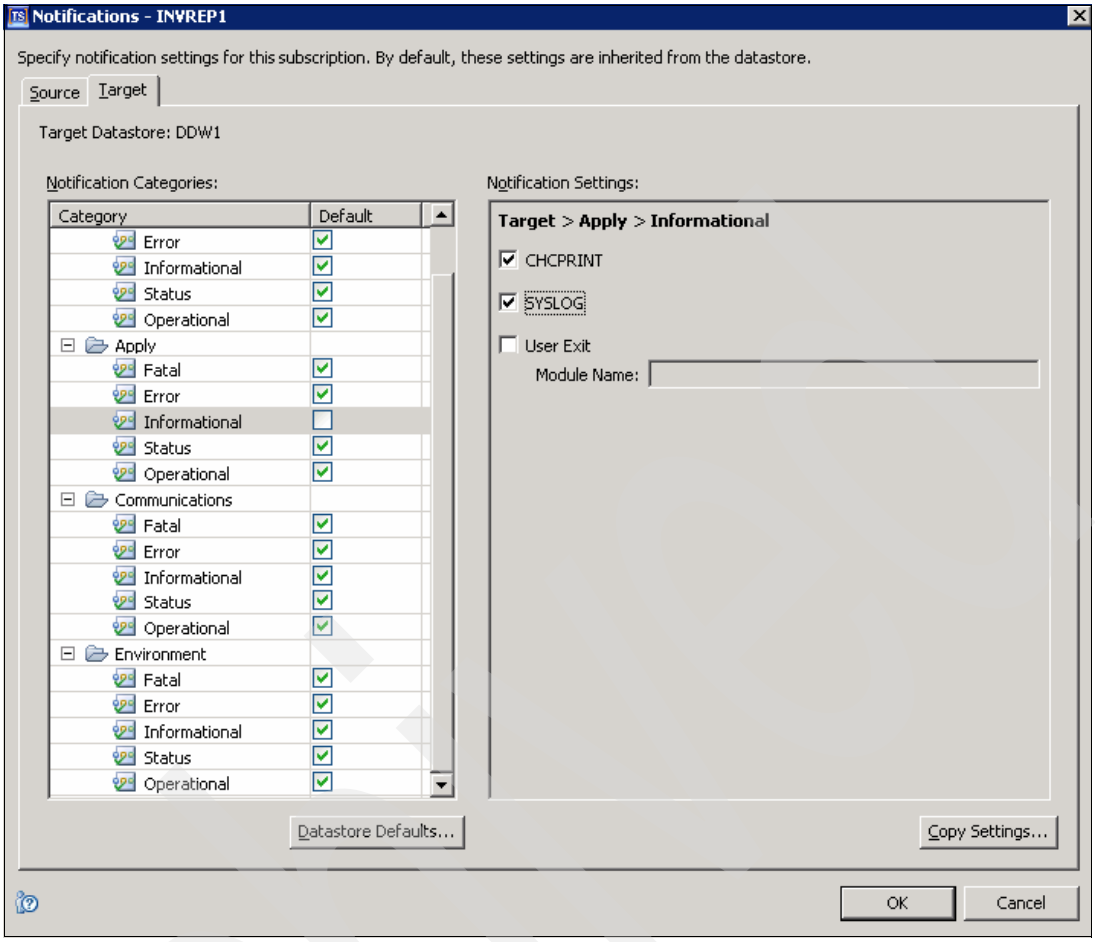


Figure 7-56 Setting the destinations for latency threshold messages

The latency threshold messages look as shown in Example 7-22.

Example 7-22 Latency threshold message format

```
CHC5104I Latency for journal jrn has crossed {above|below} the
{warning|problem} threshold of min minute(s)
```

Persistency

Persistency is a subscription attribute that can be enabled or disabled. By default, persistency is disabled. Enable this attribute for the subscriptions that you want to be automatically restarted after a normal or abnormal termination. By default, subscriptions are placed in an inactive status after such events and require manual intervention to restart them. Only subscriptions that were mirroring are automatically restarted. Subscriptions that were doing a refresh are not automatically restarted.

Persistent subscriptions can automatically restart in response to a normal or abnormal termination of the following:

- InfoSphere Change Data Capture address space - For persistent subscriptions that were active and mirroring when the termination occurred, continuous mirroring automatically restarts when the address space reinitializes. However, persistent subscriptions that were stopped as a result of a direct request are not restarted automatically during address space initialization.

- DBMS - For persistent subscriptions that were active when the termination occurred, continuous mirroring automatically restarts when the DBMS reinitializes.
- Communications - For persistent subscriptions that were active when the termination occurred, continuous mirroring automatically restarts when communications are reestablished.

InfoSphere Change Data Capture attempts to automatically restart continuous mirroring for persistent subscriptions at regular intervals. Attempts continue until an automatic restart is successful or the persistent subscription or InfoSphere Change Data Capture address space is terminated. You can set how often InfoSphere Change Data Capture attempts to automatically restart continuous mirroring for all persistent subscriptions by modifying the AUTORESTARTINTERVAL configuration control statement keyword. For more information about this keyword, see 7.3.3, “Tuning InfoSphere Change Data Capture for z/OS” on page 166.

The persistency attribute is set at the source CDC agent using the CHCMTAUT utility. In our scenario, we enabled persistency on both the INVREP1 and PRIREP1 subscriptions with the following JCL shown in Example 7-23.

Example 7-23 Enabling persistency

```
//CHCMTAUT JOB MSGLEVEL=(1,1),
//            CLASS=A,
//            MSGCLASS=X,
//            REGION=0M,
//            NOTIFY=&SYSUID
//*
//MTAUT    EXEC PGM=CHCMTAUT,
//            PARM='DDW1 CHCMTA01'
//STEPLIB  DD DSNAME=ICDC.ZDW.SCHCLOAD,DISP=SHR
//         DD DSNAME=DB2910.ZDW.SDSNEXIT,DISP=SHR
//         DD DSNAME=DB2910.ZDW.SDSNLOAD,DISP=SHR
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  PERSISTENCY SUBSCR(INVREP1) ON
  PERSISTENCY SUBSCR(PRIREP1) ON
  END
/*
//
```

Overriding refresh settings

The CHCMTAUT utility also has one other use. In 7.3.3, “Tuning InfoSphere Change Data Capture for z/OS” on page 166, we discussed the USELOADER configuration setting that controls the default refresh behavior for subscriptions using that CDC agent as the target. USELOADER=NO causes refreshes to be done using SQL statements. USELOADER=YES causes InfoSphere CDC to use the LOAD utility to do refreshes. Note that there are restrictions as to when a subscription is eligible to use the DB2 LOAD utility for refresh. The restrictions can be found in the InfoSphere CDC for z/OS user documentation.

CHCMTAUT provides a way to override the agent level default for USELOADER at the subscription level. In our scenario, we set USELOADER to YES, and then overrode the default for the PRIREF1 subscription. Our final configuration was as follows:

- INVREF1 - Uses the LOAD utility for refresh.

- INVREP1 - Never used for refresh.
- PRIREF1 - Uses SQL for refresh.
- PRIREP1 - Never used for refresh.

We used the JCL shown in Example 7-24 to override the USELOADER setting for the PRIREF1 subscription.

Example 7-24 Overriding the agent level USELOADER setting

```
//CHCMTAUT JOB MSGLEVEL=(1,1),
//            CLASS=A,
//            MSGCLASS=X,
//            REGION=0M,
//            NOTIFY=&SYSUID
//*
//MTAUT    EXEC PGM=CHCMTAUT,
//            PARM='DDW1 CHCMTA01'
//STEPLIB  DD DSNAME=ICDC.ZDW.SCHCLOAD,DISP=SHR
//         DD DSNAME=DB2910.ZDW.SDSNEXIT,DISP=SHR
//         DD DSNAME=DB2910.ZDW.SDSNLOAD,DISP=SHR
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  LOADER PUBLSH(INVREF1) USE(SYSTEM)
  LOADER PUBLSH(PRIREF1) USE(NO)
  END
/*
//
```

7.4.3 InfoSphere CDC as a DB2 workload

As a DB2 workload, InfoSphere CDC is mostly an INSERT/UPDATE/DELETE workload running in the subsystem that the target CDC agent is attached to. In our testing, we ran in two different scenarios. The first scenario had one CDC agent, acting as both source and target, attached to member DDW1 of data sharing group DDWG. The second scenario had separate source and target CDC agents, with each being attached to a member of two separate data sharing groups.

From an INSERT/UPDATE/DELETE workload perspective, the largest factor to consider is COMMITFREQ, which controls the unit of work size. See 7.3.3, “Tuning InfoSphere Change Data Capture for z/OS” on page 166 for more information about COMMITFREQ and its effect on processor usage.

However, InfoSphere CDC can also act as a SELECT workload at the source or target agent. InfoSphere CDC SELECT workloads can occur for two different reasons:

- Refresh - Regardless of the USELOADER setting, InfoSphere CDC uses a SELECT statement to gather the data required to refresh the target.
- Transformations involving lookups - If the %SELECT or %GETCOL transformation functions are used for lookups, a SELECT workload results. This workload is at the source in the case of derived columns and at the target in the case of expressions.

For example, let us look at how the DB2 processor utilization differs between INVREF1 and PRIREF1 when doing a refresh. Both subscriptions and table mappings have the same source table for refresh, namely the OLTP INVENTORY table. For these refresh tests, the

INVENTORY table had 6,303,581 rows. The main difference between INVREF1 and PRIREP1 is that PRIREP1 has a %SELECT transformation function that looks up price information from the OLTP TITLE table.

The initial SELECT statements against INVENTORY (Table 7-4) were similar for both INVREF1 and PRIREF1.

Table 7-4 SELECT against INVENTORY when doing a full refresh (6.3M rows)

Subscription	DB2 z/OS CPU (seconds)	Elapsed Time (seconds)
INVREF1	12.615503	20.23145
PRIREF1	13.132127	27.145842

However, PRIREF1 has to do a SELECT against TITLE for every row in INVENTORY (Table 7-5). This adds up to a fairly substantial processor and elapsed time cost.

Table 7-5 SELECTs against TITLE when refreshing PRICE_HIST (6.3M rows)

Subscription	DB2 z/OS processor (seconds)	Elapsed Time (seconds)
PRIREF1	210.587868	511.955332

We also ran a continuous mirroring test (Table 7-6) where we captured and replicated one million changes to the INVENTORY table. Again the processor cost of doing the INSERTs to apply the changes was very similar between INVREP1 and PRIREP1.

Table 7-6 DB2 z/OS CPU used by Apply processing during continuous mirroring (1M changes)

Subscription	DB2 z/OS processor (seconds)
INVREP1	24.236078
PRIREP1	24.976297

However, PRIREP1 also has to do a SELECT statement against the TITLE table for every change to the INVENTORY table (Table 7-7). The cost of the %SELECT transformation is again substantial.

Table 7-7 SELECTs against TITLE during continuous refresh (1M changes)

Subscription	DB2 z/OS processor (seconds)
PRIREP1	74.213765

Archive

Archived

8

Our test configuration and infrastructure

This chapter discusses the following topics and considerations:

- Hardware and software
- IBM Cognos BI for Linux on System z

Archived

© Copyright IBM Corp. 2010. All rights reserved.

8.1 Hardware and software

This section describes the hardware, operating systems, subsystems, and storage that we used for this System z co-location project. We built two distinct configurations for use during this residency. The first was the single z/OS configuration shown in Figure 8-1.

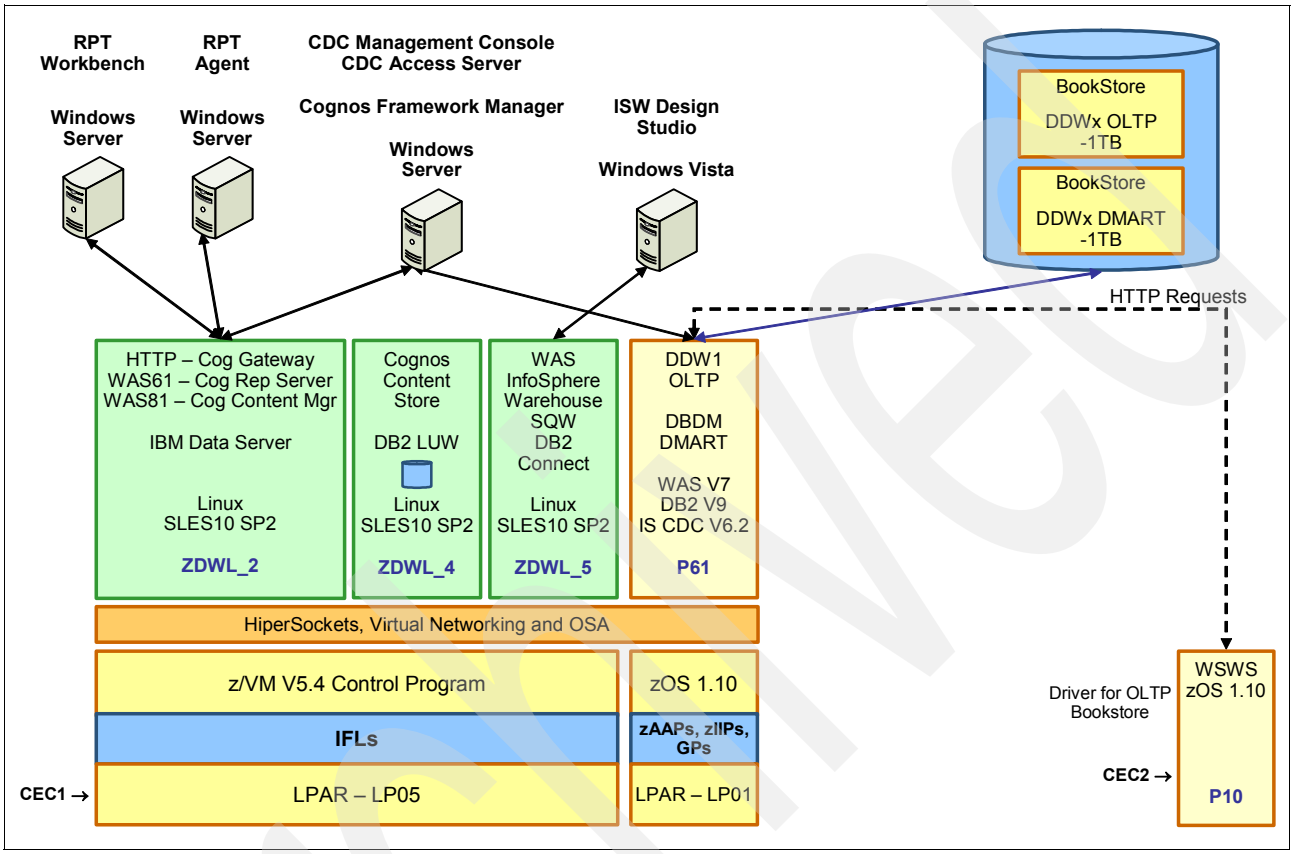


Figure 8-1 Single z/OS LPAR configuration

The second was the multiple z/OS LPAR configuration shown in Figure 8-2 on page 209.

Archived

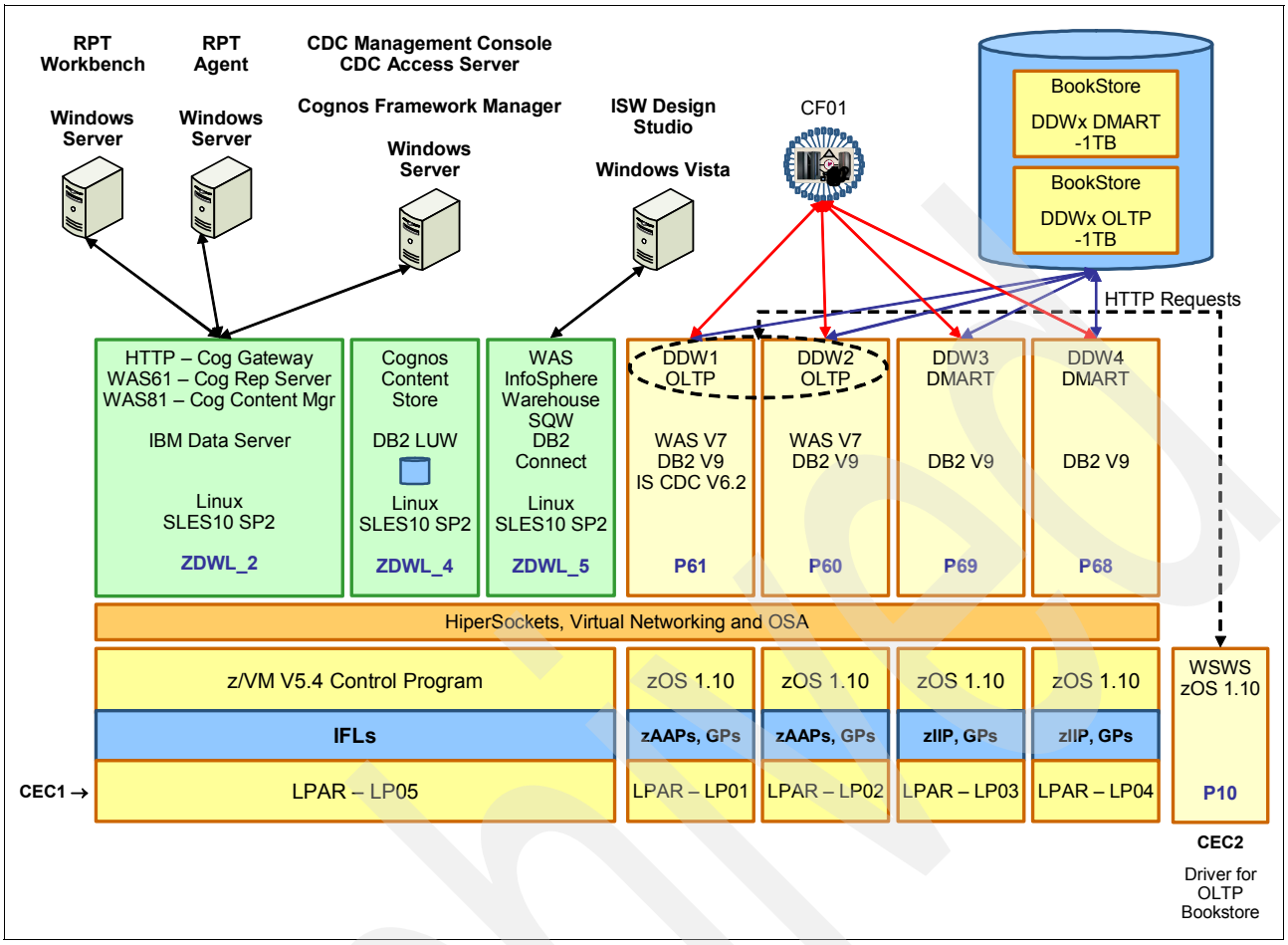


Figure 8-2 Multiple z/OS LPAR configuration

WebSphere was a key part of the configuration. Details about the WebSphere topology are not included in Figure 8-2 but are shown in Figure 8-3 on page 210.

Archive

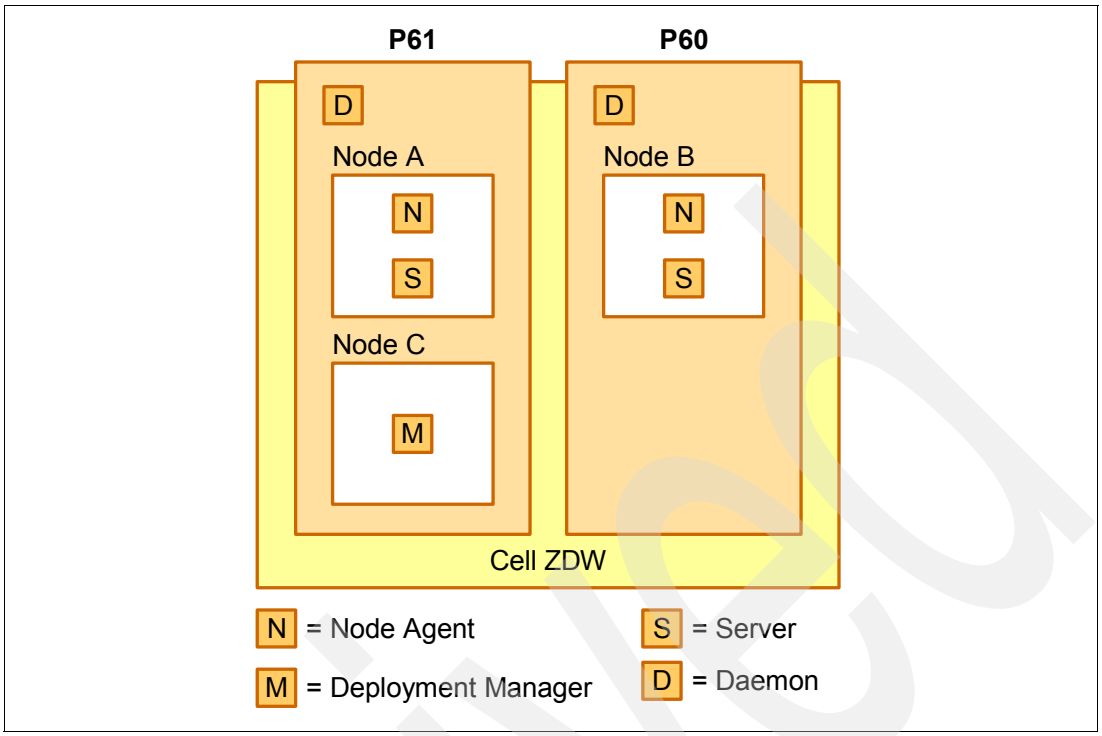


Figure 8-3 Multiple z/OS LPAR WebSphere Application Server topology

8.1.1 Hardware configuration

We used an IBM System z10 Enterprise Class processor, model 2097-E64. In addition to general purpose processors, we configured specialty engines, including an Integrated Information Processor (zIIP), an Application Assist Processor (zAAP), and an Internal Coupling Facility (ICF).

The number of general purpose and specialty engines used for the z/OS LPARs varied during the project. The engines assigned to the LPAR where z/VM and the three Linux guests ran was constant. Eight dedicated general purpose engines, no specialty engines, and 16 GB of central storage were assigned to this LPAR throughout the project.

In our multiple z/OS configuration, zAAPs were configured on the two LPARs where z/OS systems P60 and P61 were IPLed. The transactional workload was installed in WebSphere on these two systems; much of this workload could be successfully offloaded to zAAPs. zIIPs were configured on the LPARs where systems P58 and P59 were IPLed. (See the rightmost two LPARs in could be successfully offloaded to zIIPs.

System P10 ran on a single LPAR on a second z10 processor. It was used to drive the BookStore transactional workload. This was accomplished using a product called WebSphere Studio Workload Simulator or WSWS. WSWS sent HTTP requests to WebSphere running on systems P60 and P61 using a gigabit Ethernet connection.

The database resided on a single DS8300 model 2421-932. All of the logical volumes configured on the DS8300 were 3390 mod 54 types with about 55 GB of storage on each volume. Connectivity between the z10 and the DS8300 was provided by fourteen FICON channels.

The configuration included four IBM System x workstations. Rational Performance Tester 8.1 or RPT was installed on two of the IBM System x workstations. These workstations were used

to drive the IBM Cognos BI workload. The RPT Workbench component ran on a model x 3650 with 16 GB of RAM. The RPT Agent component ran on a model x 3850 with 3.25 GB of RAM. These two workstations are shown in the upper left corner of Figures 8-1 and 8-2 on page 209.

The next workstation served multiple purposes. The Change Data Capture or CDC Management Console and the **CDC → Access → Server** were both installed on this workstation. IBM Cognos Framework Manager was also installed here. This workstation was an IBM System x model 3850 with 3.25 GB of RAM.

Note: There was no technical reason for installing this set of products on a single workstation. This was done only for convenience.

Windows Vista and Infosphere Warehouse Design Studio were installed on the remaining workstation. This workstation was an IBM System x model 336 with 3.1 GB of RAM.

Two different networks connected the workstations to each other and to the z10. The workload was run over a gigabit Ethernet connection. Other activities utilized a fast Ethernet connection. Hipersocket connections were configured between the LPARs on the z10 CEC.

8.1.2 Software configuration

z/OS systems

This section describes the z/OS systems in the test environment. Systems P58, P59, P60, and P61 were members of a sysplex called PATPLX61. The LPARs on which these systems ran were connected to a single integrated Coupling Facility using ICP type links. In a production environment, two CFs would be used rather than a single CF. System P10 was a standalone system used to drive the OLTP workload. The workload was driven using a tool called WebSphere Studio Workload Simulator or WSWS.

Operating system

- z/OS 1.10
- Coupling Facility code level 16
- z/OS UNIX (UNIX System Services)
- C/C++ compiler and Language Environment
- RACF
- DFSMS
- DFSORT
- RMF™
- WLM
- TCP/IP

Subsystems

- DB2 9 for z/OS
- IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Version 4.2
- IBM InfoSphere Change Data Capture Version 6.2
- WebSphere Application Server for z/OS Version 7.0
- IBM DB2 Administration Tool for z/OS Version 7.2

DB2

Five DB2 subsystems were built for this project. Four subsystems were members of a four-way DB2 data sharing group and the fifth DB2 subsystem was a standalone non-data-sharing subsystem. Both OLTP and data mart tables were defined in the data

sharing group. Only data mart tables were defined in the non-data-sharing subsystem. It is important to understand that the data mart tables defined in the two databases were identical.

Omegamon XE for DB2 PE on z/OS includes support for a database that contains historical DB2 performance information. This set of tables was defined in the non-data-sharing subsystem (DBDM). There was no technical reason for defining the Omegamon XE for DB2 PE tables in this DB2 subsystem. This approach was taken for convenience. Detailed information about the two databases is contained in Table 8-1.

The active logs for both non-data-sharing subsystem DBDM and data-sharing group DDWG were striped across four volumes in order to achieve optimal performance. In order to stripe DB2 active logs, the volumes where the logs reside must be SMS-managed. In addition, the SMS dataclass of the logs must have extended formatting enabled and must have the volume count set to the desired number of stripes. In our case, a value of four was used. See Table 8-1.

Table 8-1 DB2 naming conventions

Entity	Data sharing	Non-data-sharing
Group Name	DDWG	n/a
Group Attach Name	DDWG	n/a
Location Name	DDWG	DBDMDDF
Generic LU Name	DDFDDWn	DDFDBDM
IRLM Group Name	IDWG	n/a
Subsystem Name	DDWn	DBDM
Command Prefix	-DDWn	-DBDM
BSDS	DB2DW.ZDW.DDWn	DB2DM.ZDW.DBDM
Active Log Prefix	DB2DWL.ZDW.DDWn	DB2DML.ZDW.DBDM
Work File Database	WORKDDWn	DSNDB07
Procedure Names	DDWnDBM1 DDWnDIST DDWnMSTR	DBDMDBM1 DBDMDIST DBDMMSTR
Subsystem ZPARM	DSNZDDWn	DSNZDBDM
TCPIP Ports	Port 6120 Resync Port 61n1	Port 6220 Resync Port 6221
IRLM Subsystem Name	IDWn	IRLM
IRLM Procedure Name	DDWnIRLM	DBDMIRLM
IRLM ID	n	1

SMS

The SMS definitions are documented in Table 8-2.

Table 8-2 SMS storage group configuration

SMS Storage Group	SMS Storage Class	SMS Data Class	DB2 Stogroup	Number of Volumes	Vol IDs	Use
BOOKOLTP	BOOKOLTP	DB2EXAD	BOOKOLTE	32	DWOnnn	OLTP user data, DDWG
BOOKDW1	BOOKDW1	DB2EXAD	BOOKDW1	48	DWDnnn	Data mart user data, DDWG
BOOKDW2	BOOKDW2	DB2EXAD	BOOKDW2	32	DMDnnn	Data mart user data, DBDM
DB2WORK1	DB2WORK1		DB2WORK1	20	DWWnnn	DB2 work files, DDWG
DB2WORK2	DB2WORK2		DB2WORK2	16	DMWnnn	DB2 Work Files, DBDM
RAWDATA	RAWDATA	CMPEXT	n/a	55	DWRnnn	Raw data, DATAGEN output
DB2LOGS1	DB2LOGS1	DB2LOGS		16	DWLnnn DMLnnn	DB2 active logs

The key characteristics of the SMS data classes that we used are:

CMPEXT Extended format, extended addressability, compaction

DB2LOGS Extended format, extended addressability, volume count of 4

DB2EXAD Extended format, extended addressability

The DB2EXAD SMS data class was used for both the transactional and the data mart user databases. Extended addressability allows the creation of VSAM data sets that are over 4 GB in size.

WebSphere Application Server

We built a WebSphere network deployment configuration with a single cell called ZDW. The cell spanned z/OS systems P60 and P61. The WebSphere Application Server deployment manager ran on system P61. A single server was configured on both system P60 and system P61. WebSphere was not installed on the remaining two z/OS systems in the sysplex, P58 and P59. The multiple z/OS LPAR WebSphere configuration is shown in Figure 8-2 on page 209.

z/OS UNIX

Our sysplex was configured with a shared file system. Therefore, the same directories were visible from z/OS UNIX (UNIX System Services) on each member of the sysplex.

8.2 IBM Cognos BI for Linux on System z

For this test setup we used one LPAR on the CPC in which we have installed z/VM. Under z/VM we have created three Linux guests, one for the IBM Cognos BI server and one for its content databases running DB2 9.5 for Linux, Unix and Windows, and the last Linux guest was for Infosphere Warehouse. See Figure 8-4 on page 214.

Archive

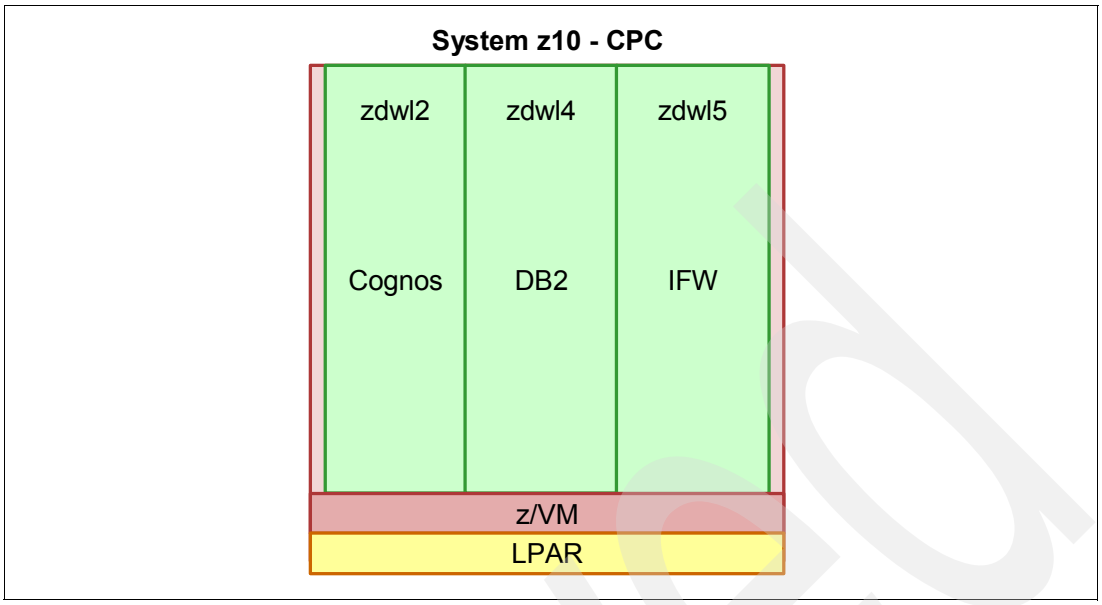


Figure 8-4 Linux infrastructure

We used Suse Linux Enterprise server 10 SP2.

As IBM Cognos BI was mainly used as an injection for DB2 on z/OS, we did not spend much time tuning the environment, especially the I/O components.

8.2.1 Our test setup

IBM Cognos BI is a 31-bit application running on Linux on System z. It runs under WebSphere which has to be a 31-bit version for the JVM. To reach the throughput we wanted to achieve here, we created two report servers running in two DVMs. So there are two WebSphere profiles, each one running one Report Server.

Content store

The content store is mandatory for IBM Cognos BI to work. It is a core component where it stores the report definitions, the user parameters, and so on.

Ever since IBM Cognos BI Version 8.4, there is the support of DB2 on z/OS as the content store. Because the configuration we set up for this book was already large, we did not want to go into more possible issues. So we did not implement our whole test with the content store on z/OS. Instead we stuck with DB2 UDB 9.5 FP4 on Linux for System z. We had a dedicated z/VM guest with four virtual processors and 4 GB of memory. Because we did not want to have any swapping, we did not configure swap at all.

Note: Do not run production without swap space. If you do so and run out of memory, you will crash.

Database creation

```
db2inst1@ZDWL4:~> db2 "create database C8STORE using codeset UTF-8 TERRITORY US"
DB20000I  The CREATE DATABASE command completed successfully.
db2inst1@ZDWL4:~> db2 connect to C8STORE
```

```
   Database Connection Information

 Database server        = DB2/LINUXZ64 9.5.4
 SQL authorization ID   = DB2INST1
 Local database alias   = C8STORE

db2inst1@ZDWL4:~> db2 grant dbadm on database to Cognos
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 UPDATE DATABASE CONFIGURATION USING APPLHEAPSZ 1024 DEFERRED
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

db2inst1@ZDWL4:~> db2 UPDATE DATABASE CONFIGURATION USING LOCKTIMEOUT 240 DEFERRED
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

db2inst1@ZDWL4:~> db2 CREATE BUFFERPOOL "C8STORE_08KBP" IMMEDIATE SIZE 1000
PAGESIZE 8K
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 CREATE BUFFERPOOL "C8STORE_32KBP" IMMEDIATE SIZE 1000
PAGESIZE 32K
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 CREATE SYSTEM TEMPORARY TABLESPACE "TSN_SYS_C8STORE" IN
DATABASE PARTITION GROUP IBMTEMPGROUP PAGESIZE 32K MANAGED BY SYSTEM USING
('CNT_SYS_C8STORE') BUFFERPOOL "C8STORE_32KBP"
DB20000I  The SQL command completed successfully.

db2 CREATE USER TEMPORARY TABLESPACE "TSN_USR_C8STORE" IN DATABASE PARTITION GROUP
IBMDEFAULTGROUP PAGESIZE 8K MANAGED BY SYSTEM USING ('CNT_USER_C8STORE')
BUFFERPOOL "C8STORE_08KBP"
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 CREATE REGULAR TABLESPACE "TSN_REG_C8STORE" IN DATABASE
PARTITION GROUP IBMDEFAULTGROUP PAGESIZE 8K MANAGED BY SYSTEM USING
('CNT_REG_C8STORE') BUFFERPOOL "C8STORE_08KBP"
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 ALTER TABLESPACE "TSN_REG_C8STORE" DROPPED TABLE RECOVERY ON
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 CREATE SCHEMA db2 Cognos AUTHORIZATION Cognos
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 GRANT CREATETAB,BINDADD,CONNECT,IMPLICIT_SCHEMA ON DATABASE
TO USER Cognos
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 GRANT CREATEIN,DROPIN,ALTERIN ON SCHEMA DB2 Cognos TO USER
Cognos WITH GRANT OPTION
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> db2 GRANT USE OF TABLESPACE "TSN_USR_C8STORE" TO Cognos
DB20000I  The SQL command completed successfully.
```

Archived

```
The rest of the database, the tables and indexes are created automatically at
first IBM Cognos startup. The database must be referenced in cogconfig.sh with the
right creditential though.
```

Logging database

Log messages are an important diagnostic tool for investigating the behavior of IBM Cognos BI. In addition to error messages, log messages provide information about the status of components and a high-level view of important events. For example, log messages can provide information about attempts to start and stop services, completion of processing requests, and indicators for fatal errors. Audit logs, which are available from a logging database, provide information about user and report activity.

Database creation

```
db2inst1@ZDWL4:~> db2 "create database C8AUDIT using codeset UTF-8 TERRITORY US"
DB20000I  The CREATE DATABASE command completed successfully.
db2inst1@ZDWL4:~> db2 connect to C8AUDIT

   Database Connection Information

 Database server        = DB2/LINUXZ64 9.5.4
 SQL authorization ID   = DB2INST1
 Local database alias   = C8AUDIT

db2inst1@ZDWL4:~> db2 grant dbadm on database to Cognos
DB20000I  The SQL command completed successfully.

db2inst1@ZDWL4:~> su - Cognos
Password:
Cognos@ZDWL4:~> db2 connect to C8AUDIT

   Database Connection Information

 Database server        = DB2/LINUXZ64 9.5.4
 SQL authorization ID   = Cognos
 Local database alias   = C8AUDIT

db2inst1@ZDWL4:~> db2 connect to C8AUDIT

   Database Connection Information

 Database server        = DB2/LINUXZ64 9.5.4
 SQL authorization ID   = DB2INST1
 Local database alias   = C8AUDIT

db2inst1@ZDWL4:~> db2 create
db2inst1@ZDWL4:~> db2 "create bufferpool BP32K PAGESIZE 32K "
DB20000I  The SQL command completed successfully.
db2inst1@ZDWL4:~> db2 create system temporary tablespace tempts32 pagesize 32K
bufferpool BP32K
DB20000I  The SQL command completed successfully.
db2inst1@ZDWL4:~> db2 grant use of tablespace tempts32 to user Cognos
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0159N  The statement references "TEMPTS32" which identifies a(n) "SYSTEM
TEMPORARY TABLESPACE" rather than a(n) "USER TABLESPACE".  SQLSTATE=42809
```

Archived

```
db2inst1@ZDWL4:~> db2 create user temporary tablespace tempts4 pagesize 4K
bufferpool BP4K
DB20000I  The SQL command completed successfully.
db2inst1@ZDWL4:~> db2 grant use of tablespace tempts4 to user Cognos
DB20000I  The SQL command completed successfully.
Cognos@ZDWL2:/opt/Cognos/c8/logs> db2 connect to C8AUDIT user Cognos using 123456

   Database Connection Information

 Database server        = DB2/LINUXZ64 9.5.4
 SQL authorization ID   = Cognos
 Local database alias   = C8AUDIT

Cognos@ZDWL2:/opt/Cognos/c8/logs> db2 list tables

Table/View                      Schema          Type  Creation time
------------------------------- --------------- ----- --------------------------

  0 record(s) selected.

Cognos@ZDWL2:/opt/Cognos/c8/logs> db2 list tablespaces

           Tablespaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = Database managed space
 Contents                             = All permanent data. Regular table space.
 State                                = 0x0000
   Detailed explanation:
     Normal

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x0000
   Detailed explanation:
     Normal

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = Database managed space
 Contents                             = All permanent data. Large table space.
 State                                = 0x0000
   Detailed explanation:
     Normal

 Tablespace ID                        = 3
 Name                                 = TEMPTS32
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x0000
   Detailed explanation:
     Normal
```

Archived

```
Tablespace ID                        = 4
Name                                 = TEMPTS4
Type                                 = System managed space
Contents                             = User Temporary data
State                                = 0x0000
  Detailed explanation:
    Normal

Tablespace ID                        = 5
Name                                 = COGTS4K
Type                                 = Database managed space
Contents                             = All permanent data. Regular table space.
State                                = 0x0000
  Detailed explanation:
    Normal

Tablespace ID                        = 6
Name                                 = COGTS8K
Type                                 = Database managed space
Contents                             = All permanent data. Regular table space.
State                                = 0x0000
  Detailed explanation:
    Normal
```

Now one can start the application server, with IBM Cognos app deployed, it will automaticaly create the tables in the database we have juste defined.

```
Cognos@ZDWL2:/opt/Cognos/c8/bin> db2 list tables

Table/View                      Schema          Type  Creation time
------------------------------- --------------- ----- --------------------------
COGIPF_ACTION                   Cognos          T     2009-11-04-13.36.46.211349
COGIPF_AGENTBUILD               Cognos          T     2009-11-04-13.36.46.515362
COGIPF_AGENTRUN                 Cognos          T     2009-11-04-13.36.46.497453
COGIPF_EDITQUERY                Cognos          T     2009-11-04-13.36.46.441826
COGIPF_MIGRATION                Cognos          T     2009-11-04-13.36.46.533108
COGIPF_NATIVEQUERY              Cognos          T     2009-11-04-13.36.46.248668
COGIPF_PARAMETER                Cognos          T     2009-11-04-13.36.46.310878
COGIPF_POWERPLAY                Cognos          T     2009-11-04-13.36.46.567202
COGIPF_POWERPLAY_DIM_USAGE      Cognos          T     2009-11-04-13.36.46.585845
COGIPF_POWERPLAY_LEVEL_USAGE    Cognos          T     2009-11-04-13.36.46.601519
COGIPF_POWERPLAY_MEASURE_USAGE  Cognos          T     2009-11-04-13.36.46.616558
COGIPF_RUNJOB                   Cognos          T     2009-11-04-13.36.46.325899
COGIPF_RUNJOBSTEP               Cognos          T     2009-11-04-13.36.46.365157
COGIPF_RUNREPORT                Cognos          T     2009-11-04-13.36.46.402940
COGIPF_SYSPROPS                 Cognos          T     2009-11-04-13.36.46.631618
COGIPF_THRESHOLD_VIOLATIONS     Cognos          T     2009-11-04-13.36.46.550931
COGIPF_USERLOGON                Cognos          T     2009-11-04-13.36.46.230868
COGIPF_VIEWREPORT               Cognos          T     2009-11-04-13.36.46.480257

  18 record(s) selected.
```

Archived

8.2.2 Move content store to z/OS

Running IBM Cognos BI on the mainframe makes it possible to have most of it on z/OS. It may not be a priority, but it can be handy. If your backup and recovery policy is well tested on z/OS and DB2, there is no need to worry about IBM Cognos BI.

There are however some specifics to create the content store in DB2 for z/OS. As usual, in DB2 for z/OS one is not connecting to a database, but to a subsystem. Tables are access through their owner / schema name.

The easiest way is to get a specific ID in z/OS and DB2 with DBADM rights on the two databases we are going to create. Yes, two databases. This is also a specific of having the content store on z/OS. You have to create the Content database and the Delivery database for IBM Cognos BI to work smoothly and nicely. Moreover, the philosophy on how to create those two database are a little bit different. Before moving the content store to z/OS, one needs to create the new content store databases.The next paragraphs will guide you through the process of creating the two databases and their respective tablespaces and tables.

Content store database creation

We need only to create the database and the tablespaces (and auxiliary ones). The tables and indexes will be created automatically in those tablespaces the first time the application is started.

Note: SQL scripts are in the `COG_ROOT/configuration/schema/content/db2zOS` directory.

We will assume for the rest of the following information:

- The user ID for IBM Cognos content store is Cognos1.
- The content database name is CSTORE.
- Cognos1 is DBADM on CSTORE.
- Cognos1 has USE permissions on BP1 and BP32K.

Table 8-3 on page 219 shows the variables used by the SQL script.

Table 8-3 Variables used by the SQL script

Variables	Values	Comments
CMSCRIPT_DATABASE	CSTORE	Not more than 8 characters
CMSCRIPT_STOGROUP	SYSDEFLT	If you use implicit tablespace creation, it has to be SYSDEFLT
CMSCRIPT_CS_ID	C8	2 digits only
CMSCRIPT_LARGE_BP	BP32K	
CMSCRIPT_REGULAR_BP	BP01	
CMSCRIPT_TABLESPACE	BASETS	Not more than 6

In an IBM Cognos BI configuration (COG_ROOT/bin/cogconfig.sh), there are two additional variables, shown in Table 8-4.

Table 8-4 Two additional variables in an IBM Cognos BI configuration

Variables	Values	Comments
CMSCRIPT_CREATE_IN	CSTORE.C8BASETS	This variable is used in the SQL script as the parameter for the CREATE TABLE [...] IN DATABASE clause. It is the concatenation of CMSCRIPT_DATABASE, CMSCRIPT_CS_ID and CMSCRIPT_TABLESPACE
CMSCRIPT_USERNAME		With the methodology we used here, this one is not in use.

Figure 8-5 on page 220 illustrates the way to implement it in the IBM Cognos BI configuration tool.

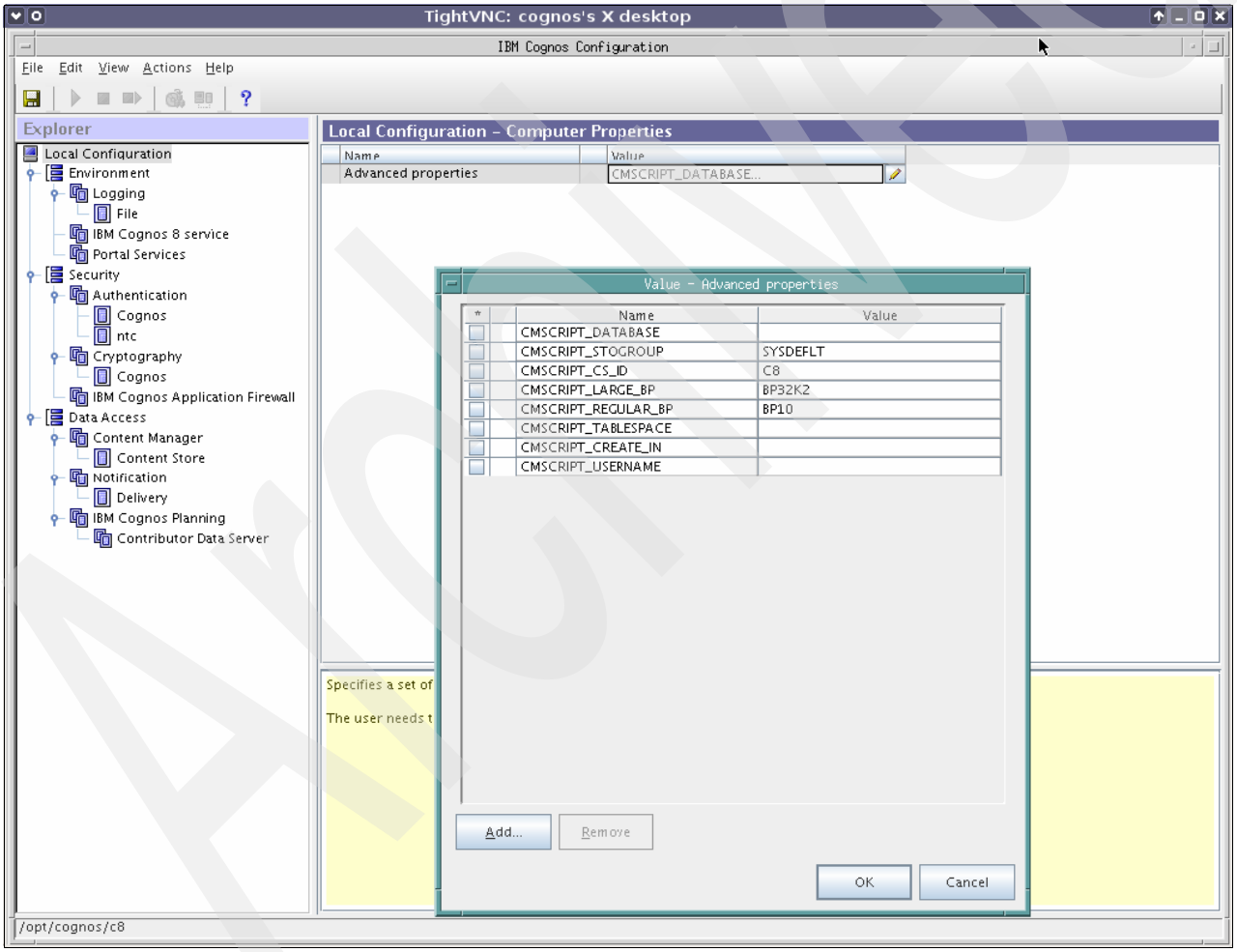


Figure 8-5 Content store variable implementation in cogconfig.sh

Delivery database creation

The delivery database is mandatory when you use z/OS as the back end for IBM Cognos BI content databases. In opposition to the content database, this one is created totally by you, using the given script. It is a little bit less flexible than the previous one because you do not

have the hand on the table or tablespace names. They also have to be prefixed (their schema/owner) by the user who will be accessing those tables.

You will find the scripts in `$COG_ROOT/configuration/schema/delivery/zosdb2`.

Update NCCOG to the database name you have been provided with in the two files `NC_TABLESPACES.sql` and `NC_CREATE_DB2.sql` and change STOGROUP in `NC_TABLESPACES.sql`.

Run the script shown in Example 8-1 after having created a connection to the DB2 subsystem with the correct user.

Example 8-1 Running SQL scripts in DB2 for z/OS from Linux

```
db2 -tvf NC_TABLESPACES.sql -z NC_TABLESPACES.out
db2 -tvf NC_CREATE_DB2.sql -z NC_CREATE_DB2.out
```

8.2.3 WebSphere for Linux on z for IBM Cognos BI

IBM Cognos BI uses only the servelet of WebSphere Application Server. IBM Cognos BI is mainly using either static pages or CGI scripts. The CGI scripts are calling processes written in C/C++, which are much faster. As a matter of fact, there is no need to use the plug-in between HTTP Server and WebSphere. In cogconfig, you configure the port number the servlet is listening on. The IBM Cognos BI code directly accesses this port number without using WebSphere Application Server tooling. IBM Cognos BI does not use the data sources interface either—not for the content store databases (content, delivery, and/or logging), and not for the reporting databases (the ones data are read from to do the reports).

WebSphere parameters for IBM Cognos BI for Linux on System z

In our setup we have two dispatchers, which means we have two distinct Java virtual machines. This also means that we have two WebSphere Application Server profiles, and each profile has its own application server. One handles IBM Cognos Report Service only and the other handles IBM Cognos Report Service and IBM Cognos Content Manager. As a reminder, this last service is the one accessing the content store.

Both WebSphere Application Server profiles share most of their configuration parameters. The only one that is a bit different is the heap size. When one is handling more work, its maximum heap size has to be bigger.

Java heap size

The profile handling both services is set up with a heap size between 512 MB and 768 MB (which is the maximum for a 32-bit application), and the other is set up with a heap size between 384 MB and 512 MB.

Environment entries

Remember that IBM Cognos BI is a 32 bit application and it does not use the WebSphere Application Server data source interface, so to connect to DB2 it has to have the 32-bit DB2 libraries in its environment entries path. See Figure 8-6 on page 222.

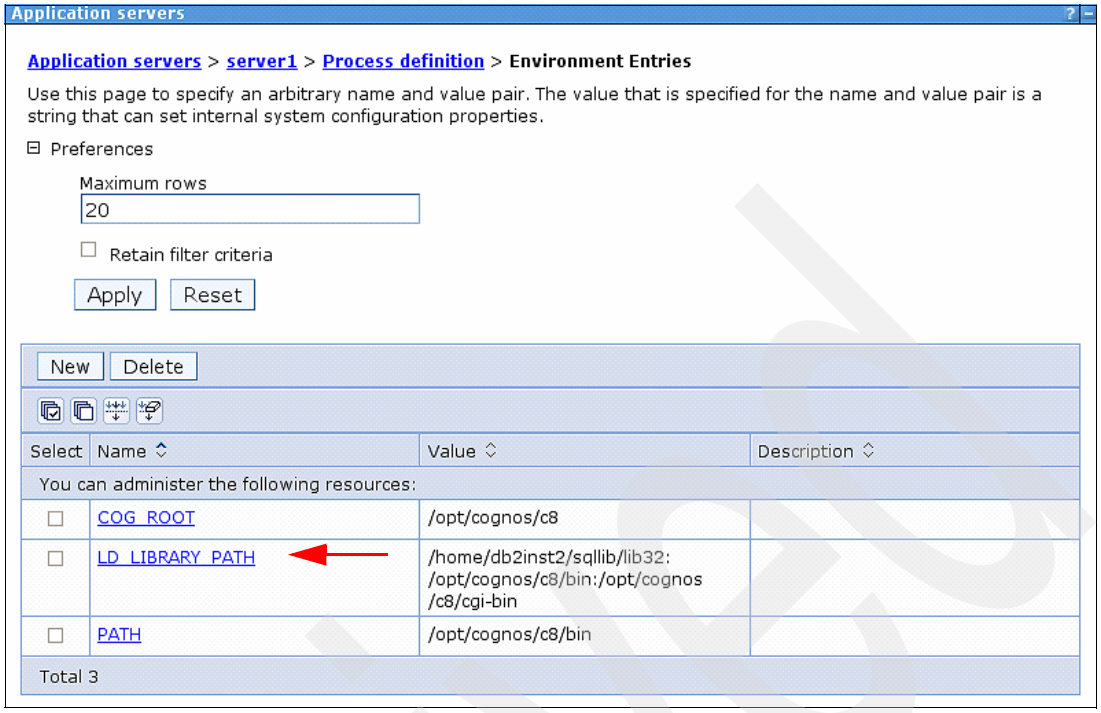


Figure 8-6 DB2 32-bit libraries configuration

Web container

We needed to increase the number of web containers for the servers we were using. It was raised 10 times compared to the default so that the minimum was 50 and the maximum was 500. See Figure 8-7 on page 223.

Archive

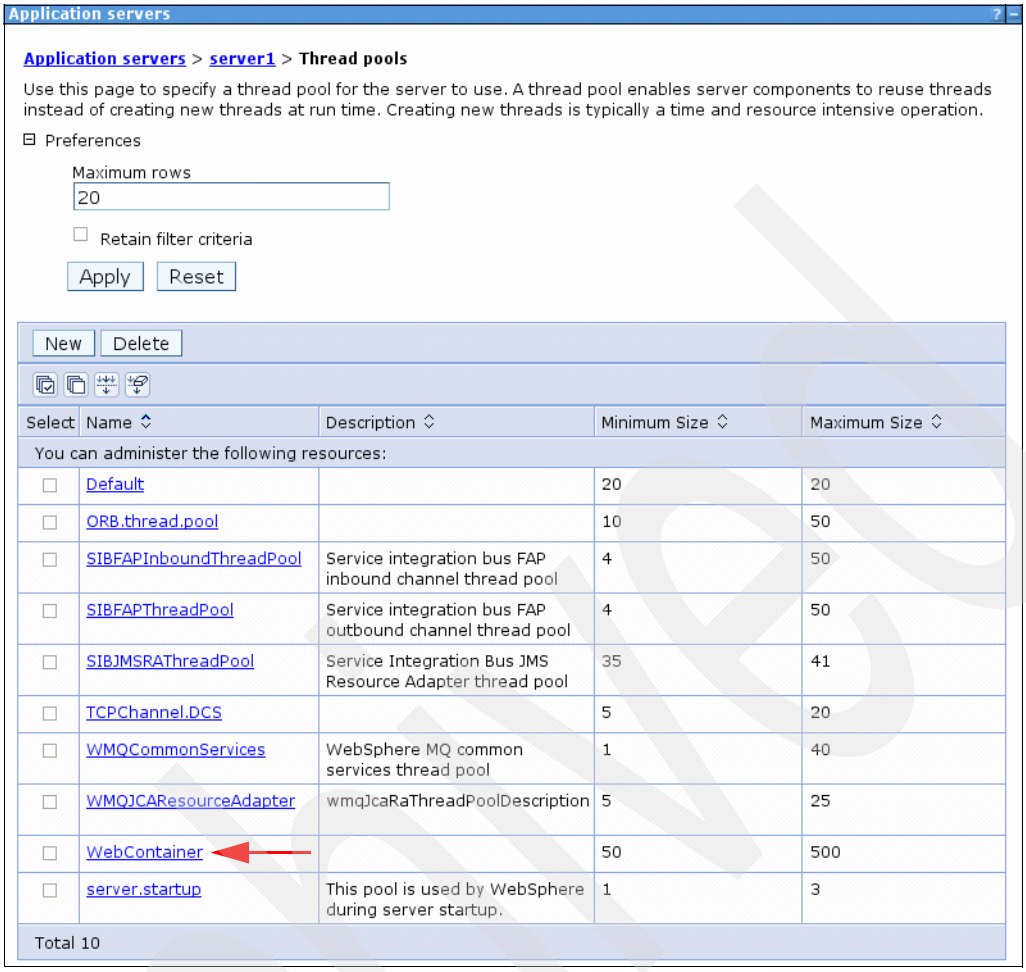


Figure 8-7 Web container configuration

Attention: Remember to change the web container configuration on each application server of your IBM Cognos BI environment.

IBM Cognos BI parameters

We did not spend much time tuning IBM Cognos BI, because the test was more about co-location of the transactional database with the data warehouse database and the benefit of such an architecture.

IBM Cognos BI has, however, a notion of *peak* and *non-peak* periods. Sometimes we were running the workload for the measurement in the morning, and sometimes at night. IBM Cognos BI changes its behavior (for running batch report creation in a noninteractive way) between the peak and non-peak period. This translates into the number of processes, and by extension to resources, in the system. To avoid such behavior we configured the start of the peak time and the non-peak time for the same time, for example at midnight; see Figure 8-8 on page 224.

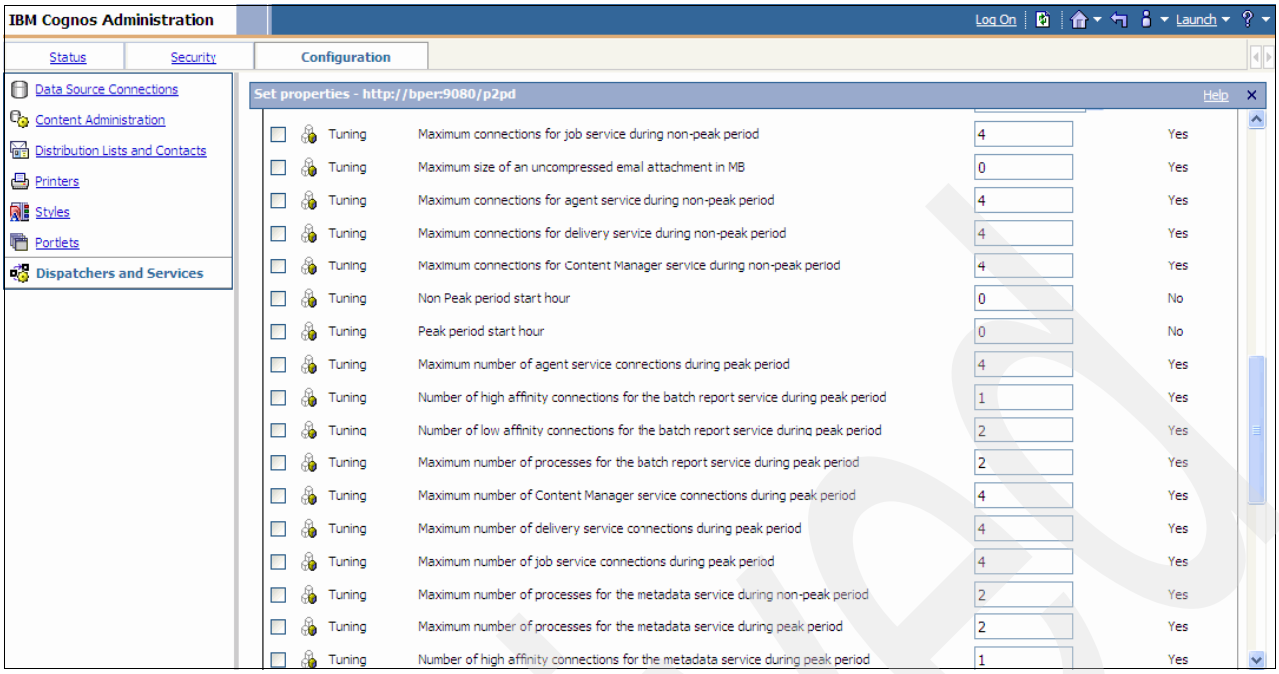


Figure 8-8 IBM Cognos BI peak and non-peak time and high/low affinity connection configuration

We left the default setting for the high/low affinity connection for the report service. This configuration parameter sets the number of BIBusTKServerMain processes. It leads to a maximum of two processes, each with 10 threads. As we had two application servers in the same environment, we had up to four processes, each with 10 threads.

Archive

Part 3

Implementation considerations

For our project, we assembled a team of subject matter experts. This part provides guidance from the team on resource management, co-location, data sharing, and performance, based on their past experience.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

9

Performance monitoring and management of queries overview

Many enterprises using IBM System z have organized their IT staff by product or functional specialties, each with its own product terminology and associated set of management and reporting options. This chapter provides a common terminology foundation for the DB2 system administrators, the System z operating system administrators, and the DW and BI administrators. This should enable them to share service requirements, workload management implementation dependencies, and reports and tools to maximize efficient and effective service delivery.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

9.1 Overview

Resource management and performance monitoring of IT workloads are keys to providing satisfactory service to the business community. This is especially true when transactional and data warehouse workloads are housed within the same System z hardware as their processing needs, and the associated resource requirements likely have very different execution characteristics and service delivery objectives. With System z, a rich infrastructure exists to monitor, manage, and report activities at historically very granular level with facilities such as z/OS SMF, and additional reporting tools such as RMF and the Tivoli Omegamon family of products.

In recent years, many enterprises have also established Business Intelligence Competency Centers (BICC) or Business Analytics Competency Centers (BACC) to improve BI solution adoption, implementation, and delivery. This virtual or fixed team of cross-functional IT and line-of-business professionals focuses on building BI expertise, standardizing the technology building blocks, and sharing best practices across the organization's expanding BI projects. This chapter also helps BICC and BACC staff understand the mainframe's technology monitoring infrastructure, their role in ensuring that BI tool interfaces to the infrastructure are effectively implemented, and how the IT monitoring tools may assist them with improving service to the business community. For the remainder of this chapter, the term BACC includes the BI administrator and the BICC.

Performance management of the DW query environment with IBM Cognos BI for Linux on System z and IBM DB2 for z/OS is the basis for examples in this chapter. Reviewing prior chapters for basic product architecture, features, and terminology may be beneficial:

- Chapter 2, “Why System z for data warehousing” on page 9 - System z hardware and software including DB2, z/OS, z/VM, and Linux
- Chapter 6, “BookStore workload 2 - IBM Cognos BI reporting” on page 81 - Business Intelligence with IBM Cognos BI

9.2 Performance management

The mainframe has a well-established infrastructure for performance management of IT workloads that has evolved over almost 40 years. There are built-in operating system functions such as z/OS SMF to capture very granular usage data at a very low level for an individual transaction and/or by user ID with minimal additional system cost. Real-time, automated workload management by WLM uses this data to allocate system resources dynamically to meet installation-defined service objectives for transactions and users. IT administrators also access this data with tools such as RMF or Tivoli Omegamon to perform real-time monitoring or problem analysis, and to generate batch reports for historical analysis of problems, assess WLM success, and capacity planning.

9.2.1 Defining performance

The effective use of the System z performance management infrastructure requires definitions of what is good or acceptable performance for key business functions supported by the application. A simple definition of application performance is how quickly a transaction carries out a specified task. However, this is a very vague and subjective definition, and must be refined.

In a diverse workload environment, the ability to predict and track transaction behavior is critical to an organization's ability to proactively identify problems before a serious failure occurs that prevent SLAs from being met. Although the ultimate goal is to monitor performance in the production environment, an effective monitoring process involves nearly all stages of the application life cycle. This includes early design decisions through development and testing, and finally to deployment and maintenance.

For effective performance monitoring and service delivery, the definition of how the system should perform must meet the following three criteria:

- First, the definition must be *quantitative*. That is, it must facilitate objective measurement.
- Secondly, it must be *consistent*. Long-term performance monitoring requires comparison of measurements over extended periods of time, although execution environments, server configurations, or applications may change.
- Finally, it must be *realistic*. The performance model must account for the real constraints of the application and not simply a desired goal. Not all application functions perform the same amount of work and, therefore, they will not perform in the same way.

In addition, the application, especially one with a multitier architecture such as IBM Cognos BI must be enabled for granular, quantitative monitoring. Many other multitier BI applications are not, and this may limit efficient, proactive performance management.

The BACC plays a primary role in defining what performance the business requires for specific BI application functions accessing the DW and communicating these requirements to IT throughout the application life cycle. These iterative discussions should include:

- Educating the business community about the performance variability that may occur due to widely varying data volumes for a given business report due to normal business cycles
- Influencing the design and implementation of the BI application to meet performance goals
- Enabling and requiring IBM Cognos BI product features or interfaces such as wlm_set_client_info to be implemented so that performance management may occur with z/OS WLM automation
- Educating IT that IBM Cognos BI and other BI workloads likely have widely varying resource requirements within a single application as well as across many applications. Some examples:
  - IBM Cognos BI is a rich, full function, flexible application builder, not just a simple reporting tool. Some implementations may choose to use saved report processing or cubing technology to provide more consistent response time but this may also produce a heavier batch load off shift.
  - A given report or transaction may have varying resource requirements due to basic business demographics and business cycles.

9.2.2 Performance metrics

This section discusses various performance metrics, components, and expectations that are used to define a quantitative, consistent, and realistic performance monitoring model for System z applications.

Transactions

The first metric is that of a transaction: a single, complete unit of work (UOW) that is the unit that all of the other metrics typically measure. In the broadest sense, a transaction is a complete business task from the point of view of the ultimate consumer, either a human user

at a browser or an automated process that consists of a single request and response indicating the completion of the request.

This logical business transaction (LBT) or end-to-end transaction typically encompasses multiple components to complete a request. These components comprise a browser, an HTTP server, one or more application servers, backend servers, database servers, as well as internal networking components and the external Internet. Figure 5-2 illustrates the end-to-end transactions' infrastructure in our project.

End-to-end transaction monitoring

As transactions become more complex by utilizing multiple components in single or multiple servers, establishing transactional boundaries and measuring performance has also become more complex. While end-to-end transaction monitoring and measurement is of great interest to many organizations, many multitier architecture applications do not pass a single unique transaction identifier for each transaction from the web browser to the logical and physical components involved in servicing the transaction. Without this unique identifier to associate the begin and end times in each component, end-to-end monitoring and service metrics may be difficult to gather accurately for DW workloads. Also, problem analysis is obviously more complex.

Another consideration is that the LBT transaction may actually be multiple server transactions, because some applications such as IBM Cognos BI make several calls to the application and database servers including multiple unit-of-work (UOWs) to produce one or more reports to respond to a single browser request or business transaction. For key LBTs, it may be necessary to understand the ratio between the LBT and the individual server transactions to insure multiserver-related software settings (such as maximum threads) are consistently set across the involved servers.

A discussion of end-to-end transaction monitoring and measurement from web browser to the z/VM Linux HTTP component to the LDAP server to the IBM Cognos BI application servers to the z/OS DB2 server is beyond the scope of this book. This chapter only deals with the portions of the transaction (IBM Cognos BI or DB2) that can be quantitatively measured and uniquely identified across multiple components, either in a single LPAR or across LPARS:

- IBM Cognos BI Version 8.4 with wlm _set_client interfaces and z/OS DB2 and z/OS WLM
- z/OS DB2, WLM, and RMF

Response time

The most common metric for measuring performance is response time. Response time is the time—often averaging seconds to minutes or even longer for a DW workload—required to complete a transaction.

Figure 9-1 on page 231 illustrates a basic response time breakdown.

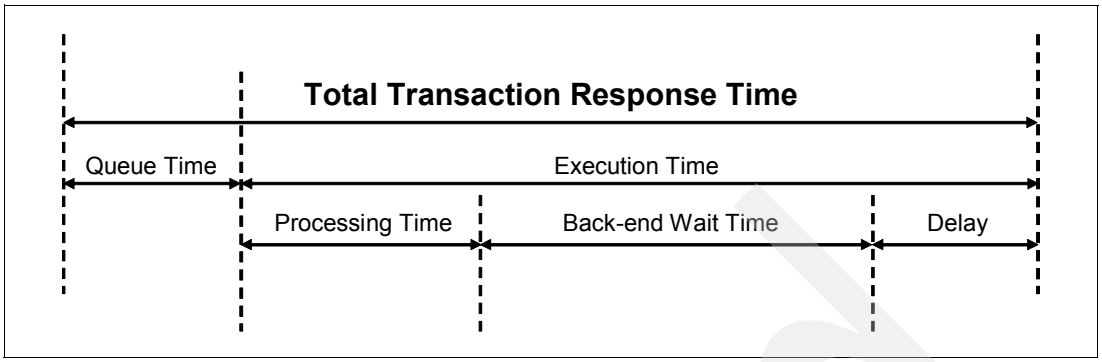


Figure 9-1 Components of response time

The response time consists of two main parts:

- The time the transaction waits for an execution thread, referred to as the *queued time* or *wait time*.
- The time taken to perform the transaction, or *execution time*.

IBM Cognos BI execution time consists of the actual IBM Cognos BI application processing time, data source wait time, and any delay time. IBM Cognos BI processing time versus data source wait time may vary tremendously from one IBM Cognos BI transaction to another, or from one business analytics application to another. Key factors influencing this are IBM Cognos BI functionality invoked and data source volumes that must be processed to perform the LBT.

Delay time accounts for the amount of time when a transaction cannot process due to the constraint of a system resource. Delays can be caused by a number of different system resources. Most often, delays are caused by processor availability, real memory availability, or input/output (I/O).

Throughput

In the System z environment (and realistically all execution environments), the individual transaction response time should be balanced with the total number of transactions that can be run in a specific period of time. This is referred to as *throughput* and is typically measured in transactions per unit of time (transactions per minute or tpm). Response time and throughput are often related, especially when a system resource is totally consumed.

The importance of throughput is based on one of the key advantages of the mainframe environment, that is, its ability to manage multiple, diverse workloads while maintaining acceptable response times. Throughput takes this into consideration

Number of clients

The number of clients can have multiple definitions. It is important to clearly state which group is being referenced in any performance-related discussion with IBM Cognos BI. The common ones are:

- Named users - The total population of people who can access (log in) to the application
- Active users - A subset of the named users. They are actually connected to the application. Their transactions may or may not be actually executing, that is, they may be reviewing transaction output and analyzing results. This is known as *think time,* and much of the total LBT transaction response time for the LBT DW transaction may be spent in think time.

- Concurrent users - A subset of the active users. They are actually making requests to the application server and causing activity throughout the configuration,

An IBM Cognos BI rule-of-thumb for the number of named users to concurrent users is 1% to 3% of named users with 1% frequently used.

System resource metrics

The two key system resources that affect applications are central processing unit (CPU) and real memory. Processor utilization can be stated in either CPU time (typically in ms) or as a percentage of the overall processor capacity.

Processor utilization

In performance management, processor utilization is often monitored. However, it is less useful because it can be very dynamic and difficult to compare one time period to another. Correlating transaction throughput and response time need with processor utilization is more valuable.

Memory utilization

Several metrics are used in monitoring memory utilization. From a system standpoint, paging rate indicates the number of times (usually in seconds) that a required segment of memory has to be read back into real memory from auxiliary memory. IBM Cognos BI and WebSphere Application Server use a significant amount of real memory compared to other System z hosted transaction servers. System paging can have an impact on the transaction response time.

From a server standpoint, the key memory metrics are:

- Average and maximum heap size indicating the amount of memory available to the Java virtual machine (JVM)
- Garbage collection intervals indicating the time period between the garbage collections
- Garbage collection time indicating the amount of time required to complete a garbage collection cycle

The garbage collection percentage is calculated as the garbage collection time divided by the garbage collection interval multiplied by 100. The desired garbage collection percentage is between 2% and 5% for WebSphere Application Server applications.

Workload Manager metrics

Two additional metrics are available from the z/OS Workload Manager (WLM) that indicate the performance of a service class against the established goal for each category of z/OS requests. These are the Performance Index (PI) and Workflow Percentage.

As a single, simple value, the PI is a key metric for triggering review of the server performance. The PI is a value of 1 when the overall workload performs at or near the goal. It is less than 1 when the overall workload outperforms the goal, and greater than 1 when it does not meet the goal.

The Workflow Percentage shows how well the service class performs by indicating how close the actual response time is to the maximum response time. A value of 100% indicates that the current average response time has no queue time or delays.

Interaction between IBM Cognos BI data source queries, DB2, and WLM are accomplished with the wlm_set_client_info feature that became available in IBM Cognos BI Version 8.4. Additional details about implementation are in Chapter 11, “Resource management of data

warehouse mixed workloads” on page 263 and test results are in Chapter 16, “Multiple z/OS LPAR experiments” on page 369.

9.3 Monitoring facilities and tools

Business Analytics on System z may encompass multiple servers and operating systems. As described in Chapter 6, “BookStore workload 2 - IBM Cognos BI reporting” on page 81, z/VM, Linux and z/OS were used in our project. All three have monitoring facilities to capture hardware and software resource usage, and commands and tools are available to report the usage real time or historically. In addition, products such as IBM Cognos BI, DB2, and WebSphere that are included in the Business Analytics implementation generate additional usage data and have their own monitoring and reporting tools. This data is needed for identifying and solving performance concerns as well as understanding capacity requirements.

9.3.1 Primary tools

The tools are briefly described in Table 9-1 with links to short overviews on usage and interpretation.

Table 9-1 Monitoring and reporting tools by platform

Operating System	Data Collection and Monitoring Options	Performance History Reporting Tools	Key Online Resource Management Tools and Commands	Additional Information

Archive

Linux	An embedded monitoring facility in all Linux distributions • Accessed real time with commands that can display the usage data at the workstation or pipe to a file		• strew/as - running processes and relationships • top – overall system data (CPU, memory) as well as per process • vmstat – overall system data (CPU, memory, I/O)	• Linux resource management and analysis tools • CPU time accounting in a z/VM environment
	sysstat • The open source package available at `http://pagesperso-orange.fr/sebastien.godard/` • Collects detail system data (CPU, memory, I/O, network, process) at user specified intervals for a length of time in a binary file. • The reporting tool is included. • This package is highly preferred for any Linux problem analysis by IBM.	• System Activity Reporter (SAR) is the text-based reduction and reporting tool. Sample reports available at SAR Tutorial. • Graphs may be produced using other spreadsheet tools or open source pkgs.	• sadc – gathers usage data in binary file • mpstat – processor utilization • iostat – I/O utilization These are text-based displays.	`http://pagesperso-orange.fr/sebastien.godard/`
	Tivoli OMEGAMON XE for z/VM and Linux agent • Provides a Linux agent that collects additional detail system usage data (CPU, memory, I/O, network, process). • Interfaces to z/VM Performance Toolkit (PTK) to merge with the z/VM Monitor and PTK data. • As with the entire OMEGAMON XE family of products, usage data is stored and maintained in a relational Tivoli data warehouse (TDW) according to user-specified pruning options.	• System usage reports by server or guest by prior defined system stored intervals may be printed from OMEGAMON web views. • IBM Cognos BI reports may also be developed to provide additional insight for performance trends and capacity planning. An IBM Cognos Framework Manager model of the TDW is provided for flexible reporting.	• Included as a friendly, web-based graphical user interface (GUI) access tool, the Tivoli Enterprise Portal (TEP) provides real-time access to Linux, z/VM monitor, and PTK usage data as well as historical usage data. • Across the entire OMEGAMON XE family of products, TEP provides default guided workspace views; customized views may also be easily developed. • Alerts, suggested response actions, and user-defined actions are also provided in TEP.	• http://www-01.ibm.com/software/tivoli/products/omegamon-xe-zvm-linux/ • Review all links in the z/VM and Linux section on how to interpret CPU data because they also apply to the OMEGAMON XE views.
	IBM Cognos Log Service - Embedded component of IBM Cognos BI • Provides tracing and logging of IBM Cognos BI activities at user-specified levels (minimal, basic, request, trace, full) to log files and an optional logging database. • Includes viewers to organize and display the content on the log files.	• Sample IBM Cognos BI audit reports are supplied as well as the Framework Manager model. Additional IBM Cognos BI reports may be created.	IBM Cognos Administration - Component of IBM Cognos BI • GUI displays provide real-time and historical monitoring of system performance metrics. •This is a secured function and required permissions must have been granted to access IBM Cognos Administration. •Threshold and alerts may be set for system metrics.	• *IBM Cognos BI Administration and Security Guide*, Chapter 5 “Setting Up Logging” and Chapter 6 “System Performance Metrics”

	• Generates real-time alerts in IBM Cognos Administration when certain error msgs show up in a log file or when thresholds are exceeded. • Creates audit logs in a relational logging database that collect user and report activities that may be reviewed with supplied IBM Cognos BI reports. This logging database is required for historical activity reporting. Audit logging is lost if the communication connection between the logging database and the log server fails.	• The sample reports provide insight into usage within IBM Cognos BI. These are described in “Sample Audit Model and Audit Reports in Chapter 5 of the *IBM Cognos BI Administration and Security Guide*.	• IBM Cognos BI system metrics such as session counts, queue length and time, JVM usage, request counts, and time (success, failure, total), response time, and service times are available. • Merging of hardware system usage data with IBM Cognos BI CPU usage data is not available.	

Archived

z/VM	An embedded real monitoring facility is available via real time commands.		Many subcommands of the CP INDICATE and CP QUERY commands are available to display system resource usage.	• Performance and Monitor Data Collection • Understanding z/VM processor utilization
	Monitor - optional to implement component of z/VM that gathers system data (processor, memory, I/O, network) by z/VM user ID as well as overall system usage into a binary data stream that userid MONWRITE will record into a file. It is highly recommended that this component be implemented because the data is needed for any problem determination.	Performance Toolkit (PTK) for VM - Reduction and reporting tool that provides text-based online real-time monitor as well as historical displays or reports of past usage.	PTK provides many menu-based displays. Some common commands in the tool include MENU, USER, and LPAR to view guest, LPAR, and processor (CEC) usage data.	
	Tivoli OMEGAMON XE on z/VM and Linux - Part of the OMEGAMON XE family of products for System z that complements Monitor and PTK data with additional Linux guest usage data.	See above Linux section.	• TEP is included as a friendly, web-based graphical user interface (GUI) access tool. It provides real-time and historical displays of system usage by guest virtual machines such as Linux as well as overall z/VM system and hardware usage. • Across the entire OMEGAMON XE family of products, TEP provides default guided workspace views; customized views may also be easily developed. • Alerts, suggested response actions, and user-defined actions are also provided in TEP.	• Tivoli OMEGAMON XE on z/VM and Linux Data Sheet. • Review above z/VM and Linux links to interpret processor data properly.

Archived

z/OS	System Management Facility (SMF) – component of z/OS that provides a standardized method of recording system and job related usage data (CPU, memory, I/O, network, software usage, error conditions, and so on) to files and data sets. SMF itself collects usage data as well as accepts and records usage data from other z/OS software products. SMF record types distinguish the producer of the data. Data gathered is very granular and accomplished with low overhead from system level to individual process and thread levels. With Data Warehousing workloads, these other products likely include the ones below.	Each product that generates SMF data provides reporting capability with one or more products. These are described below.	Each product that generates SMF data provides text based online displays for data captured and action capability. Additional products are available to provide GUI displays. These are described below.	
	RMF - base z/OS performance and management product. Data gatherer component collects additional usage data (CPU, memory, I/O, network, error conditions, job) for z/OS and sysplex environments. SMF records generated are type 70-79 with type 70 for processor activity and type 72 for workload activity.	• RMF Postprocessor provides RMF Monitor I reports that may also be tailored via parameters. • RMF Spreadsheet Reporter, running on Windows, may be used to extract reports from RMF and converts them to a common spreadsheet format for further use in other business processes.	• RMF Monitor II and III provide menu-based TSO/E ISPF text online displays for real-time snapshot monitoring (II) and short term performance analysis (III). • RMF PM is a Windows workstation graphically based tool that displays the same data and allows one to monitor continuously all z/OS systems from one central location.	http://www-03.ibm.com/systems/z/os/zos/features/rmf/product/components.html
	Tivoli OMEGAMON XE on z/OS - its z/OS agent uses the System z zIIP specialty processor and RMF for some aspects of data collection. Very granular data is captured.	• Reports are available through OMEGAMON XE web workspace views.	• Omegamon XE on z/OS provides a friendly, web-based graphical access to z/OS system usage data real-time as well as historical tracking. • Like the entire OMEGAMON family of products, guided workspace views are provided as defaults, tailored views may be easily developed.	IBM Tivoli OMEGAMON XE on z/OS Data sheet

Archived

			• User-defined alerts when thresholds are exceeded and suggested actions are included. User-defined actions may also be automated.	
	DB2 - generates SMF record types 100-102 for its related work. Very granular usage data (CPU, locks, buffer pool, SQL call counts by type, thread activity, and so on) is gathered from the DB2 subsystem level to the individual user thread level with low overhead. Tracing capability is also provided but may have higher overhead. • Tivoli OMEGAMON XE for DB2 Performance Expert (OMEGAMON XE for DB2) on z/OS produces text reports and provides GUI displays for the captured data. This is highly recommended for problem determination and performance management. A DB2 performance warehouse of the reduced SMF data is included and may be used for historical analysis of DB2 activity.	• OMEGAMON XE for DB2 provides standard historical performance reports that may be tailored via parameters.	• DB2 commands such as DISPLAY may be used in the text-based online TSO/E ISPF DB2 Interactive (DB2I) option 7 DB2 Commands menu. Native TSO command line or the z/OS console may also be used to submit commands • OMEGAMON XE for DB2 provides text-based menu displays of real-time DB2 activity for the 3270 terminal environment. Very limited history (since the prior DB2 start) information is also provided in the 3270-based online environment; historical reporting is used instead. • OMEGAMON XE for DB2 also provides a friendly GUI interface either web-based or for DB2 usage data that is similar across the OMEGAMON XE family with guided workspace views that may be customized, and thresholds and alerts.	• DB2I - Online interface • Products Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Overview

9.4 Resource management of DB2 data warehouse queries

Resource management of query workloads has been identified as one of the important and challenging data warehouse performance considerations. Here we provide a common overview of resource management, for the BI, DB, and systems administrators. We cover the high-level steps in implementing query resource management, primarily focused on the z/OS DB2 tier. Chapter 11, “Resource management of data warehouse mixed workloads” on page 263, which is targeted for the systems administrator (though written so that other administrators can follow), covers each of these steps in more detail. Additionally, other data warehouse workloads, such as refresh, maintenance, extraction and the DB2 subsystems themselves, are also covered.

9.4.1 Steps for resource management

Here are the high level steps:

1. Determine query workloads

   Determine the various query applications and workloads, associated users, business requirements, and importance.

2. Determine means to differentiate work

   Determine the options for differentiating (qualifying) this work into appropriate buckets to manage it according to the business requirements, and monitor it as deemed necessary.

3. Determine and define WLM constructs

   Given the input from the BI administrator, DBA and DW management, the z/OS performance administrator determines the necessary WLM construct definitions, then builds and deploys the new WLM service definition.

4. Analyze and maintain resource management

   Continue to analyze the resource management effectiveness and maintain the WLM service definition and related resource management activities.

Steps 1 and 2 should occur very early in the DW/BI definition process, along with data security and governance. Next we provide an overview of each of the steps.

Step 1 - Determine query workloads

This requires input from several areas in the IT shop, including the administrators, applications staff and management. This could be driven by the BI administrator or the systems administrator. The idea here is to get key application information, such as source, users, performance requirements and Business Importance. This information should be documented in a format that the key administrators and management can appreciate. We suggest a table such as Table 9-2.

Table 9-2 Query application workload characteristics

Application	Type of work	Source	Users	Performance requirements	Potential qualifiers for classification
Sales - IBM Cognos BI Online Analysis and Reporting	Online Query	DRDA	Companies key knowledge workers	Consistent response times for smaller resource consumption queries. It is critical to give higher priority to the power users over others. Importance: High	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI users IBM Cognos BI Report
Sales - IBM Cognos BI Online Analysis and Reporting	Online Query	DRDA	Intermediate to novice knowledge workers	Consistent response times for smaller resource consumption queries. The larger consumption work gets discretionary resource (after power user work). Importance Medium to Low	DB2 sub-system DB2 connection ID IBM Cognos BI package name IBM Cognos BI users IBM Cognos BI Report
BookStore Service Center Operational BI	Online Query	DRDA	Service center reps	As service center reps. are interacting with clients via phone or chat, these Operational BI reports need to have a response time of <4s. Importance: High	DB2 subsystem DB2 connection ID Sub-application name User Report name

Application	Type of work	Source	Users	Performance requirements	Potential qualifiers for classification
BookStore Sales IBM Cognos BI Scheduled Reporting	Scheduled Batch Queries	DRDA	Proxy ID for Scheduled Reporting	These scheduled reports run post the daily refresh. It is critical that a specific subset of these reports be completed by 7 a.m. daily. Importance: High	DB2 sub-system DB2 connection ID IBM Cognos BI package name IBM Cognos BI users IBM Cognos BI Report

It should cover all the applications and interfaces that would be directing queries to the data warehouse, including potential local queries such as QMF or Batch.

As part of this exercise, it is valuable to consider the work profiles of the different query applications and workloads. In a sophisticated multitiered DW environment, some portions of work requests will hit local cache, or cubes, not hitting the z/OS DB2 tier. It might be valuable to capture all the query workload information in one table, as preparation for not only resource management of the z/OS DB2 tier, but other tiers, such as Linux.

A more complete sample, including other data warehouse workloads besides query, can be found in Table 11-1, “Data warehouse related work” on page 265.

Step 2 - Determine a way to differentiate work

This is perhaps the most important, yet under-valued, consideration for resource management of diverse workloads. In an environment where a single query triggered by a single user can potentially consume a significant portion of a system’s computing resources, one needs to ensure that work is getting qualified, hence managed, properly. Here is where discussion among the administrators is clearly valuable. For each application, what are the qualifiers available to isolate individual work requests for resource management, and how does the combined administration team want to exploit them?

As you saw in Table 9-2 on page 239, we listed some sample means to qualify work. Traditionally, the core work qualifiers for DB2 queries have been DB2 subsystem name and/or in the case of DRDA, DB2 connection ID (the user ID (aka DB2 AUTHID) utilized in a data source definition) and/or DB2 package (note this is different from an IBM Cognos BI package).

DB2 client information strings are gaining use, due to the additional and valuable means to qualify. This is especially so with the availability of DB2’s WLM set client information stored procedure.

With regard to the client information data, DB2 offers the following four special registers, associated with a DRDA unit of work (work request):

CLIENT_ACCTNG	The DB2 accounting string.
CLIENT_APPLNAME	User transaction name. Potential attribute to do more granular classification within an application. In an IBM Cognos BI application, there is the concept of an IBM Cognos BI “package”, which represents a data model for a given set of users.
CLIENT_USERID	The user ID submitting the request.
CLIENT_WRKSTNNAME	The user workstation name.

In Table 9-3 on page 241, we list all the available WLM qualifiers for classification of DRDA (WLM subsystem DDF) transactions. We have highlighted the client information qualifiers in blue.

Table 9-3 WLM qualifier (classification) filters available

Qualifier	Type	Description
Accounting Information	AI	Set by client info API: Client accounting string
Correlation Information	CI	Driver program name by default but application can set
Collection Name	CN	Collection name of the first SQL package accessed by the DRDA requester in the unit of work
Connection Type	CT	Always DIST for DDF server threads
Package Name	PK	Name of the first DB2 package accessed
Plan Name	PN	Always DISTSERV for DDF server threads accessed via DRDA requesters
Procedure Name	PR	Name of the procedure called if first request in unit of work
Process Name	PC	Set by client info API: Client transaction name
Subsystem Collection Name	SSC	Usually the DB2 data sharing group name
Subsystem Instance	SI	DB2 server's MVS subsystem name
Sysplex Name	PX	Name assigned to sysplex at IPL
User ID	UI	DDF server thread's primary AUTHID
Subsystem Parameter	SPM	Set by client info API: Assigned the concatenation of client user ID and workstation name

Though there is a large set of traditional qualifiers (non-client info) for a given application, many of them result in the same value, hence the usefulness of client info for additional granularity.

Some applications, such as IBM Cognos BI, make it easier to exploit these client strings, via their administration interface. Additionally, IBM Cognos BI offers *session variables*, which contain attributes of a given work request, that can be passed into the WLM set client info stored procedure. The primary qualifier we utilized for differentiating the IBM Cognos BI work requests, for resource management, was the client user ID. For an example of this exploitation, see 13.4, "Leveraging WLM set client information in IBM Cognos BI" on page 329. For more complete details about the overall use of DB2 client information strings, see Chapter 13, "Utilizing DB2 client information for resource management and monitoring" on page 321.

It is important to consider that these same client strings are not only utilized for WLM resource management and monitoring, but for end-to-end monitors, such as Tivoli, and problem determination products, such as Optim's Data Studio. An approach should be agreed upon that will work for all viewpoints of the various administrators.

Exploiting client user IDs for management and monitoring: If client user IDs are deemed a valuable means for resource management and monitoring, then consider early discussion about user ID naming conventions and groupings, with the appropriate IT personnel.

Step 3 - Determine and define WLM constructs

Based on the input information in Steps 1 and 2, the performance administrator would now determine how to build WLM constructs to match. The primary WLM constructs are *service classes*, *report classes, workloads,* and *classification rules.* Using the administrator-defined

classification rules, all incoming work is qualified (using qualifiers like the ones previously mentioned) into a service class for resource management, and, optionally, into a report class, for resource monitoring. WLM workloads represent groupings of service classes and are another mechanism for resource reporting.

Service classes are the core WLM construct. Each service class has a business *importance* and performance *goal* associated with it. Importance is simply a 1-5 setting, with 1 being most important to the business. WLM has three different types of goals:

Response Time	Manage each work request to complete within a certain time.
Velocity	Manage each work request by how much delay can be tolerated.
Discretionary	If all our important work is satisfied (meeting its goals), then allow this work to consume the discretionary resources.

One other key WLM concept we briefly introduce here, is *period aging*. A service class can have multiple periods within it. Each period has it' own goal and importance. Work requests always start out being managed in period 1. When multiple periods are defined, each work request moves to subsequent periods after it has consumed the current period's *duration* of service. Duration is defined in z/OS *service units*. Service units can either be associated with I/O, memory, or processor usage. For DRDA queries, the duration is only tied to processor usage, hence a direct relationship between processor seconds consumed and the duration.

Multiperiod service classes are highly valuable for providing consistent response times for the smaller resource consumption queries and recommended for diverse query data warehouse workloads.

Administrator collaboration on resource management

In preparation for determining and building the necessary WLM constructs, here is a list of items worth collaboration between the administrators.

- What WLM goal is most appropriate for a particular class of query work? Does some work have strict response time goal requirements?
- For multiperiod service classes, what are appropriate durations (amount of processor usage) to isolate one period from another.
- Review the final pecking order of WLM service class importance. For an example, see 11.2.10, “Service class relative importance” on page 278.
- How will WLM reporting and monitoring mechanisms (report classes and workloads) be utilized versus DB2 versus application reporting mechanisms? Is there value in doing some alignment, allowing collaboration and cross-checking on resource usage or performance debugging?
- Application versus system (WLM) resource management. Some query applications have their own means for resource management or prioritization of work requests. Be sure that any application-level prioritization is aligned with the system-level resource management.
- Resource limits. WLM, DB2, and some applications each have their own controls for limiting or governing resource consumption for given classes of work. For example, DB2 has its Resource Limit Facility (RLF). WLM has concepts such as discretionary goals or (more stringent) *Resource Groups*. Collaboration on if and how these controls should be utilized is important.

Step 4 - Analyze and maintain resource management

In 11.3, “Analysis and verification of a working WLM policy” on page 279, we provide details on how to analyze and maintain WLM resource management. Some quality data that can be shared among administrators is the resource consumption of the different query applications

and workloads and how they are doing versus the goals defined. Additionally, quality collaboration on the previous steps should help expedite the performance debug process. With any new application or major workload change, Steps 1-4 should be revisited.

Resource management summary

Data warehousing and analytics are two of the fastest growing computing paradigms in the industry. Additionally, key advances are being made in System z end-to-end resource management and monitoring. Over the next several years, one should expect frequent changes and new applications, workloads and technology in this space. Hence, this also signifies a frequent need for administrators recollaborating and revising their data warehouse and business intelligence end-to-end resource management strategy and implementation.

9.4.2 Distributed query flow

DB2 distributed database functionality is based on the Distributed Relational Database Architecture (DRDA)—an open, vendor-independent architecture for providing connectivity between a client and a database server. DRDA has been adopted by The Open Group as an industry standard interoperability protocol. Connectivity is independent from hardware and software architecture as well as from vendors and platforms.

DRDA is an important data warehouse component. Many of today's business analytics run as distributed applications. SQL arriving via TCP/IP and using DRDA protocol becomes zIIP eligible. Redirecting this work to a zIIP specialty engine helps to reduce the cost of using z/OS as a data sever platform with DB2 for z/OS. It is equally important to understand how to monitor a distributed SQL statement when examining the performance of a business analytics distributed application.

DRDA supports SQL as the standardized API for execution of applications and defines flows and logical connections between the application and a relational database management system (DBMS), program preparation, and BIND SQL statements for target DBMS.

By definition, distributed means that the application requires data from a remote DBMS rather than a local DBMS. Using DRDA connectivity, you can access data on DB2 for z/OS from client applications running on remote and different platforms, and from DB2 for z/OS you can access remote non-DB2 for z/OS systems. Figure 9-2 on page 244 shows the flow of a remote DRDA request and how monitoring Class 1 and Class 2 times is done.

Archived

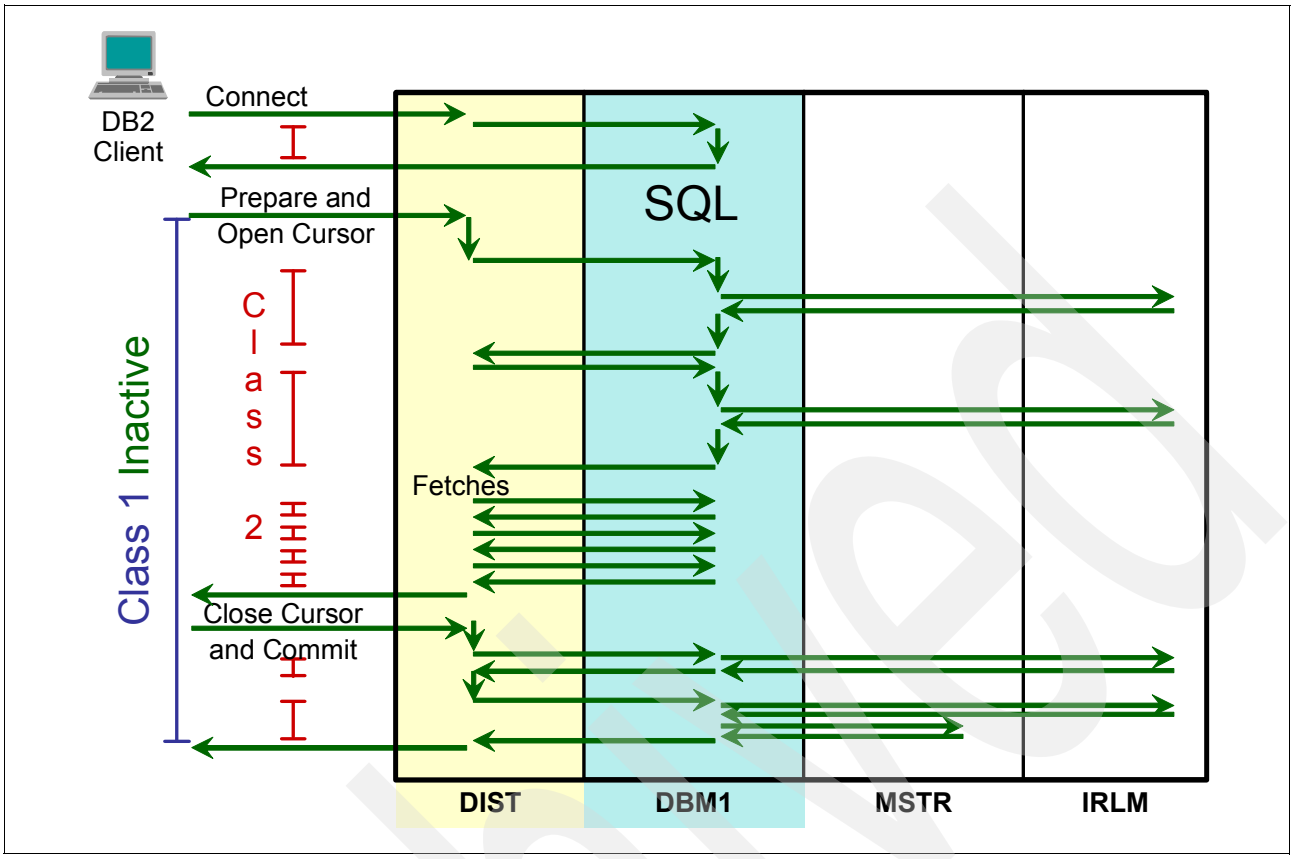


Figure 9-2 Distributed flow with accounting trace

Enclaves introduced a much more direct way of managing transactions that span multiple regions. An enclave is an entity that encapsulates the execution units (TCBs and SRBs) that execute programs on behalf of the same work request. See Figure 9-3.

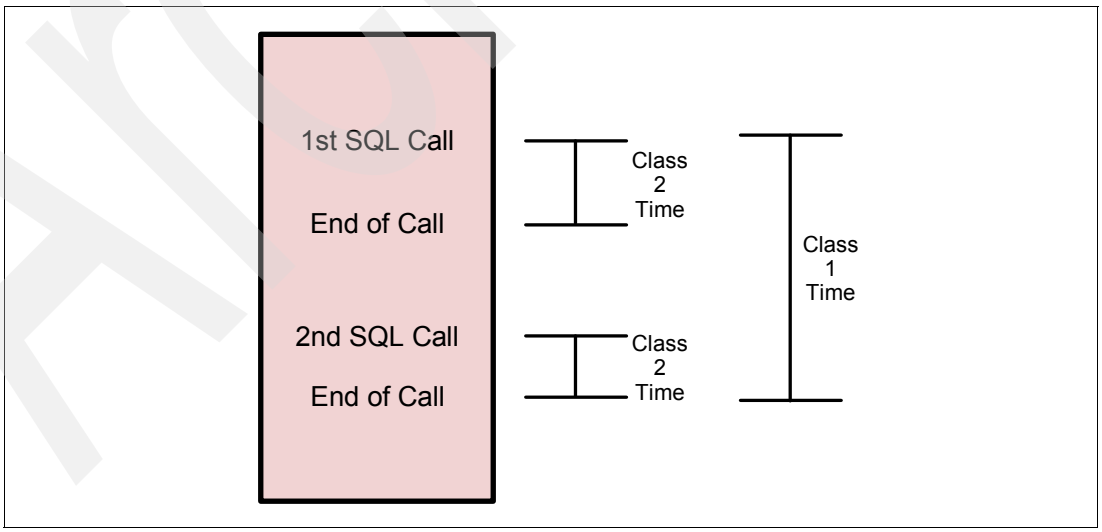


Figure 9-3 Enclave

You can also think of an enclave as a BUoW without address space boundaries. It is close to the user view of a transaction.

There are two types of enclaves, independent and dependent.

- Independent enclaves - These are true WLM transactions separately classified and managed in a service class. Distributed Data Facility (DDF) uses only independent enclaves. They represent a new transaction that has no association with a particular address space from a priority and account point of view.
- Dependent enclaves - These are a logical extension of an existing address space and inherit service classes from the owner address space. They continue an existing transaction that is running under dispatchable units not associated with the current home address space.

When an enclave spans a system boundary in a sysplex, it is called a multisystem enclave. Multisystem enclaves are made with the original enclave and foreign enclaves. The characteristics are:

- Enclaves are created by WLM on behalf of an address space owner, and one address space can own many enclaves.
- Enclaves maintain the priority and account information of the associated dispatchable unit.
- One enclave can include many dispatchable units (SRBs and Tasks) executing concurrently in multiple address spaces called participants, although it is much more common to have only a dispatchable unit per enclave. Enclave SRBs are preemptible (like tasks) by higher priority work. All its dispatchable units are managed as a group.
- WLM manages enclave dispatching and I/O priority.
- Enclaves do not own storage.
- Processor usage accounting in SMF records:
  - For a dependent enclave, the processor service consumed is accumulated in the SMF30 record of the owning address space and the SMF72 record of the owning address space service class period.
  - For an independent enclave (representing an individual WLM transaction), transaction count and resource usage are recorded in the SMF72 record (service class and report class). SMF30 records are accounted to the address space where the enclave runs.
- Multisystem enclaves are the sysplex scope of an enclave. This implementation requires a Coupling Facility.

Archive

Archived

10

Considerations for data warehousing with DB2 on System z

This chapter provides guidance on utilizing DB2 for parallel query and the associated resource considerations. It also discusses some of the components of DB2 for z/OS that could have a direct effect on a data warehousing and business intelligence environment, including considerations for a data warehouse.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

10.1 Overview

Data warehousing has always been in DB2's genetic makeup; it just has not always been called data warehousing. In the beginning, it was referred to as Decision Support.

Just as warehousing has been changing and maturing, so has DB2. Instead of determining what happened in the past, users want to use all that information to make immediate decisions today. Instead of making information kept in the warehouse available to just a few, it is being exploited by larger numbers of users, including customers. The focus has switched to getting the correct information to the correct person at the correct time. Warehouse information today must arrive rapidly and accurately.

DB2 has been continually enhanced to support data warehousing and the changes to data warehousing. The trick is to know how to get the most out of DB2 so that you are getting the most from your data warehouse. What is the best way to set up buffer pools in a warehouse environment, what about sort and the sort work files, not to mention certain DSNZPARM settings that should be examined to insure the settings chosen are the best choices for a data warehouse? Then there is data-sharing, the focus of this co-location book. Are there special considerations that should be taken into account when data warehousing is going to be part of your data-sharing group? What about the Coupling Facility, anything different that needs to happen with it? Storage management and virtual storage considerations; the list could keep going on.

10.2 Parallelism

DB2's query parallelism targets queries that are I/O intensive (table space scans and large index scans) and CP intensive (joins, sorts, and complex expressions). Its objective is to reduce the elapsed time of a query by taking advantage of available I/O bandwidth and processor power. For these reasons, parallelism could be significant to data warehousing. It is one of the most straightforward ways to reduce the elapsed time of a long-running query—by splitting a query across multiple processors doing multiple concurrent I/Os. It accomplishes all this with no code changes by allowing a query to run across two or more general purpose engines (CPs) with redirect of some portion of the work to a zIIP specialty engine if a zIIP is available.

Parallelism comes in three very different types: I/O, CPU, and sysplex query parallelism.

- I/O query parallelism was the first variation of parallelism. It was delivered in DB2 Version 3 and allowed a single CP to process multiple I/Os; fetching multiple pages into the buffer pool in parallel. Today, I/O parallelism is infrequently observed: it is also *not* zIIP eligible.
- Processor query parallelism was the second to arrive. It became available in DB2 Version 4 and is by far the most commonly observed among the three. A query is broken down into multiple parts with each part running on a different general purpose (CP) and zIIP specialty engine if one is enabled. Each processor running its portion of a query has the ability to perform I/O processing in parallel also.
- The final form of parallelism is sysplex query parallelism. Introduced with DB2 Version 5, it not only spreads a query across multiple processors, it will also take advantage of processors available to other DB2 members of a data-sharing group.

DB2 parallelism is not a given. A number of actions must be performed before the optimizer makes a decision to consider parallelism as an access path. First, all three forms of parallelism require DB2 to know that parallelism should be considered for a package or SQL

statement. In a dynamic SQL environment, the SQL type most likely used in a data warehousing environment, the special register CURRENT DEGREE is used to enable or disable parallelism. If the CURRENT DEGREE is set to “1” or “1 “, parallelism is disabled. If CURRENT DEGREE is set to “ANY” then parallelism is enabled. The default value for CURRENT DEGREE is set on the installation panel DSNTIP8 or by the DSNZPARM keyword CDSSRDEF on the DSN6SPRM macro. The value of the CURRENT DEGREE special register can be modified via the SET CURRENT DEGREE SQL statement. For static SQL, a BIND or REBIND of a package specifying the DEGREE keyword can be used to set the CURRENT DEGREE special register for that package.

It is recommended that the default for CURRENT DEGREE be normally set to “1”, disabling parallelism. Parallelism should be enabled on a per task basis to ensure that valuable processor resources are not wasted.

Note: For this study we followed the recommendation. The default for the CURRENT DEGREE special register was set to “1”. The SET CURENT DEGREE SQL statement was used to alter the special register to “ANY”.

The next parallelism control to consider is the number of CPs that DB2 will be allowed to use. MAX DEGREE on the installation panel DSNTIP8 or the DSNZPARM keyword PARAMDEG on the DSN6SPRM macro can be used to set the maximum number of CPs DB2 can use for processor query parallelism. The default for this value is 0 (zero). Zero allows DB2 to choose the degree of parallelism. In most cases, however, time should be taken to determine the best value for MAX DEGREE. A guideline for choosing a starting point for MAX DEGREE is: choose a value somewhere midpoint between the maximum number of CPs available and the maximum number of partitions that will be processed. If queries tend to be more processor intensive, make this value closer to the number of available CPs. If the queries will be I/O intensive, make this number closer to the number of partitions. This should be monitored and adjusted accordingly.

Note: This study used a MAX DEGREE **(**DSNZPARM PARAMDEG) equal to 8. It was initially set to 1 then changed to 5 for a short time before we settled on 8.

Once parallelism has been enabled, to actually get the optimizer to consider a query for a parallel access path, VPPSEQT, the parallel sequential threshold for buffer pools, must be some value greater than zero. The number used for this threshold is a percentage. When specified, it allows for a percentage (VPPSEQT) of the sequential steal threshold (VPSEQT) to be used for parallel processing. VPSEQT is a percentage of the VPSIZE (virtual pool size) value. For example, if VPSIZE is set to the hypothetical value of 100 pages and VPSEQT is set to 80 (%), then 80 pages will be available from the pool for sequential processing. Now, if VPPSEQT is set to 50 (%), the default, 40 pages, or half of the VPSEQT pages, are available for parallel processing.

Note: With the exception of BP0, the buffer pool used by the DB2 catalog, all other buffer pools had VPPSEQT (the parallel sequential threshold) set to its maximum, 100%, to insure full use by DB2’s parallelism.

As just mentioned, some actions have to take place that enable, or turn on, parallelism. However, even if everything is done in a DB2 subsystem to make parallelism available, it is not a forgone certainty that it will be used. There are a number of reasons that could prevent DB2 from selecting parallelism even after it has been enabled.

- Processor parallelism is not considered if only one engine exists. If there is any chance that parallelism could be part of your query execution, configure with at least two engines,

including zIIPs. One zIIP and one CP should satisfy the multiple engine requirement of processor parallelism.

- Processor parallelism can be disabled using the resource limit facility (RLF). Setting RLFFUNC to 4 in the RLF table for a plan, package, or authid prevents parallelism for that object type.
- Setting the buffer pool threshold VPPSEQT to 0 selectively disables parallelism. This will be discussed shortly. However, for now, VPPSEQT must be set to a number greater than 0 in order for DB2 to take advantage of processor parallelism.
- Cursor with hold prevents a query from taking advantage of parallelism
- Ambiguous cursor also prevents parallelism from taking place.

The query is broken into multiple parts, with each part running under its own Service Request Block (SRB), and each part performing its own I/O. Although there is some additional processor cost associated with parallelism when DB2 first decides to take advantage of query parallelism for the setup of parallelism, there is still a close correlation between the degree of parallelism achieved and the query's elapsed time reduction.

The second reason parallelism can be especially significant to warehousing is its potential to reduce the cost of doing business on the System z platform. The use of SRB in the previous paragraph is significant to this point. When taking advantage of parallelism in DB2, the parallel child tasks have the potential to be redirected to a System z Integrated Information Processor (zIIP). Software charges are unaffected by the additional processor capacity made available by the addition of zIIP processors, or specialty engines, to System z.

One of the easiest ways to improve the amount of zIIP redirect is by enabling parallelism. There is also the situation when you are not processing real-time queries but rather batch work. Taking advantage of DB2's parallelism in your batch jobs could not only increase the amount of redirect to a zIIP, it could also utilize a resource during a time when the usual DRDA redirect is low.

There is a specialized case of parallelism called a *star schema*, a relational database's way of representing multidimensional data, which is often popular with data warehousing applications. A star schema is usually a large fact table with lots of smaller dimension tables. For example, you might have a fact table for sales information. This sales table would be the largest table holding most of the data. The dimension tables could then represent products that were sold, the stores where those products were sold, the date the sale occurred, any promotional data associated with the sale, and the employee responsible for the sale. Using star joins in DB2 requires enabling the feature through a DSNZPARM keyword. You also should check a few other ZPARMs before using star joins because they can affect a star join's performance.

10.3 Co-location data-sharing

Co-locating the OLTP and data warehouse databases in a data-sharing group delivers additional capabilities beyond the more common non-data-sharing configurations. Since data resides in the same data-sharing group, a single application connection can now access data from both the operational and data warehouse databases. Additionally, co-location data-sharing more easily facilitates “in-database” transformations, providing ETL capabilities without leaving the database. This is also commonly referred to as ELT. See 7.2.4, “InfoSphere Warehouse refresh scenario in a data sharing environment” on page 142 for an implementation of this using InfoSphere Warehouse on System z.

10.3.1 ETL

Traditionally data is extracted from the operational database and moved to a different server for the transformation process. Changes are then applied to the data, which is subsequently uploaded to the data warehouse. This process requires data movement across multiple servers. Also, data is written to external files in the intermediate stages. With operational and data warehouse data co-locating in a data-sharing group, data can be moved directly from the operational database to the data warehouse. No externalization to files is necessary. Security is improved since data does not flow outside the Sysplex. The ETL function can be shortened significantly if the current process is gated by data movement. However, this is not always the case as the loading process takes a long time to complete. This is especially true for the situations where rows are inserted rather than loaded. In addition, the number of indexes of a table plays an important role in the data ingestion time.

This direct data movement between databases is most appropriate if two conditions are met. With this approach transformation is provided via SQL statements, and complex logic may not be possible. For data ingestion, the number of rows processed in each ETL cycle could be very high and these rows will be inserted into the data warehouse as a single unit of work. This could lead to data contention between the ETL process and the data warehouse queries. It may be necessary to break up the data processed in each ETL cycle into multiple parts so that processing one part at a time presents fewer opportunities of resource contention. To understand more about these considerations as well as follow an example of the implementation technique, read 7.2.4, “InfoSphere Warehouse refresh scenario in a data sharing environment” on page 142.

10.3.2 Accessing OLTP and data warehouse data with a single query

One of the advantages offered by co-location of OLTP and data warehouse databases in the same data-sharing group is the capability of accessing data from both databases within a single query. In most cases, information required for analysis is readily available from a data warehouse. But in some instances information related to recent activities is required to complete the analysis. If there are frequent refreshes to a data warehouse, this may not be necessary. But obtaining the latest real time information from the operational database could offer business advantages, especially in the realm of operational business intelligence.

Consider the situation of an online book store. A user is browsing the catalog and decides to purchase a book. It will be valuable to provide a list of books that other customers who bought this book also purchased. A table can be created offline to store this information. For each book listed in the catalog, there are a number of rows in the table storing the list of other books that customers tend to purchase as well. This table will be stored in a data warehouse. When a customer buys a book, a query is built to access this table to obtain the recommended book list. If this list is returned directly to the user without any filtering, it could create duplicates because some of the books were purchased previously by this user. To overcome this problem, it is necessary to build a history of the books bought by this user. This history can be kept in a data warehouse, but it needs to be updated regularly. A better option is to access the data directly from the operational database.

A query can be built to access data from the order table in the operational database and the recommended book list table in the data warehouse to produce this filtered list. See 6.3.4, “Operational BI reporting” on page 110 for an example. This improves data quality returned to the user and increases customer satisfaction. Since both tables reside in the same data-sharing group, performing a join or subquery across multiple tables from the two databases will perform as if the tables were coming from a single database only. This is an advantage only available to a data-sharing group. If this is not available, data has to be pulled

from the databases individually, and then the join processing is performed by the application. This scheme increases query response time and processor cost significantly.

There are concerns dealing with business intelligence queries accessing operational data directly. Most important of all is the possibility of processor monopolization by a business intelligence query. With the guidelines provided in this book, such concerns should subside. With careful construction of an appropriate WLM policy, it is straightforward to prevent the data warehouse workloads from interfering with the operational workloads. In this book, information is provided to demonstrate that protection of operational workloads can easily be achieved.

Another area of apprehension is the potential resource contention between operational transactions and business intelligence queries. With careful design, a business intelligence query should access only a small quantity of data from the operational database. One scenario is the access of customer data by an operational BI query. This type of query typically accesses just one customer of data since a customer representative is dealing with one user at a time. The number of rows fetched from an operational table is expected to be quite small. In addition, it is expected that an isolation of uncommitted reads will be used. Contention with online transactions should be minimal.

10.3.3 Global buffer pools

Data warehouse workloads are mainly read only. Generally the only exception comes from the ETL process. Some ETL programs read log records from the operational database and apply the changes to tables in the data warehouse. In this study, changes were limited to access the inventory table only. The changes to this table were then applied to the inventory history table. This produces a history of the inventory of books so that analysis could be performed.

Although a log reading program runs in a data warehouse LPAR, it does not create any group buffer pool dependency with the operational LPARs. Instead of accessing a table directly, it reads the log records through a DB2-provided API. As such, there is no interference to the operational workloads.

On the data apply side, the processing does create group buffer dependency with the data warehouse workloads. In our study, changes were written to the inventory history table creating write interest. Queries accessing this table from another data warehouse LPAR generated read interest. Since a small number of DB2 objects were related to this table, only one additional group buffer pool was created to support the ETL processing. The amount of work performed by the ETL process was not substantial, and the data-sharing overhead due to this processing was minimal.

The net is that co-locating operational and data warehouse processing in the same data-sharing group workloads generally does not lead to any additional group buffer pool dependency against the operational database objects. Unless data warehouse queries access operational tables directly, no additional contention is expected. Even when accesses are performed by the data warehouse queries, additional data-sharing traffic is small because these queries create read interest only for the data warehouse LPARs.

ETL processes typically run on the data warehouse LPARs. They will lead to some data-sharing overhead in a multiple LPAR environment, but it is not expected to be significant. In contrast to a single LPAR environment, group buffer pools are needed for those objects that are refreshed through SQL inserts. Refreshes through offline loading do not create group buffer pool dependency.

10.3.4 Lock structure

In a production environment, ETL activities could introduce a noticeable amount of global locking activities between the ETL LPAR and the query LPAR. This depends on the volume of updates to the data warehouse. Since the operational and data warehouse workloads share the same data-sharing group, it is wise to monitor the lock structure activities. If false contention rate is creeping higher, it may be necessary to increase the lock structure size.

It should be noted that the introduction of additional global locking activities only takes place when the ETL and the queries run on different LPARs. If there is only one LPAR for the data warehouse workload, it is not expected to introduce any additional global locking activities even though it shares the same data-sharing group with the operational workloads. Similarly, performing data warehouse refreshes through offline loading does not lead to an increase in lock structure activities.

10.4 DB2 work files

It is better to define more work files than fewer. If you are deciding between a few large work files and many small work files, you will find that DB2 sort usually performs better with lots of small work files. Temp space and work files now all share the same work files. So, in addition to sort, declared global and created temporary tables, some merge, stat and outer joins, materialized views, materialized nested table expressions, non-correlated subqueries and triggers now share the work files. If these objects are in use in your shop, you may want to consider allocating more work files in DB2 9 than in previous versions.

While you are defining all those small work files, you should minimize I/O contention by placing as many of the work files as possible on different I/O paths and different disks. You also should take advantage of Parallel Access Volumes (PAV) whenever possible. PAVs could significantly reduce your contention. Besides the obvious problems that occur with I/O contention, DB2 will reduce the prefetch quality to four or lower if the buffer pool is constrained. A high number of synchronous reads may also be an indication of an insufficient number of work files.

When defining the sort work table spaces in DB2 9, remember that these table spaces will now be segmented. Use SEGSIZE 16 for all work files defined with zero secondary extents and SEGSIZE 4 for all others. This should minimize wasted space, with SEGSIZE 4 working out to be a more reasonable size for DGTTs.

We also recommend that you do not specify a secondary extent for any of the table spaces defined to DSNDB07 (or whatever your sort work database is called if you are a data-sharing shop). Secondary extents will work with sorts but why have sort do extra work processing multiple extents?

DB2 9 also favors the 32 K work files over the 4 K work files for sort keys greater than 100 bytes. However, if an adequate number of 32 K work files is not available, DB2 uses the 4 K work files as reported in IFCID 2. In previous releases, the 32 K buffers were only used for long sort keys to avoid tag sorts. Because they were not used frequently, most have few 32 K work files defined prior to DB2 9. With DB2 9, that has all been reversed. Ensure that you have plenty of 32 K work files defined. DB2 does not support 8 K and 16 K work files.

Note: For this co-location study, 100 4 KB work files were allocated with a SEGSIZE of 16 and a primary space allocation of 2047 MB (zero secondary allocation). Ten 10 32 KB work files with a SEGSIZE of 16 and a primary space allocation of 2002 MB (zero secondary allocation) were used for the non-data-sharing subsystem.

For the data-sharing environment, each DB2 member used 50 4 KB work files with a primary space allocation of 2047 MB (zero secondary allocation) and a SEGSIZE of 16, ten 32 KB work files with a primary space allocation of 2002 MB (zero secondary allocation) and a SEGSIZE of 16.

Finally, in DB2 9, the work file database and TEMP database have converged into a single database. However, now that the two are combined, another issue has arisen. Declared Global Temporary Tables (DGTTs) cannot span multiple table spaces. This is the exact opposite from how we would like to define the work files. For this reason, APAR PK70060 has DGTT favoring work files that are defined with secondary extents while RDS sort will favor work files with zero secondary extents defined. If you are running DB2 9, you should definitely be planning to install this APAR if you do anything with DGTTs. This work file enhancement is available immediately in Conversion Mode (CM).

10.5 Buffer pools

Accessing data from disk is almost always more expensive than reading it directly from memory. That is why data caches are so important. DB2's access of data is no different. For DB2, its data caches come in the form of multiple buffer pools. DB2 provides the capability to define up to 80 buffer pools in varying page sizes. There are:

- 50 4 KB buffer pools ranging from BP0 to BP49
- 10 8 KB buffer pools ranging from BP8K0 to BP8K9
- 10 16 KB buffer pools ranging from BP16K0 to BP16K9
- 10 32 KB buffer pools ranging from BP32K to BP32K9 (there is no BP32K0)

All buffer pools are defined in central memory above the 2 GB bar. Allocating larger buffer pools, if storage is monitored via the IFCID 225 record, normally has more positive effect than negative. However, there are better and worse ways to allocate DB2's buffer pools.

BP0, for example, should be reserved for the DB2 catalog's use. In most cases, it does not need to be as big as pools holding user data or user indexes. A buffer pool also needs to be dedicated to the sort work database DSNDB07. The pool used to support sort will have unique characteristics different in most cases from other data and index buffer pools. Pool characteristics refer to the different buffer pool thresholds, and other settings, that can be externally set for each buffer pool.

The pools used for the co-location study were in most cases larger, with some pools dedicated to certain table spaces and index spaces. For the most part, pool characteristics, with the exception of BP7 used for DSNDB07, were all very similar. Care was taken to separate certain table spaces and indexes into certain pools.

Archive

Note:

BP0, BP8K0, and BP16K0 are defined with a VPSIZE of 2000, 1000, and 500 respectively. Because these particular buffer pools are only used by the DB2 catalog, it was decided to leave PGFIX set to NO and not page fixing any pages for these buffer pools.

BP7, used as the sort buffer pool, was defined with the PGFIX attribute set to NO, page fixing NO buffers used by the sort buffer pool.

BP1 through BP6 and BP8 through BP30 had the page fix attribute set to YES, page fixing all pages in these 28 buffer pools. These buffer pools where used in support of user table spaces and user index spaces. Example of space allocations for these buffer pools are:

- BP1 1,612,482 buffers
- BP2 568,603 buffers
- BP3 205,349 buffers
- BP4 272,529 buffers
- BP5 279,834 buffers
- BP6 205,007 buffers
- BP13 335,000 buffers
- BP14 50,000 buffers
- BP15 25,000 buffers
- BP16 25,000 buffers
- BP17 275,000 buffers
- BP18 45,000 buffers
- BP19 13,000 buffers
- BP28 18,520 buffers
- BP29 1,570,000 buffers

The remaining buffer pools were all defined with values between 100 and 6,000 buffers.

Minimal buffer pool tuning was performed during the course of the co-location study.

32 K sort buffers were not exploited.

10.5.1 Buffer pool page fixing

To help reduce processor overhead, DB2 V8 conversion mode (CM) introduced a buffer pool feature that can have a significant effect on processor usage by long-term page fixing selected DB2 buffer pools.

When a page is brought into storage, DB2 fixes and releases the page for I/O processing as required by the channel. The processor cost for this operation using the 64-bit instruction can be as high as 10%. To avoid this processor cost for every page being touched by DB2, the ALTER BUFFERPOOL command has an option to page fix one or more entire buffer pools. After monitoring buffer pool activity and storage usage, it is suggested to first page fixing the pools with the highest I/O rates. If you need more granularity, start with the pools with the poorest hit ratios.

TIP: We suggest using relative “I/O Intensity” to determine which buffer pools are best candidates for PGFIX(YES). The higher I/O Intensity is, the better. The following formula can be used to calculate I/O Intensity:

```
((Sync Reads + Async Pages Read) by (SPF + Async Pages Read) by (LPF + Async
Pages Read) by (DPF + Sync Writes + Async Pages Written)) / VPSIZE
```

In either case, always start with the pools most critical to the overall performance of your DB2 subsystem. It is also suggested to just page fix a few pools at a time, then monitoring your storage usage before page fixing additional pools. There usually is no need to page fix all of the buffer pools, although the more buffer pools that are long-term page fixed, the greater the possible processor savings. However, like most things in DB2, nothing is free. You must ensure that the storage page fixed does not exceed your real storage. DB2 needs to remain 100% backed by real storage and the buffer pools, in most cases, can account for the greatest percentage of real storage use.

To change the PGFIX option, which is NO by default, use the following ALTER BUFFERPOOL command:

```
ALTER BUFFERPOOL   (bpname)   PGFIX ( NO | YES )
```

When altering the PGFIX option, the buffer pool does not get long-term page fixed in real storage until that buffer pool's next allocation. To have the page fixed pool take effect sooner than later, some actions need to take place that will force the buffer pools to be reallocated. For any buffer pool other than the three pools used by the catalog, perform the following three steps:

1. ALTER BUFFERPOOL (bpname) PGFIX(YES)
2. ALTER BUFFERPOOL (bpname) VPSIZE(0)
3. ALTER BUFFERPOOL (bpname) VPSIZE(integer value)

10.6 Coupling Facility

Decisions about Coupling Facilities should made by collaboration between DBAs and z/OS system programmers. Here are some considerations.

- DB2 requires more CF storage than the sum of all the structures. DB2 requires additional CF storage for failover and GBP duplexing.
- Do not allocate a structure to a CF arbitrarily. One technique is allocating the primary GBPs to one CF, while the lock structure is assigned to another CF (secondary GBPs can sit there as well).
- More than one CF is needed for HA purposes.
- At least one of the CFs should be external for HA purposes.
- z/OS system programmers own the CFRM policy, but they should work with DBAs to jointly define the contents for the DB2 structures.
- Although CF usage is more OLTP oriented, active data warehousing elevates the usage of CF in DW workloads as well.

10.7 DSNZPARM recommendations

DB2 9 for z/OS introduces many enhancements that benefit data warehousing systems. So some of the previous recommendations for DB2 V8 system parameters need to be adapted and complemented by additional recommendations.

This section summarizes the recommended values for the DB2 9 system parameters and DBM1 virtual storage estimates for huge data warehouse environments such as SAP Business Warehouse. We recommend that you keep system parameters set to their default values if not otherwise stated in this section.

This section is divided into highly recommended parameters, recommended parameters and buffer pool setting recommendations.

10.7.1 Highly recommended parameters

- NUMLKUS 2097152
- NUMLKTS 1000000
- IRLMRWT 600
- DEADLOK 5.1 (with relatively few entries in the IRLM lock table, this parameter can also be set to 1.1)
- EVALUNC YES
- CONTSTOR YES
- CONDBAT: Set CONDBAT to the same value as MAXDBAT
- MONSIZE 500000
- NPGTHRSH 10
- STARJOIN 2
- SJTABLES 4
- MAX_OPT_CPU 2
- MXQBCE 1023
- MXDTCACH 128 MB (up to 256 MB if sufficient real storage)
- EDMPOOL: Value recommended by DB2 installation CLIST should be accepted, but at least 64000 KB
- EDMDBDC: Value recommended by DB2 installation CLIST should be accepted, but at least 150,000 KB
- IMPDSDEF NO
- IMPTSCMP YES
- TBSBPOOL BP2
- IDXBPOOL BP3
- TBSBPLOB BP40
- TSQTY 144
- IXQTY 144
- MGEXTSZ YES

10.7.2 Recommended parameters

- EDMSTMTC 300,000 KB
- MAXKEEPD <= 8000
- SRTPOOL >= 28,000 KB
- MAXRBLK >= 100,000
- CHKFREQ 2 minutes
- DSMAX 20,000
- RETLWAIT 1

Archived

- UTIMOUT 3
- URCHKTH 1
- URLGWTH 100 K
- XLKUPDLT TARGET
- SMFACCT 1, 2, 3
- SYNCVAL 30
- SEQCACH SEQ
- LBACKOUT NO (may be set to AUTO if ramifications are known)
- PARAMDEG: Number of available CPUs
- PCLOSET 25
- PCLOSEN 15
- MLMT 4 GB (needs to be backed by real storage)
- ACCUMUID 11
- LRDRTHLD 10
- SYSTEM_LEVEL_BACKUPS YES
- STATROLL YES
- MINSTOR YES
- CACHEDYN_FREELOCAL 1
- OPTIXOPREF ON (introduced by APAR PK51734)

10.7.3 Buffer pool setting recommendations

If there is sufficient real storage available, it is recommended to set the buffer pool attribute PGFIX=YES. Note that this must not be set if the buffer pools are not always backed by real storage. If there are compressed indexes, these indexes should be backed by a separate buffer pool with PGFIX=NO. There is no benefit in page fixing this buffer pool, because there is a special I/O work area for these objects that is always page-fixed.

10.7.4 DBM1 virtual storage consumption below the 2 GB bar

With DB2 V9, some storage structures reside below the 2 GB bar. The average sizes of these structures in SAP environments are as follows. Note that the actual sizes may vary. Therefore, it is beneficial to leave sufficient headroom of 300 MB.

- Per DB2 thread: 2.5 MB
- Per open data set: 1.5 KB
- System storage: Non-data-sharing 124 MB
- data-sharing 237 MB
- RID pool: MAXRBLK * 0.25
- EDM pool: EDMPOOL * 0.75
- SQL statement in local DSC: 0.4 * (M * 3.5 KB + MAXKEEPD * 1.1 * 40 KB)

  (M is the larger value of MAXKEEPD * 1.1 and the max. number of statements in the EDM statement cache. Avg. statement size in EDM statement cache is 30 KB)
- Storage cushion: LPVTSOS + LPVTCRIT + LPVTMVS with

- LPVTMVS (storage for MVS) = (DSMAX * 1300) / 1024 * 1024 + 40K
- LPVTSOS (cushion warning) = MAX (5% of Extended Region Size, LPVTMVS)
- LPVTCRIT (storage for must complete) = (CTHREAD + MAXDBAT + 1) * 20K

As a general starting point, you may assume to accommodate roughly 400 threads in an individual DB2 subsystem. To control the number of database connections that are opened from a single Java node in the JDBC connection pool, you can adapt the following properties of the Java system data source:

- sysDS.maximumConnections
- sysDS.initialConnections

Note: When an application requests a new connection and no connection is available in the pool, then the connection pooling mechanism creates sysDS.initialConnections new connections.

- sysDS.connectionLifetime
- sysDS.maxTimeToWaitConnection
- sysDS.runCleanupThreadInterval

10.8 Automatic identification of latch contention and DBM1 below-the-bar virtual storage

DB2 V9 adds a monitor for automated health checking with the intent to improve the reliability, availability, and serviceability (RAS) of your DB2 subsystems.

Two issues that have sometimes impacted DB2 in the past have been processor stalls, which cause latch contention, and DBM1 below-the-bar storage shortages. These issues were identifiable by the following processes:

- DB2 V7 introduced a serviceability command, DISPLAY THREAD(*) SERVICE(WAIT), to help identify processor stalls.
- The IFCID 225 records identify DBM1 storage constraint issues.

However, these processes were manual, and DB2 did not provide an automated means to identify them.

With DB2 9 in conversion mode, a built-in monitor runs from restart to shutdown and checks the health of the system in one-minute intervals. The built-in monitor identifies processor stalls (for system, DBAT, and allied agents) that result in latch contention. The monitor attempts to clear the latch contention by a temporary priority boost via WLM services to the latch holder. This should allow you to run closer to 100% processor utilization by reducing the chances that less important work can hold a latch for an extended period of time, causing important work to stall.

In addition, DBM1 storage below the 2 GB bar is monitored for critical storage increases, and messages are sent when thresholds are reached.

You can view the health of your system by issuing the command shown in Example 10-1.

Example 10-1 DISPLAY THREAD(*) TYPE(SYSTEM) output

```
-DISPLAY THREAD(*) TYPE(SYSTEM)
DSNV401I  -DB9B DISPLAY THREAD REPORT FOLLOWS -
DSNV497I  -DB9B SYSTEM THREADS -  778
DB2 ACTIVE
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
DB9B     N  *     0 002.VMON  01 SYSOPR            0069     0
V507-ACTIVE MONITOR, INTERVALS=429522, STG=7%, BOOSTS=0, HEALTH=100%
. . .
```

When DBM1 storage below the 2 GB bar reaches a threshold of 88, 92, 96, or 98 percent of the available storage, messages (DSNV508I, DSNV510I, DSNV511I, and DSNV512I) are issued reporting the current DBM1 storage consumption and the agents that consume the most storage. Example 10-2 shows a sample of the DSNV508I, DSNV510I, and DSNV512I messages.

Example 10-2 Sample DSNV508I, DSNV510I, and DSNV512I messages

```
DSNV508I -SE2O DSNVMON - DB2 DBM1 BELOW-THE-BAR 09
STORAGE NOTIFICATION
91% CONSUMED
87% CONSUMED BY DB2

DSNV510I -SE2O DSNVMON - BEGINNING DISPLAY OF LARGEST
STORAGE CONSUMERS IN DBM1
DSNV512I -SE2O DSNVMON - AGENT 1: 094
NAME    ST A  REQ ID           AUTHID   PLAN
----    -- -  --- --           ------   -----
SERVER  RA * 18461 SE2DIA004     R3USER   DISTSERV
LONG 172OK VLONG 388K 64BIT 2056K

DSNV512I -SE2O DSNVMON - AGENT 2: 095
NAME    ST A  REQ ID           AUTHID   PLAN
----    -- -  --- --           ------   -----
SERVER  RA * 927O  SE2DIA001     R3USER   DISTSERV
LONG 1672K VLONG 388K 64BIT 2056K
```

You can easily write automation based on these messages and take proactive actions to prevent the problem from becoming serious.

Furthermore, the DISPLAY THREAD command is extended to include STORAGE as an option. You can now issue the command shown in Example 10-3.

Example 10-3 DISPLAY THREAD(*) SERVICE(STORAGE) output

```
DISPLAY THREAD(*) SERVICE(STORAGE)
-DISPLAY THREAD(*) SERVICE(STORAGE)
DSNV401I  -DB9A DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -DB9A ACTIVE THREADS -
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
RRSAF    T        8 DB9AADMT0066 STC      ?RRSAF   0081     2
V492-LONG 140 K VLONG 28 K 64 1028 K
RRSAF    T     4780 DB9AADMT0001 STC      ?RRSAF   0081     3
V492-LONG 140 K VLONG 28 K 64 1028 K
BATCH    T  *    41 PAOLOR7C     PAOLOR7  DSNTEP91 002D    63
V492-LONG 716 K VLONG 404 K 64 1028 K
```

```
TSO      T  *     3 PAOLOR7      PAOLOR7           0068    64
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -DB9A DSNVDT '-DIS THREAD' NORMAL COMPLETION
***
```

The health monitor feature should correct some problems, help provide early warning about other problems, and at least provide additional diagnostic information. This automated monitor should help lower the cost of ownership for DB2 and help ensure that clients maintain healthy DB2 systems.

Tip: In order to properly detect these processor stalls, we recommend that you run the started task for DB2 MSTR in the SYSSTC dispatching priority.

To avoid failover, DB2 attaches a monitor task for each system address space in the following order:

1. MSTR
2. DBM1
3. DIST, if present

The tasks are active for intervals and check for agents that are stalled on latches and conditions of below-the-bar cache.

Archived

Archived

11

Resource management of data warehouse mixed workloads

In this chapter we discuss z/OS resource management implementation considerations and guidance for adding data warehouse workloads to a System z environment. We cover the following topics:

- Implementation considerations and a process for determining appropriate WLM service definitions for either standalone data warehouse workloads or growing one's current transactional-based WLM service definition to include data warehouse workloads.
- Sample WLM workloads, service classes and classification rules for data warehouse workloads. The resulting sample service definition can be found in Appendix D, "Sample WLM service definition" on page 433.
- Considerations for analyzing performance and verification of a working WLM policy.
- Additional System z resource management considerations for data warehousing.

If you are not familiar with WLM concepts, we suggest you first read Appendix F, "WLM refresher" on page 491.

Another key resource for this chapter, which we reference throughout, is *System Programmer's Guide to: Workload Manager,* SG24-6472. We have included some of the applicable materials here.

© Copyright IBM Corp. 2010. All rights reserved.

11.1 Introduction

The evolution of data warehousing (DW) has spawned a diverse set of workloads, each having unique service requirements. Some examples are Operational BI (or tactical queries), online analytical processing, scheduled reporting, data warehouse refresh and deep analytics, such as data mining. Additionally, the user base has grown beyond senior executives, mid-level management and business analysts to include client-facing personnel, such as service representatives. The coexistence of this mixed workload and how to distribute resources has been identified as one of the most important performance considerations in managing a data warehouse and business intelligence computing system.

This is actually one of the core reasons this book was developed. WLM is known for its ability to manage diverse workloads efficiently, but to exploit that sophistication, you must implement it in accordance with your business objectives. Underestimating the importance of having proper resource management in place for these types of mixed workloads can lead to inefficient use of resources and unhappy users.

Additionally, we'll provide guidance on how to analyze WLM performance of the data warehouse workloads and maintain the WLM service definition. In the subsequent experiment chapter we will show examples of using various policy definitions.

11.2 WLM service definition with data warehouse workloads

We designed this section as a potential process flow a performance administrator might consider when building a WLM service definition for either:

- A new dedicated system for a set of data warehouse workloads.
- Expanding their current transactional (OLTP) service definition to include data warehouse workloads.

Optimally, this process should include collaboration with the following administrators:

- Business Intelligence
- DB2 Data Base (DBA)
- z/OS performance

Together, these administrators should all have a consistent understanding of the relative business importance of the various data warehouse workloads, how they relate to the potential existing workloads, and so on (OLTP).

For each step, we discuss some considerations, followed by what we included in our sample service definition or implemented in our experiments. Our resulting sample service definition clearly does not cover the complete set of diverse workloads and products in a client's shop, though we did attempt to include a reasonable set of data warehouse-related definitions to get started.

A resource management implementation process should be started very early in the implementation of a data warehouse and associated Business Intelligence applications. Consider starting it at the same time security and data governance processes are started. Here is a sample set of implementation steps, and ones we will cover in this section.

- Determining the warehouse workloads, sources and business requirements

- Determining appropriate WLM service definitions:
  - Workloads
  - Service classes
  - Determining multiperiod goals and durations
  - Classification
  - Report classes
  - Consider additional protection of work (processor or storage critical, Resource Groups)
  - Multiple policies
  - Relative importance across mixed workloads

Building a WLM service definition is an iterative process. Anticipate more iterations and larger changes when first running new workloads. Then it will be smaller changes to hone goals, period durations, and classification.

11.2.1 Determining the warehouse workloads, sources, users, and relative importance

Before getting started with the actual WLM definitions, you first need to ascertain some key information relative to the workloads. This will be utilized as input to the actual WLM definition. The more thorough a job done here, the easier it is later. Here are some key questions that should have collaboration among the administrators:

- What applications, processes and servers compromise the data warehouse workloads?
- Where does the work arrive from?
- Who are the users?
- What is the relative importance to the business?
- What are the specific performance requirements?
- How can the work be differentiated so WLM can properly classify for management and monitoring?

We strongly suggest capturing this information in a table such as Table 11-1. An important part of this process is to understand the relative importance of the different applications, packages within applications, and the different users running these applications. After this process, one could further evolve this table and relate it directly to the WLM service definition.

Table 11-1 Data warehouse related work

Application or Process or Server	Type of work	Source	End-users	Performance requirements	Potential qualifiers for classification
BookStore Sales IBM Cognos BI Online Analysis and Reporting	Online Query	DRDA	Key critical knowledge workers	Consistent response times for smaller resource consumption queries. It is critical to give higher priority to the business key knowledge workers. Importance: High	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI users IBM Cognos BI report

Application or Process or Server	Type of work	Source	End-users	Performance requirements	Potential qualifiers for classification
BookStore Sales IBM Cognos BI Online Analysis and Reporting	Online query	DRDA	Intermediate to novice knowledge workers	Consistent response times for smaller resource consumption queries. The larger consumption work gets discretionary resource, after higher importance users. Importance: Medium to Low	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI users IBM Cognos BI report
BookStore Service Center IBM Cognos BI Operational BI	Online query	DRDA	Service center reps	As service center reps. are interacting with clients via phone or chat, these Operational BI reports need to have a response time of <4s. Importance: High	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI users IBM Cognos BI report
BookStore Sales IBM Cognos BI Scheduled Reporting	Scheduled batch-like query	DRDA	Proxy ID for scheduled reporting	These scheduled reports post the daily refresh. It is critical that a specific subset of these reports are completed by 7 a.m. daily. Importance: High	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI users IBM Cognos BI report
QMF Reporting	Online query	TSO	Power user Standard user	Consistent response times for smaller resource intensive queries. It is critical to give higher priority to the power users over others. High to Low	TSO user ID
BookStore Daily Refresh - SQW	Batch refresh	DDF	Proxy ID?	Daily refresh should be complete within 3 hours. High during window	SQW DB2 connection ID
BookStore Dynamic Refresh - CDC	Dynamic refresh	STC	n/a	Dynamic refresh should not fall behind Transactional workload by > 10 minutes. High	CDC started task name
BookStore DW maintenance	Batch	JES2	n/a	Typically run within a batch window, though occasionally concurrent with prime time High during window	JES2 batch jobnames
BookStore DW Extractions	Feed other BI apps	DRDA	Key knowledge workers	.....	....
DB2 Server @spaces	DB Server	STC	n/a	Highest	Started task names

- The work items in black were added to the table to be more representative of perhaps a typical customer shop, but they were not implemented for this project.
- We utilize the terms “Online query” versus “Batch query” in our sample. Our definition of “Online” is a user at a terminal waiting for a response. Batch work is more like background, where a user is not waiting at a terminal for a response. Also, *Batch* does not necessarily signify JES2 batch jobs. In the case of the IBM Cognos BI batch scheduled reporting, this work all arrives via DDF.

- At this point, one might not know what potential *qualifiers* are available, for the given type of work. We suggest working with the DBA and BI administrators to determine what might be useful. Recall that this is not only valuable for resource management, but also for monitoring and reporting.
- Once the associated WLM service definition is defined, we believe there is value in extending this table to relate it to your WLM service definition (service classes). This could be kept as a live document, and referenced and utilized when key workload management questions arise or *major* workloads or WLM service definition modifications are made. It would also be valuable to document why certain decisions were made.

We did not attempt to cover all possible workloads or applications, but enough to give a reasonable sampling of key DW workloads. To round out the view, we also thought we should include a similar view of the Transactional workloads utilized in our experiments.

Table 11-2 Transactional workloads

Application or process or server	Type of work	Source	Users	Performance requirements	Key qualifiers
BookStore Web Application Browse/Buy Online	Online transaction	WebSphere Application Server- CB	Clients	Response Time < 0.5 sec High	URL
BookStore Web Application Search Order	Online transaction	WebSphere Application Server - CB	Customers	Response Time < 1 sec Medium	URL
WebSphere server @spaces	Application server	STC	n/a	Highest	Started task names

11.2.2 Determining WLM workloads

Recall that a WLM workload is a logical grouping of service class performance statistics. It is useful for a high-level view of resource consumption across the overall system or sysplex.

After examining some current customer service definitions and the key types of workloads in a data warehouse environment, we decided our primary objective for defining our WLM workloads would be to isolate the transactional and data warehouse workloads. This would allows us to have a high-level understanding of the resources being consumed by the different categories of work.

WLM workloads for mixed transactional and data warehouse

DWMAINT	All the batch maintenance type processes in support of the data warehouse databases and workloads. This excludes Data Warehouse Refresh (ETL) and Scheduled Reporting.
DWQUERY	All the online query work accessing the data warehouse
DWREFRESH	All processes or jobs associated with refreshing (including ETL) the data warehouse
DWSCHREP	All scheduled (batch-like) query reporting accessing the data warehouse
DWSTC	All started tasks serving the data warehouse workloads
OPERATNL	All the transactional (OLTP) transaction workloads

OPERBAT	All the batch processes in support of the transactional (OLTP) workloads
OPERSTC	All started tasks serving the transactional workloads
SERVICES	Additional system-like services outside of SYSTEM
TSO	Represents traditional TSO workloads. Excludes QMF, which would be included in DWQUERY.
NOWKLD	Represents work that came into the system, but perhaps was not classified properly. This could be utilized as a way to red-flag work coming into the system that is not managed properly. It should always consume zero or near-zero resources. For our study, we associated the default DDF service class with this workload.
SYSTEM	The SYSTEM workload is predefined by WLM itself. It includes Service classes SYSSTC, SYSSTCx, SYSTEM, and SYSOTHER.

Other considerations might include adding workloads for ODS type processing or non-production test workloads.

11.2.3 Determining WLM service classes

Determining service classes and the classification of work into them are the most critical decisions in setting up optimal z/OS resource management. Recall that a WLM service class definition represents how we would like WLM to manage resources, for that group of work. In defining these, here are some key questions:

- What additional service classes do we need to add for the data warehouse workloads?
- For each new service class
  - Do we desire or require multiple periods?
- For each service class or period:
  - What is an appropriate WLM goal, duration, and importance?
  - What workload should the service class be associated with?
  - Do we need to utilize additional protection of work options, such as CPU Cortisol or Resource Groups?

To help in answering these questions, let us review some key general, DB2-specific, and data warehouse-specific recommendations for defining service classes. These come from the various WLM resources mentioned, plus our own experiences.

General recommendations

- Keep the number of overall active service class periods to the minimum required. For a given z/OS image, preferably less than 30.
- Work that has similar goals, resource requirements, and business importance can share the same service class.
- Define goals based on peak periods of time for that given work.
- Service classes should only be defined for work when sufficient demand exists for them in the system. You can measure the demand either by the service consumption or by the amount of ending transactions during high utilization periods of this work. It probably does not make sense to define service classes with a demand of less than 1% of measured service units.

Archived

- Define goals that are not only in line with business requirements, but also within the current machine and subsystem capabilities.
- DB2 server address spaces, xxxxMSTR, xxxxDBM1 and xxxxDIST, should be defined to a service class with a high velocity goal. If multiple DB2 subsystems are defined, consider differentiating them into separate service classes. The xxxxIRLM address spaces should be defined to the SYSSTC service class.
- The DB2 stored procedure address spaces should be defined to service classes that reflect the requirements of the stored procedure, in comparison to other application work.
- When appropriate, response time goals are more desirable than velocity. For the performance administrators and users, it is a more tangible goal and RMF provides more tangible statistics, such as response time distributions. Secondly, response time goals require less maintenance with hardware or software migrations.
- To consider using a response time goal, it is generally suggested that the workload have at least 10 completions within a 20-minute period of time.
- Avoid using goals to prioritize work. Use the degree of importance to indicate the importance of the work, and set the goal to what is actually needed. Example: Do not use Velocity 50 versus Velocity 30 to differentiate priority between different service classes or service class periods.
- Do not expect work of higher importance to always get a higher dispatching priority than lesser import work. Dispatch priorities depend on the goal and characteristics of the work. It is common to see low-velocity goal work with a dispatching priority higher than high-velocity work due to differences in the work. If you really need to manage dispatching priorities, use the CPU Critical function (discussed later).

Data warehouse workloads

- For DB2 query (data warehouse and business intelligence) workloads, where the individual query resource requirements and business importance vary, consider the following:
  - Multiple service classes to differentiate users, applications, and application packages
  - Multiple service class periods to differentiate small consumers from large consumers, and to yield consistent response times for the smaller consumers
  - Percentile response time goals for early periods that have frequent completions of smaller consumption work, such as Operational BI, metadata requests, or short reports
  - Velocity goals for later periods that have less-frequent completions, and larger, varying resource consumption characteristics
  - Discretionary goal as a last period for planned or unplanned (exceeds some reasonably anticipated amount of service) very large consumption work
- For query workloads, where a few long-running queries can significantly dilute an average, it is strongly suggested to go with a percentile response time goal.
- For the data warehouse DB2 subsystem, ensure that the DB2 zparm CMTSTAT is set to INACTIVE.
- For each of the other key data warehouse processing activities, such as refresh, scheduled reporting, warehouse maintenance, and so on, it is suggested to isolate higher importance from lower importance work into separate service classes. Example: Post a data warehouse refresh; there are key reports that a company's knowledge workers or other applications are awaiting to make timely tactical business decisions. There is likely another set of reports that are more in the "nice to have" category, which are less-frequently accessed. Having all the reports serviced under one service class could lead to delays for the critical ones.

Optimal WLM management of DB2 DDF CP parallelism.

For DB2 V9, if exploiting CP parallelism for DDF workloads, we recommend applying DB2 APAR PM06953 - Single Enclave support for CP parallelism. This APAR exploits WLM work-dependant enclaves, which provide single enclave resource management of all tasks associated with a parallel query. Without this APAR, on DB2 V8 and above, all parallel tasks of a DDF CP parallelized query are managed as individual enclaves. When managed as individual enclaves, parallel queries move through WLM service periods in a suboptimal and inconsistent manner.

Service classes for the data warehouse workloads

Using the table we created earlier, and the considerations just reviewed, we decided to create the following DW-related service classes:

DWDB2SYS	Data warehouse DB2 system address spaces. Single period, Velocity goal.
DWDDFHI	DDF query for high importance users and applications. Multiperiod mix of percentile response time and velocity goals. Providing higher priority, with more consistent response times, for shorter consumption work.
DWDDFMD	DDF query for medium importance or novice users, applications. Multiperiod mix of percentile response time, velocity and discretionary goals. Providing higher priority for more consistent response times, for shorter consumption work.
DDFDFLT	Place holder for unexpected work that comes in through the DDF subsystem.
DWREFHI	Data warehouse batch-style refresh high-importance
DWREFLO	Data warehouse batch-style refresh lower importance
DWSCHDHI	High importance scheduled reports; example: After a batch refresh
DWSCHDLO	Lower importance scheduled reports; example: Less urgent reports, seldom used, but still run.
DWBATHI	High importance batch maintenance activities
DWBATLO	Lower importance batch maintenance activities

At this point we have determined what our service classes and relative importance would be, but we still need to answer these key questions:

- How do you set velocity goals for the data warehouse service classes determined to be velocity oriented?
- For our data warehouse query DDF service classes (DWDDFHI and DWDDFST):
  - How do we determine the settings of period durations?
  - What do we set the individual service class period response time and/or velocity goals to?

In the next section we discuss some considerations, what we did for our experiments, and finally what we provided as part of our sample service definition.

11.2.4 Determining data warehouse service class goals and period durations

As preparation and working through this step, we highly recommend reading 5.2, “Setting Goals” in *System Programmer’s Guide to: Workload Manager*, SG24-6472.

Setting goals for Velocity goal data warehouse service classes

The general WLM guidance is to utilize historical data and conduct analysis of the achieved execution Velocities during the peak periods for the given service class. Utilize those achieved Velocities as input to resetting a more appropriate Velocity goal. Of course, if no historical data exists, then start with values from something similar, perhaps from your transactional WLM service definition. Otherwise, set a value to start with and then hone from there. Hone using the velocity actually obtained, during peak periods, with which you are achieving satisfactory performance.

For our project experiments we utilized values from our transactional service definition (DB2 @spaces) or simply went with Velocity=50 to start with. In both cases we ended up honing these values once our actual workloads were running. See our final sample service definition policy in Chapter 12, “DB2 data sharing workload balancing and fault tolerant configuration” on page 291.

Setting goals and durations for the multiperiod data warehouse DDF service classes

Based on analysis of several customer data warehouse query workloads, we suggest considering 3-4 periods for the highest importance DDF query service classes, and perhaps 2-3 periods for the medium to lower importance DDF query service classes. In both cases consider the initial periods to be percentile response time goals, and later periods to be Velocity and perhaps the last period of the medium to lower importance service class to be discretionary.

In determining the response time goals, consider the first period to be one or a few seconds to accommodate the queries that require immediate responsiveness, such as metadata, Operational BI, or trivial queries. In considering additional response time periods, consider whether there are user or service level requirements for a percentage of queries or reports to meet some response time objective. The other, more important, consideration, is what is obtainable given the computing resources and application or data base tuning. The best way to understand the latter and then further hone potential response time goals, is again via collecting and analysis of historical data.

Regarding durations, for the higher importance data warehouse DDF query service class, we recommend defining the periods in a manner that the durations (and associated response time goals) are defined such that 70-80% of transactions (queries) complete in the first period. Define the last period with duration and goal such that ~1% of the transactions complete, and define the middle periods to capture the rest in a similar trend (the higher the period, the lower the # of completions). For the medium to lower importance data warehouse DDF query service class, we recommend a similar approach: yet again, the first period should only be for very short consumption work.

In setting velocity goals for the later periods, we suggest starting with velocities of 30 or less, and then hone using historical data, as discussed previously. Data warehouse query workloads have the potential to put high demands on processing resources, hence the suggestion to start with 30 or less.

In Appendix C, “Using DB2 accounting data for setting WLM period durations” on page 421 we provide a technique, utilizing DB2 accounting data, to help determine reasonable period durations as well as potential response time goals for the early periods. It is the method we

utilized to initially set the DDF query service classes for our experiments and also what was utilized to conduct similar analysis of customer data, leading to our sample service definitions.

11.2.5 Sample data warehouse query service classes

Based on our experiences and analysis of several customer z/OS DB2 data warehouse profiles, we derived the values shown in Figure 11-1.

Description	Service Class	Period	Imp	Duration (SU)	Type	Goal
High Importance OLAP DDF Queries	DWDDFHI	1	2	25,000	Response Time	90% < 3 sec
		2	2	200,000	Response Time	80% < 15 sec
		3	3	1,000,000	Velocity	30
		4	4		Velocity	20
Medium to Low Importance OLAP DDF Queries	DWDDFMD	1	3	25,000	Response Time	90% < 3 sec
		2	4		Velocity	5
		3	5		Discretionary	
High Importance Scheduled Reports	DWSCHDHI	1	2		Velocity	40
Medium Importance Scheduled Reports	DWSCHDMD	1	4		Velocity	10

Figure 11-1 DDF service classes

DB2 DDF work requests are managed via WLM as independent enclaves. Only processor service units are accounted to an independent enclave. The I/O and MSO service units are accounted to the DB2 server address space. This is another reason why we were able to utilize DB2 Class 1 accounting processor time to help set period durations.

The scheduled reports could also arrive via the WLM DDF subsystem (not depicted in Figure 11-1).

For our total sample service definition, see Appendix D, “Sample WLM service definition” on page 433.

11.2.6 Classification

Recall that WLM uses classification rules to map incoming work requests to a specific service class and report class. This classification is based on work qualifiers. A work qualifier identifies a work request to the system. The first qualifier is essentially predetermined,

because it is based on which WLM subsystem receives the work request. For example, a DB2 DRDA work request will automatically get qualified to the WLM DDF subsystem. A JES2 batch job will automatically get qualified to the WLM JES2 subsystem. Each subsystem has a predefined set of attributes, which work requests can be further qualified, for classification into a service class and report class. For details on attributes available for all the WLM subsystems, reference the *System Programmer's Guide to: Workload Manager*, SG24-6472.

Without classifying work into appropriate service classes, all the other WLM definitions are meaningless. From our experience, classification is often an opportunity for refinement, especially with regard to query workloads. This became more relevant with the recent advancements in utilizing DB2 client information for more granular classification of such workloads. Additionally, other performance monitors, and accounting and capacity planning tools are often based on the resulting WLM resource usage statistics.

We recommend gaining an understanding of the various qualifiers available for classification of the incoming work. Work with the DB2 DBAs, the BI administrator and potentially the applications staff to determine the appropriate qualifiers for differentiation, and how to implement them.

Today, a common approach to differentiate DDF requests within a DRDA application is to utilize multiple data source definitions in the application server, each data source having a different AUTHID associated with it. Then programmatically use one of the specific data sources for each of the groups being differentiated. Maintenance of this is suboptimal.

With the recent availability of DB2's WLM Set Client information stored procedure there are additional, perhaps more useful options, using DB2 client strings. The *client strings* are text attributes associated with the connection, which we can use for classification of the workload. The client strings can be set as a part of the data source definition or alternatively you can set them programmatically within the application so that every transaction can have a different value. We can use the client strings for workload classification, and we can also use them for end-to-end auditing (the fields are written in DB2 accounting reports). The recommended client strings are client user ID (attribute SPM in Table 11-3), which is different from an MVS TSO user ID, and client transaction name (attribute PC in the table).

Table 11-3 has the complete list of the standard DDF qualifiers available. Note the three associated with the DB2 client information (AI, PC, SPM), which we have highlighted in blue. IBM Cognos Administration makes it relatively easy to exploit the use of these DB2 client strings. We implemented this for our experiments. This implementation and more information regarding WLM set client information stored procedure is described in detail in Chapter 13, "Utilizing DB2 client information for resource management and monitoring" on page 321.

Table 11-3 WLM DDF qualifier (classification) filters available

Qualifier	Type	Description
Accounting Information	AI	Set by client info API: Client accounting string
Correlation Information	CI	Driver program name by default but application can set
Collection Name	CN	Collection name of the first SQL package accessed by the DRDA requester in the unit of work
Connection Type	CT	Always DIST for DDF server threads
Package Name	PK	Name of the first DB2 package accessed
Plan Name	PN	Always DISTSERV for DDF server threads accessed via DRDA requesters
Procedure Name	PR	Name of the procedure called if first request in unit of work

Qualifier	Type	Description
Process Name	PC	Set by client info API: Client transaction name
Subsystem Collection Name	SSC	Usually the DB2 data sharing group name
Subsystem Instance	SI	DB2 server's MVS subsystem name
Sysplex Name	PX	Name assigned to sysplex at IPL
User ID	UI	DDF server thread's primary AUTHID
Subsystem Parameter	SPM	Set by client info API: Assigned the concatenation of client user ID or workstation name

Figure 11-2 shows sample DDF classification rules exploiting DB2 client strings. This is not what we utilized for our experiments, but serves as a sample.

```
* Subsystem Type DDF - Rules for DDF (Transaction only)

  Classification:

    Default service class is DDFDFLT
    Default report class is RDDFDFLT

      Qualifier  Qualifier      Starting       Service  Report
    # type       name           position       Class    Class
    - ---------- -------------- ---------      -------- --------
    1 SI         DWPR                          DWDDFMD  RDWPDFLT
    2 . UI       . COGNOS1                     DWDDFMD  RCOGPDFL
    3 . . PC     . . PH3DW            25       DWDDFMD  RCBKPKG
    4 . . . SPM  . . . userK*                  DWDDFHI  RCBKCRIT
    4 . . . SPM  . . . userS*                  DWDDFHI  RCBKSERV
    4 . . . SPM  . . . userM*                  DWDDFMD  RCBKMED
    4 . . . SPM  . . . SCHEDHI                 DWSCHDHI RCSCHDHI
    4 . . . SPM  . . . SCHEDMD                 DWSCHDMD RCSCHDMD
    2 . UI       . SQWREF                      DWDDFREF RSQWREF
    1 SI         DWTS                          DWDDFMD  RDWTSTDF
```

Figure 11-2 Sample WLM classification rules for DDF

Some notes on Figure 11-2:

- This set of rules uses up to four levels of classification for a given work request. WLM classifies work, using these rules, in a top-down fashion. If a work request matches the level 1 qualifier, then the level 2 qualifier is checked, and so on. When reaching a level that does not match, the work request is classified to the last-matching service class and report class. Within a given level, WLM also searches top-down.
- Level 1, SI (subsystem instance) and level 2, UI (user ID, the DRDA connection user ID, which is also known as the DB2 primary AUTHID) are common DDF qualifiers currently utilized in IT shops. `DWPR` represents a production DW DB2. `Cognos1` is the z/OS user ID utilized to connect IBM Cognos BI to DB2.
- At level 3, we utilize PC (client transaction name) to capture and qualify on the IBM Cognos BI package name. An IBM Cognos BI package represents a data model for a given set of business users, for example Sales or Marketing for a particular data mart. There are other potential IBM Cognos BI attributes available that could be passed in. Utilizing the IBM Cognos BI package is one potential implementation. For our single LPAR experiments, we chose to pass in the IBM Cognos BI report name as the client transaction name. This allowed for quicker validation of the mix of IBM Cognos BI reports that were run for each experiment.

- At level 4 we utilized the SPM (concatenation of client user and workstation name) to differentiate our IBM Cognos BI users as follows:
  - userK* - All users having IDs starting with “userK” are recognized as the company’s key knowledge workers. They are classified to service class DWDDFHI, our high importance DDF query service class. Additionally, they are classified to report class RCBKCRIT.
  - userS* - All users having IDs starting with “userS” are recognized as the company’s service center representatives. They are also classified to service class DWDDFHI, yet to report class RCBKSERV.
  - userM* - Intermediate to Novice knowledge workers, classified to the DWDDFMD service class and the RCBKMED report class.
  - SCHEDHI & SCHEDMD - These represent potential “proxy” IBM Cognos BI user IDs, utilized for submission of some IBM Cognos BI batch scheduler work. Potential use here would be for perhaps regularly scheduled refreshes.

While we did explicitly test the use of *WLM set client information* stored procedure to exploit the SPM qualifier (for IBM Cognos BI client user IDs), we did not specifically test the use of a “proxy” user ID for the IBM Cognos BI batch scheduler. For a sample view of the actual values of some of these qualifiers for an IBM Cognos BI DDF query, see Figure 11-4 on page 282.

For an alternative set of classification rules, utilizing DB2 set client information, see Figure 15-2 on page 352.

11.2.7 Report classes

Report classes allow the definition of sufficient granularity to obtain more detailed information for analysis.

Some common usage scenarios are for monitoring and reporting by application, package, department, or users.

We advise you to use report classes extensively. They are the base for a deeper understanding of the work running in service classes. One of the greatest strengths of WLM is its ability to monitor the work running in the system, and report classes are the way.

We advise collaboration with the DBA to determine if there is any benefit in aligning any of the DB2-related WLM report classes with common ways the DBA conducts performance reporting or monitoring. This could be valuable for cross-validation of performance analysis and/or capacity planning exercises.

Report classes can be utilized as a first step, prior to defining a set of work to be managed via a service class. You can first characterize resource consumption of a set of work. Analysis of the report class could validate the need and be utilized as input in defining the new service class.

Report classes can be useful for conducting internal testing, benchmarking, or proofs of concept. For our experiments, we created a separate report class for each of our IBM Cognos BI reports. This allowed us to cross-check performance statistics between DB2 accounting and z/OS RMF.

Report classes are mutually exclusive. A particular piece of work will only get classified to one report class.

11.2.8 Protecting work

WLM offers several features to protect work from being hurt by other work's behavior. We briefly discuss two of the more commonly utilized functions, CPU Critical and Resource Groups. For more in-depth discussion, once again, reference *System Programmer's Guide to: Workload Manager*, SG24-6472.

CPU Critical

Here we discuss the feature available in WLM to guarantee the dispatching priority of selected service classes. When you assign a service class with CPU Critical = YES, you ensure that less important work will generally have a lower dispatch priority. This protection can be valuable for work that is CPU-sensitive.

Generally it is possible that work in service classes with lower importance gets higher dispatch priority than work with a higher importance. Why? The processor adjustment algorithm attempts to optimize processor usage by:

- Service consumption of the work
- Importance of the work
- Goal definition of the work

As a result, a service class with low importance, low goal definitions, and low service consumption can get the highest available dispatch priority in the system. This is not a problem as long as the work behaves well. But it can hurt work of higher importance when the work starts to use large amounts of processor in an unpredictable manner.

The purpose of the CPU Critical feature is to eliminate this problem, though it is suggested to not overuse this feature. The more use of CPU Critical, the less is left for WLM to efficiently balance the available processor resources across the service classes in order to meet goals. In other words, WLM not only tries to meet the goals of the highest importance work, but also of all the work in the system.

In one of our mixed workload experiments, we exhibit the use of CPU Critical to further protect our highest importance BookStore transactional workload.

Resource groups

Resource Groups can be used to define a minimum and maximum amount of processor service for particular work. Work is assigned to a Resource Group by specifying the Resource Group in the service class definition. A Resource Group can include multiple service classes, but a service class can only be assigned to one Resource Group.

The maximum capacity of a Resource Group is enforced. WLM takes various actions to stop work in a Resource Group from exceeding the capacity, including swapping, changing dispatching priorities, and capping the processor consumption.

The minimum capacity is only used when work in the Resource Group is not meeting its goals, in which case WLM attempts to provide the minimum service defined in the Resource Group, *even if that causes more important work to miss its goals*.

The general recommendation is to try and avoid using Resource Groups. Try to set appropriate goals and allow WLM to manage the resources. Resource Groups restrict the ability of WLM to make changes to meet the goals you have defined. However, they are necessary in some circumstances. Your workload might have some characteristics that mean that normal WLM management does not work well. In that case, Resource Groups can be useful to ensure that work receives the service it requires.

There are three types of Resource Groups you can define:

Type 1 Capacity is specified in unweighted processor service units per second (SU/s). The maximum and minimum limits have *sysplex* scope.

Type 2 Capacity is specified as a percentage of logical partition (LPAR) capacity from 0-99. The sum of all Resource Group minimums should not exceed 99. The maximum and minimum limits have *system* scope.

Type 3 Capacity is specified as a percentage of a single general purpose CP. 100 represents the capacity of one processor. The number can range from 0 to 999999, but the sum of all Resource Group minimums should not exceed the number of processors x 100. The maximum and minimum limits have *system* scope.

Considerations relative to data warehouse workloads

In recent years, there appears to be a trend toward utilizing Resource Groups for limiting resource consumption of data warehouse workloads, in a mixed transactional or data warehouse environment. This could lead to inefficient use of resources, for example starving queries, while processing cycles are available. We suggest using Resource Groups only if a combination of WLM goals and CPU Critical does not accomplish the required protection. The other potential usage is in the case when very specific control of resources is required for perhaps chargeback purposes.

The Resource Group capacity settings are only applicable for general purpose processors. zIIP and zAAP specialty engines are not included. A potential downside of using Resource Group maximums, for zIIP-eligible work, is that it also limits the use of the zIIP processors.

We did not make use of Resource Groups in our sample service definition.

11.2.9 Multiple policies

Your business needs might require different performance goals at different times. In determining the need for multiple policies, here are some sample questions you might consider:

- In your data warehouse environment are queries, refreshes, batch reporting, database maintenance, and so on happening concurrently? Does the relative importance of these change at different times?
- In a mixed transactional or data warehouse environment, what workloads are running simultaneously? Does the relative importance of these change at different times?

Optimally, one policy could be utilized all the time, but often that is not feasible. Different policies can be defined to override the base policy. The following values can be overridden:

- Service class goals
- Service class processor protection
- Service class Resource Group assignment
- Resource Group attributes

Classification rules, application environments, or scheduling environments cannot be overridden with a different policy. These are consistent for the total WLM service definition. Also, service classes and report classes cannot be deleted.

For our sample service definition, we created the following three additional policies:

- CL_RFRSH - In this policy we override the goal for the DWDDFREF service class, which services one of our fictitious high importance daily refreshes as well as other intra-day refreshes. During the time our high importance daily refresh runs, we override the DWDDFREF service class, increasing the Velocity goal and Importance.
- CL_POWR - This policy override further differentiates the importance and goals between our power knowledge workers (DWDDFHI) and intermediate or novice knowledge workers (DWDDFMD).
- CL_WCRIT - This policy overrides the WASTCHI service class, which serves our high importance critical online BookStore Browse and Buy application. The overridden change sets CPU Critical to YES for the WASTCHI service class. This was primarily used for our experiments, but left in the sample.

From examining some customer WLM service definitions, for mixed transactional/data warehouse workloads, we've noticed some use of multiple policies to dynamically change Resource Group max attributes at different points in the day. This perhaps is the safest way of protecting some workloads (ex. High Importance transactional), though the downside is it limits WLM's ability to efficiently utilize all resources all the time. As mentioned previously, we'd suggest exploring traditional goals and functions such as CPU Critical prior to implementing resource groups in this manner.

11.2.10 Service class relative importance

Once all service class and service class period definitions are determined, it is valuable to place them in a table, sorted by WLM Importance. This allows you to easily view relative Importance and validate it against your business objectives. See Table 11-4 for our set of sample service classes for our mixed transactional and data warehouse environment.

Table 11-4 Sample set of key transactional and data warehouse service classes - base policy

Service Class	Workload	Contents	Per	Dur	Imp
SYSTEM	SYSTEM	Key system address spaces	N/A	N/A	Above All
SYSSTC	SYSTEM	Key system STC address spaces, WebSphere, IRLM, RMF, TCPIP, RRS, etc.	N/A	N/A	Above the rest
OPDB2SYS	OPERSTC	OLTP DB2 system address spaces			1
DWDB2SYS	DWSTC	DW DB2 system address spaces			1
STCOHIGH	OPERATNL	OLTP WebSphere server address spaces	1		2
WASTCHI	OPERATNL	High Imp OLTP transactions	1		2
DWSTCHI	DWSTC	Near-real-time refresh of the dynamic DW tables (InfoSphere Change Data Capture)	1		2
DWDDFHI	DWQUERY	High Imp; example: metadata access, operational BI, trivial reports	1	25K	2
DWDDFHI	DWQUERY	High Imp - Small consumption queries	2	200K	2
DWDDFHI	DWQUERY	High Imp - Medium to large consumption queries	3	1M	3
WASTCLOW	OPERATNL	Medium Imp - OLTP transactions	1		3
DWDDFMD	DWQUERY	Medium Imp - Trivial consumption queries	1	25K	3
DWSCHDHI	DWSCHED	High Imp - Scheduled reports	1		3

Service Class	Workload	Contents	Per	Dur	Imp
DWDDFHI	DWQUERY	Large + consumption queries	4		4
DWDDFMD	DWQUERY	Medium Imp - Small consumption queries	2	500K	4
DWDDFREF	DWREFRSH	Batch refresh	1		4
DWSCHDMD	DWSCHED	Medium to Low Imp - Scheduled reports	1		5
DWDDFMD	DWQUERY	Low Imp - Large + consumption queries	5	N/A	Disc

The table represents service class relative importance, for a mixed transactional and data warehouse environment. Eliminating the transactional (OLTP) related service classes could also represent a standalone data warehouse workload environment.

11.3 Analysis and verification of a working WLM policy

When a workload or application is performing below expectations, the collective performance personnel have to determine if there is a need for application or database performance tuning or for more resource capacity or perhaps WLM policy tuning. With a quality service definition in place, the performance administrator should be able to quickly determine whether there is any opportunity with regard to the latter.

IBM (and other companies) offers a variety of products and tools designed to help clients monitor and analyze the behavior of the system workload. The primary tool we utilized was the IBM Resource Management Facility (RMF). Here is a brief description of each of the RMF tools and reports we utilized, followed by some examples of usage.

Resource Management Facility (RMF) Workload Activity report

This RMF post-processor report is the primary means for analyzing z/OS workload performance after the fact. This was our primary means of analysis of our experiments.

Resource Management Facility (RMF) Monitor III

There are many RMF Monitor III reports that are useful for interactively monitoring workload performance. We specifically utilized it to display classification data available for the active WLM enclaves associated with our IBM Cognos BI reports.

Resource Management Facility (RMF) Spreadsheet Reporter

The Resource Management Facility (RMF) Spreadsheet Reporter is an extension of the RMF Postprocessor and runs on your Windows workstation. It gives you the capability to extract reports from the RMF Postprocessor output and convert them into common spreadsheet formats. You can find it online at:

http://www-03.ibm.com/systems/z/os/zos/features/rmf/tools/rmftools.html

11.3.1 Resource Management Facility (RMF) Workload Activity report

For details on generating an RMF Workload Activity report, see *Resource Management Facility Users Guide* at:

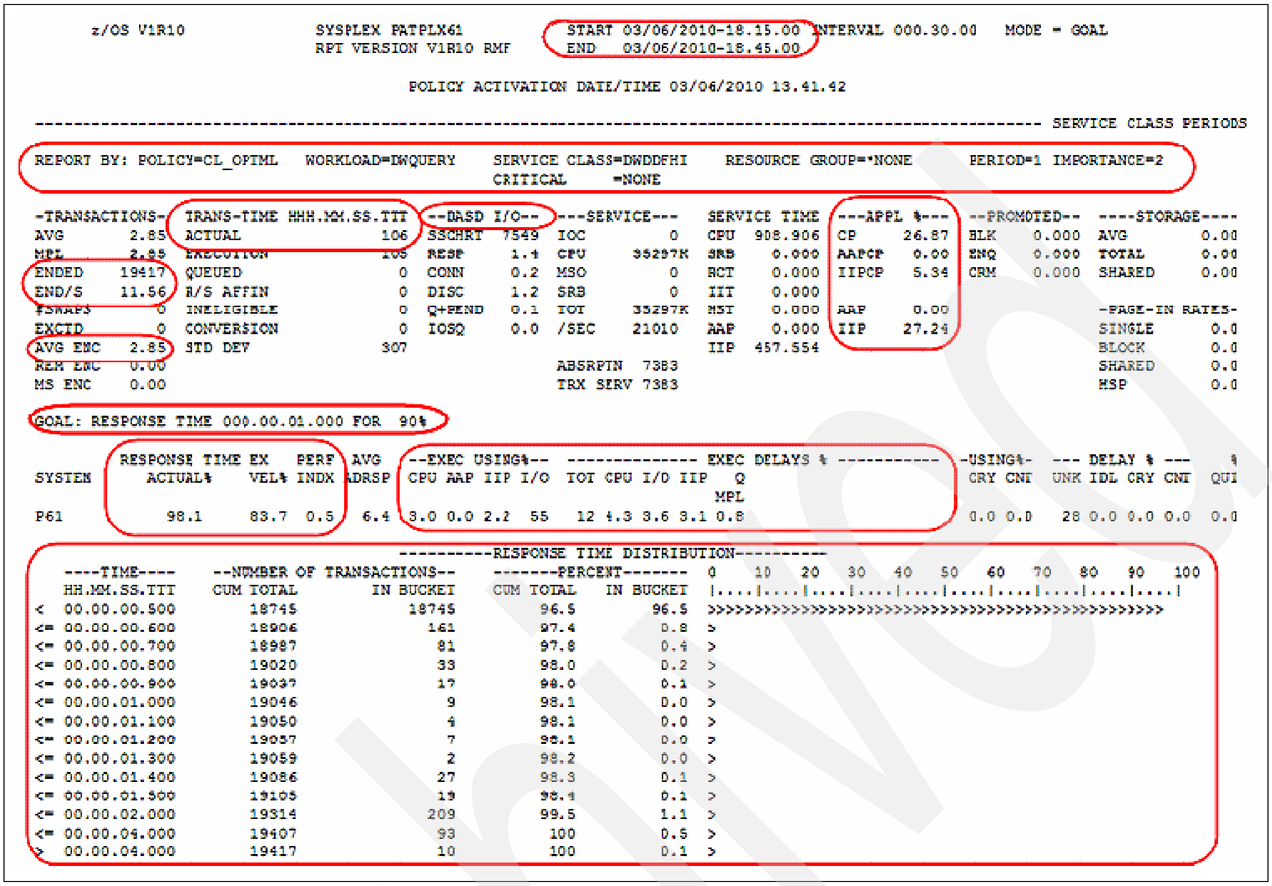
http://publibz.boulder.ibm.com/epubs/pdf/erbzug90.pdf

Figure 11-3 on page 281 is an excerpt of an RMF Workload Activity report, from one of our experiments. It captures statistics for period 1 of service class DWDDFHI. We have highlighted in red some of the key fields of interest, and here are some brief descriptions:

- START & END: Start and end time of the reported interval.
- REPORT BY: The POLICY and WORKLOAD this service class period belongs to. Additionally, the relative IMPORTANCE, RESOURCE GROUP (if applicable), and whether or not this service class is designated as CPU CRITICAL.
- ENDED: This value is the number of transactions ended within the interval.
- END/S: Computed throughput (transactions/second) for the interval.
- AVG ENC: The average # of independent enclaves during the interval.
- ACTUAL: The average response time for transactions ended.
- DASD I/O: I/O characteristics for this service class period, during the interval.
- APPL%: Percentage of processor time used by transactions running on the different processor types. CP - General purpose, AAP - zAAPs, IIP - zIIPs, AAPCP - zAAP eligible running on CP, IIPCP - zIIP eligible running on the CP.
  Note: APPL% shows processor utilization based on a uniprocessor capacity. This means the values can exceed 100%.
- GOAL: The specified WLM goal for this service class period.
- RESPONSE TIME ACTUAL%: The achieved goal for this service class period, during the interval.
- EX VEL%: The achieved execution velocity for this service class period, during the interval. The achieved Velocity is reported, regardless of whether a Velocity goal was set.
- PERF INDX: The performance index for the period represents how close a period came to reaching the goal. The PI is 1.0 if goal is reached. If PI < 1, goal is exceeded. If PI > 1, then goal is missed.
- EXEC USING% - EXEC DELAYS%: Of the WLM samples collected for this period, the percentages of those samples found in each of the categories listed. When a service class period is not meeting its goal, this could be useful in determining what resources it is being delayed on.
- RESPONSE TIME DISTRIBUTION: WLM maintains counts of how many transactions were completed within a particular time. This is useful for obtaining more granular response time characterization of the work completing within this service class period.

See Figure 11-3 on page 281.

Archived



```
      z/OS V1R10              SYSPLEX PATPLX61          START 03/06/2010-18.15.00 INTERVAL 000.30.00    MODE = GOAL
                              RPT VERSION V1R10 RMF     END   03/06/2010-18.45.00

                                     POLICY ACTIVATION DATE/TIME 03/06/2010 13.41.42

 ------------------------------------------------------------------------------------------------ SERVICE CLASS PERIODS

 REPORT BY: POLICY=CL_OPTML    WORKLOAD=DWQUERY     SERVICE CLASS=DWDDFHI     RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=2
                                                    CRITICAL      =NONE

 -TRANSACTIONS- TRANS-TIME HHH.MM.SS.TTT  --DASD I/O-- ---SERVICE---    SERVICE TIME  ---APPL %---  --PROMOTED--  ----STORAGE----
 AVG      2.85  ACTUAL                106 SSCHRT  7549 IOC          0   CPU  908.906  CP     26.87  BLK   0.000   AVG         0.00
 MPL      2.85  EXECUTION             105 RESP     1.4 CPU     35297K   SRB    0.000  AAPCP   0.00  ENQ   0.000   TOTAL       0.00
 ENDED   19417  QUEUED                  0 CONN     0.2 MSO          0   RCT    0.000  IIPCP   5.34  CRM   0.000   SHARED      0.00
 END/S   11.56  R/S AFFIN               0 DISC     1.2 SRB          0   IIT    0.000
 #SWAPS      0  INELIGIBLE              0 Q+PEND   0.1 TOT     35297K   HST    0.000  AAP     0.00                -PAGE-IN RATES-
 EXCTD       0  CONVERSION              0 IOSQ     0.0 /SEC     21010   AAP    0.000  IIP    27.24                SINGLE       0.0
 AVG ENC  2.85  STD DEV               307                               IIP  457.554                              BLOCK        0.0
 REM ENC  0.00                                         ABSRPTN   7383                                             SHARED       0.0
 MS ENC   0.00                                         TRX SERV  7383                                             HSP          0.0

 GOAL: RESPONSE TIME 000.00.01.000 FOR  90%

           RESPONSE TIME EX    PERF  AVG   --EXEC USING%-- -------------- EXEC DELAYS % ----------- -USING%-  --- DELAY % ---    %
 SYSTEM     ACTUAL%     VEL%  INDX  ADRSP  CPU AAP IIP I/O  TOT CPU I/O IIP   Q                      CRY CNT  UNK IDL CRY CNT  QUI
                                                                             MPL

 P61          98.1      83.7   0.5    6.4  3.0 0.0 2.2  55   12 4.3 3.6 3.1 0.8                      0.0 0.0   28 0.0 0.0 0.0  0.0

                                        ----------RESPONSE TIME DISTRIBUTION----------
     ----TIME----     --NUMBER OF TRANSACTIONS--     -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
     HH.MM.SS.TTT     CUM TOTAL        IN BUCKET     CUM TOTAL   IN BUCKET  |....|....|....|....|....|....|....|....|....|....|
 <   00.00.00.500         18745            18745          96.5        96.5  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 <=  00.00.00.600         18906              161          97.4         0.8  >
 <=  00.00.00.700         18987               81          97.8         0.4  >
 <=  00.00.00.800         19020               33          98.0         0.2  >
 <=  00.00.00.900         19037               17          98.0         0.1  >
 <=  00.00.01.000         19046                9          98.1         0.0  >
 <=  00.00.01.100         19050                4          98.1         0.0  >
 <=  00.00.01.200         19057                7          98.1         0.0  >
 <=  00.00.01.300         19059                2          98.2         0.0  >
 <=  00.00.01.400         19086               27          98.3         0.1  >
 <=  00.00.01.500         19105               19          98.4         0.1  >
 <=  00.00.02.000         19314              209          99.5         1.1  >
 <=  00.00.04.000         19407               93           100         0.5  >
 >   00.00.04.000         19417               10           100         0.1  >
```

Figure 11-3   RMF Workload Activity for service class DWDDFHI

For a complete description of the fields in RMF reports, see *z/OS V1R11 Resource Measurement Facility (RMF) Report Analysis* at:

http://publibz.boulder.ibm.com/epubs/pdf/erbzra90.pdf

11.3.2 Resource Management Facility (RMF) Monitor III

We utilized RMF Monitor III to better understand the qualifiers (classification filters) associated with our IBM Cognos BI queries. Additionally, this allowed us to functionally validate the use of WLM set client information stored procedure for setting some of those qualifiers.

Figure 11-4 on page 282 is an RMF Monitor III Classification Data view of a WLM enclave associated with one of our IBM Cognos BI reports. To obtain this view, the RMF Monitor III data gatherer must first be started on the z/OS system. From a TSO ISPF session, here are the necessary steps:

1. Select the RMF option
2. Select option 3 - Monitor III
3. Select option 1 - Overview
4. Select option 2 - SYSINFO
5. Move the cursor down to the * ENCLAVE entry and press Enter.
6. Use PF10 or PF11 to move to the RMF Monitor III interval of interest. Each parallel DDF query has two enclaves associated with it (for example, ENC0002 and ENC0001). One

Archived

enclave represents the parent and one represents the associated child SRBs. Selecting either of the two related enclaves yields the same classification information. See Figure 11-4.

```
                RMF Enclave Classification Data

The following details are available for enclave ENC00002
Press Enter to return to the Report panel.

                                                          More:     +
  Subsystem Type: DDF    Owner: DBDMDIST   System: P61
  Accounting Information . . :
   SQL09054Linux/390         BIBusTKServerMain   cognos1  Report
   8

  Collection Name  . . . . . : NULLID
  Connection Type  . . . . . : SERVER
  Correlation Information  . : BIBusTKServe
  LU Name  . . . . . :
  Netid  . . . . . :
  Package Name . . . . . : SYSSH100
  Plan Name  . . . . . : DISTSERV
  Procedure Name . . . . . :
  Process Name . . . . . : /content/package @name='PH3DW' /
  Transaction Class/Job Class:
  Transaction Name/Job Name  :
  Userid . . . . . : COGNOS1
  Scheduling Environment . . :
  Subsystem Collection Name  :
  Subsystem Instance . . . . : DBDM
  Subsystem Parameter  . . . :
   userK43     9.12.43.13
```

Figure 11-4 RMF Monitor III classification data for a DDF query initiated by IBM Cognos BI

11.3.3 Resource Management Facility (RMF) spreadsheet reporter

We utilized the RMF spreadsheet reporter Workload Activity Trend report to plot key service class period statistics over time. See Figure 11-5 on page 283 for a sample plot, depicting the Performance Index (PI) and the Ended Transactions per sec (throughput) for our WASTCHI service class in one of our WLM mixed workload experiments.

Archived

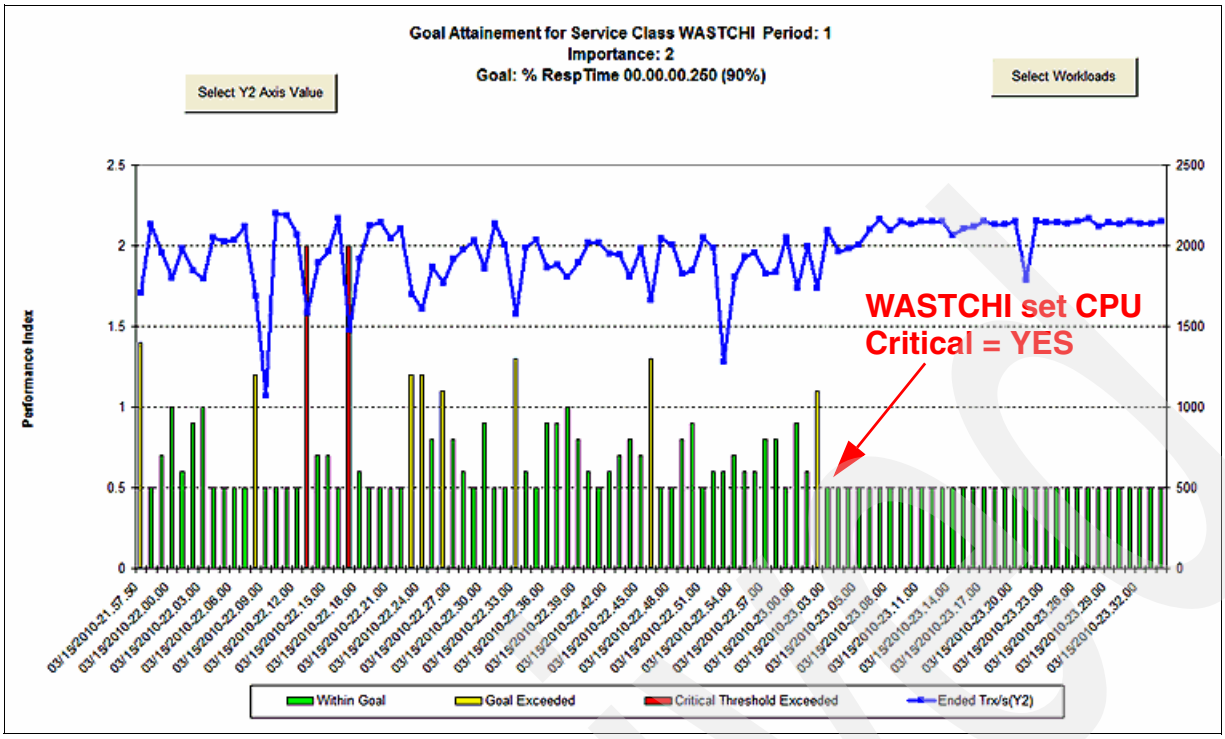


Figure 11-5 WASTCHI service class with and without CPU Critical

Additionally, we utilized the RMF spreadsheet reporter Workload Overview Report to capture, in spreadsheet format, all the key WLM metrics for each of our experiments in Chapter 15, “Single z/OS LPAR topology experiments” on page 347. Depending on the experiment, we abstracted a subset of these key metrics into a summary table. Figure 11-6 is a sample of one of our service class results tables from that chapter.

Service Class												
Service Class Definition					Goal Results		KPIs		Processor Utilization			
Per	Imp	Duration (SUs)	Goal Type	Goal	Actual	PI	Trans per min	Avg Resp Time	CP	IIP	Total	IIPCP
1	2	10,000	Resp time	90%< 1s	98%	0.5	693	0.1	27	27	54	5
2	2	100,000	Resp time	80%<10s	95%	0.7	70	3.1	39	49	87	12
3	3	1,000,000	Velocity	40	56	0.7	11	28.1	49	71	120	17
4	4		Velocity	10	16	0.6	0.2	330.9	17	30	47	7
								Totals:	132	176	308	41

Figure 11-6 Results table using RMF spreadsheet reporter Workload Overview Report as input

Another key tool for WLM analysis is the WLM Service Definition Formatter. We found this very valuable to easily evaluate and understand a WLM service definition. This tool can be downloaded from the following IBM website:

http://www.ibm.com/servers/eserver/zseries/zos/wlm/tools/sdformatter.html

11.3.4 Recommendations for maintaining the WLM service definition

One main objective of building a WLM service definition is to specify how resources should be allocated when they are in short supply. When things work properly, and you have a mix of work in the system, the system should be able to run smoothly even at 100% busy for periods of time. The key is to know which work can be delayed when things get busy.

Here are some general WLM recommendations for tuning your service definitions:

- Try to ensure that even your lowest priority workloads get at least some service. When work receives no service at all, you start to see additional problems. For example, a job requiring processor service to respond to the cancel command. If a job receives no processor service at all, you will probably have problems cancelling the job. Too many low importance service classes can cause this problem. The classes at the end of the queue might never get service. The best solution in this case is to combine service classes so that there is a reasonable amount of work running in each service class. For example, if you have five low importance service classes, you might find that the fourth and fifth do not get any service. If you combine them all into one service class (or make them discretionary), they all share the service that was received by the first three service classes.
- Make sure that your goals are reasonable. If your goals are too high, WLM might waste time on high-importance workloads that already receive their needed resources with acceptable performance, instead of helping lower importance work that needs attention. Goals that you set too high can result in erratic performance, because WLM makes changes to try to meet the goals. Ideally, you want WLM to find a setting that meets all your goals and only changes things when the workload changes. This results in the most consistent and predictable performance.
- Remember that the system is finite. It can seem wrong to take resources from more important work and give them to less important work. However, that is often necessary to allow the less important work to perform adequately. As long as the more important work achieves its business objectives, it makes sense to do it. This is what WLM is designed to do.
- Consider fine-tuning service classes whose goals are too aggressive and unachievable, regardless of infinite resources available. If goals are consistently missed, yet users are content with the performance, the goal should be set more in line with what is being achieved. Service classes with velocity goals often fall into this category. Properly adjusting these goals will remove some of the false negatives (yellow or red flagged service classes), reported by WLM monitoring tools, such as Tivoli's Omegamon. Additionally, it will save WLM some processing cycles in trying to satisfy unreasonable goals.
- While WLM will do its best to align resources according to the service policy in place, WLM cannot create additional resources. If the service policy is properly aligned with the business objectives, and important work is not meeting its service objectives, then application tuning, database tuning, or a resource upgrade are required.
- Do not hesitate to change multiple things at once if many service classes are not meeting their goals. System programmers tend to be cautious, and they like to change one thing at a time and observe the results. However, the performance of all of the service classes in a system is interrelated, and if you miss many goals, you might not be able to detect a result from changing a single goal. You cannot tell whether the change was beneficial or not. It is often better to make relatively large adjustments until you get close to what you need. When the WLM policy is mostly working, you can make smaller adjustments to fine-tune the system.

- There is no single perfect policy. The WLM policy does not directly change the priorities of work. WLM uses the policy to measure how well the work is running and work out whether changes are necessary. In many cases, different policies can give much the same result. It often does not make sense to obsess over small details. Conversely, sometimes a small change can make a big difference. If you have a service class with period 1 duration of 100000, and a lot of work in the service class was completing in just over 100000 service units, a small increase in the duration might make a dramatic difference to the performance by allowing the work to finish in period 1.
- Consider that setting up a single processor system is more difficult than multiple processor systems. On a single processor system, the processor is 100% busy for short periods of time even when the average utilization is low, which can cause problems such as erratic response times. A single processor system is more sensitive to problems with the WLM policy. Refer to Chapter 12 of *System Programmer's Guide to: Workload Manager,* SG24-6472 for more information.

Additional considerations for data warehouse service definitions

- Ensure that work is being classified as expected. It is often discovered, especially for DDF requests, that work is not getting classified as expected. Check the workload activity report to see if unexpected work is getting charged to default service classes. Additionally, DB2 provides service class information in its Accounting data (field QWAWLME) as well as in the DB2 Activity Monitor.
- Verify the expected distribution of work across multiperiod service classes, serving the query workloads. If all work requests are consistently falling into a small subset of the multiple periods, then period durations should be reexamined.
- For preexisting data warehouse service definitions, reevaluate whether response time goals could replace some periods, which were previously defined as velocity goals. Over the past 10 years, the DB2 data warehouse environment has matured quite a bit. There have been new workloads, such as operational BI, more users, increased query arrival rates, and significant DB2 and z/OS performance enhancements. This has dramatically changed resource consumption characteristics, yet often WLM service definitions have not been reviewed.
- Be sure to reevaluate query service class period durations after a processor migration. The SRM constants, which are utilized to determine an LPAR's service unit or second ratio, are based more on transactional (OLTP) workloads. Due to the read-only nature of data warehouse query workloads, they typically achieve much better performance than the difference in processor SRM constants.

11.3.5 Process for review of WLM service definitions

The following paragraphs, from *50 TB Data Warehouse Benchmark on IBM System z*, SG24-7674, provide a guideline for a WLM policy review process that is represented by Figure 11-7 on page 286.

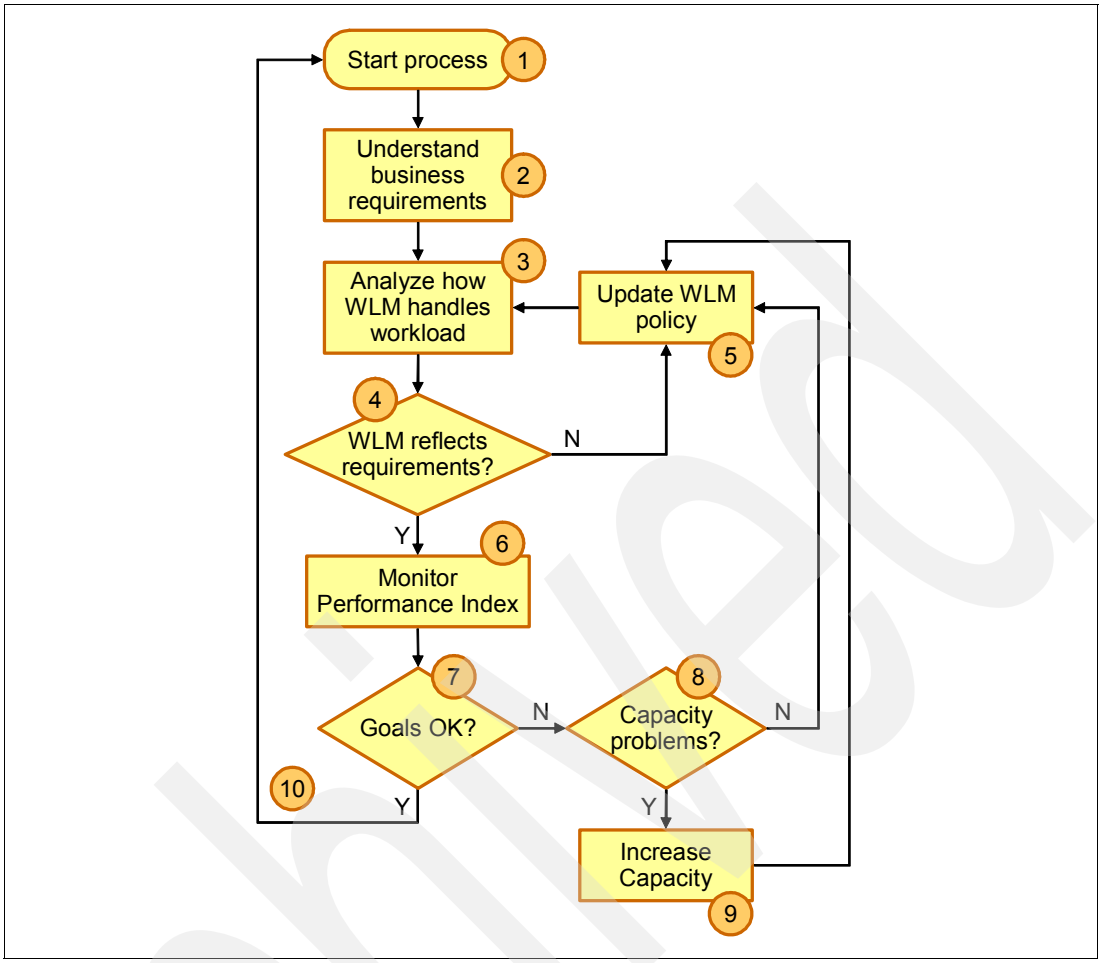


Figure 11-7 WLM Service Class definitions review

1. Start the process.
2. Understand business requirements. Requirements can be formally documented in SLAs or you may need to meet the people that hold the knowledge of the business requirements in terms of importance. Inevitably all applications are important for the business, but you must be able to classify workload or applications in order of importance. You may use a scale that reflects the WLM levels of importance: from 1 (highest) to 6 (discretionary).
3. Analyze how WLM handles workload. Get information about the available service classes, reporting classes, classification rules and other components of the WLM policy. Sometimes you may find that your installation is using 2 different policies for different moments of the business day, for instance day and night time. If your installation is a sysplex, consider that the WLM policy has sysplex scope: a single WLM is active for all the members of the sysplex at a given time.
4. Verify that the WLM classification rules are actually classifying the workload as expected by the business requirements. This step is a validation of the classification rules, given that the necessary service classes were defined.
5. If the WLM definitions do not match the business requirements, you need to update these definitions. A change needs to be validated again as described in the previous point.
6. If the WLM definitions are in line with the business requirements, you can move to the next stage: monitoring of the goal achievements through the service classes performance index.

7. Even if the WLM definitions reflect the business requirements, goals could be defined as too aggressive or too low. If goals are OK, and because you already verified at this stage that they are in line with the applications' expectations, you have a well-tuned system. You may plan for a recurrent review of this process, starting by reviewing whether the business requirements have evolved.
8. If the goals are not being achieved, you may be facing a capacity problem: WLM cannot create extra processor cycles and you may need to increase capacity to cope with the workload needs. If failing to achieve the defined goals is not related to a capacity problem, you need to review them and update the WLM policy accordingly.
9. Increased capacity should be followed by a WLM policy review, because a new hardware configuration, for instance an increase in processor capacity but with fewer processors, can introduce changes to how the resources can be distributed among tasks.
10. Once service classes are in line with business requirements and the goals are achieved, you just need to plan for a periodic review.

11.4 Additional resource management considerations

Much has been written on the following topics, in many sources. It would be unwise to try to replicate all that information here, but we thought we would briefly provide some of our collected insights on these various resource management concepts, especially relative to data warehouse workloads.

11.4.1 zIIP specialty engines

zSeries Integrated Information Processor (zIIP) is a specialty engine that is capable of executing eligible database workloads. The benefits of using a zIIP processor to execute DB2 code consist of the processor cycles that are saved on the general purpose processor. It also saves IBM software charges in WLC, because you can have additional capacity without affecting the total MSU/hour of the server. Work that can run on a zIIP includes:

- Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM): For applications running on z/OS, UNIX, Intel®, or Linux on System z that access DB2 for z/OS V8 and above, via DRDA over a TCP/IP connection, DB2 gives z/OS the necessary information to have portions of these SQL requests directed to the zIIP.
- Data warehousing applications: Requests that utilize DB2 for z/OS V8 and above for parallel queries, including complex star schema parallel queries, might have portions of these SQL requests directed to the zIIP.
- DB2 for z/OS V8 and above utilities: You can redirect a portion of DB2 utility functions that are used to maintain index structures (LOAD, REORG, RUNSTATS, and REBUILD INDEX) that typically run during batch to the zIIP.

Unlike zAAP work, zIIP work is managed as an extension of general purpose processor work and it flows over to general purpose processors.

Setting IIPHONORPRIORITY

The IIPHONORPRIORITY statement in SYS1.PARMLIB(IEAOPTxx) determines whether or not zIIP-eligible work is allowed to run on standard CPs. The default is YES, which signifies that if a zIIP processor is unavailable, zIIP-eligible work can run on standard CPs. Optimally, from a cost-efficiency perspective, we would rather have all zIIP-eligible work to run on zIIP engines. By specifying IIPHONORPRIORITY=NO, zIIP-eligible work will not run on standard CPs, so in the case of none of the zIIP processors being available, zIIP-eligible work will wait.

So in specifying IIPHONORPRIORITY, a key consideration is whether the zIIP-eligible work can tolerate the potential latency. With the theory that data warehouse workloads might be less likely to require subsecond response times, we conducted an experiment of our mixed transactional and dw workload, comparing IIPHONORPRIORITY YES to NO.

For our experiment (see 15.9, “zIIP-eligible work to only run on zIIP processors” on page 364) our conclusion was that IIPHONORPRIORITY=NO was a reasonable setting. In our experiment, the only workload exploiting zIIPs was data warehouse queries. In a production shop, you need to consider whether all zIIP-eligible workloads could tolerate potential latency with IIPHONORPRIORITY=NO.

An additional consideration is when running on IBM System z10 BC processor models. In this case, the zIIP processors have a faster cycle time than the general purpose processors, hence another reason why IIPHONORPRIORITY=NO might be more efficient.

11.4.2 zAAP on zIIP

z/OS V1.11 has been enhanced with a new function that can enable System z Application Assist Processor (zAAP) eligible workloads to run on System z Integrated Information Processors (zIIPs). This function allows zIIP- and zAAP-eligible workloads to run on the zIIP.

This function is targeted for customers without enough zAAP- or zIIP-eligible workloads to justify a specialty engine today. The combined eligible workloads may make the acquisition of a zIIP cost effective.

This capability is available with z/OS V1.11 (and z/OS V1.9 and V1.10 with PTF for APAR OA27495). This capability does not provide an overflow so that additional zAAP-eligible workloads can run on the zIIP; it enables the zAAP-eligible work to run on zIIP when no zAAP is defined. Therefore, this new capability is not available for z/OS LPARs if zAAPs are installed on the server.

This function is enabled via the ZAAPZIIP parameter within IEASYSxx. The default in z/OS V1.11 is YES. The default in z/OS V1.9 and V1.10 is NO.

For more information about this function, see the following reference:

ftp://ftp.software.ibm.com/common/ssi/sa/st/n/zsq03036usen/ZSQ03036USEN.PDF

Co-locating transactional and data warehouse workloads likely increases the chances of using both zAAP and zIIP specialty engines. When migrating to a new System z server, consider merging zAAPs and zIIPS into one larger pool of zIIPs. One pool will likely result in more efficient use of resources, especially if shared across multiple LPARs.

11.4.3 LPAR group capacity limit

A major consideration with warehouse co-location, where multiple LPARS are being utilized, needs to be the control of available capacity usage.

While setting controls for individual LPARs via the HMC does allow a degree of control, circumstances such as an LPAR being deactivated can very easily skew these limits.

With the logical partition (LPAR) Group Capacity Limit you can specify LPAR group capacity limits allowing you to define each LPAR with its own capacity and one or more groups of LPARs on a server. This is designed to allow z/OS to manage the groups in such a way that the sum of the LPARs' processor utilization within a group will not exceed the group's defined

capacity. Each LPAR in a group can still optionally continue to define an individual LPAR capacity limit.

LPAR group capacity limits may help provision a portion of a System z server to a group of LPARs allowing the processor resources to float more readily between those LPARs, resulting in more productive use of “white space” and higher server utilization.

By using this method we can ensure that the data warehouse LPARs are restricted to certain resource levels and as such cannot overwhelm the available resource or skew the client’s MSU usage targets.

The LPAR group capacity limit requires that all LPARs managed in the group are running at z/OS V1R8 or later.

11.4.4 Blocked workload support

Blocked workload support is intended to allow small amounts of processor service to be allocated to workloads that are processor-starved. The idea behind this function is: By providing some processing cycles to these processor-starved pieces of work, it would lessen the chance of these lower importance requests holding a shared resource with a higher importance request. If your system is experiencing time-outs or delays due to scenarios like this, then consider exploiting this support. For more information, see the Washington System center FLASH at:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10609

Archived

Archived

12

DB2 data sharing workload balancing and fault tolerant configuration

Configuring a fault tolerant environment for applications accessing a DB2 data sharing group using TCP/IP requires expertise in z/OS V1R8 Communications Server, DB2 9 for z/OS, and the DB2 Version 9.5 for LUW or later data server clients. This chapter develops the steps required to configure and verify sysplex workload balancing and fault tolerant settings.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

12.1 Introduction

DB2 sysplex workload balancing functions provided by the DB2 Connect Server and the IBM DB2 Driver for JDBC and SQLJ, and Connection Concentrator support are critical components of the DB2 data-sharing fault-tolerant environment. DB2 sysplex workload balances work across the DB2 group using connection concentration. The connection concentrator balances application connections by multiplexing transactions across a set of managed server connections to DB2. To balance transactions (individual units of work delimited by a commit) across individual members of the DB2 group, unique member IP addresses are configured for each member using a DVIPA called the member-specific DVIPA. Member IP addresses are not used by remote applications since they route connections to a specific member. Setting up a member IP address as a DVIPA enables routing to work even if a member fails over to another LPAR using VIPA takeover. As soon as the member is started after a failure, any distributed in-doubt threads can be quickly recovered.

The TCP/IP sysplex distributor configured with DVIPA and automatic VIPA is the other critical component of the DB2 data-sharing fault-tolerant environment. The functionality of TCP/IP sysplex distributor is that one IP entity advertises ownership of an IP address by which a particular service is known. In this fashion, the single system image of DB2 by remote applications is the use of a special IP address enabled for connection distribution across the group. This IP address is called a distributed DVIPA. Further, in sysplex distributor, the IP entity advertising the distributed DVIPA and dispatching connections destined for it is itself a system image within the sysplex, referred to as the distributing stack. To support a distributed DB2 single system image, a group IP address is configured and used by all application servers to access the DB2 group. All DB2 members are configured with the same group DVIPA that is distributed across the group using a distributed DVIPA supported by the sysplex distributor. DB2 calls the distributed DVIPA the DB2 location or group dynamic VIPA.

The objectives of the setting described on this chapter are:

- To provide a Workload Balanced and Fault Tolerant configuration for the access to our Data Sharing Group
- To create a subgroup composed of a subset of Data Sharing members dedicated to our IBM Cognos BI workload

Figure 12-1 shows our DB2 data sharing connection configuration overview.

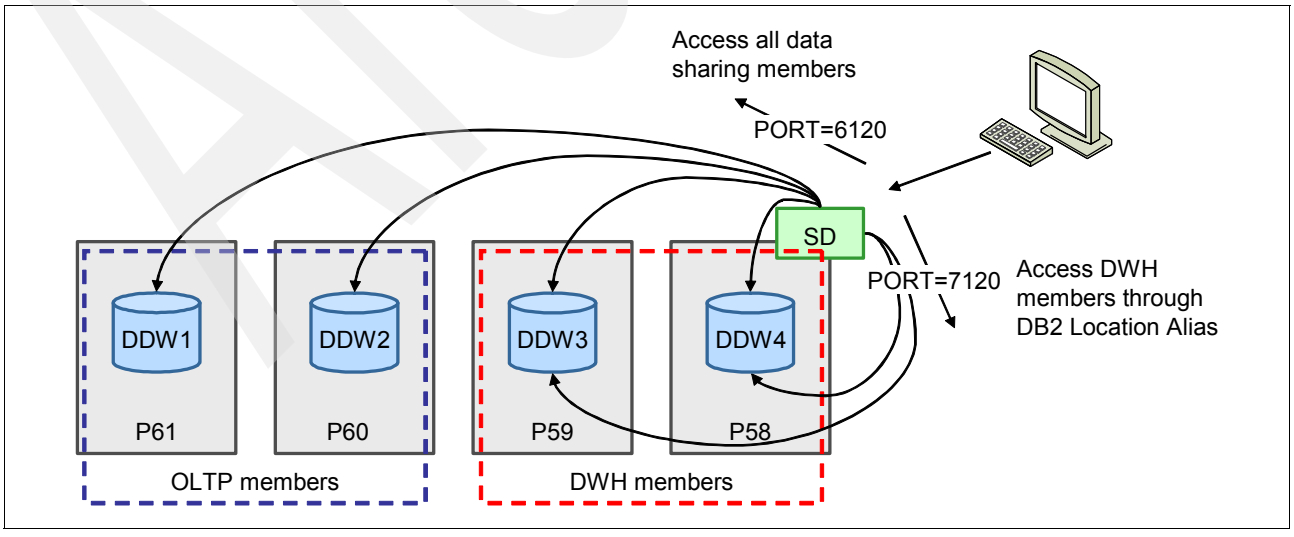


Figure 12-1 Connections configuration overview

For more information, refer to the following publications:

- *DB2 9 for z/OS Data Sharing: Distributed Load Balancing and Fault Tolerant Configuration* at*:*

  http://www.redbooks.ibm.com/abstracts/redp4449.html

- *DB2 9 for z/OS: Distributed Functions* at:

  http://www.redbooks.ibm.com/abstracts/SG246952.html

- *Communications Server for z/OS V1R11 TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance at:*

  http://www.redbooks.ibm.com/abstracts/SG247800.html

12.2 Configuration

DB2 provides an initial contact port and a resync port. The resync port is used by a database client in two situations:

- One is when the SQL connection fails leaving in-doubt threads, and the client and server need to resynchronize after the error.
- The other is for other connections used to interrupt SQL processing on a different application connection.

Obviously, resynchronization needs to occur with the specific DB2 instance with which the client was in session, so this instance must be reachable via a specific IP address (the member-specific DVIPA in this case). To address the case when the DB2 itself terminates and is restarted on another image, possibly with another DB2 instance, each DB2 instance configures its resync address with a port number unique to that instance: at a minimum, we need to allocate a port on which the members of the data sharing group will be listening and a distinct per member resynch port.

The initial contact point to the DB2 group is with the DB2 port 6120 using the distributed DVIPA bound to each member DVIPA.

In DB2 9 for z/OS the specific and group IP addresses can be configured in the DB2 boot strap data set allowing DB2 to listen to any IP address, thereby eliminating the need to use the BIND parameter on the PORT reservation statement in a TCP/IP configuration. The location of a particular DB2 instance at any point in time is thus transparent to the database application servers. Example 12-1 shows one of the DSNJU003 executions we used in our settings.

Example 12-1 Updating BSDS with DVIPA and Group DVIPA information

```
//DDW4JU03 EXEC PGM=DSNJU003
//STEPLIB  DD DISP=SHR,DSN=WORKLOAD.ZDW.DB2910.TEST.SDSNLOAD
//         DD DISP=SHR,DSN=DB2910.ZDW.SDSNLOAD
//SYSUT1   DD DISP=SHR,DSN=DB2DW.ZDW.DDW4.BSDS01
//SYSUT2   DD DISP=SHR,DSN=DB2DW.ZDW.DDW4.BSDS02
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
 DDF IPV4=192.168.70.105,GRPIPV4=192.168.70.101
/*
```

The access to a DB2 data sharing group can be isolated to a subset of members; load distribution for both transactions and connections are performed across the subset. If all members of the subset are down, connection failures will occur even if other members not in

the subset are started and capable of processing work. By designating subsets of members, you can:

- Limit the members to which application servers can connect. System and database administrators might find this useful for any number of purposes.
- Ensure that initial connections are established only with members that belong to the specified subset. Without subsets, requesters can make initial and subsequent connections to any member of the data sharing group.
- Provide requesters with information about only those members in the subset. With subsets, a member that receives an initial connection request can return to the requester a list of members that are currently active, able to perform work, and represented by the location alias.

Each DB2 member configures its alias port number, which is unique across all members participating in the subset by updating the BSDS information.

We created a subset identified by the alias DDWGCOG that is listening in the port 7120. Example 12-2 shows one of the jobs we used for updating one of the members of the subgroup.

Example 12-2 Updating DB2 BSDS with ALIAS information

```
//DDW4JU03 EXEC PGM=DSNJU003
//STEPLIB  DD DISP=SHR,DSN=WORKLOAD.ZDW.DB2910.TEST.SDSNLOAD
//         DD DISP=SHR,DSN=DB2910.ZDW.SDSNLOAD
//SYSUT1   DD DISP=SHR,DSN=DB2DW.ZDW.DDW4.BSDS01
//SYSUT2   DD DISP=SHR,DSN=DB2DW.ZDW.DDW4.BSDS02
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
 DDF ALIAS=DDWGCOG:7120
/*
```

All the members, including the participants of this subgroup, listen at port 6120, while only the members that are dedicated to a IBM Cognos BI workload can be reached via the port 7120. This situation is represented in Figure 12-2.

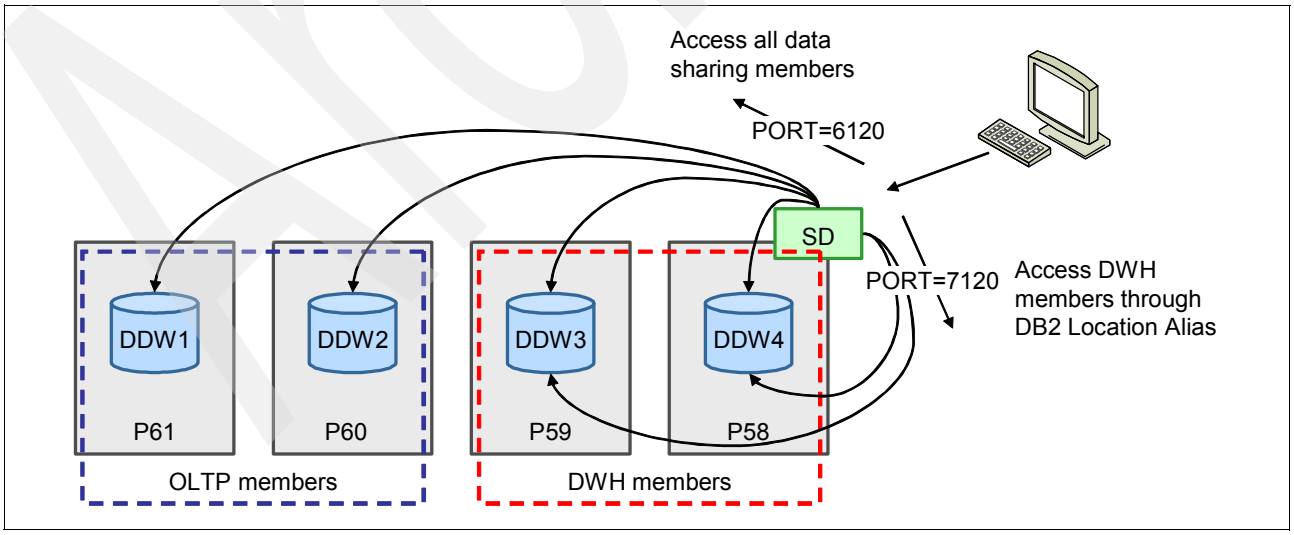


Figure 12-2 Connections configuration overview: DB2 Location ALIAS

Example 12-3 shows how part of this information can be retrieved by the DISPLAY DDF DETAIL DB2 command, in this case executed on the member DDW4, one of the members of the DDWGCOG subgroup.

Example 12-3 DDW4 -DIS DDF DETAIL output command

```
16.39.28 STC19296  DSNL080I  -DDW4 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:  391
   391             DSNL081I STATUS=STARTD
   391             DSNL082I LOCATION           LUNAME           GENERICLU
   391             DSNL083I DDWG               USIBMT6.DDFDDW4  -NONE
   391             DSNL084I TCPPORT=6120  SECPORT=6122  RESPORT=6151  IPNAME=-NONE
   391             DSNL085I IPADDR=::192.168.70.101
   391             DSNL086I SQL    DOMAIN=DDW4
   391             DSNL086I RESYNC DOMAIN=DDW4
   391             DSNL087I ALIAS              PORT  SECPORT
   391             DSNL088I DDWGCOG            7120  0
   391             DSNL089I MEMBER IPADDR=::192.168.70.105
   391             DSNL090I DT=I  CONDBAT=    400 MDBAT=  100
  391 DSNL092I ADBAT=   27 QUEDBAT=      0 INADBAT=      0 CONQUED=      0
   391             DSNL093I DSCDBAT=     16 INACONN=     41
   391             DSNL100I LOCATION SERVER LIST:
   391             DSNL101I  WT IPADDR           IPADDR
   391             DSNL102I  37 ::192.168.70.103
   391             DSNL102I  43 ::192.168.70.105
   391             DSNL102I  23 ::192.168.70.102
   391             DSNL102I  33 ::192.168.70.104
   391             DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

12.3 Data sharing subgroup attach

As discussed in the previous section, the use of a DB2 location alias allows you to restrict DRDA work to a subset of data sharing members. Figure 12-3 illustrates the configuration utilized for the execution of some of the experiments described in this book.

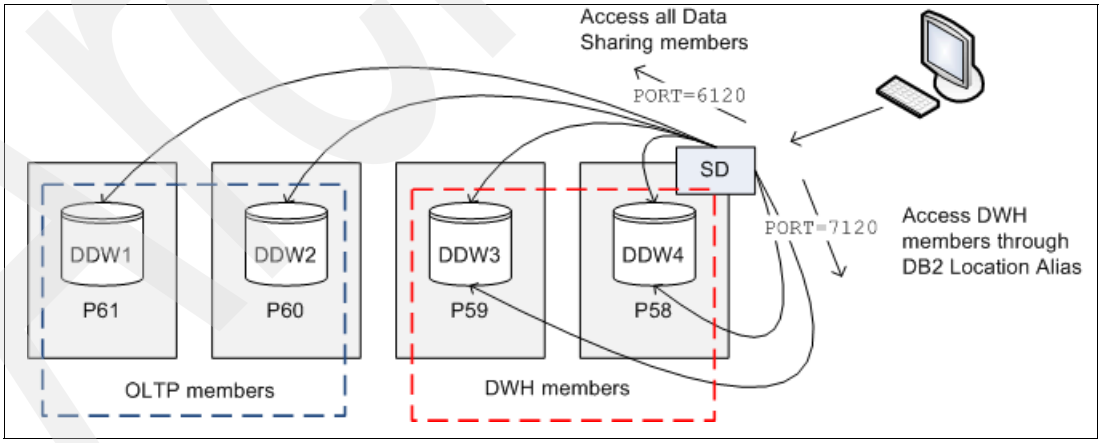


Figure 12-3 DB2 Location Alias example

In this example, DRDA connection requests through the DB2 location alias will reach only the members DDW3 and DDW4. We implemented sysplex high availability and workload balancing support using member-specific dynamic VIPA for each member. One of the DWH members could be started on an LPAR where an OLTP member resides, as illustrated in

Figure 12-4. In this co-location example, DRDA connection requests will reach only the members defined in the location alias.

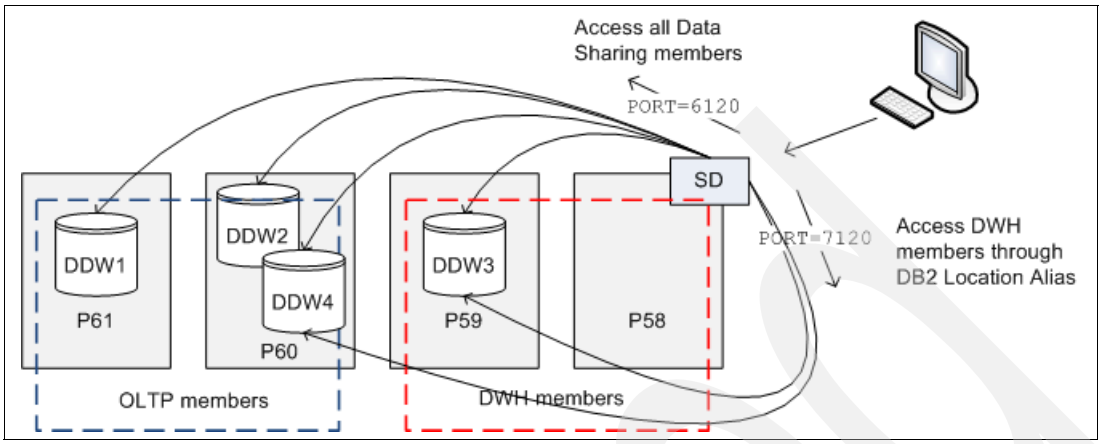


Figure 12-4 DB2 member failover and DB2 Location Alias

The exploitation of location aliases is restricted to DRDA connection requests and cannot be used for the following connection types:

- TSO Attach
- CAF
- RRSAF
- JDBC
- ODBC
- DB2 utilities

The advantages of subgroup attach for non-DRDA connection requests can be of great help in some co-location configurations.

Some examples include member co-location configurations where it is preferable to route ELT processes, utilities, or reporting jobs to the DWH members only with minimal impact on the OLTP ones. Refer to Chapter 3, “Co-location topologies” on page 45 for more configuration examples.

In DB2 for z/OS V8, group attachments were attempted in sequence given by the order of the DB2 subsystem initialization and some clients relied on this behavior in order to isolate some workload to specific members. DB2 for z/OS V9 changed the attachment order method and members were selected at random.

APAR PK79327: NEW FUNCTION SET MEMBER INELIGIBLE FOR RANDOMIZED GROUP ATTACH introduced improvements to the randomization algorithm to more evenly distribute group attachment requests. It also introduced the DB2 subsystem parameter `RANDOMATT`.

`RANDOMATT=NO` may be specified to exclude a member from being selected at random. To satisfy a group attachment request, DB2 first randomly searches for an active DB2 subsystem, excluding those specifying `RANDOMATT=NO`. If all such subsystems are inactive, DB2 will search for any active DB2 subsystem within the group, in the order defined by the z/OS subsystem initialization, regardless of the setting of `RANDOMATT`. To achieve non-randomized group attachments for all members, as in DB2 Version 8, in DB2 V9 specify `RANDOMATT=NO` for all members of the group.

Implementing data sharing member subgroup attach

The usermod ANSUBS9 allows users of DB2 9 for z/OS to implement subgroup attach. This provides non-DRDA connection requests with a function like Location Alias. This usermod, provided as file ANSUBS9.$HIZ9910, can be obtained by request from IBM. It enforces rules for the IEFSSNxx DB2 subsystem definition more aggressively than base DB2 9 and you may need to review your current definitions. This usermod does not support RANDOMATT=YES/NO for subgroup attach; the subgroup RANDOM attach is always used.

Note: Data sharing member subgroup is supported by DB2 V10 natively. The V10 code does support RANDOMATT=YES/NO

With this usermod, the MVS subsystem definition allows the optional inclusion of a subgroup attach name to assist in the control of DB2 connection requests. The subgroup attach name is added to the existing definition parameter, as shown in Example 12-4.

Example 12-4 Adding a subgroup attach definition example

```
DDW1,dsn3ini, 'DSN3EPX,-DDW1,S,DDWG,DDWO'
```

In this example, DDW1 indicates the member name, DDWG indicates the group name, and DDWO indicates the subgroup name. As an example, consider a configuration with two subgroups within the data sharing group DDWG that provides a way of directing attach requests only to the two OLTP members DDW1 and DDW2 via the subgroup DDWO, and to the two DWH members DDW3 and DDW4 via the subgroup DDWH. Figure 12-5 describes this disposition.

Figure 12-5 Data sharing subgroups implementation schema

Example 12-5 shows the subsystem initialization commands for implementing this configuration.

Example 12-5 Definition of data sharing subgroups

```
DDW1,dsn3ini, 'DSN3EPX,-DDW1,S,DDWG,DDWO'
DDW2,dsn3ini, 'DSN3EPX,-DDW2,S,DDWG,DDWO'
DDW3,dsn3ini, 'DSN3EPX,-DDW3,S,DDWG,DDWH'
DDW4,dsn3ini, 'DSN3EPX,-DDW4,S,DDWG,DDWH'
```

The benefits of data sharing subgroups are evident in the case of co-location of an OLTP and a DWH member on the same LPAR. Co-location could be the configuration of choice for your environment or it could be the result of a member that needs to fail over to an LPAR that is reserved for OLTP processing in normal circumstances. This last situation is shown in Figure 12-6. In this case, jobs and processes that normally run on the DWH LPAR can also be

failed over to the OLTP LPAR and processing will occur (provided they use the subgroup definition for attach) only against the DWH member DDW4.

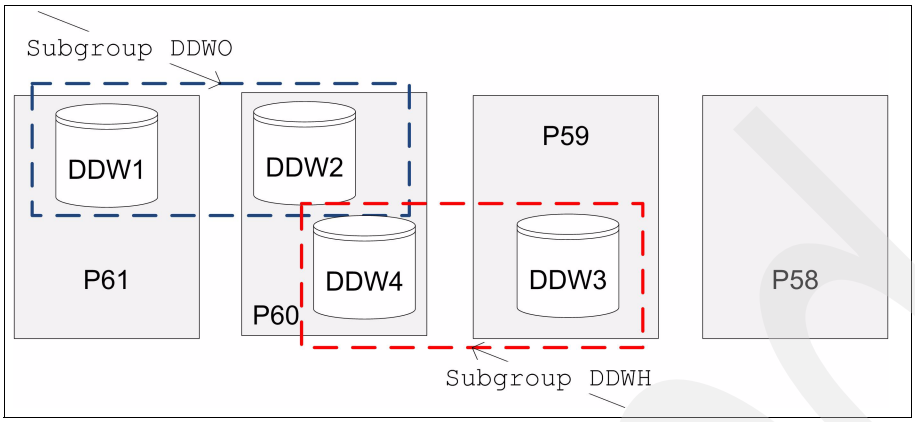


Figure 12-6 Failover of data sharing member to an OLTP LPAR

Important: If all of the members in a subgroup are inactive, the subgroup attach will fail even if other members of the data sharing group are active.

Subgroup attach is subject to certain rules and conditions:

- It can only belong to one group attach.
- It must have a group attach.
- It cannot have the same name as a group attach.
- A member may only belong to at most one subgroup attach.
- A member does not need to belong to a subgroup attach.

These rules are explicitly enforced and new messages warn a user of an invalid initialization parameter as well as fail the subsystem initialization. Subgroup attach names also follow the same rules as group attach names such that they must be 1-4 characters in length and should not be the same as an existing DB2 member name (not enforced).

Important: Dynamic reconfiguration of subgroup attach is not supported and an IPL is required to clear subgroup attach data (as is the case with group attach).

The DB2 command -DISPLAY GROUP DETAIL has been modified to display all defined members of all subgroup attach groups defined to the given member. See Example 12-6 for a sample overview.

Example 12-6 -DISPLAY GROUP DETAIL and subgroup information

```
---------------------------------------------------------------------
DISPLAY SUBGROUP ATTACH INFORMATION FOR GROUP ATTACH DDWG
---------------------------------------------------------------------
MEMBERS OF SUBGROUP ATTACH DDWO
DDW1 DDW2
MEMBERS OF SUBGROUP ATTACH DDWH
DDW3 DDW4
```

Attention: -DISPLAY GROUP DETAIL support for subgroup information is part of DB2 V10. The usermod ANSUBS9 (for DB2 for z/OS V9) does not provide this facility.

Subgroup attach is subject to the same rules concerning RANDOM ATTACH as group attach. Using the -SET SYSPARM command to change RANDOM ATTACH - NO/YES will change the member information for both group attach and subgroup attach.

Conclusions

Subgroup capability for local attaches allows non-DRDA requests to be directed to a subgroup of data sharing members providing an equivalent level of member selectivity. If you are exploiting DB2 location aliases for a DRDA workload, you can extend this functionality in DB2 for z/OS 9 to other workloads by implementing this usermod. By doing so you can better plan and distribute any workload in a co-located environment. The utilization of this usermod extends the concept of data sharing subgroup to all kinds of workloads.

12.4 Dynamic VIPA and Distributed DVIPA (Sysplex Distributor)

The z/OS Communication Server provides two types of Virtual IP Addressing (VIPA):

- Static
- Dynamic

A static VIPA is an IP address that is associated with a particular TCP/IP stack. Using either ARP takeover or a dynamic routing protocol (such as OSPF), static VIPAs can enable mainframe application communications to continue unaffected by network interface failures. As long as a single network interface is transactional on a host, communication with applications on the host will persist.

Dynamic VIPAs (DVIPAs) can be defined on multiple stacks and moved from one TCP/IP stack in the sysplex to another automatically. One stack is defined as the primary or owning stack and the others are defined as backup stacks. Only the primary stack is made known to the IP network. TCP/IP stacks in a sysplex exchange information about DVIPAs, their existence, and their current location, and the stacks are continuously aware of whether the partner stacks are still functioning. If the owning stack leaves the XCF group (such as resulting from some sort of failure), then one of the backup stacks automatically takes its place and assumes ownership of the DVIPA. The network simply sees a change in the routing tables (or in the adapter that responds to ARP requests).

In this case, applications associated with these DVIPAs are active on the backup systems, thereby providing a hot standby and high availability for the services. DVIPA addresses identify applications independently of which images in the sysplex the server applications execute on and allow an application to retain its identity when moved between images in a sysplex.

Sysplex Distributor is a function that allows an IP workload to be distributed to multiple server instances within the sysplex without requiring changes to clients or networking hardware and without delays in connection setup. It enables us to implement a dynamic VIPA as a single network-visible IP address that is used for a set of hosts belonging to the same sysplex cluster. A client on the IP network sees the sysplex cluster as one IP address, regardless of the number of hosts in the cluster.

With Sysplex Distributor, clients receive the benefits of workload distribution provided by both the Workload Manager (WLM) and the Quality of Service (QoS) Policy Agent. In addition,

Sysplex Distributor ensures high availability of the IP applications running on the sysplex cluster by providing continued operation if a LAN fails, or an entire IP stack leaves the XCF group, or a z/OS image is lost.

For high availability and workload balancing in a DB2 data sharing environment you must define TCP/IP and DB2 to take advantage of the function of Sysplex Distributor to make an initial connection to an available member of the data sharing group and to define the DB2 data sharing group and members with dynamic virtual IP addressing (DVIPA) to allow connection requests to succeed despite LPAR outages.

In a high availability configuration the DB2 data sharing group has a distributed DVIPA (also known as the group DVIPA). The requesters in the network specify this group DVIPA to initiate a connection to the data sharing group. The Sysplex Distributor function of TCP/IP identifies an available member of the data sharing group. Sysplex Distributor generally routes initial connection requests evenly across the available members of the group. Subsequently the workload balancing functions available in the DRDA requesters use the server list returned by DB2 and route transactions or connection requests across the members of the data sharing group.

You can use these steps as a guideline when you implement DVIPA and Sysplex Distributor support in your data sharing environment:

- Identify available IP addresses for the group DVIPA and each member-specific DVIPA.
- Update the TCP/IP profile settings with appropriate VIPADynamic statements.
- Add group and member-specific DVIPAs to the `/etc/hosts` file.
- Update BSDS for each DB2 member.

Contact your TCP/IP network administrator to identify what addresses are available for group DVIPA and member-specific DVIPAs. If you are planning for a new data sharing group, and if your existing DB2 requesters use an IP address that is specific to DB2, try to make that IP address the group DVIPA, as it will ease the migration.

Attention: Be very careful when updating the TCP/IP PROFILE data set in order to avoid outages.

We used the following IP addresses in our environment:

- 192.168.70.101for the Group DVIPA
- 192.168.70.102 as the member-specific DVIPA for DDW1
- 192.168.70.103 as the member-specific DVIPA for DDW2
- 192.168.70.104 as the member-specific DVIPA for DDW3
- 192.168.70.105 as the member-specific DVIPA for DDW4

Example 12-7 shows the VIPADYNAMIC section of the TCP/IP PROFILE used on the LPAR where the Sysplex Distributor is normally running.

Example 12-7 TCP/IP PROFILE file extract for the LPAR where Sysplex Distributor runs

```
....
; Sysplex Distributor (SD) definitions.
VIPADYNAMIC
  VIPARANGE DEFINE 255.255.255.255 192.168.70.102 ; DDW1
  VIPARANGE DEFINE 255.255.255.255 192.168.70.103 ; DDW2
  VIPARANGE DEFINE 255.255.255.255 192.168.70.104 ; DDW3
  VIPARANGE DEFINE 255.255.255.255 192.168.70.105 ; DDW4
```

```
  VIPADEFINE 255.255.255.255 192.168.70.101 ; Group DVIPA
  VIPADISTRIBUTE DEFINE 192.168.70.101 PORT 6120 6122 7120 DESTIP ALL
ENDVIPADYNAMIC
```

Example 12-8 shows the configuration used on the other LPARs.

Example 12-8 TCP/IP PROFILE for the LPARs where Sysplex Distributor is not running

```
....
; Sysplex Distributor (SD) definitions.
VIPADYNAMIC
  VIPARANGE DEFINE 255.255.255.255 192.168.70.102 ; DDW1
  VIPARANGE DEFINE 255.255.255.255 192.168.70.103 ; DDW2
  VIPARANGE DEFINE 255.255.255.255 192.168.70.104 ; DDW3
  VIPARANGE DEFINE 255.255.255.255 192.168.70.105 ; DDW4
  VIPABACKUP 100 192.168.70.101 ; Group DVIPA
  VIPADISTRIBUTE DEFINE 192.168.70.101 PORT 6120 6122 7120 DESTIP ALL
ENDVIPADYNAMIC
```

Changes on the TCP/IP stack can be activated by using the OBEYFILE command. Example 12-9 shows an example of the OBEYFILE command being executed on one of our partitions.

Example 12-9 TCP/IP OBEYFILE command example

```
/VARY TCPIP,,OBEYFILE,DSN=TCPIP.PROFILES(ZDWP58)
```

The output of this command is represented in Example 12-10.

Example 12-10 TCP/IP OBEYFILE command output example

```
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCPIP.PROFILES(ZDWP58)'
EZZ0641I IP FORWARDING NOFWDMULTIPATH SUPPORT IS ENABLED
EZZ0350I SYSPLEX ROUTING SUPPORT IS ENABLED
EZZ0615I MULTIPATH SUPPORT IS DISABLED
EZZ0759I CANNOT MODIFY DYNAMICXCF ON IPCONFIG AFTER IT HAS BEEN ENABLED
EZZ0619I LINK VIPLOA016E51 USES DUPLICATE HOME ADDRESS 10.1.110.81
EZZ0619I LINK EZASAMEMVS USES DUPLICATE HOME ADDRESS 10.1.110.81
EZZ0619I LINK IQDIOLNKOA016E51 USES DUPLICATE HOME ADDRESS 10.1.110.81
EZZ0619I LINK EZAXCF60 USES DUPLICATE HOME ADDRESS 10.1.110.81
EZZ0619I LINK EZAXCF61 USES DUPLICATE HOME ADDRESS 10.1.110.81
EZZ0619I LINK EZAXCF59 USES DUPLICATE HOME ADDRESS 10.1.110.81
EZZ0338I TCP PORTS 1 THRU 1023 ARE RESERVED
EZZ0338I UDP PORTS 1 THRU 1023 ARE RESERVED
EZZ0059I VARY OBEY COMMAND FAILED: SEE PREVIOUS MESSAGES
```

In addition to defining the IP addresses to TCP/IP, the member and group DVIPA corresponding host names must be defined prior to starting DDF. DDF recovery processing may require the use of these names during in-doubt resolution after a subsystem failure. You define the host names by configuring the hlq.HOSTS.LOCAL data set, the `/etc/hosts` file in the hierarchical file system (HFS), or the domain name server (DNS).

In our test environment, we set this information in the `/etc/hosts` file; an extract of our settings is shown in Example 12-11.

Example 12-11 Extract of /etc/hosts settings

```
/etc/hosts
 ===>
```

```
**************************
....
192.168.70.101   DDWGCOG
192.168.70.102   DDW1
192.168.70.103   DDW2
192.168.70.104   DDW3
192.168.70.105   DDW4
```

After adding the new definitions in /etc/hosts, we made them active without outage by refreshing the resolver using the command shown in Example 12-12.

Example 12-12 Making effective changes on /etc/hosts by refreshing the resolver information

```
/Modify Resolver,Refresh
```

Example 12-13 shows the output in the System Log after execution of the refresh command.

Example 12-13 Feedback in System Log of refreshing the resolver

```
MODIFY RESOLVER,REFRESH
EZZ9298I DEFAULTTCPIPDATA - TCPIP.TCPDATA(GLOBAL)
EZZ9298I GLOBALTCPIPDATA - None
EZZ9298I DEFAULTIPNODES - None
EZZ9298I GLOBALIPNODES - None
EZZ9304I NOCOMMONSEARCH
EZZ9293I REFRESH COMMAND PROCESSED
```

12.5 Stress tests and workload distribution

Workload distribution and fault tolerant capabilities of the Sysplex Distributor are based on the information provided by a server list that each member of the DB2 data-sharing group returns on connection boundaries. This list contains the IP address and the workload balancing weight for each DB2 member.

The server list is returned on the first successful connection to the DB2 database. Therefore, the initial database connection should be directed at the group DVIPA owned by the Sysplex Distributor. If at least one DB2 member is available, the Sysplex Distributor routes the request to the database. After the client has received the server list, the client directly accesses a DB2 member based on information in the server list.

The values returned by the server list are reported in the messages DSNL100I and DSNL102I that are returned when executing the DIS DDF DETAIL command. Example 12-14 shows an example.

Example 12-14 Server list information in the DIS DDF DETAIL command output

```
15.23.58 STC19939  DSNL080I  -DDW4 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:  953
...
   953             DSNL100I LOCATION SERVER LIST:
   953             DSNL101I  WT IPADDR            IPADDR
   953             DSNL102I  23 ::192.168.70.103
   953             DSNL102I  22 ::192.168.70.102
   953             DSNL102I  16 ::192.168.70.104
   953             DSNL102I     ::192.168.70.105
   953             DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Archived

In order to monitor and plot the server list and server weights during our tests, we used a REXX program that executed and parsed the DIS DDF command at regular intervals. This program is included in this section as Example 12-15.

Example 12-15 Parsing server list info from the DIS DDF DETAIL command output: REXX

```
/* REXX */
/*
CONFIGIURATION INFORMATION:
DS GROUP NAME = DDWG
GROUP DVIPA (NOT USED --> LOCAL CONNECTION)
   192.168.70.101
MEMBER DVIPAS:
   192.168.70.102 --> DDW1
   192.168.70.103 --> DDW2
   192.168.70.104 --> DDW3
   192.168.70.105 --> DDW4                                          */

SLEEPTIME = 30      /* HOW MANY SECONDS BETWEEN DISPLAYS */
DSGROUP   = "DDWG"  /* DS GROUP */

SAY "TIME      DDW1 DDW2 DDW3 DDW4"

DO FOREVER
  CALL    SYSCALLS 'ON'
  ADDRESS SYSCALL
  "SLEEP" SLEEPTIME
  CALL    SYSCALLS 'OFF'

  CALL DISPLAY_DDF DSGROUP

  DDW1 = "N/A"
  DDW2 = "N/A"
  DDW3 = "N/A"
  DDW4 = "N/A"

  DO XX = 1 TO CMD_OUTPUT.0
     IF POS("DSNL102I",CMD_OUTPUT.XX) > 0 THEN DO
        WT = ""
        IP = ""
        PARSE VALUE CMD_OUTPUT.XX WITH "DSNL102I" WT "::" IP " " .
        /* SAY "-->" CMD_OUTPUT.XX */
        SELECT
          WHEN IP = "192.168.70.102" THEN DO
             DDW1 = WT
          END
          WHEN IP = "192.168.70.103" THEN DO
             DDW2 = WT
          END
          WHEN IP = "192.168.70.104" THEN DO
             DDW3 = WT
          END
          WHEN IP = "192.168.70.105" THEN DO
             DDW4 = WT
          END
          OTHERWISE NOP
          END
     END /* IF POS("DSNL102I",CMD_OUTPUT.XX) > 0 THEN DO */
  END /* DO XX = 1 TO CMD_OUTPUT.0 */
```

Archived

```
  SAY TIME() RIGHT(DDW1,4," ") RIGHT(DDW2,4," ")  ,
             RIGHT(DDW3,4," ") RIGHT(DDW4,4," ")

END

EXIT

/*-------------------------------------------------------------------- */
/* ISSUE DISPLAY UTILITY COMMAND, CAPTURE OUTPUT INTO STEM             */
/*-------------------------------------------------------------------- */
DISPLAY_DDF:
ARG DSGROUP
X     = OUTTRAP(CMD_OUTPUT.)
QUEUE   "-DISPLAY DDF DETAIL"
QUEUE   "END"
ADDRESS TSO "DSN SYSTEM (" || SPACE(DSGROUP) || ")"
IF RC /= 0 THEN,
   DO
      DO I = 1 TO CMD_OUTPUT.0
         SAY CMD_OUTPUT.I
      END
      EXIT 666
   END
X     = OUTTRAP(OFF)
RETURN;
```

Example 12-16 shows part of the output of this program.

Example 12-16 Program output: parsing server list information

```
READY
TRACEDDF
TIME      DDW1 DDW2 DDW3 DDW4
14:38:20  17   18   14   14
14:38:50  17   17   14   14
14:39:20  17   17   14   14
14:39:50  18   17   14   14
14:40:20  18   17   14   14
14:40:50  18   17   14   14
14:41:20  18   17   14   14
14:41:50  18   17   14   14
14:42:20  18   17   14   14
14:42:50  18   18   14   14
14:43:20  18   18   14   14
14:43:50  18   18   14   14
14:44:20  17   18   14   14
14:44:50  17   18   14   14
14:45:21  17   18   14   14
14:45:51  17   18   14   14
14:46:21  17   18   14   14
14:46:51  17   18   14   14
```

This information can be used to create a chart like the one in Figure 12-7 on page 305. This chart shows how WLM gives more priorities to servers where there is more capacity: during this example, we ran the LPAR where the member DDW4 was running to 100% processor utilization.

Archived

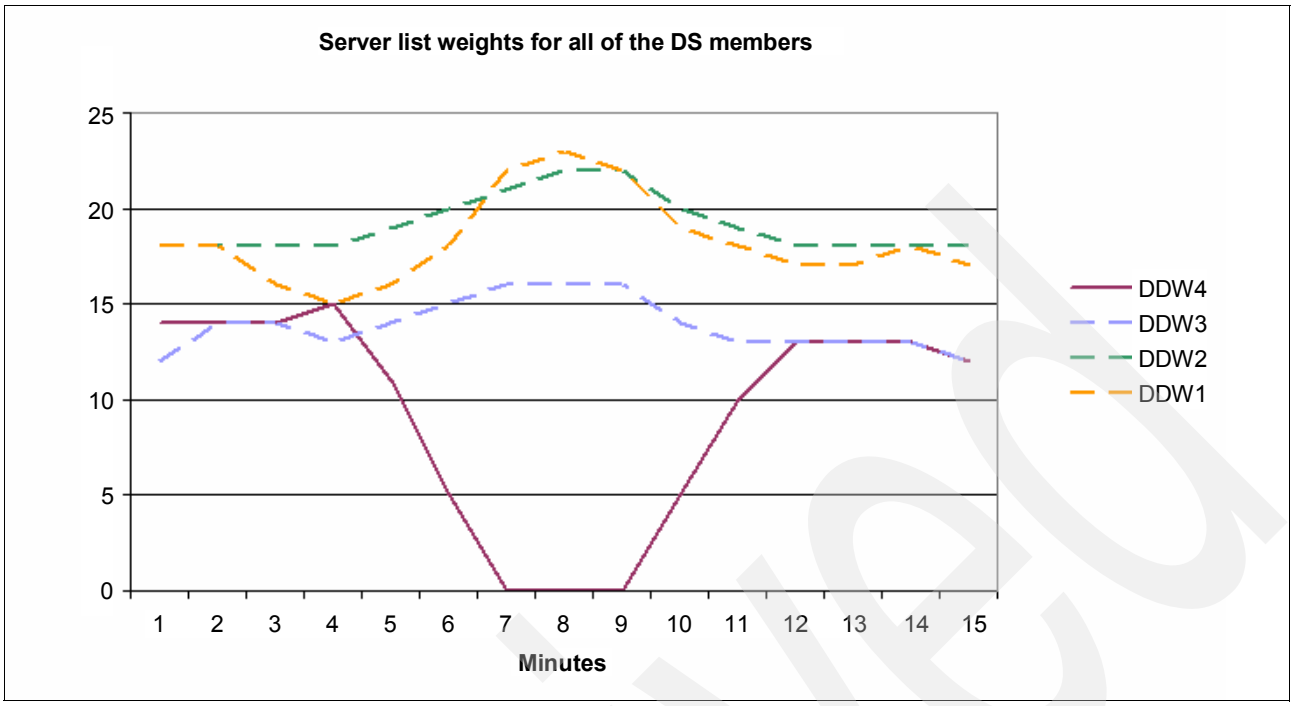


Figure 12-7 Plotting server list information

As a result of the change in the weights, one can expect workload to be routed to other members than DDW4. In order to verify this, we executed the script shown in Example 12-17. This script creates a new connection to the Sysplex Distributor DVIPA using port 7120, which is used only for reaching the data-sharing members reserved for the IBM Cognos BI workloads DDW3 and DDW4.

Example 12-17 Create and destroy connections to DDWGCOG ALIAS

```
#!/usr/bin/ksh
echo "Start workload test"
echo "Parameters:"

# Target DB2 for z/OS alias
MFDB2="DDWG"

# Userid at Server
HOSTuser="res10"

# Password at Server
HOSTpasswd="xxxxxx"

# Number of execution of test query
count=100000

# The following line defines the query to be executed during the benchmark
# Adapt the 'stmt' variable to the test, i.e. I/O or CPU bound

stmt="select current time, current member from
sysibm.sysdummy1;"

echo $stmt
echo "Connecting to "  $MFDB2
while [[ $count -gt 0 ]];do
```

Archive

```
  db2 +o "Connect to " $MFDB2  " user " $HOSTuser " using " $HOSTpasswd
  (( count -= 1 ))
  db2 -xto $stmt
  db2 +o terminate
  sleep  1

done

# End program
# -----------
```

This script was executed on the Linux for z partition; an example of the output can be seen in Example 12-18.

Example 12-18 Connection creation and destruction script: Output example

```
Start workload test
Parameters:
select current time, current member from sysibm.sysdummy1;
Connecting to  DDWG
15.50.35 DDW4
15.50.37 DDW4
15.50.39 DDW4
15.50.43 DDW2
15.50.45 DDW2
15.50.47 DDW2
15.50.49 DDW2
15.50.53 DDW1
15.50.55 DDW1
15.50.57 DDW1
15.50.59 DDW1
15.51.01 DDW3
15.51.03 DDW3
```

Figure 12-8 on page 307 shows a graphic representation of these results. Compare this chart with Figure 12-7 on page 305 and observe how the distribution of connections is influenced by the server list information.

Archived

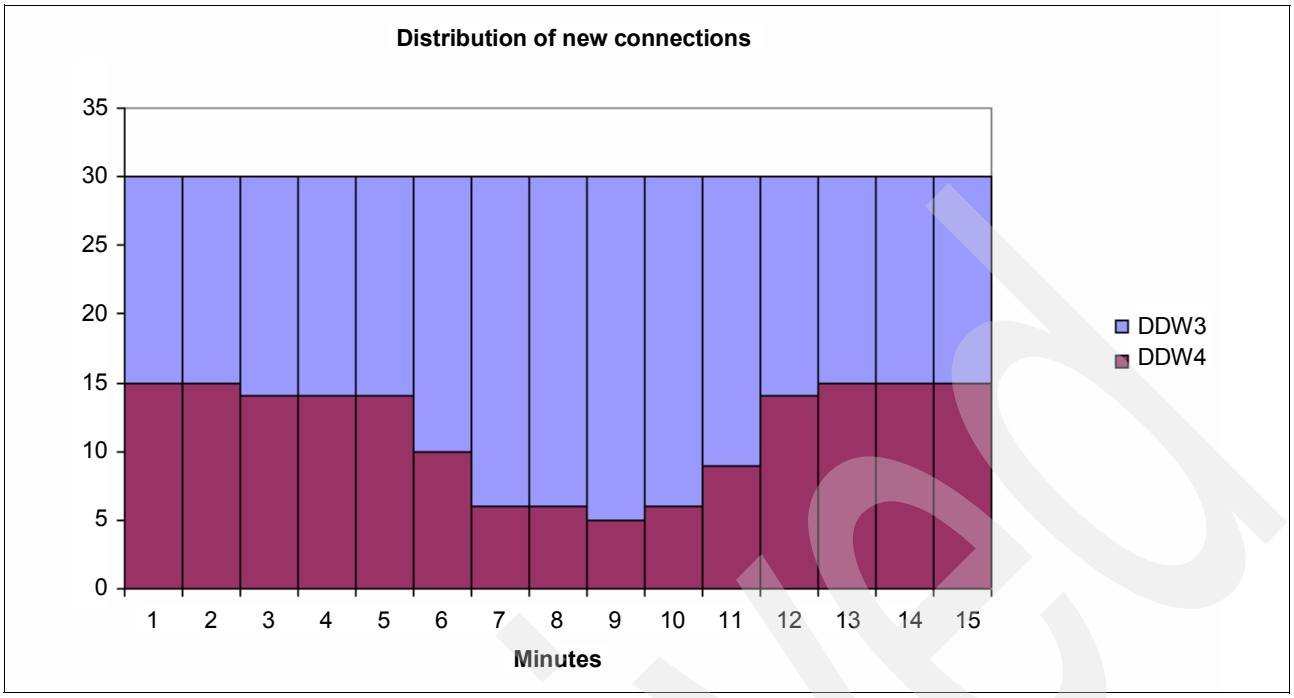


Figure 12-8 Plotting connection distribution

Figure 12-9 shows another scenario in which the processor utilization of the LPAR on which DDW4 was running was pushed to 100% processor utilization.

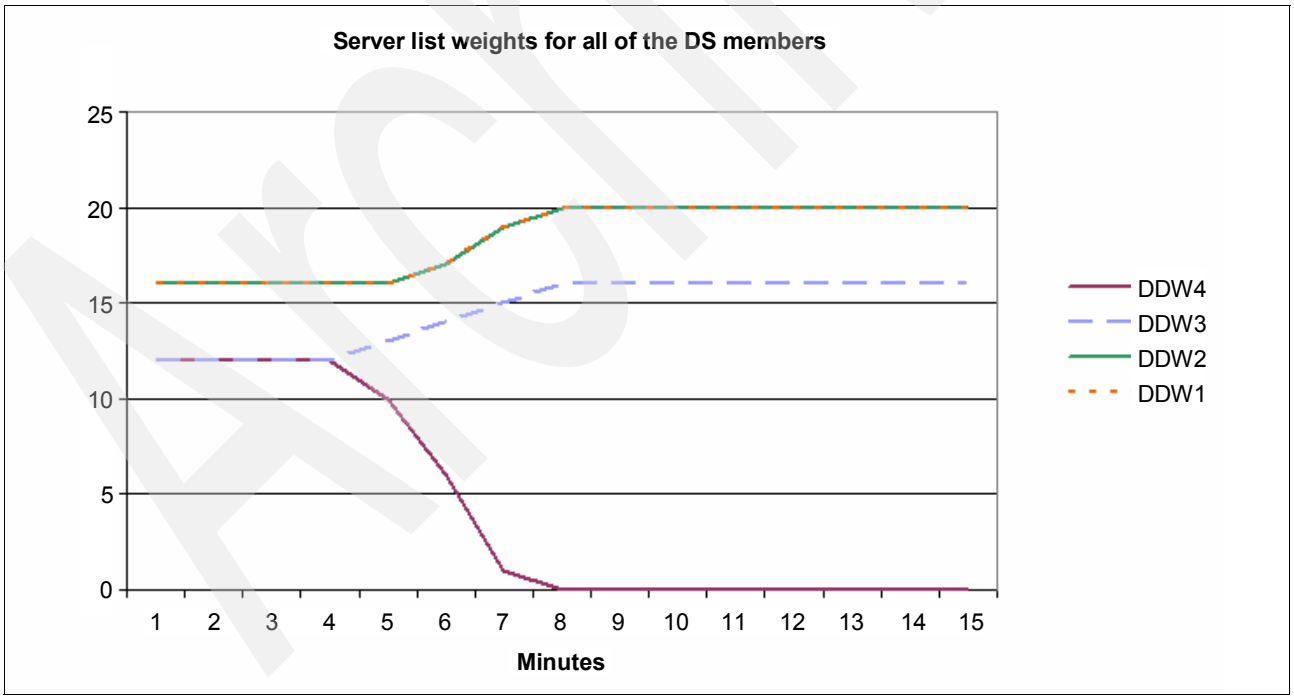


Figure 12-9 Server list weights

During the same period, we used a variation of the script in Example 12-17 but the connections were done using port 6120; it is reaching all the members of the data-sharing group. The distribution of the connections is represented in Figure 12-10 on page 308.

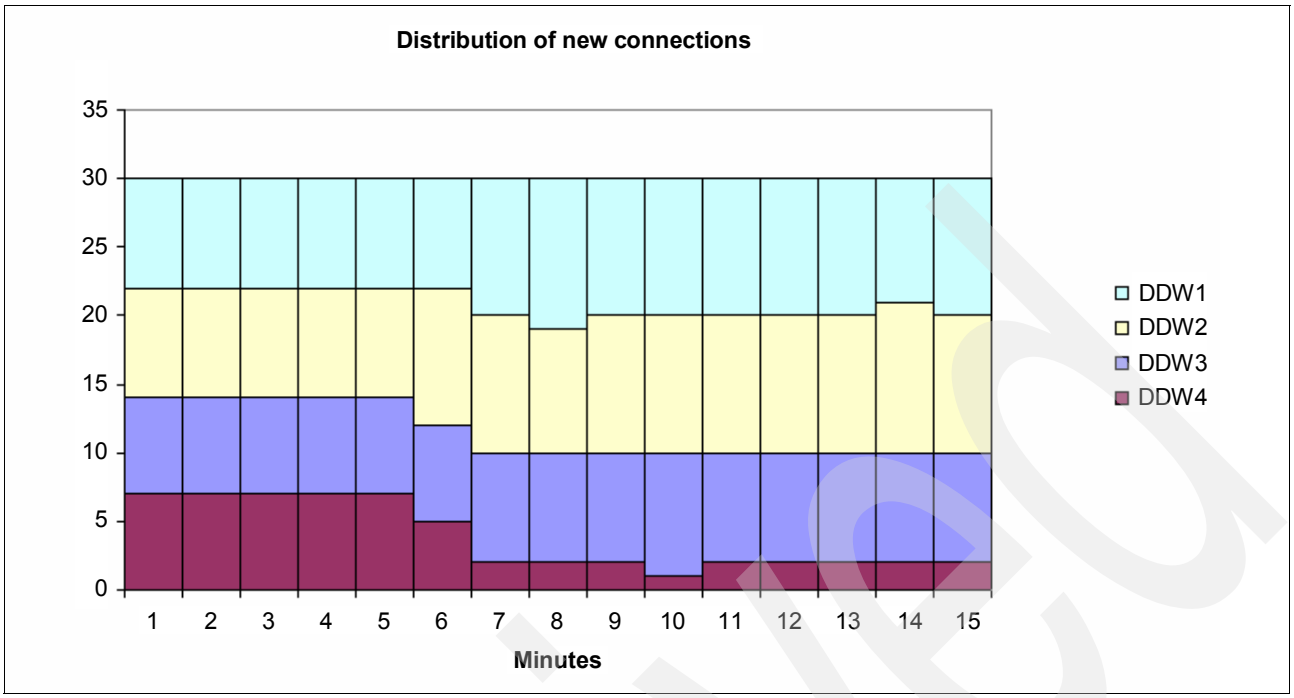


Figure 12-10 Connections distribution

12.5.1 Failover example

Consider as initial situation the scenario described in Figure 12-11: four members of the DS group are active and IBM Cognos BI reports are executed using the DB2 Location Alias on port 7120.

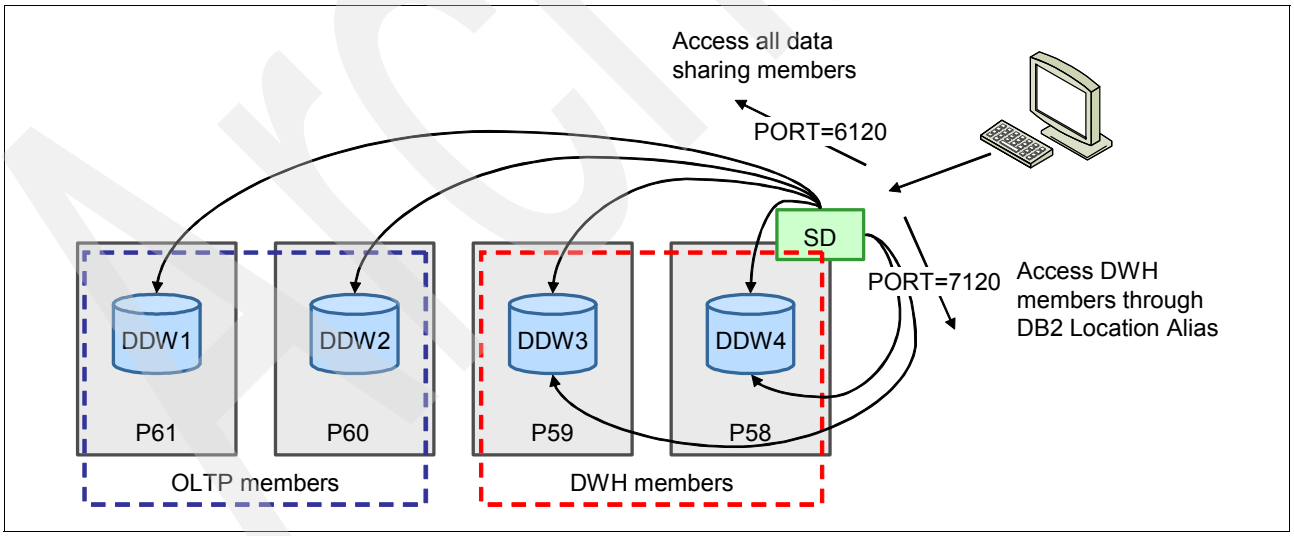


Figure 12-11 Connection distribution schema

Example 12-19 on page 309 shows an extract of the DIS DDF DETAIL command where the four members are listed with their weights as assigned by WLM into the server list.

Example 12-19 DSNL100I example: four members of the DS group are up

```
WT IPADDR            IPADDR
 18 ::192.168.70.102
 18 ::192.168.70.103
 14 ::192.168.70.105
 14 ::192.168.70.104
```

DDW4, running in LPAR 58, was stopped; it deregistered from WLM and is not present any more in the server list. Messages on the System Log indicate that its member-specific DVIPA (192.168.70.104) was deleted from TCP/IP. See Example 12-20 for details.

Example 12-20 Member-specific DVIPA deleted from TCP/IP when stopping a DB2 member

```
-DDW4 STOP DB2
DSNY002I  -DDW4 SUBSYSTEM STOPPING
EZD1298I DYNAMIC VIPA 192.168.70.105 DELETED FROM TCPIP
EZD1206I DYNAMIC VIPA 192.168.70.105 WAS DELETED USING IOCTL BY DDW4DIST
 ON TCPIP
DSNL005I  -DDW4 DDF IS STOPPING
DSNL006I  -DDW4 DDF STOP COMPLETE
```

Example 12-21 shows that the server list will not include the stopped member's specific DVIPA and as consequence no connections will be routed to it.

Example 12-21 Server list example showing a missing member

```
LOCATION SERVER LIST:
  WT IPADDR            IPADDR
  23 ::192.168.70.103
  22 ::192.168.70.102
  18 ::192.168.70.104
 DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Member DDW4 is then started in another LPAR, P60. The resulting configuration is shown in Figure 12-12.

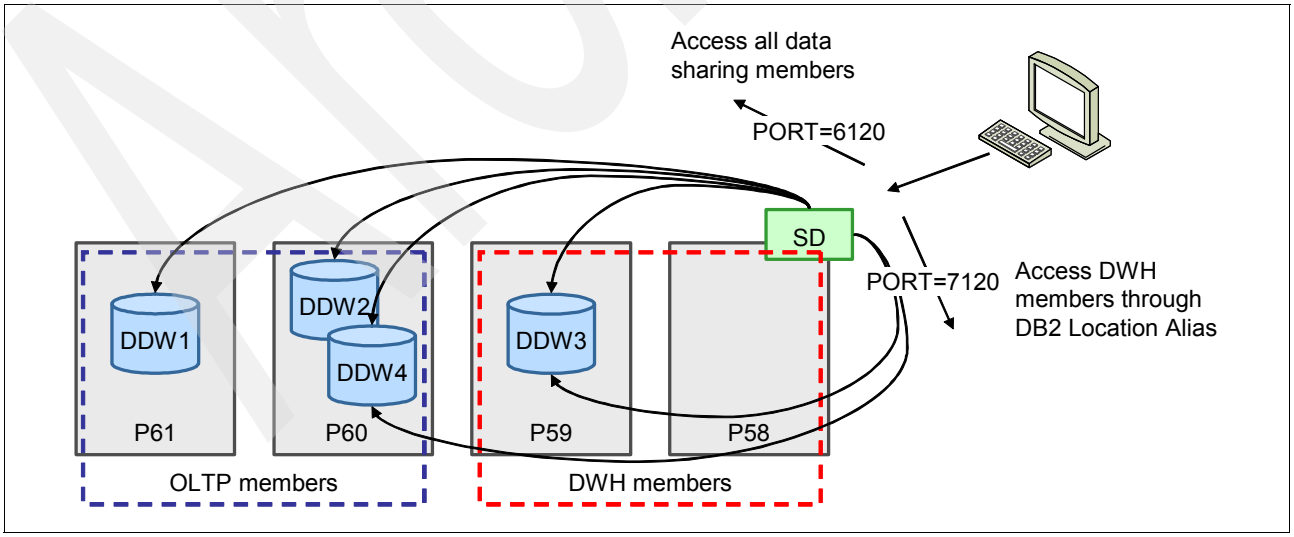


Figure 12-12 Starting a member in another LPAR

Archived

As a consequence of starting member DDW4, its member-specific DVIPA is created again into TCP/IP as observed in the system log; refer to Example 12-22 for details.

Example 12-22 Start member register member-specific DVIPA

```
DSN3029I  -DDW4 DSN3RRSR RRS ATTACH PROCESSING IS AVAILABLE
EZD1204I DYNAMIC VIPA 192.168.70.105 WAS CREATED USING IOCTL BY DDW4DIST
 ON TCPIP
DSNL519I  -DDW4 DSNLILNR TCP/IP SERVICES AVAILABLE 275
           FOR DOMAIN DDW4 AND PORT 6120
DSNL004I  -DDW4 DDF START COMPLETE 276
```

Now the member-specific DVIPA is reported on the server list and is able to accept connections; refer to Example 12-23 for details. Weights are provided by WLM based on the capacity of the LPAR; because DDW4 and DDW2 are running on the same LPAR, they share the same weight, as indicated in the example.

Example 12-23 Server list example

```
-DDW3 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:  280
 STATUS=STARTD
 LOCATION           LUNAME            GENERICLU
 DDWG               USIBMT6.DDFDDW3   -NONE
 TCPPORT=6120  SECPORT=6122  RESPORT=6141  IPNAME=-NO
 IPADDR=::192.168.70.101
 SQL    DOMAIN=DDW3
 RESYNC DOMAIN=DDW3
 ALIAS              PORT  SECPORT
 DDWGCOG            7120  0
 MEMBER IPADDR=::192.168.70.104
 DT=I  CONDBAT=   2000 MDBAT=  300
 ADBAT=    0 QUEDBAT=      0 INADBAT=      0 CONQUED=
 DSCDBAT=      0 INACONN=      0
 LOCATION SERVER LIST:
  WT IPADDR           IPADDR
  23 ::192.168.70.102
  18 ::192.168.70.103
  11 ::192.168.70.103
  11 ::192.168.70.105
```

Connections using the DB2 Location alias were being executed during the restart of DDW4. These attempts were routed to the other member on the alias, DDW3. There was a short period between messages DSNL519I and DSNL004I in which requests were sent to DDW4, because it was on the server list again, but this member was not capable of handling them. This is working as designed and all the connections will be handled after message DSNL004I. Refer to Example 12-24 for details.

Example 12-24 DDF startup sequence

```
EZD1204I DYNAMIC VIPA 192.168.70.105 WAS CREATED USING IOCTL BY DDW4DIST
 ON TCPIP
DSNL519I  -DDW4 DSNLILNR TCP/IP SERVICES AVAILABLE 306
           FOR DOMAIN DDW4 AND PORT 6120
DSNL030I  -DDW4 DSNLAGNT DDF PROCESSING FAILURE FOR 307
LUWID=C0A84608.P21D.000000000000
AUTHID=        , REASON=00D31032
DSNL030I  -DDW4 DSNLAGNT DDF PROCESSING FAILURE FOR 308
LUWID=C0A84608.P005.000000000000
AUTHID=        , REASON=00D31032
DSNL030I  -DDW4 DSNLAGNT DDF PROCESSING FAILURE FOR 309
```

```
LUWID=C0A84608.P21F.000000000000
AUTHID=        , REASON=00D31032
DSNL004I  -DDW4 DDF START COMPLETE 310
```

12.5.2 Installation of Data Server Runtime Client

This section describes the steps required for installing and configuring the IBM Data Server Runtime Client. This package is a superset of the Data Server Driver package. It includes many DB2-specific utilities and libraries, and includes the DB2 Command Line Processor (CLP) tool.

At the moment of writing, the latest client packages for IBM Data Servers can be downloaded from:

http://www-01.ibm.com/support/docview.wss?rs=4020&uid=swg21385217

Once the software is downloaded and transferred to the Linux on z server, it is uncompressed in an appropriate place using the **tar** command as shown in Example 12-25; the uncompressed software is stored in the folder `./rtcl`.

Example 12-25 Uncompressed Data Server Runtime Client installation software

```
ZDWL2:/opt/DataServer_code/rtc97 # tar -xvf
ibm_data_server_runtime_client_linux390x64_v97.tar
```

You have the option of installing the product using commands or a graphical interface if you have a working X Server in your workstation. For our installation, we used the free Xming server; for more details refer to:

http://sourceforge.net/projects/xming/

This method requires X forwarding traffic to be active; refer to the ssh documentation for more details.

Example 12-26 shows how we started the execution of the **db2setup** command.

Example 12-26 Starting the Data Server Runtime Client installation

```
ZDWL2:/opt/DataServer_code/rtcl/rtcl # ./db2setup
DBI1190I  db2setup is preparing the DB2 Setup wizard which will guide
      you through the program setup process. Please wait.
```

Figure 12-13 on page 312 shows the first panel of the installation process as shown by the X server on one of our workstations.

Archived

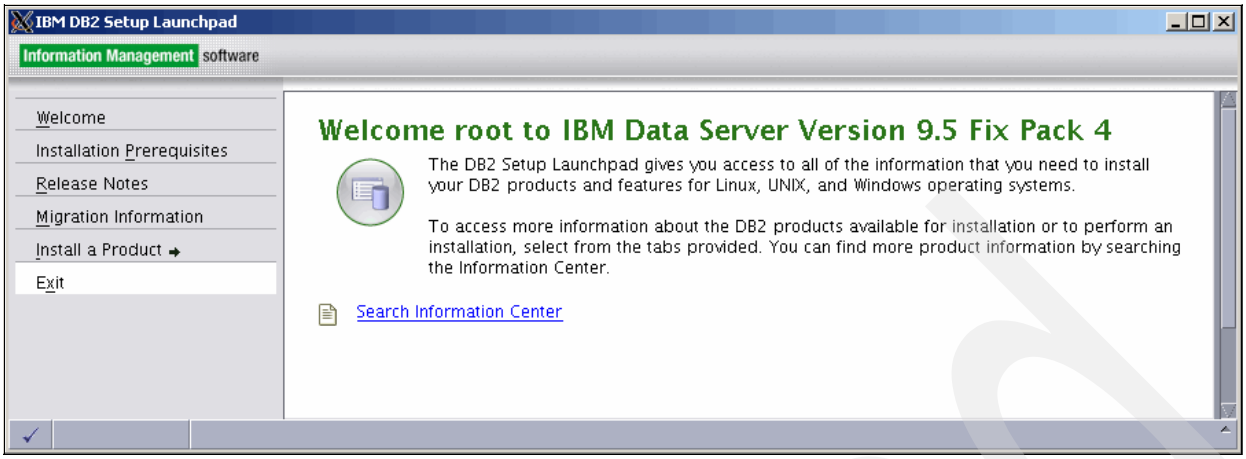


Figure 12-13 Initial panel of the DS Runtime Client installation

Figure 12-14 shows one of the panels presented during the installation. This section does not include all of them; refer to the DS Client documentation for more details about the installation process.

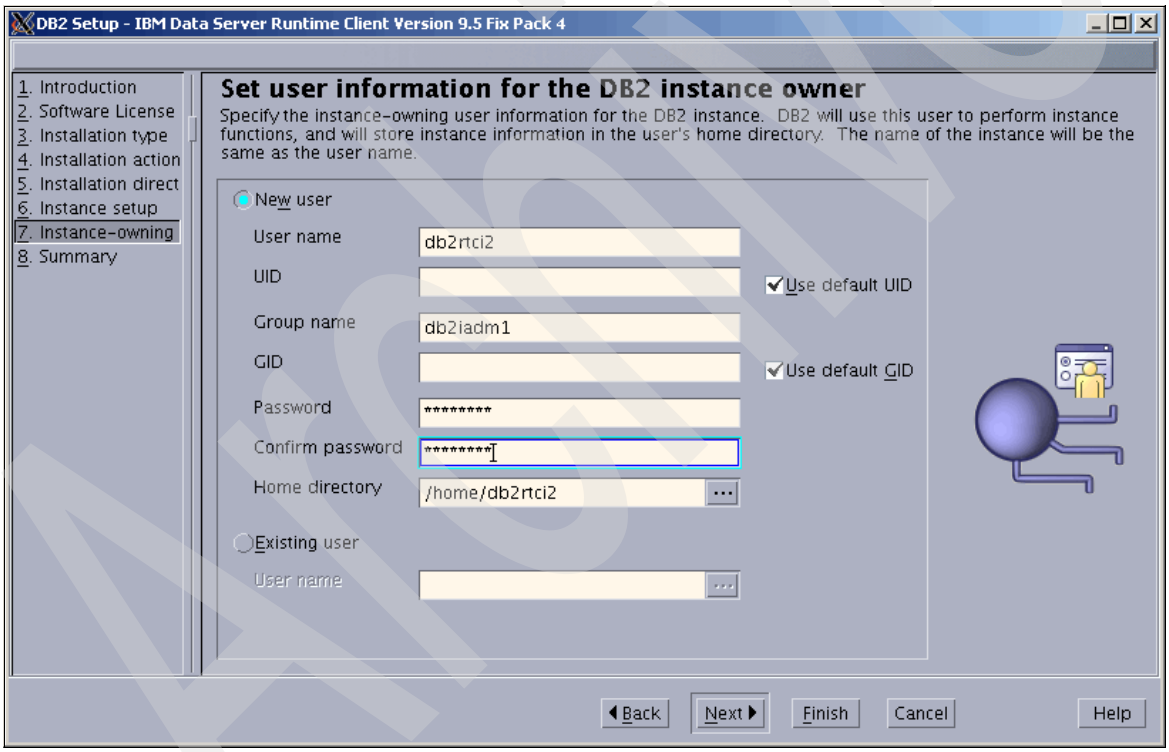


Figure 12-14 DS Runtime Client installation process: example

You should ensure that you have the correct license entitlements for DB2 products and features installed on this machine. Each DB2 product or feature comes with a license certificate file (also referred to as a license key) that is distributed on an Activation CD, which also includes instructions for applying the license file. If you purchased a base DB2 product, as well as separately priced features, you might need to install more than one license certificate. The Activation CD for your product or feature can be downloaded from Passport Advantage if it is not part of the physical media pack you received from IBM. For more

information on licensing, search the Information Center[1] using terms such as licensing or db2licm.

Refer to “What's New” in the DB2 Information Center[2] to learn about the new functions for DB2 9.5.

Verify that you have access to the DB2 Information Center based on the choices you made during this installation. If you performed a typical or a compact installation, verify that you can access the IBM website using the Internet. If you performed a custom installation, verify that you can access the DB2 Information Center location specified during the installation.

It is good practice to review the response file created at `/root/db2rtcl.rsp`. Additional information about response file installation is available in the DB2 documentation under “Installing DB2 using a response file”.

Logged in the system as the instance owner user, you can verify the installation level using the **`db2level`** command as shown in Example 12-27.

Example 12-27 Verification of software level and installation path with the db2level command

```
db2rtci2@ZDWL2:~/sqllib> db2level
DB21085I  Instance "db2rtci2" uses "64" bits and DB2 code release "SQL09054"
with level identifier "06050107".
Informational tokens are "DB2 v9.5.0.4", "s090429", "MI00305", and Fix Pack
"4".
Product is installed at "/opt/ibm/db2/rtcV9.5b".

db2rtci2@ZDWL2:~/sqllib>
```

12.5.3 Cataloging the data warehouse subsystem into the Data Server Client

The Data Server Runtime Client requires a DB2 Connect license in order to allow connections to a DB2 for z/OS server. You can add a license using the command **`db2licm`** as shown in Example 12-28.

Example 12-28 Adding a DB2 connect license

```
db2rtci2@ZDWL2:/opt/ibm/db2/FP4/db2conn> db2licm -a ./db2conpe.lic

LIC1402I  License added successfully.

LIC1426I  This product is now licensed for use as outlined in your License Agreement.  USE
OF THE PRODUCT CONSTITUTES ACCEPTANCE OF THE TERMS OF THE IBM LICENSE AGREEMENT, LOCATED IN
THE FOLLOWING DIRECTORY: "/opt/ibm/db2/rtcV9.5b/license/en_US.iso88591"
db2rtci2@ZDWL2:/opt/ibm/db2/FP4/db2conn>
```

You can also use the **`-l`** option of this command for verification; this is shown in Example 12-29.

Example 12-29 Verification of licenses using db2licm -l

```
db2rtci2@ZDWL2:/opt/ibm/db2/FP4/db2conn> db2licm -l
Product name:                     "DB2 Connect Personal Edition"
License type:                     "Client Device"
Expiry date:                      "Permanent"
Product identifier:               "db2conpe"
```

[1] http://publib.boulder.ibm.com/infocenter/db2luw/v9r5
[2] http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.wn.doc/doc/c0052035.html

```
Version information              "9.5"
```

We catalogued a node called DDWGCOG using the Sysplex Distributor DVIPA and DB2 Location alias port 7120 in order to be able to reach only the two members dedicated to the IBM Cognos BI workload. The node catalog commands can be seen in Example 12-30.

Example 12-30 Data Server runtime client catalog node command

```
db2rtci2@ZDWL2:/> db2 "catalog tcpip node DDWGCOG remote 192.168.70.101 server 7120 ostype
mvs"
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
```

Use the **list node directory** command, as shown in Example 12-31, for verifying the node settings.

Example 12-31 List node directory command example

```
db2rtci2@ZDWL2:/> db2 "list node directory"

 Node Directory

 Number of entries in the directory = 1

Node 1 entry:

 Node name                      = DDWGCOG
 Comment                        =
 Directory entry type           = LOCAL
 Protocol                       = TCPIP
 Hostname                       = 192.168.70.101
 Service name                   = 7120
```

After cataloguing the node, we catalogued the database DDWGCOG, DB2 Location Alias name, using the commands in Example 12-32.

Example 12-32 Data Server Runtime Client catalog database command

```
db2rtci2@ZDWL2:/> db2 "catalog database DDWGCOG as DDWGCOG at node DDWGCOG authentication
server"
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
```

Use the **list db directory** command for verification; see Example 12-33 for details.

Example 12-33 List db directory example

```
db2rtci2@ZDWL2:/> db2 "list db directory"

 System Database Directory

 Number of entries in the directory = 1

Database 1 entry:

 Database alias                       = DDWGCOG
 Database name                        = DDWGCOG
 Node name                            = DDWGCOG
```

```
 Database release level              = c.00
 Comment                             =
 Directory entry type                = Remote
 Authentication                      = SERVER
 Catalog database partition number   = -1
 Alternate server hostname           =
 Alternate server port number        =
```

Example 12-34 shows how you can verify a connection to DB2 from z/OS using the new definitions.

Example 12-34 Connection to DB2 using the new catalogued db

```
db2rtci2@ZDWL2:/> db2 "connect to DDWGCOG user res10"
Enter current password for res10:
   Database Connection Information

 Database server        = DB2 z/OS 9.1.5
 SQL authorization ID   = RES10
 Local database alias   = DDWGCOG
```

12.5.4 The db2dsdriver configuration file

You need to create or update the `db2dsdriver.cfg` configuration file in order to enable client side workload balancing and failover.

The configuration file `db2dsdriver.cfg` contains database directory information and client configuration parameters in a human-readable format. This file is an ASCII file containing various keywords and values that can be used to make a connection to a supported database through ODBC, CLI, .NET, OLE DB, or open source (PHP or Ruby) and the applications using the keywords. The keywords are associated with the database alias name, and affect all the applications that access the database. You can also use this configuration file to specify sysplex-related settings.

For more information: For more details, refer to the *db2dsdriver* configuration file at:

`http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.swg.im.dbclient.config.doc/doc/c0054555.html`

If you have an existing IBM data server client (IBM Data Server Runtime Client or IBM Data Server Client), you can copy the existing database directory information into the `db2dsdriver.cfg` configuration file using the **`db2dsdcfgfill`** command. Using this command, the configuration file is populated based on the contents of the local database directory, node directory, and Database Connection Services (DCS) directory of a specific database manager instance. An execution of this command can be seen in Example 12-35.

Example 12-35 Executing the db2dsdcfgfill command

```
db2rtci1@ZDWL2:~/sqllib/cfg> db2dsdcfgfill
SQL01535I  The db2dsdcfgfill utility successfully created the db2dsdriver.cfg configuration
file.
```

Example 12-36 on page 316 shows the `db2dsdriver.cfg` file used in this project.

Example 12-36 db2dsdriver.cfg file example used in the co-location infrastructure

```
<configuration>
  <DSN_Collection>
   <dsn alias="DDWGCOG" name="DDWGCOG" host="192.168.70.101" port="7120">
     <parameter name="Authentication" value="Server"/>
   </dsn>
     <dsn alias="DDWG" name="DDWG" host="192.168.70.101" port="6120">
     <parameter name="Authentication" value="Server"/>
   </dsn>
  </DSN_Collection>
<databases>
 <database name="DDWGCOG" host="192.168.70.101" port="7120">
  <parameter name="DisableAutoCommit" value="1"/>
  <wlb>
   <parameter name="enableWLB" value="true"/>
  </wlb>
  <acr>
   <parameter name="enableACR" value="true"/>
  </acr>
 </database>
 <database name="DDWG" host="192.168.70.101" port="6120">
  <parameter name="DisableAutoCommit" value="1"/>
  <wlb>
   <parameter name="enableWLB" value="true"/>
  </wlb>
  <acr>
  <parameter name="enableACR" value="true"/>
  </acr>
  </database>
  </databases>
  <parameters>
 <parameter name="CommProtocol" value="TCPIP"/>
 </parameters>
</configuration>
```

The WLB and ACR sections must be under the <databases> part of the file. Refer to the following section of the DB2 Information Center for more details about configuring client side transaction-level load balancing:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.qb.dbconn.doc/doc/t0054740.html

12.5.5 How to verify that the Client WLB is working

The process involved in getting client side WLB is prone to error and there is no available tool, at the moment of writing, that can be used for validating your current settings. In addition, errors in the configuration file are silently ignored.

As a guidance you can use the messages in the `db2diag.log` file. Example 12-37 shows how the file provides feedback about WLB.

Example 12-37 db2diag.log showing that sysplex WLB is enabled

```
2009-11-19-20.27.23.067176-300 I37477C302         LEVEL: Info
PID     : 2464                 TID  : 1778351024  PROC : BIBusTKServerMain
INSTANCE: db2rtci1             NODE : 000
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrParseSrvlst, probe:10
MESSAGE : Sysplex Workload Balancing is enabled.
```

Archived

Important: Syntax errors in the db2dsdriver.cfg file are ignored. If you want to make sure that your settings are getting picked up, update your database configuration using DIAGLEVEL 4 through CLP (or manually update db2cli.ini if using the thin data server drivers where CLP is not available) and check db2diag.log for error messages.

Example 12-38 shows how to verify the current DIAGLEVEL settings.

Example 12-38 Verify the current DIAGLEVEL parameter

```
db2rtci1@ZDWL2:~/sqllib/cfg> db2 get dbm cfg | grep -i diag
 Diagnostic error capture level              (DIAGLEVEL) = 3
 Diagnostic data directory path               (DIAGPATH) = /home/db2rtci1/sqllib/db2dump
```

In order to change the DIAGLEVEL settings to 4, you can use the commands shown in Example 12-39.

Example 12-39 Update the DIAGLEVEL parameter

```
db2rtci1@ZDWL2:~/sqllib/cfg> db2 update dbm cfg using diaglevel 4
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
```

Changes to DIAGLEVEL are reported in the db2diag.log file, as shown in Example 14-37.

Example 12-40 db2diag.log feedback on changing the DIAGLEVEL parameter

```
FUNCTION: DB2 UDB, config/install, sqlfLogUpdateCfgParam, probe:30
CHANGE  : CFG DBM: "Diaglevel" From: "3"  To: "4"
```

Example 12-41 shows how the contents of the db2dsdriver.cfg file are reported in the log.

Example 12-41 db2diag.log reporting db2dsdriver.cfg settings

```
2009-11-19-20.27.23.062499-300 I34717C1308       LEVEL: Info
PID     : 2464                 TID  : 1778351024  PROC : BIBusTKServerMain
INSTANCE: db2rtci1             NODE : 000
FUNCTION: DB2 UDB, Remote Client Config API, rccConfig::getInstance, probe:75
DATA #1 : String, 1022 bytes
<configuration>
  <DSN_Collection>
    <dsn alias="DDWGCOG" name="DDWGCOG" host="192.168.70.101" port="7120">
      <parameter name="Authentication" value="Server"/>
    </dsn>
    <dsn alias="DDWG" name="DDWG" host="192.168.70.101" port="6120">
      <parameter name="Authentication" value="Server"/>
    </dsn>
  </DSN_Collection>
  <databases>
    <database name="DDWGCOG" host="192.168.70.101" port="7120">
      <parameter name="DisableAutoCommit" value="1"/>
      <WLB>
      <parameter name="enableWLB" value="true"/>
      </WLB>
      <ACR>
      <parameter name="enableACR" value="true"/>
      </ACR>
    </database>
    <database name="DDWG" host="192.168.70.101" port="6120">
```

```
      <parameter name="DisableAutoCommit" value="1"/>
      <WLB>
      <parameter name="enableWLB" value="true"/>
      </WLB>
      <ACR>
      <parameter name="enableACR" value="true"/>
      </ACR>
    </database>
  </databases>
  <parameters>
    <parameter name="CommProtocol" value="TCPIP"/>
  </parameters>
</configuration>
```

Once client side WLB is enabled, you can get server list information in the db2diag.log file when working using DIAGLEVEL 4. You can use this information for verification of WLB being activated. Refer to Example 12-42 for an illustration.

Example 12-42 db2diag.log showing server list information

```
2009-11-19-10.18.36.145313-300 I49094C695        LEVEL: Info
PID     : 9996                 TID  : 1794984880  PROC : BIBusTKServerMain
INSTANCE: db2rtci1             NODE : 000
APPID   : 192.168.70.8.57161.091119151821
FUNCTION: DB2 UDB, DRDA Application Requester, sqljrDumpTransportPool, probe:0
DATA #1 : <preformatted>
DbName=DDWGCOG           , TotTpts=1, InUse=0, InWait=0, Caller=1 , Conn=0x62ab5b00
  Host=192.168.70.104, Port=7120, TotTpts=1, InUse=0 Weight=66, ratio=0.519685, totCon=1
    Host=192.168.70.104, Port=7120, InUse=0, LCnt=0, MCnt=0, HCnt=1, Conn=0x62ab5b00,
Ctx=0x62ad42b0, tid=1
  Host=192.168.70.105, Port=7120, TotTpts=0, InUse=0 Weight=61, ratio=0.480315, totCon=
```

Another source of analysis information are the DRDA traces. Before analyzing DRDA traces, you must understand that DRDA is an open standard for the definition of data and communication structures. For example, DRDA comprises a set of rules about how data should be organized for transmission and how communication of that information should occur.

DRDA uses distributed data management as a methodology that allows data stored in a relational database system to be accessed by another system. This requires a consistent set of protocols among different database vendors. Because database vendors store and represent data differently, they are unable to share data in heterogeneous database environments. Without DDM, each implementation must be written for each different relational database. What these systems need is a methodology they can all use. And that is exactly what the DDM architecture provides.

The DDM architecture makes the sharing and accessing of data between computer systems possible by providing a common language and a set of rules that enable different systems to communicate and share data.

The DDM architecture can be broken down into small, well-defined, and standardized units called objects. Objects are data processing entities. All objects are composed of a contiguous string of bytes with a specific format that describes the object. For the purposes of this section and accepting a great degree of simplification, it is enough to consider that objects are identified by code points. Code points are 2-byte hexadecimal values that indicate where the class description of the object can be found in a dictionary that is also part of the DRDA standards.

For a more detailed description, refer to the following publications:

- *"DRDA, Version 4, Volume 3: Distributed Data Management (DDM) Architecture, The Open Group"* available free of charge at:

  http://www.opengroup.org

- *DB2 9 for z/OS: Distributed Functions*:

  http://www.redbooks.ibm.com/abstracts/SG246952.html

Archived

Archived

13

Utilizing DB2 client information for resource management and monitoring

In this chapter we provide the following information:

- What is DB2 client information and why is it valuable in a data warehouse environment
- What are the implementation options for passing the DB2 client information
- Leveraging the WLM set client information option for an IBM Cognos BI reporting application
- The sequence of steps in flowing DB2 client information on a DB2 thread
- Snapshots of common Resource Management Facility (RMF) and Tivoli monitors and reports exhibiting the value of DB2 client information

Archived

© Copyright IBM Corp. 2010. All rights reserved.

13.1 DB2 client information

It is very likely for Data Warehousing (DW) and Business Intelligence (BI) environments to provide a graphical user interface for reporting and analyzing data that is stored in a data warehouse, especially since using a native 3270 panel does not provide the capabilities of modern BI tools to visualize all kinds of different result sets. Therefore, the vast majority of queries in those IT architectures originate from many different platforms and applications, utilizing tools such as IBM Cognos BI for your convenience. Additionally, the majority of queries being issued in those environments predominantly use dynamic SQL, since all possible combinations of reporting requests against a data warehouse are simply unforeseeable to include your applications using static SQL.

For static SQL, isolating problem queries is less an issue since all the relevant information required to locate the package in trouble can be obtained by the detailed thread information, like the name of plan being executed, the correlation name and the primary authorization ID. For dynamic queries issued on distributed platforms, the plan can be listed as DISTSERV, only telling you that the request causing performance problems came in from a distributed platform. As additional information, the correlation ID can contain information about the program executing on the client side (for example, db2bp.exe for the DB2 Command Line Processor).

Therefore, when dealing with distributed dynamic SQL statements causing high resource consumption, identifying the details of the environment issuing this query could become a time-consuming task. Setting client information for physical workstations only partially solves the problem since this information does not provide details about a specific user if generic application user IDs are used. While environments using static SQL can be well tuned and unpredictable resource consumptions typically are less of a problem, the challenge persists for environments with a large number of users firing dynamic SQL at the database server.

Managing workloads in large mixed-user environments is one of the strengths of the System z architecture. Concepts such as period aging can keep short-running queries short by preventing the system from being monopolized by queries that come with the flavor of a large resource consumption. Since this concept has worked very well for queries whose originators could be assigned attributes that are known in z/OS WLM, workloads originating on distributed platforms still could cause problems in assigning the required priority inside of z/OS. Distributed threads have not been able to maximize the potential WLM capabilities provided by z/OS, mostly since there was not enough information available to differentiate the threads.

The solution to address both challenges is the introduction of new special registers that can provide information about a distributed request. The additional data you can provide via the special registers is as follows:

- CURRENT CLIENT_USERID (contains the user ID for the client)

  In a distributed environment, if the value set by the API exceeds 16 bytes, it is truncated to 16 bytes. If one of these APIs is not used to set the value of the special register, an empty string is returned when the special register is referenced.

- CURRENT CLIENT_WRKSTNNAME (contains the workstation name for the client)

  In a distributed environment, if the value set by the API exceeds 18 bytes, it is truncated to 18 bytes. If one of these APIs is not used to set the value of the special register, an empty string is returned when the special register is referenced.

- CURRENT CLIENT_APPLNAME (contains the application name for the client)

  In a distributed environment, if the value set by the API exceeds 32 bytes, it is truncated to 32 bytes. If one of these APIs is not used to set the value of the special register, an empty string is returned when the special register is referenced.
- CURRENT CLIENT_ACCTNG (contains the accounting string for the client)

  In a distributed environment, if the value set by the API exceeds 200 bytes, it is truncated to 200 bytes. If one of these APIs is not used to set the value of the special register, an empty string is returned when the special register is referenced.

The value of the special registers can be changed by using one of the following application programming interfaces (APIs):

- DB2Connection.setDB2ClientUser(String info) (JDBC)
- The WLM_SET_CLIENT_INFO stored procedure
- Set Client Information (sqleseti)
- The RRS DSNRLI SIGNON, AUTH SIGNON, CONTEXT SIGNON, or SET_CLIENT_ID function

Next we discuss two of these implementations.

13.1.1 Using JDBC Universal Driver methods

One way of passing information about a client request to set the special registers mentioned above is to use Java Database Connectivity (JDBC) universal driver methods.

Providing the attributes using JDBC does not require a lot of effort. The client registers can be set as shown in Table 13-1.

Table 13-1 Setting client information attributes using JDBC Universal Driver methods

Special register	Set client register via
CLIENT_USERID	Void DB2Connection.setDB2ClientUser(String userid)
CLIENT_WRKSTATION	Void DB2Connection.setDB2ClientWorkstation(String workstation)
CLIENT_APPLICATION	Void DB2Connection.setDB2ClientApplicationInformation(String application)
CLIENT_APPLICATION_ACCOUNTING	Void DB2Connection.setDB2ClientAccountingInformation(String application_accounting)

All information, except for the accounting string, is displayed on the DISPLAY THREAD command as well, as we have shown in Example 13-4 on page 325.

For any results on where these values are shown in any reporting, refer to the examples we provide in 13.1.2, “Using the WLM_SET_CLIENT_INFO stored procedure”.

13.1.2 Using the WLM_SET_CLIENT_INFO stored procedure

The stored procedure WLM_SET_CLIENT_INFO is made available in DB2 on all IBM platforms. It appeared in DB2 for z/OS Version 8. It resides in schema SYSPROC and allows you to pass further information to the DB2 for z/OS server instance to incorporate additional data in monitoring outputs as well as to incorporate its attributes in any WLM policies (see “Leveraging WLM set client information in IBM Cognos BI” on page 329 for further information on how to combine the information provided by WLM_SET_CLIENT_INFO and WLM).

The registers cannot be set by accessing the special registers using the SET command. However, the value of these registers can be retrieved in the same way as a value for other special registers such as CURRENT TIMESTAMP can be retrieved.

Note: The existing behavior of the CLIENT_ACCTNG register is unchanged. It will continue to get its value from the accounting token for DSN requesters, and from the accounting string for SQL and other requesters.

The values of the information passed to the DB2 for z/OS server can be viewed in the output of an accounting trace. If the special registers mentioned above have not been set in any way, the information listed in the IDENTIFICATION section of an accounting report contains the value BLANK for fields ENDUSER, TRANSACT, WSNAME. and DDCS ACC.SUFFIX in the JDBC driver correlation information. See Example 13-1 for an accounting report where these values have not been set before.

Example 13-1 Accounting report information without invoking WLM_SET_CLIENT_INFO

```
---- IDENTIFICATION--------------------------------------------------------------------------------------------
 ACCT TSTAMP: 10/19/09 18:03:11.29    PLANNAME: ADB           WLM SCL: 'BLANK'         CICS NET: N/A
 BEGIN TIME : 10/19/09 18:02:58.38    PROD TYP: N/P                                    CICS LUN: N/A
 END TIME   : 10/19/09 18:03:11.29    PROD VER: N/P           LUW NET: DEIBMIPS        CICS INS: N/A
 REQUESTER  : DWHD913                 CORRNAME: PBECKER       LUW LUN: IPWAS913
 MAINPACK   : ADB2CON                 CORRNMBR: 'BLANK'       LUW INS: C4F5DD7030A5    ENDUSER : 'BLANK'
 PRIMAUTH   : PBECKER                 CONNTYPE: TSO           LUW SEQ:            1    TRANSACT: 'BLANK'
 ORIGAUTH   : PBECKER                 CONNECT : TSO                                    WSNAME  : 'BLANK'

[...]

---- INITIAL DB2 COMMON SERVER OR UNIVERSAL JDBC DRIVER CORRELATION -------------------------------------------
 PRODUCT ID     : COMMON SERV
 PRODUCT VERSION: V9 R1 M0
 CLIENT PLATFORM: NT
 CLIENT APPLNAME: app.exe
 CLIENT AUTHID  : RES2
 DDCS ACC.SUFFIX: 'BLANK'
```

To populate these values, the WLM_SET_CLIENT_INFO stored procedure can be invoked from the client side using the syntax shown in Example 13-2.

Example 13-2 Syntax for invoking the stored procedure WLM_SET_CLIENT_INFO

```
call sysproc.wlm_set_client_info (client_userid, client_wrkstation,
client_application, client_application_accounting)
```

The variables being used during the invocation have the following attributes[1]:

```
CLIENT_USERID                    VARCHAR (255)
CLIENT_WRKSTATION                VARCHAR (255)
CLIENT_APPLICATION               VARCHAR (255)
CLIENT_APPLICATION_ACCOUNTING    VARCHAR (255)
```

[1] For actual truncation values, see 13.1, “DB2 client information”.

Example 13-3 exhibits the output of an accounting report if WLM_SET_CLIENT_INFO has been invoked using the values PBECKER as client_userid, MYMACHINE as client_wrkstation, MYTRANS as client_application, and MYACCOUNT as client_application_accounting.

Example 13-3 Accounting report information after invoking WLM_SET_CLIENT_INFO

```
---- IDENTIFICATION-------------------------------------------------------------------------------------------
 ACCT TSTAMP: 10/17/09 22:49:54.21    PLANNAME: db2bp.ex        WLM SCL: STCCMD          CICS NET: N/A
 BEGIN TIME : 10/17/09 22:49:51.51    PROD TYP: COMMON SERV                              CICS LUN: N/A
 END TIME   : 10/17/09 22:49:54.21    PROD VER: V9 R1 MO        LUW NET: G93830B6        CICS INS: N/A
 REQUESTER  : ::FFFF:9.56.48#1        CORRNAME: db2bp.ex        LUW LUN: G73C
 MAINPACK   : SQLC2F0A                CORRNMBR: e               LUW INS: C4F398FE47F3    ENDUSER : PBECKER
 PRIMAUTH   : RES2                    CONNTYPE: DRDA            LUW SEQ:           2     TRANSACT: MYTRANS
 ORIGAUTH   : RES2                    CONNECT : SERVER                                   WSNAME  : MYMACHINE

[...]

---- INITIAL DB2 COMMON SERVER OR UNIVERSAL JDBC DRIVER CORRELATION ------------------------------------------
 PRODUCT ID     : COMMON SERV
 PRODUCT VERSION: V9 R1 MO
 CLIENT PLATFORM: NT
 CLIENT APPLNAME: db2bp.exe
 CLIENT AUTHID  : RES2
 DDCS ACC.SUFFIX: MYACCOUNT
```

Note that in an accounting report, the values are truncated as follows:

- ENDUSER is truncated after 16 characters.
- TRANSACT is truncated after 32 characters.
- WSNAME is truncated after 18 characters.
- ACC.SUFFIX is truncated after 22 characters.

The same information is also available in the output of a DISPLAY THREAD COMMAND as we show in Example 13-4.

Example 13-4 Output of DISPLAY THREAD(*) command

```
DSNV401I  -D913 DISPLAY THREAD REPORT FOLLOWS -
 DSNV402I  -D913 ACTIVE THREADS -
 NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
 SERVER   RA *   110 db2bp.exe    PBECKER  DISTSERV 008D    75
  V437-WORKSTATION=MYMACHINE, USERID=PBECKER,
       APPLICATION NAME=MYTRANS
  V442-CRTKN=9.57.156.163.58896.091102210830
  V445-G9399CA3.HOE6.C507A105CE89=75 ACCESSING DATA FOR
   ::FFFF:9.57.156.163
         PT *     0 db2bp.exe    PBECKER  DISTSERV 0084    81
         PT *     0 db2bp.exe    PBECKER  DISTSERV 0084    78
         PT *     0 db2bp.exe    PBECKER  DISTSERV 0084    77
         PT *     0 db2bp.exe    PBECKER  DISTSERV 0084    76
 TSO      T  *     3 PBECKER      PBECKER           0094    82
 DISPLAY ACTIVE REPORT COMPLETE
 DSN9022I  -D913 DSNVDT '-DIS THREAD' NORMAL COMPLETION
 ***
```

The WLM_SET_CLIENT_INFO procedure is another way to set the CLIENT INFO registers, instead of using the client-specific APIs. It provides more flexibility to the IBM Cognos (version 8.x and higher) Business Intelligence administrator and the application programmer.

In determining what variables are passed into the four fields, we suggest collaboration between the BI, DBA and z/OS administrators. This ensures that expectations are consistent

for the various potential resource monitors, managers, and accounting tools, keying off these fields.

Note: The stored procedure WLM_SET_CLIENT_INFO has been made available via APAR PK74330. The PTF for DB2 for z/OS Version 8 is UK46551. For DB2 9 for z/OS refer to UK46552. In DB2 for z/OS Version 8, WLM_SET_CLIENT_INFO requires new-function mode. Additionally, APAR PK93084 should be applied, which rectifies the security property setting of the initial WLM_SET_CLIENT_INFO stored procedure.

13.2 Mapping of special registers to DB2 and WLM

As previously discussed, the special register values are passed into multiple products. Table 13-2 depicts how they are mapped for being set by either the sqleseti API or the WLM set client information stored procedure.

Table 13-2 Special register mappings

sqleseti - Set client information	SQL Special Registers	Stored Procedure Parameter	DB2 Accounting Data Fields	WLM Class.
SQL_ATTR_INFO_ACCTSTR	CLIENT_ACCTNG	CLIENT_APPLICATION_ACCOUNTING	QMDAAINFO - accounting string	AI
SQL_ATTR_INFO_APPLNAME	CLIENT_APPLNAME	CLIENT_APPLICATION	QWHCEUTX - end user transaction name	PC
SQL_ATTR_INFO_USERID	CLIENT_USERID	CLIENT_USERID	QWHEUID - end user id	SPM
SQL_ATTR_INFO_WRKSTNNAME	CLIENT_WRKSTNNAME	CLIENT_WRKSTATION	QWHCEUW- end user workstation name	SPM

Attention: The SQL special registers have names to reflect their historical and anticipated role. When using WLM set client information stored procedure to set these registers, the BI administrator has flexibility to pass values that might not necessarily reflect the original name. While this could be advantageous, it must be clearly understood by all parties or products exploiting the resulting accounting and WLM data. For example, Optim products have their own expectations for the contents of these fields.

13.3 WLM_SET_CLIENT_INFO in IBM Cognos BI

IBM Cognos BI administrators can use the **`connection`** command blocks for calling DB2's WLM_SET_CLIENT_INFO stored procedure.

IBM Cognos BI command blocks and connection pooling

Ever since IBM Cognos BI Version 8.1, a DB2 database connection has provided the option to define blocks of statements that would be executed whenever IBM Cognos BI has to attach to DB2. Each data source connection defined in the IBM Cognos BI content store can have one or more connections associated with it. An IBM Cognos BI administrator can define a DB2 CLI connection that includes connection command blocks. Access to connections can be controlled by security settings for groups or specific users. A command block of

statements could also include calling the stored procedure WLM_SET_CLIENT_INFO and populating the required values as variables.

An instance of IBM Cognos BI Report Server maintains its connections in a connection pool. If a user session needs access to a database in order to run a report, the user session is associated with a connection from the pool. A connection manager returns a proxy to the “real” physical connection and the report is run in this context.

Figure 13-1 shows the flow for a session attach sequence. Whenever a new session is associated, “Run open session command block” is executed. Because the stored procedure is invoked here, execution for a report always has the correct WLM context set.

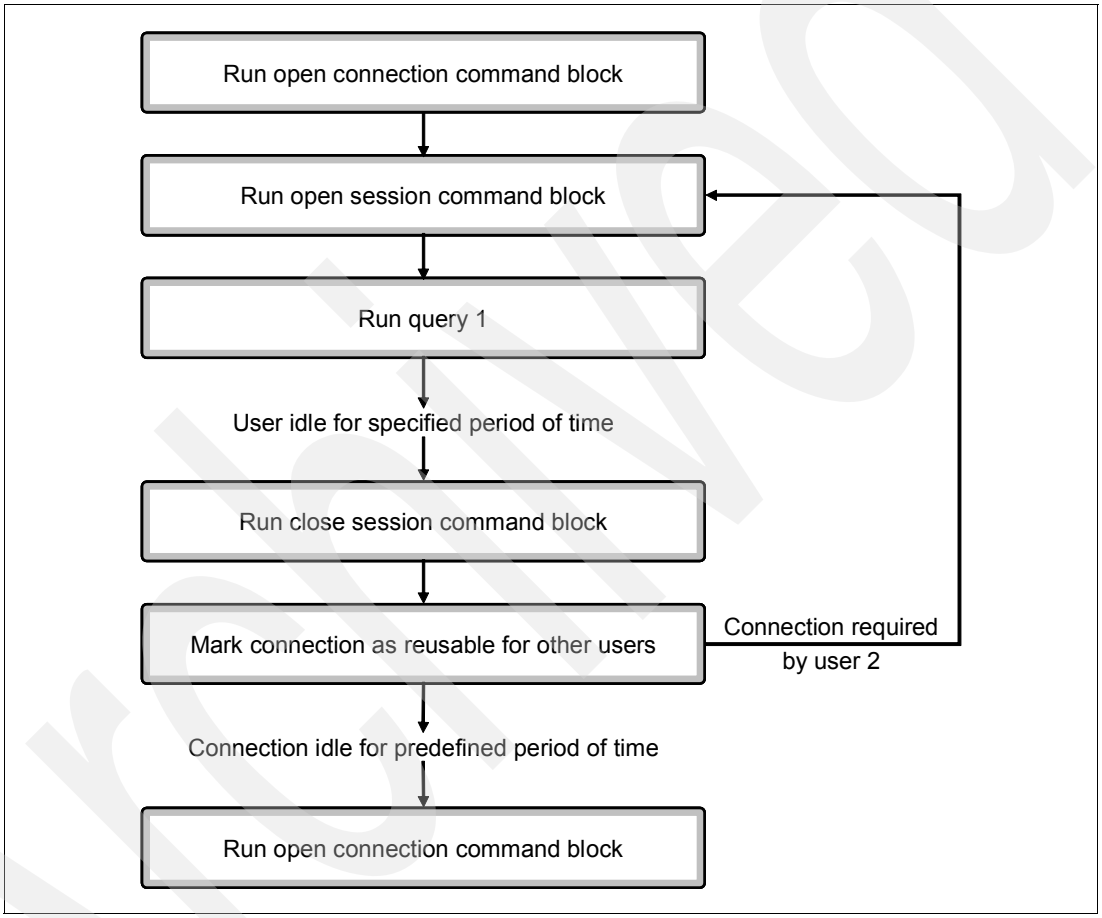


Figure 13-1 IBM Cognos BI command blocks and connection pooling

This execution sequence can guarantee that each IBM Cognos BI report will be associated with any environment variables being set within the open session command block. Here is where we exploit the WLM_SET_CLIENT_INFO, which we discuss in more detail soon.

In Figure 13-2 on page 328 we exhibit the sequence of DB2 client information flowing via WLM SCI, from IBM Cognos BI to the IBM Data Server Driver to DB2 and configured without auto commit.

Figure 13-2 on page 328 shows the sequence of connections from IBM Cognos BI to DB2 on z/OS and the way the special registers are passed to DB2 accounting.

Archive

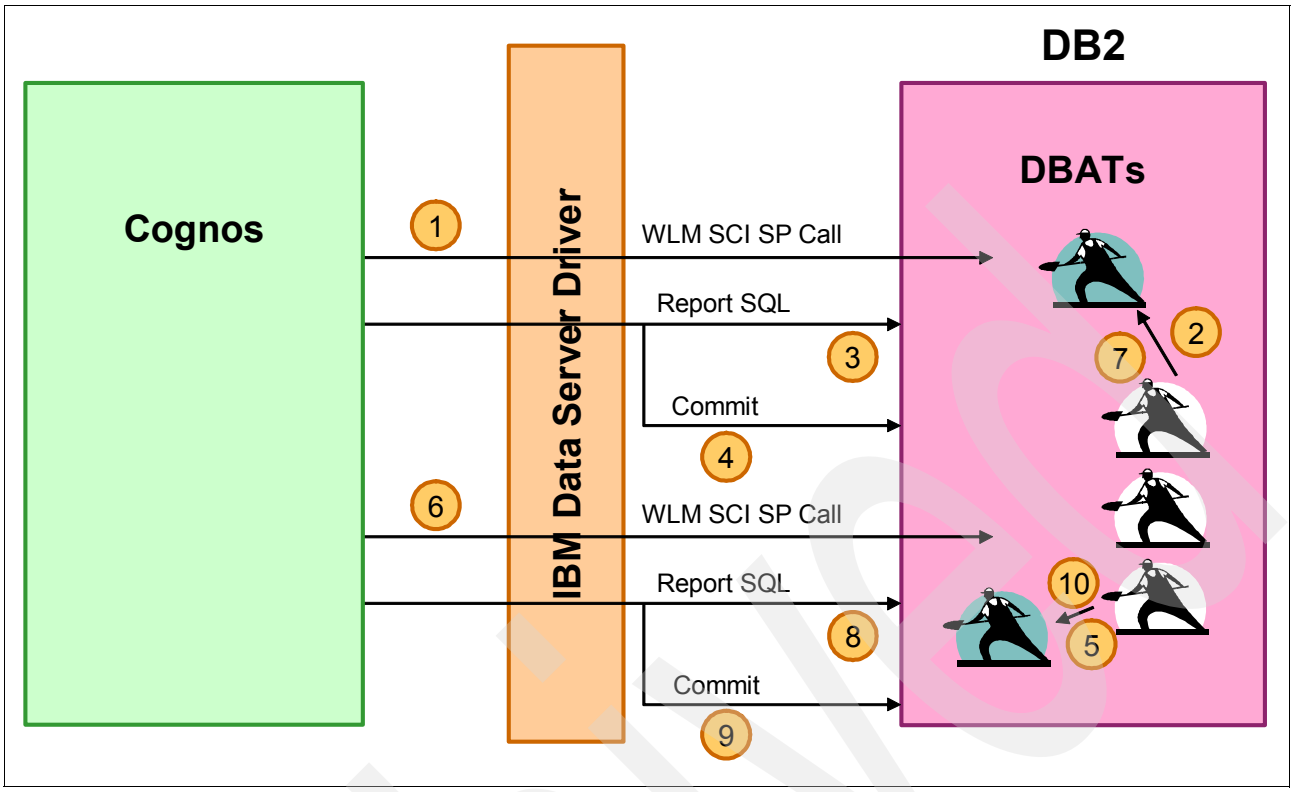


Figure 13-2 IBM Cognos BI, Data Server Drivers, DB2 DBAT & WLM_SCI without auto commit

1. At the start of a report, the wlm_sci stored procedure is called.
2. A DBAT is assigned and the special registers are set accordingly.
3. The report starts; SQL queries are run accordingly with the special registers.
4. At the end of the report, IBM Cognos BI does a commit.
5. It frees up the DBAT, registers are reset, and the DBAT goes back to the pool.
6. A new request and report comes in.
7. A new DBAT is allocated when the WLM_SCI stored procedure is called.
8. The report runs.
9. IBM Cognos BI does the commit at the end of the report.
10. The DBAT is freed.

IBM Cognos BI session variables and macros

IBM Cognos BI maintains variables and macros related to the context of a given session. They are called IBM Cognos BI *session variables* and *macro* functions. Some of the session variables are based on data coming from authentication providers. Cognos supports NTLM, Active Directory, LDAP, and SAP, among others. Table 13-3 provides samples of these, along with sample contents.

Table 13-3 Sample of IBM Cognos BI session variables

IBM Cognos BI maintained variables	Authentication provider mapping	Sample value settings
HTTP_HOST	N/A	9.12.43.13:9380

IBM Cognos BI maintained variables	Authentication provider mapping	Sample value settings
machine	N/A	ZDWL2
account.defaultName	uid	usera40
account.personalInfo.surname	sn	usera40
account.personalInfo.userName	cn	usera40
SERVER_NAME	N/A	workstation ip @
modelPath	N/A	Path and package name
report	N/A	IBM Cognos BI report name

In this table, in the LDAP directory, the user usera40 has the same information stored in all the fields (uid, sn, and cn).

13.4 Leveraging WLM set client information in IBM Cognos BI

In this section we provide a sample scenario of using the client information for z/OS and DB2 resource management and monitoring. We provide the following three steps:

1. Background and objective - Sample scenario. What are we trying to achieve?
2. IBM Cognos BI command block implementation of WLM set client information
3. Associated z/OS WLM classification Implementation

13.4.1 Background and objective

Our simple sample BookStore company has the following three groups of data warehouse query users. The following description provides the background for our need to differentiate them.

Key knowledge workers

Consists of the company's most important proven knowledge workers, responsible for both strategic and tactical business decisions based on analysis of data. These users will all be designated user IDs that start with "userK". We want all IBM Cognos BI DB2 work requests initiated by these users to be managed in our high importance WLM service class DWDDFHI, and we want to report on them using WLM report class RCOGCRIT.

Intermediate and novice knowledge workers

Consists of the company's newer knowledge workers, responsible for tactical business decisions. These users all have user IDs that start with "userM". We want all IBM Cognos BI DB2 work requests initiated by these users to be managed in our medium importance WLM service class DWDDFMD, and we want to report on them using WLM report class RCOGUSER.

Service center representatives

Consists of the company's service center representatives that are interacting with customers regarding purchasing or billing inquiries, and are the company's predominant Operational BI users. These users will all have user IDs that start with "userS". We want all IBM Cognos BI DB2 work requests initiated by these users to be managed in our high importance WLM service class DWDDFHI, and we want to report on them via WLM report class RCOGSERV.

Additionally, we have two unique IBM Cognos user IDs, designated for scheduled batch reporting. These are more like proxy user IDs, as opposed to real users. The potential usage here is relative to regular scheduled reporting that occurs directly after a data warehouse refresh. The user IDs and desired WLM classes are as follows:

- High importance scheduled reporting: user = “SCHEDHI”. This work is to be managed in the DWSCHDHI service class, and reported in the RCSCHDHI report class.
- Medium importance scheduled reporting: user = “SCHEDMD”. This work is to be managed in the DWSCHDMD service class, and reported in the RCSCHDMD report class.

13.4.2 IBM Cognos BI command blocks to call WLM set client information

In using the IBM Cognos BI Version 8.4 command block facility to call WLM’s set client information (SCI) stored procedure, we followed a process documented in “Passing IBM Cognos BI Context to a Database” of the following IBM Cognos BI infocenter documentation:

http://publib.boulder.ibm.com/infocenter/c8bi/v8r4m0/index.jsp?topic=/com.ibm.swg.im.cognos.rdm_crn.8.4.1.doc/rdm_crn_i_PassingCognosContext.html

The following steps were taken:

1. Launch the IBM Cognos Administration tool.
2. Select **Configuration** and an appropriate data source.
3. Within the data source, select **Properties**, for the appropriate database connection.
4. Open up the **Commands** list.
5. Set the Open session commands “command block”. (Refer to Example 13-5 for the XML we utilized.)

When you define a connection via the wizard, you have the option to “Set the commands”. We edited the “Open session commands” command block, as shown in Example 13-5.

Example 13-5 Calling WLM set client info stored procedure from an IBM Cognos BI command block

```
<commandBlock>
  <commands>
    <sqlCommand>
      <sql> CALL SYSPROC.WLM_SET_CLIENT_INFO(#sq($account.defaultName)#,#sq($SERVER_NAME)
#.#sq(modelPath)#,#sq($report)#
      </sql>
    </sqlCommand>
  </commands>
</commandBlock>
```

We used IBM Cognos BI session variables to pass in values associated with the current session work request.

To better clarify, here is an example of the data flow, for each of the WLM SCI stored procedure parameters, using IBM Cognos BI session variables:

CLIENT_USERID → $account.defaultName → ex. userK22

CLIENT_WRKSTNAME → $SERVER_NAME → ex. 9.13.15.21

CLIENT_APPLNAME → $modelPath → ex. /content/package @name=’PH3DW’ /

Note: PH3DW is the name of our BookStore IBM Cognos BI package.

CLIENT_ACCTSTR → $report → ex. Report8

Important: As previously mentioned, all four client variables have their specific size limits. Should the values pushed into these exceed those limits, one must understand the implications of the truncation. In our implementation, all our session variable values were within the size limits. It is quite possible that customers will have longer package, report, and model names.

13.4.3 WLM classification using DB2 client information

In this section we describe our z/OS WLM classification setup to exploit the client information that we just set. Let us first recall how the SQL special client registers map to the WLM classification qualifiers; see Table 13-4.

Table 13-4 Mapping of SQL special registers to WLM qualifiers

SQL Special Registers	WLM Classification qualifiers
CLIENT_USERID	SPM
CLIENT_WRKSTNNAME	SPM
CLIENT_APPLNAME	PC
CLIENT_ACCTNG	AI

Note: In DB2 V8 and beyond, CLIENT_USERID is concatenated with CLIENT_WRKSTNNAME, completing the string passed to the “SPM” WLM qualifier.

All DB2 DRDA work requests automatically qualify into WHELM’s DDF (Distributed Data Facility) subsystem. Then user-defined rules are utilized to further qualify the requests into Service classes and report classes. Figure 13-3 is an excerpt of our WLM classification rules for the DDF WLM subsystem.

```
* Subsystem Type DDF - Rules for DDF (Transaction only)

  Classification:

    Default service class is DDFDFLT
    Default report class is RDDFDFLT

      Qualifier  Qualifier       Starting       Service  Report
    # type       name            position       Class    Class
    - ---------- --------------- ---------      -------- --------
    1 SI         DWPR                           DWDDFMD  RDWPDFLT
    2 . UI       . COGNOS1                      DWDDFMD  RCOGPDFL
    3 . . PC     . . PH3DW           25         DWDDFMD  RCBKPKG
    4 . . . SPM  . . . userK*                   DWDDFHI  RCBKCRIT
    4 . . . SPM  . . . userS*                   DWDDFHI  RCBKSERV
    4 . . . SPM  . . . userM*                   DWDDFMD  RCBKMED
    4 . . . SPM  . . . SCHEDHI                  DWSCHDHI RCSCHDHI
    4 . . . SPM  . . . SCHEDMD                  DWSCHDMD RCSCHDMD
    2 . UI       . SQWREF                       DWDDFREF RSQWREF
    1 SI         DWTS                           DWDDFMD  RDWTSTDF
```

Figure 13-3 WLM DDF Classification rules exploiting DB2 client information

In this set of DDF classification rules, we utilize a combination of traditional qualifiers (SI - DB2 subsystem identifier, and UI - z/OS user ID used for connections), along with two of the DB2 client information qualifiers (PC and SPM). Incoming DDF work requests get filtered through these rules in a top-to-bottom manner. Here is a sample flow:

Note: Cognos1 is the user ID we utilized for our IBM Cognos BI data source connection to DB2.

Example 1: IBM Cognos user *userM34* submits a report against the IBM Cognos BI package PH3DW, whose source DB2 subsystem instance is *DWPR*.

Going top-to-bottom: At level 1, we have a match with the DB2 subsystem instance, hence we then compare against sublevel 2, where we have a match with the connection ID (DB2 AUTHID) for the IBM Cognos BI application. Hence, we move to sublevel 3, where we have a match with the IBM Cognos BI package being accessed, hence we then serially move through the level 4s until we find the match at *userM**. This work request classifies to service class DWDDFMD and report class RCBKMED.

Example 2: An IBM Cognos BI user submits a report against an IBM Cognos BI package, which is not listed in our rules, yet uses the same DB2 source subsystem (*DWPR*) and connection ID *Cognos1*). Since there are no matches at level 3 for *PH3DW*, this work request will get classified to the last matching level 2 (*Cognos1*), hence into DWDDFMD and RCOGPDFL (Report Class for IBM Cognos BI production default) classes.

Example 2 also exhibits the value of defining default report classes to highlight work in the system that potentially is currently not being managed or monitored appropriately.

Notice the “Starting position” entry of 25 for the IBM Cognos BI package name. As we learned earlier, the IBM Cognos BI $modelPath session variable contains the path and package name. The actual package name starts at the 25th character within this string. There are other means in WLM to filter longer strings like this, but we did not explore them here.

For more details about the WLM service class and other definitions relative to this workload, see 11.2, “WLM service definition with data warehouse workloads” on page 264.

13.5 Monitor views of work requests using DB2 client information

Here we examine some excerpts from common z/OS DB2 monitoring tools, for a given work request resulting from one of our IBM Cognos BI reports exploiting DB2 client information.

Figure 13-4 on page 333 relates the DB2 client information between the Tivoli Omegamon XE for DB2 Performance Expert and RMF Monitor III Enclave Classification Data monitors. The reader will also notice the correlation between some of the other traditional qualifiers. Notice also, the “Report8” in the *Accounting Information* section in the RMF monitor. This was as a result of passing IBM Cognos BI session variable $report into the WLM SCI stored procedure CLIENT_ACCTG field.

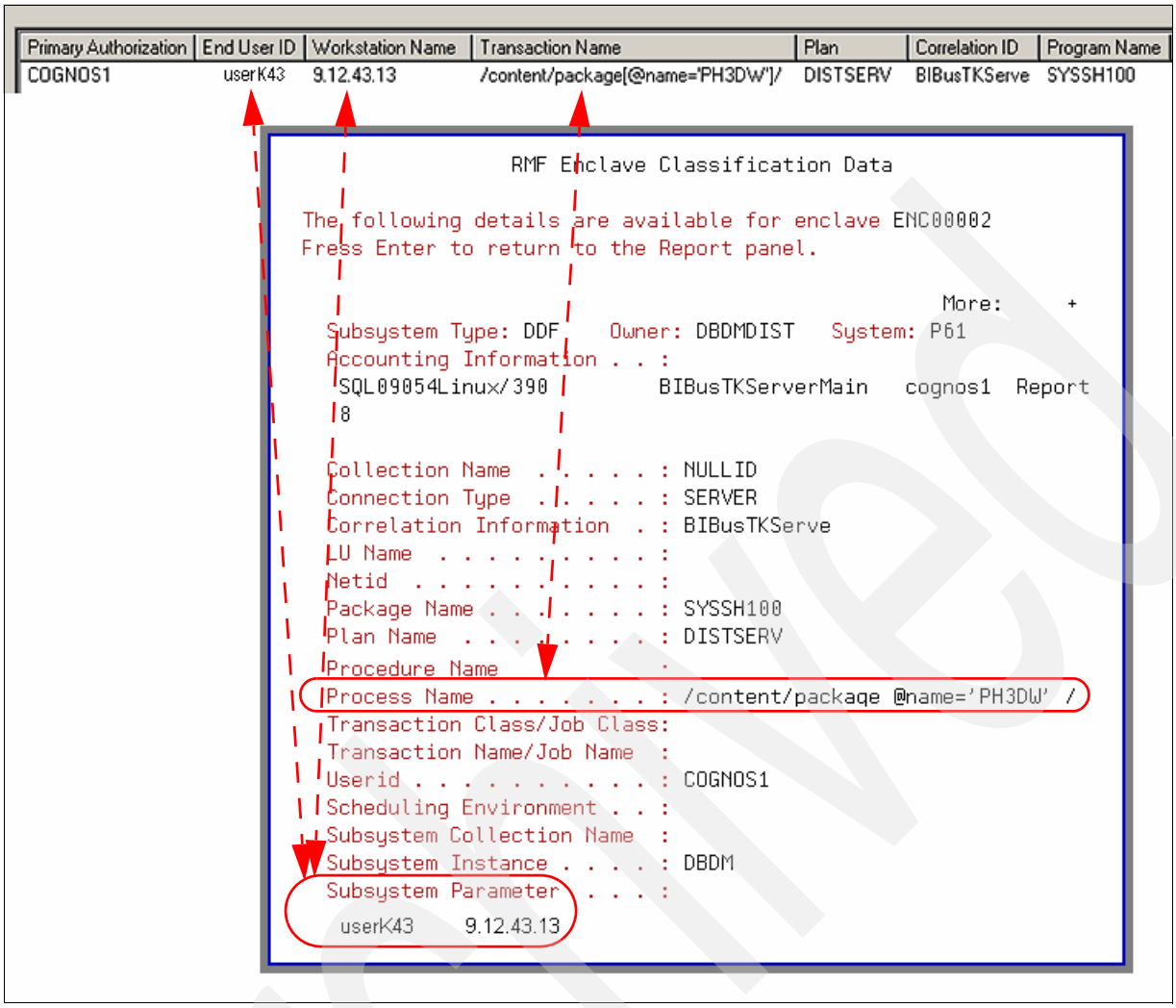


Figure 13-4 Omegamon PE V3.2 and RMF V1R10 Monitor III comparison

Archived

Bridging Reports of query workloads using DB2 client information

Here we bridge some common reporting data, again utilizing common reporting tools. We captured an RMF Workload activity report and a DB2 accounting report for the same reporting interval.

```
REPORT BY: POLICY=CL_OPTML                          REPORT CLASS=RCBKCRIT
                                                     DESCRIPTION =Cognos BKStore Key Knowledge workers

 -TRANSACTIONS-   TRANS-TIME HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE---    SERVICE TIME   ---APPL %---
 AVG        2.34  ACTUAL           3.13.425  SSCHRT 654.4   IOC          0   CPU  550.147   CP       67.51
 MPL        2.34  EXECUTION        3.13.424  RESP     1.6   CPU      21365K   SRB    0.000   AAPCP     0.00
 ENDED         3  QUEUED                 1  CONN     0.9   MSO          0   RCT    0.000   IIPCP    21.68
 END/S      0.01  R/S AFFIN              0  DISC     0.6   SRB          0   IIT    0.000
 #SWAPS        0  INELIGIBLE             0  Q+PEND   0.0   TOT      21365K   HST    0.000   AAP       0.00
 EXCTD         0  CONVERSION             0  IOSQ     0.0   /SEC     86105   AAP    0.000   IIP     154.21
 AVG ENC    2.34  STD DEV                0                                  IIP  382.631
 REM ENC    0.00                                           ABSRPTN     37K
 MS ENC     0.00                                           TRX SERV    37K
```

Figure 13-5 Excerpt from RMF - Workload Activity Report: Report Class

```
1   LOCATION: DBDMDDF                  OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1)                PAGE: 1-1
      GROUP: N/P                              ACCOUNTING REPORT - LONG                  REQUESTED FROM: NOT SPECIFIED
     MEMBER: N/P                                                                                    TO: NOT SPECIFIED
  SUBSYSTEM: DBDM                          ORDER: TRANSACT-ENDUSER                     INTERVAL FROM: 02/01/10 09:45:
DB2 VERSION: V9                               SCOPE: MEMBER                                          TO: 02/01/10 09:46:

TRANSACT: /content/package[@name='PH3DW']/  ENDUSER: userK43
...........
AVERAGE       APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)    CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT    HIGHLIGHTS
------------  ----------  ----------  ----------    -------------------   ------------  --------    --------------------------
ELAPSED TIME  3:13.42478  3:12.35317         N/P    LOCK/LATCH(DB2+IRLM)      1.161706   3158.67    #OCCURRENCES    :         3
 NONNESTED    3:13.42478  3:12.35317         N/A    SYNCHRON. I/O             2.429475    741.67    #ALLIEDS        :         0
 STORED PROC     0.000000    0.000000        N/A     DATABASE I/O             2.429475    741.67    #ALLIEDS DISTRIB:         0
 UDF             0.000000    0.000000        N/A     LOG WRITE I/O            0.000000      0.00    #DBATS          :         3
 TRIGGER         0.000000    0.000000        N/A    OTHER READ I/O           19.758688   4157.67    #DBATS DISTRIB. :         0
                                                    OTHER WRTE I/O            0.653232    199.33    #NO PROGRAM DATA:         0
CP CPU TIME    55.821906   55.821720         N/P    SER.TASK SWTCH            0.475550      5.33    #NORMAL TERMINAT:         3
 AGENT          2.232871    2.232696        N/A     UPDATE COMMIT            0.000000      0.00    #DDFRRSAF ROLLUP:         0
  NONNESTED     2.232871    2.232696        N/P     OPEN/CLOSE               0.473832      1.33    #ABNORMAL TERMIN:         0
  STORED PRC    0.000000    0.000000        N/A     SYSLGRNG REC             0.000000      0.00    #CP/X PARALLEL. :         3
  UDF           0.000000    0.000000        N/A     EXT/DEL/DEF              0.000000      0.00    #IO PARALLELISM :         0
  TRIGGER       0.000000    0.000000        N/A     OTHER SERVICE            0.001718      4.00    #INCREMENT. BIND:         0
 PAR.TASKS     53.589035   53.589024        N/A    ARC.LOG(QUIES)            0.000000      0.00    #COMMITS        :         3
 SECP CPU       0.050131         N/A        N/A    DRAIN LOCK                0.000000      0.00    #SVPT REQUESTS  :         0
SE CPU TIME   2:07.51243  2:07.51221         N/A    PAGE LATCH                0.008284      3.00    #SVPT ROLLBACK  :         0
 NONNESTED     2.600241    2.600185        N/A    NOTIFY MSGS               0.000000      0.00    MAX SQL CASC LVL:         0
 STORED PROC     0.000000    0.000000        N/A    GLOBAL CONTENTION         0.000000      0.00    UPDATE/COMMIT   :      0.00
 UDF             0.000000    0.000000        N/A    COMMIT PH1 WRITE I/O      0.000000      0.00    SYNCH I/O AVG.  : 0.003276
 TRIGGER         0.000000    0.000000        N/A    ASYNCH CF REQUESTS        0.000000      0.00
 PAR.TASKS    2:04.91204  2:04.91202         N/A    TOTAL CLASS 3            24.486935   8265.67
...........
AVERAGE SU          CLASS 1       CLASS 2    DYNAMIC SQL STMT      AVERAGE      TOTAL    MISCELLANEOUS          AVERAGE
------------  --------------  ------------    ----------------    --------    --------    ----------------    --------
CP CPU           2167839.67     2167833.00    REOPTIMIZATION          0.00           0    MAX STO LOB VAL (KB)      0.00
...........
SE CPU           4951938.00     9802882.00
```

Figure 13-6 Excerpt from Omegamon XE for DB2 PE - IBM Cognos BI BookStore package PH3DW

Using DB2 client information to qualify DRDA queries, here are some key performance metrics bridging two common reporting views of the DBA and z/OS system administrator. For the following, refer to Figure 13-5 and Figure 13-6.

1. The DB2 client qualifiers provide the ability to capture WLM and *DB2 performance statistics by IBM Cognos BI package - Enduser*. During this reporting interval, only one key knowledge worker, userK43, submitted IBM Cognos BI reports against IBM Cognos BI package PH3DW.

2. The # of RMF transactions that completed (ENDED) within the reporting interval is the same as the # of DBATs. In this case, it was three IBM Cognos BI reports, each having one SQL statement.
3. The RMF average response time (ACTUAL) is the same as the DB2 Accounting Class 1 average elapsed time.
4. The RMF CPU SERVICE units, which encompass general purpose as well as specialty processors, is also captured in DB2 accounting, though the DB2 accounting report provides a breakdown by general purpose (CP CPU) and specialty engine (SE CPU).
5. The RMF total SERVICE TIME - CPU is the total cp seconds, including general purpose and specialty engines, for this WLM report class, for this interval. To obtain just the general purpose cp seconds, one has to subtract out the RMF IIP (labeled #6). This delta would then equal the DB2 accounting report's CP CPU TIME.
6. The RMF total SERVICE TIME - IIP is the total cp seconds for the zIIP specialty engine. This is equal to the DB2 accounting report's SE CPU TIME.

Here are some caveats regarding the bridging of statistics just discussed. In order for the performance statistics to bridge as they have, the following conditions must be met:

- For DRDA parallel queries, APAR PM06953 (Single Enclave support for CP parallelism) must be applied.
- DB2 zparms need to have CMTSTAT=INACTIVE. This is the DB2 default and is already highly advised for any level of sophisticated WLM resource management of data warehouse queries.
- The DB2 application does not hold resources beyond a commit. This is DB2's recommended practice.

Additional thoughts

For examples of performance experiments, exhibiting the resource management value of using WLM SCI for WLM classification, see Chapter 15, "Single z/OS LPAR topology experiments" on page 347.

Recall the flexibility with WLM set client information, in that the administrator can decide what information is to be passed into each given parameter. Within this project alone, we experimented with three different permutations on what we passed into CLIENT_APPLNAME and CLIENT_ACCTSTR. One of those alternative approaches was using the IBM Cognos BI session variable, $report name, for the CLIENT_APPLNAME. This yielded the ability to generate DB2 accounting reports and charts, based on the IBM Cognos BI report name. This was beneficial for our project and would be beneficial in proofs of concept, benchmarking or internal performance testing and validation, where the workload under test is predefined.

In addition to z/OS and DB2 management and monitoring, the use of client information is also potentially valuable for performance problem determination (such as Optim's Data Studio, Tivoli ITCAM, and Guardium), accounting and chargeback.

Archived

14

I/O considerations

This section contains the following topics:

- Description of DB2 OLTP and data warehouse workloads from an I/O perspective
- Discussion of the IBM DS8000 storage subsystems

Archived

© Copyright IBM Corp. 2010. All rights reserved.

14.1 Introduction

Because each implementation is different and each one presents particular challenges, I/O subsystem planning and performance cannot be done using preplanned procedures or cookbooks. A co-location environment is not an exception. You should understand the principles involved and create a customized solution for each scenario.

14.2 DB2 workloads

To better understand and position the performance of your particular database system, it is helpful to first learn about the following common database profiles and their unique workload characteristics

- DB2 online transaction processing (OLTP)
- Decision support systems (DSSs)

DB2 online transaction processing (OLTP)

OLTP databases are among the most mission-critical and widely deployed of all. The primary defining characteristic of OLTP systems is that the transactions are processed in real time or online and often require immediate response back to the user. Examples include:

- A point of sale terminal in a retail business
- An automated teller machine (ATM) used for bank transactions
- A telemarketing site processing sales orders and checking the inventories

From a workload perspective, OLTP databases typically:

- Process a large number of concurrent user sessions.
- Process a large number of transactions using simple SQL statements.
- Process a few rows at a time.
- Are expected to complete transactions in seconds or even subseconds.

OLTP systems process the day-to-day operation of businesses and, therefore, have strict user response and availability requirements. They also have extremely high throughput requirements and are characterized by large numbers of database inserts and updates. They typically serve hundreds, or even thousands, of concurrent users.

Decision support systems (DSSs)

DSSs differ from the typical transaction-oriented systems in that they most often use data extracted from multiple sources for the purpose of supporting user decision-making. The types of processing consist of:

- Data analysis applications using predefined queries
- Application-generated queries
- Ad hoc user queries
- Reporting requirements

DSS systems typically deal with substantially larger volumes of data than OLTP systems due to their role in supplying users with large amounts of historical data. Whereas 100 GB of data is considered large for an OLTP environment, a large DSS system might be 1 TB of data or more. The increased storage requirements of DSS systems can also be attributed to the fact

that they often contain multiple, aggregated views of the same data. While OLTP queries are mostly related to one specific business function, DSS queries are often substantially more complex. The need to process large amounts of data results in many processor-intensive database sort and join operations. The complexity and variability of these types of queries must be given special consideration when estimating the performance of a DSS system.

How DB2 assigns I/O priorities

DB2 informs z/OS about which address space's priority is to be associated with a particular I/O request, and WLM handles the management of the request.Table 14-1 describes which enclave or address space is associated with I/O read requests.

Table 14-1 How read I/O priority is determined

Request type	Synchronous reads	Prefetch reads
Local	Application's address space	Application's address space
DDF or sysplex query parallelism (assistant only)	Enclave priority	Enclave priority

Table 14-2 describes to which enclave or address space DB2 is associated with the I/O write requests.

Table 14-2 How write I/O priority is determined

Request type	Synchronous writes	Deferred writes
Local	Application's address space	ssnmDBM1 address space
DDF or sysplex query parallelism (assistant only)	DDF address space	ssnmDBM1 address space

14.3 IBM DS8000 storage subsystem

The IBM System Storage DS8000 series is a high-performance, high-capacity series of disk storage subsystems. It offers balanced performance and storage capacity that scales linearly up to hundreds of terabytes.

14.3.1 Introduction to the IBM DS8000 series features and functions

The RAID 6 capability is optional for the DS8000 series and requires device adapter cards. For DS8000 machines already installed, a feature conversion to these device adapter cards is available. For new DS8000 machine orders requiring RAID 5, RAID 6, and RAID 10 support, these device adapter cards are preinstalled.

With the DS8000 LPAR Models it is possible to create two storage system logical partitions (LPARs) in a single DS8000. These LPARs split the system resources equally (50/50) or in a variable way (75/25), and can be used in completely separate storage environments.

The IBM System Storage DS8000 series highlights include:

- Robust, flexible, enterprise class, and cost-effective disk storage
- Exceptionally high system availability for continuous operations
- Centralized and simplified management

- IBM POWER5+™ processor technology
- Physical Capacities from 1.1 TB to 460/1024 TB (FC/SATA)
- Multiple storage system LPARs for completely separate storage environments
- Point-in-time copy function with FlashCopy®, FlashCopy SE, or Remote Pair FlashCopy, and Remote Mirror and Copy functions with Metro Mirror, Global Copy, Global Mirror, Metro/Global Mirror, z/OS Global Mirror, and z/OS Metro/Global Mirror with new Incremental Resync capability
- Support for a wide variety of platforms, including System i and System z
- Designed to increase storage efficiency and utilization, the DS8000 can be part of an energy-efficient data center

Storage capacity

The physical capacity for the DS8000 is purchased through disk drive sets. A disk drive set contains 16 identical disk drive modules (DDMs) that have the same physical capacity and the same revolutions per minute (rpm). For additional flexibility, feature conversions are available to exchange existing disk drive sets.

In the first frame, there is space for a maximum of 128 disk drive modules (DDMs) and each Expansion Frame can contain 256 DDMs. With a maximum of 384 DDMs, the DS8100 Turbo Model 931, using 450 GB drives, provides up to 172 TB of storage capacity. With a maximum of 1024 DDMs, the DS8300 Turbo Models 932 and 9B2, using 450 GB drives, provide up to 460 TB of storage capacity with four expansion frames. When using 1 TB SATA drives, you can scale up your storage to 1024 TB or 1 PB.

The DS8000 can be configured as RAID 5, RAID 6, RAID 10, or as a combination (some restrictions apply for Full Disk Encryption (FDE) and Solid State Drives).

Virtual Capacity

With the Thin Provisioning feature introduced in Licensed Machine Code 5.4.3x.xx and the FlashCopy SE feature released in Licensed Machine Code 5.3.xx.xx, Virtual Capacity was introduced to the DS8000. The total Virtual Capacity associated with space-efficient volumes of both types can be larger than the underlying physical capacity, reflecting the concept of Thin Provisioning or over-provisioning of storage.

IBM Standby Capacity on Demand offering for the DS8000

Standby Capacity on Demand (Standby CoD) provides standby on demand storage for the DS8000 that allows you to access the extra storage capacity whenever the need arises. With CoD, IBM installs up to four disk drive sets (64 disk drives) in your DS8000. At any time, you can logically configure your CoD drives, concurrently with production, and you are automatically charged for the capacity.

Tip: For more details about the DS8000 controller family, refer to *IBM System Storage DS8000: Architecture and Implementation*, SG24-6786.

14.3.2 Performance for System z

The DS8000 series supports the following IBM performance enhancements for System z environments:

- Parallel Access Volumes (PAV) enable a single System z server to simultaneously process multiple I/O operations to the same logical volume, which can help to significantly reduce device queue delays. This is achieved by defining multiple addresses per volume. With

Dynamic PAV, the assignment of addresses to volumes can be automatically managed to help the workload meet its performance objectives and reduce overall queuing. PAV is an optional feature on the DS8000 series.

- HyperPAV is designed to enable applications to achieve equal or better performance than with PAV alone, while also using fewer Unit Control Blocks (UCBs) and eliminating the latency in targeting an alias to a base.
- Multiple Allegiance expands the simultaneous logical volume access capability across multiple System z servers. This function, along with PAV, enables the DS8000 series to process more I/Os in parallel, helping to improve performance and enabling greater use of large volumes.
- I/O priority queuing enables the DS8000 series to use I/O priority information provided by the z/OS Workload Manager to manage the processing sequence of I/O operations.
- High Performance FICON for z (zHPF) reduces the impact associated with supported commands on current adapter hardware which, together with multitrack support, improves FICON throughput on the DS8000 I/O ports.

14.3.3 Considerations for mixed workloads

The bigger capacity of the DS8000 enables you to combine data and workloads from several different kinds of independent servers into a single DS8000.

Sharing resources in a DS8000 has advantages from a storage administration and resource-sharing perspective, but it does have implications for workload planning. Resource-sharing has the benefit that a larger resource pool (for example, disk drives or cache) is available for critical applications. However, be careful to ensure that uncontrolled or unpredictable applications do not interfere with mission-critical work.

If you have a workload that is truly mission-critical, you might want to consider isolating it from other workloads, particularly if those other workloads are unpredictable in their demands.

There are several ways to isolate the workloads:

- Place the data on separate DS8000s, or separate DS8000 LPARs. This option is, of course, the best choice.
- Place the data on separate DS8000 servers. This option isolates the use of memory buses, microprocessors, and cache resource. However, before doing that, make sure that a half DS8000 provides sufficient performance to meet the needs of your important applications. Note that Disk Magic provides a way to model the performance of a half DS8000 by specifying the Failover Mode. Consult your IBM representative for a Disk Magic analysis.
- Place the data behind separate device adapters.
- Place the data on separate ranks, which will reduce contention for the use of DDMs.

Note: z/OS and Open Systems data can only be placed on separate extent pools.

DS8000 considerations for DB2

Benefits of using the DS8000 in a DB2 environment include:

- DB2 takes advantage of the parallel access volume (PAV) function that allows multiple concurrent I/Os to the same volume at the same time from applications running on a z/OS system image.

- Less disk contention when accessing the same volumes from different systems in a DB2 data sharing group using the Multiple Allegiance function.
- Higher bandwidth on the DS8000 allows higher I/O rates to be handled by the disk subsystem, thus allowing for higher application transaction rates.

When using DS8000, the following generic recommendations are useful when planning for good DB2 performance:

- Know where your data resides

  DB2 storage administration can be done using SMS to simplify disk use and control, or also without using SMS. In both cases, it is important that you know where your data resides. If you want optimal performance from DS8000, do not treat it totally like a “black box.” Understand how DB2 tables map to underlying volumes and how the volumes map to RAID arrays.

- Balance workload across DS8000 resources

  You can intermix tables and indexes and also system, application, and recovery data sets on DS8000 ranks. The overall I/O activity will be more evenly spread, and I/O skews will be avoided. You can balance workload activity across DS8000 resources by:

  – Spreading DB2 data across DS8000s, if practical
  – Spreading DB2 data across servers in each DS8000
  – Spreading DB2 data across DS8000 device adapters
  – Spreading DB2 data across as many extent pools and ranks as practical

- Take advantage of VSAM data striping

  By striping data, the VSAM control intervals (CIs) are spread across multiple devices. This format allows a single application request for records in multiple tracks and CIs to be satisfied by concurrent I/O requests to multiple volumes. The result is improved data transfer to the application. In a DS8000 with Storage Pool Striping (SPS), the implementation of VSAM striping still provides a performance benefit. Because DB2 uses two engines for the list prefetch operation, VSAM striping increases the parallelism of DB2 list prefetch I/Os. This parallelism exists with respect to the channel operations as well as the disk access.

- Large volumes

  With large volume support, which supports up to 65520 cylinders per volume, System z users can allocate the larger capacity volumes in the DS8000. From the DS8000 perspective, the capacity of a volume does not determine its performance. From the z/OS perspective, PAVs reduce or eliminate any additional enqueues that might originate from the increased I/O on the larger volumes. From the storage administration perspective, configurations with larger volumes are simpler to manage. Measurements oriented to determine how large volumes can impact DB2 performance have shown that similar response times can be obtained using larger volumes compared to using the smaller 3390-3 standard size volumes.

- DB2 burst write

  When DB2 updates a record, it first updates the record residing in the buffer pool. If the percentage of changed records in the buffer pool reaches the threshold defined in the vertical deferred write threshold (VDWQT), DB2 starts to flush and to write these updated records. These write activities to the disk subsystem are a huge burst of write I/Os, especially if the buffer pool is large and the VDWQT is high. This burst can cause a nonvolatile storage (NVS) saturation, because it is being flooded with too many writes. It shows up in the Resource Measurement Facility (RMF) cache report as DASD Fast Write Bypass (DFWBP or DFW Bypass). The term bypass is actually misleading. In the

3990/3390 era, when the NVS is full, the write I/O bypasses the NVS and the data is written directly to the disk drive module (DDM). In DS8000, when the NVS is full, the write I/O is retried from the host until NVS space becomes available. So DFW Bypass must be interpreted as DFW Retry for DS8000. If RMF shows that the DFW Bypass divided by the total I/O Rate is greater than 1%, that is an indication of NVS saturation. If this NVS saturation happens, we recommend that you set the VDWQT to 0 or 1. Setting the VDWQT to 0 does not mean that every record update will cause a write I/O to be triggered, because despite the 0% threshold set, the DB2 buffer pool will still have 40 buffers set aside and the least recently updated 32 buffers are scheduled for write, which allows multiple successive updates to the same record to be not written out every time that record is updated, thus preventing multiple write I/Os to the same record on the disk subsystem. Lowering the VDWQT will have a cost; in this case, it increases the processor utilization, which shows up as a higher DBM1 SRM CPU time.

DS8000 performance monitoring tools

There are several tools available that can help with monitoring the performance of the DS8000:

- Resource Management Facility (RMF)
- RMF Magic
- OMEGAMON
- Tivoli Productivity Center

Tip: For more details about DS8000 performance and monitoring, refer to *DS8000 Performance Monitoring and Tuning*, SG24-7146.

Archived

Archived

Part 4

Project experiment results

In this part we provide the results, observations, and conclusions of our experiments, for both single LPAR and multiple LPAR configurations.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

15

Single z/OS LPAR topology experiments

In this chapter we provide the setup, results, and analysis of a set of resource management experiments utilizing the workloads described in chapters 5, 6, and 7. The objective of the experiments is primarily to understand how well WLM and related resource management techniques can be utilized to manage transactional and data warehouse workloads that are running concurrently.

Our hope is that you will gain insight on how WLM makes trade-offs between the mixed workloads, both steady-state and under different workload variations. This information is helpful when implementing WLM in your own environments. In this chapter we provide:

- A recap of the workloads and performance objectives
- Relevant WLM implementation details
- Measurement and data collection methodology
- Description of the key performance metrics
- Workload characterization
- Key configuration settings
- Experiment results and observations
- A summary

© Copyright IBM Corp. 2010. All rights reserved.

15.1 Introduction

The experiments in a single z/OS LPAR topology were driven by the idea that many shops run a dedicated data warehouse DB2 for z/OS subsystems in the same LPAR as their traditional transactional workloads, storing the data in either IMS/DB or DB2 for z/OS. Traditional OLTP transactions are highly optimized to complete in the shortest possible amount of time, mostly subsecond. Compared to these transactions, common data warehouse queries can range from very short-running, small resource consumers to large resource consumers, higher by orders of magnitude both in terms of processor and I/O requirements.

Though all our experiments had both transactional and data warehouse query workloads, we believe the contents of this chapter are also applicable to understanding resource management for data warehouse workloads alone, which are quite diverse in themselves.

For each experiment, the following workloads were run (see Table 15-1 on page 349):

- BookStore Transactional Web Browse/Buy application
  - Simulating over 1000 users generating a request rate approximately 2000-2500 transactions/second
- BookStore Transactional Order status application
  - Simulated hundreds of users generating requests at a rate of approximately 50-100 transactions/second
- BookStore Data warehouse IBM Cognos Analysis, Reporting, Operational BI
  - Simulates hundreds of users running a mix of trivial to large resource consumption IBM Cognos BI reports
  - 50% of this work is arriving from a group of high importance critical knowledge workers. The other 50%, which is comprised of the same mix, arrives from a group of novice to intermediate knowledge workers, of medium to low importance.

The following experiments were conducted:

- Sophisticated management, utilizing DB2 client connection properties for classifying the IBM Cognos BI query workload
- Spike in the Transactional workload
- Spike in the data warehouse query workload
- Spike in the data warehouse query workload: Further protection of high importance transactional work
- zIIP-eligible work limited to zIIP processors
- Dynamic WLM management of SQW refresh

15.2 WLM implementation for experiments

Here we recap the background of our sample BookStore business performance requirements. We also provide the relevant WLM service classes and WLM classification rules utilized for the experiments.

Archive

Table 15-1 Key workloads focused on in the experiments

Application or Process or Server	Type of work	Source	Users	Performance requirements	Potential qualifiers
BookStore web application Browse/Buy	OLTP	Web-Sphere RRS	1000s of users No differentiation	Require subsecond response times. Importance: High	URL
BookStore Web application Order Status	OLTP	Web-Sphere RRS	100s of users No differentiation	Require 1 second response times. Importance: Medium	URL
BookStore Sales IBM Cognos BI Online Analysis and Reporting	Online Query	DRDA	Key knowledge workers	Consistent response times for smaller resource consumption queries.Give higher priority to the power users over the others. Importance: HIgh	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI user IBM Cognos BI report
BookStore Sales IBM Cognos BI Online Analysis and Reporting	Online Query	DRDA	Intermediate to novice knowledge workers	Consistent response times for smaller resource consumption queries. The larger consumption work gets discretionary resource (after power user work). Importance: Medium to Low	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI user IBM Cognos BI report
BookStore Service Center IBM Cognos BI Operational BI	Online Query	DRDA	Service center reps	As service center reps are interacting with clients, these Operational BI reports must have a response time of <3s. Importance: High	DB2 subsystem DB2 connection ID IBM Cognos BI package name IBM Cognos BI user IBM Cognos BI report

Transactional (OLTP) service classes

Service class WASTCHI services the BookStore Web Browse/Buy application (Table 15-2)

Table 15-2 WASTCHI service class definition

Period	Importance	Duration	Goal type	Goal
1	2	n/a	Response time	90%< 0.25s

Service class WASTCLOW services the BookStore Web Order status application (Table 15-3)

Table 15-3 WASTCHLOW service class definition

Period	Importance	Duration	Goal type	Goal
1	3	n/a	Response time	90%< 0.9s

Data warehouse query service classes

Service class DWDDFHI services the high Importance “IBM Cognos BI online analysis and reporting” users (the “Power” or expert knowledge workers) as well as the “IBM Cognos Operational BI” service center representatives (Table 15-4).

Table 15-4 DWDDFHI service class definition

Period	Importance	Duration (service units)	Goal type	Goal
1	2	10,000	Response time	90%< 1s
2	2	100,000	Response time	80%< 10s
3	3	1,000,000	Velocity	40
4	4		Velocity	10

Service class DWDDFST services the Low to Medium Importance “IBM Cognos BI online analysis and reporting” users (the novice to intermediate knowledge workers)(Table 15-5).

Table 15-5 DWDDFST service class definition

Period	Importance	Duration (service units)	Goal type	Goal
1	3	10,000	Response time	80%< 1s
2	3	100,000	Response time	80%< 15s
3	4	1,000,000	Velocity	30
4	5	8,000,000	Velocity	5
5			Discretionary	

Items of interest regarding the two DW DDF query service class definitions:

- For our experiments, we purposely made the period durations the same for periods 1-3. This way we would be able to better appreciate how WLM is differentiating between the different user groups. Recall that the same amount and mix of IBM Cognos BI report requests are being driven against both of these service classes.
- Note the differences in Importance and Goals for the DWDDFST periods versus the DWDDFHI periods.
- Note the additional 5th period of Discretionary for the DWDDFST service class. This is to service the very large consumption, lower importance queries.

> **Note:** In a production environment, we recommend at least two service classes, each with multiple periods, for differentiating user groups for resource management. We would not recommend having the same period durations for the different Importance service classes. That was done solely for our experiments, in order to better appreciate WLM’s behavior.

For a more complete set of WLM sample service definitions, see Appendix D, “Sample WLM service definition” on page 433.

Transactional workload classification rules

Figure 15-1 on page 351 shows the key classification rules implemented for service and report classes exercised in the experiments.

```
* Subsystem Type CB - Rules for WAS (Transaction only)

  Classification:

    Default service class is WASTCDFL
    Default report class is RWASTCDF

      Qualifier   Qualifier       Starting       Service    Report
    # type        name            position       Class      Class
    - ----------- -------------- ---------      --------- ---------
    1 TC          BKWASTRN                      WASTCDFL   RWASTCDF
    1 TC          WASTCLOW                      WASTCLOW   RWASTCLO
    1 TC          WASTCHI                       WASTCHI    RWASTCHI
```

Figure 15-1 Classification rules for the transactional workload service classes

Work requests for our BookStore transactional WebSphere application workloads arrived into the z/OS system through the CB (WebSphere) WLM subsystem. WebSphere WASTCHI and WASTCLOW transaction classes were created for the respective incoming BookStore Browse/Buy and Order Status applications. The WebSphere preparation of those transaction classes is covered in 5.3.1, “WLM classification of BookStore transactions” on page 72.

The incoming work is filtered into the WebSphere transaction classes using their respective URIs. The WLM classification utilizes these transaction classes (TC) to filter the work into WLM’s service classes with the same names. We also created WLM Report classes, which for this project were redundant, since they mapped directly to the service classes defined.

Data warehouse DRDA query classification rules

Figure 15-2 on page 352 shows the key classification rules implemented for DRDA query service and report classes.

Archived

```
* Subsystem Type DDF - Rules for DDF (Transaction only)

  Classification:

    Default service class is DDFDFLT
    Default report class is RDDFDFLT

      Qualifier   Qualifier      Starting      Service    Report
    # type        name           position      Class      Class
    - ----------- -------------  ---------     --------   --------
    1 SI          DBDM                         DWDDFST    DWP_DFLT
    2 . UI         COGNOS1                     DWDDFST    COG_DFLT
    3 . . SPM        . userS*                  DWDDFHI    CS_SERV
    4 . . . PC       . . OpBI*       3         DWDDFHI    CS_OPBI
    3 . . SPM      . userK*                    DWDDFHI    CK_CRIT
    4 . . . PC     . . port3         3         DWDDFHI    CK_RP3
    4 . . . PC     . . port5         3         DWDDFHI    CK_RP5
    4 . . . PC     . . port6         3         DWDDFHI    CK_RP6
    4 . . . PC     . . port8         3         DWDDFHI    CK_RP8
    4 . . . PC     . . port10        3         DWDDFHI    CK_RP10
    4 . . . PC     . . port11S       3         DWDDFHI    CK_RP11S
    4 . . . PC     . . port11M       3         DWDDFHI    CK_RP11M
    4 . . . PC     . . port 12       3         DWDDFHI    CK_RP12
    4 . . . PC     . . port 13       3         DWDDFHI    CK_RP13
    3 . . SPM      . userI*                    DWDDFST    CI_MED
    4 . . . PC     . . port3         3         DWDDFST    CI_RP3
    4 . . . PC     . . port5         3         DWDDFST    CI_RP5
    4 . . . PC     . . port6         3         DWDDFST    CI_RP6
    4 . . . PC     . . port8         3         DWDDFST    CI_RP8
    4 . . . PC     . . port10        3         DWDDFST    CI_RP10
    4 . . . PC     . . port11S       3         DWDDFST    CI_RP11S
    4 . . . PC     . . port11M       3         DWDDFST    CI_RP11M
    4 . . . PC     . . port12        3         DWDDFST    CI_RP12
    4 . . . PC     . . port13        3         DWDDFST    CI_RP13
```

Figure 15-2 Classification rules for the data warehouse query workload service classes

We utilized four levels of DDF qualifiers for the incoming work requests. The requests are first filtered, at level 1, by SI (DB2 subsystem), then at level 2, by UI (authorization user ID - the common connection user ID aka DB2 AUTHID). At level 3, work is filtered utilizing SPM, our DB2 client information qualifier for IBM Cognos BI user. This is where we differentiate between our high importance query service class, DWDDFHI and our medium-low importance query service class, DWDDFST. At level 4, work is filtered utilizing PC, our DB2 client information qualifier for IBM Cognos BI report name.

One can see that our general methodology was that all service center representatives would have user names starting with userS*. All key knowledge workers have user names starting with userK*, and all other IBM Cognos BI users (considered intermediate or novice) would start with userI*.

For this project, qualifying by IBM Cognos BI report name, via DB2 client info, proved valuable for multiple reasons. Not only were we able to capture WLM performance statistics by Report name, we also were able to capture DB2 performance statistics by report name. This allowed us to verify our workload driver as well as cross-checking statistics between z/OS and DB2. While qualifying by report name might be less applicable in a production environment, it would be valuable for things such as internal testing of new function or new H/W, proofs-of-concept, or benchmark exercises. More details about the use of DB2 client information can be found in Chapter 13, “Utilizing DB2 client information for resource management and monitoring” on page 321.

15.3 Measurement and data collection methodology

Prior to exploring the experiment results, it is important to understand the general methodology utilized for running the experiments and data collection. For each experiment the following steps were taken:

1. Start WebSphere and the two DB2 subsystems (transactional and data warehouse).
2. For the data warehouse subsystem, execute the following statements to open up all the data sets.
3. For the data warehouse subsystem, we ran a batch job of queries to open DB2 data sets and prime some of the DB2 buffer pools for consistency across experiments. Most of the dimension tables were small enough to be completely fit within a buffer pool.
4. Start the jibe simulator simulating x # of clients conducting BookStore Browse/Buy transactions.
5. Start the Rational Performance Test Schedule script that will call the IBM Cognos BI reports.
6. Wait approximately 15 minutes for both workloads to stabilize.
7. Clear SMF.
8. Start data collection on z/OS.
   - Resource Monitor Facility (RMF) - 1-minute intervals
   - DB2 traces
     - Statistics trace Class(1,3,4,6)
     - Accounting trace Class(1,2,3,7,8)
9. Start data collection on Linux guests (sar and sadf).
10. Run for a specified amount of time (typically 20-30 minutes).
11. Optional: Make a workload or policy adjustment and run for another 20-30 minutes.
12. End data collection.
    - On z/OS: Stop RMF, stop DB2 traces and switch SMF.
    - On Linux: Cancel the **`sar`** and **`sadf`** commands.
13. Stop the workloads (jibe simulator and Rational Performance test schedule).
14. Dump and post process the z/OS SMF data.
    - Creating the following reports
      - RMF Monitor 1 plus a Workload Activity report for the duration of the measurement
      - RMF Workload Activity reports for each 1-minute interval in the measurement
      - Omegamon for DB2 PE Accounting and Statistics Reports
15. Post process and summarize the Linux data performance data collected.

Our primary data of interest for these experiments was the z/OS WLM performance data, available in the RMF Workload Activity report. This is the data we focus on in more detail in the next section.

Archived

15.4 Key performance metrics

There are many tools available for analyzing System z resource management and performance. We decided to utilize the RMF Workload Activity report data as the main means for our experiment analysis. For each experiment, we pulled what we believed to be the predominant key WLM performance metrics out of this report, and built a spreadsheet similar to Table 15-6. Actually we created RMF overview records and after the experiment, pulled the overview records into the RMF spreadsheet reporter's Workload Overview report.

Table 15-6 Table of key statistics pulled from the RMF workload activity report

Service Class												
Service Class Definition					Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP or IIP	Total	AAPCP or IIPCP
1	2	10,000	Resp time	90%< 1s		0.5	693	0.1	27	27	54	5
2	2	100,000	Resp time	80%< 10s		0.7	70	3.1	39	49	87	12
3	3	1,000,000	Velocity	40		0.7	11	28.1	49	71	120	17
4	4		Velocity	10		0.6	0.2	330.9	17	30	47	7
								Totals:	132	176	308	41

For each experiment, we provide these metrics for all key service classes. Additionally, we provide the processor utilization for each of the processor types (general purpose, zAAP, and zIIP).

The following field definitions are provided as a reference for analyzing the experiment results. Additionally, we include some considerations on how this data might be useful for analysis beyond the scope of these experiments.

Service class definitions

We briefly define these here. They are covered in more detail in the WLM refresher section (Appendix F, "WLM refresher" on page 491).

Serv Class The Service Class being reported. Note: Similar data could also be abstracted from the RMF Workload activity report, by Report Class, if the Report Class was "homogeneous". "Homogeneous" signifies all the work within a given report class, within the same service class.

Per The Service class "period".

Imp Importance: What is the relative (relative to everything else running under the current WLM policy) importance of work in this service class period.

Duration (SUs) For a given unit of work, the number of service units (SUs) accumulated, prior to moving to the next service class period. Duration is the only metric not provided in the RMF Workload Activity report. We just copied these directly from our service policy definition.

For our experiments, all of the IBM Cognos BI query work requests arrived via DRDA, hence were WLM-managed as *independent enclaves*. For independent enclaves, the number of service units (SUs) consumed is directly associated with the number of processor seconds consumed (both general purpose and zIIP). I/O and MSO SUs are not included.

Note: In our configuration, the number of SUs/cpu sec was approximately 38,000 SU/sec. This ratio comes directly from the RMF Workload Activity report.

Goal Type and Goal This is the WLM Goal for this particular service class period.

Goal results

How are we doing against our goal definitions? Exceeding our goal? Meeting our goal? Underachieving?

Actual The actual (measured) service class period goal for the RMF interval? For example, for Period 1, the goal was 90% of this work to complete in less then 1 second. The actual results yielded 100% of the work ended in less than 1 second.

PI Performance Index. A value of 1 or less indicates we are meeting our goal. A value greater then 1, we are missing our goal.

Key performance indicators (KPIs)

Trans per min This is *unit of work* completions per minute. In our case, it represents IBM Cognos BI queries/minute. This is simply calculated by dividing -TRANSACTIONS- ENDED (within the RMF workload activity report service class period) by the RMF interval minutes. This metric is often referred to as workload "throughput".

Avg Resp Time Average response time for all query completions within the RMF interval. For service classes that have response time goals, a more detailed characterization of response times is provided via response time distribution data, also provided in the RMF Workload Activity report.

Processor utilization

Here we capture a view of the processing consumption (by processor type) by service class periods. This type of data, plotted over time, is often fed into z/OS performance analysis and capacity planning tools. For our experiment results, we are summarizing this data by service class and service class periods, but it is also beneficial to summarize this data by WLM Workload or report classes. We now explain the usefulness of this data and the types of questions it helps answer.

- Across all workloads or service classes, who is consuming the processing cycles of the z/OS system? Is this what I anticipated? Is this what I desire? Might I consider additional report classes to gain a more granular understanding of where the processing resources are going?
- For a given multiperiod service class, what is the distribution of processing cycles being consumed across the different periods? Is this what I expected? Is this what I desire? Note: For z/OS DB2 query workloads, it is quite common to see an 80/20 relationship. 80% of the query completions come in period 1, though they consume 20% or less of the resources. Meanwhile, the remaining 20% of the query completions come in the remaining periods, yet they are responsible for 80% of the processing consumption.
- Dealing with a shift in workload demands. For example, suppose a high importance application starts to regularly demand more processing resource (perhaps a growth in users), yet the current policy is not aligned well with that increased demand. Having a good understanding of the current processing resource usage distribution will help determine what, if any, service class modifications might yield a solution, without acquisition of more processing power. Related to this, the better work is properly classified, the more valuable this information becomes.
- By either workload, service class, or report class, how much specialty engine offloading are we achieving? How much potential specialty engine processing is running on our general purpose processors? Should we reevaluate our HONORPRIORITY settings? Should we consider merging our zAAP and zIIP pools to just zIIPs for an upcoming processor migration?

The following fields are taken from the APPL % column, for a given service class period, of an RMF Workload Activity report.

CP The average percentage of a single general purpose processor being consumed, during this RMF interval. For example, referencing Figure 15-6 on page 354, period 4 of service class DWDDFHI, on average, is consuming 49% of a single general purpose processor, over that interval.

AAP The average percentage of a single zAAP processor being consumed, during this RMF interval.

IIP The average percentage of a single zIIP processor being consumed, during this RMF interval.

AAPCP The average percentage of a single general purpose processor being consumed, with work that was potentially offloadable to a zAAP, had a zAAP processor been available.

IIPCP The average percentage of a single general purpose processor being consumed, with work that was potentially offloadable to a zIIP processor if one had been available.

15.4.1 BookStore data warehouse DB2 query profile

To provide you with a reference for the DB2 query workload utilized in these experiments, Table 15-7 depicts the arrival rates and key DB2 performance statistics for the queries generated by the IBM Cognos BI reports.

Table 15-7 DB2 query performance statistics associated with the IBM Cognos BI reports

Category of Query	Aggregate Query Arrival rate	DB2 Avg Class 1 Total CP Time (sec)	DB2 Avg Class 1 Elapsed Time (sec)	DB2 Avg GetPage count
Trivial	612/min	0.15	0.8	15K
Small	86/min	2	6	56K
Medium	72/hr	22	40	298K
Large	8/hr	203	275	15,227K
XLarge	7/hr	607	3903	34,332K

Notes for Table 15-7:

- Our workload objective was to get a reasonably representative distribution of Trivial, Small, Medium, Large queries, by processing needs. This distribution was roughly based on one of our customer workload profiles.
- Category of Query was our own categorization based on DB2 cp time consumption.
- The XLarge query is not part of the base workload. It is only utilized in the “Spike the data warehouse” experiment.
- In these statistics we omitted the transactions associated with calls to WLM set client information stored procedure and DB2 catalog table look-ups prompted by IBM Cognos BI.

Archived

15.4.2 Key configuration settings

The key z/OS and DB2 configuration settings shown in Table 15-8 were utilized for all experiments, except when noted otherwise.

Table 15-8 Key configuration settings

CPs	zAAPs	zIIPs	Memory	zIIP on CP?	DB2 zparm for Parallelism
7	2	4	32GB	IIPHONORPRIORITY = YES	PARAMDEG=8 (Max Degree=8)

15.5 Sophisticated management with DB2 client attributes

15.5.1 Objectives

- To understand the effects of utilizing DB2 client connection properties to isolate higher importance key knowledge workers from medium importance intermediate-novice knowledge workers.
- To obtain baseline resource usage and performance profiles of the transactional and data warehouse workloads.

15.5.2 Results

Table 15-9 Processor utilization

zOS Processor Utilization	CP	zAAP	zIIP
Base Transactional + DW Query	*94*	*61*	*89*

Archived

Table 15-10 Key service class performance statistics

DWDDFHI					Base Transactional + DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	2	10K	RT	90%< 1s	98%	0.5	693	0.1	27	27	54	5
2	2	100K	RT	80%< 10s	95%	0.7	70	3.1	39	49	87	12
3	3	1M	Vel	40	56	0.7	11	28.1	49	71	120	17
4	4		Vel	10	16	0.6	0.2	330.9	17	30	47	7
								Totals:	132	176	308	41

DWDDFST					Base Transactional + DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	3	10K	RT	80%< 1s	98%	0.5	667	0.1	25	27	53	5
2	3	100K	RT	80%< 15s	98%	0.5	68	3.6	39	51	90	12
3	4	1M	Vel	30	43	0.7	12	38.3	48	79	126	16
4	5	8M	Vel	5	7	0.7	0.1	717.9	8	14	23	3
5			Disc		N/A	0	0	0.0	0	0	0	0
								Totals:	120	171	291	36

WASTCHI					Base Transactional + DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	2		RT	90%< 0.25s	98%	0.5	2033	0.012	174	77	251	10

WASTCLOW					Base Transactional + DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	3		RT	90%< 0.9s	98%	0.6	45	0.186	11	12	24	1

Observations and conclusions

First we examine the general resource consumption of the different service classes. Examining the zIIP, zAAP, CP, and total processing usage for each of the service classes provides a base understanding for the distribution of processing resource amongst the different workloads. Additionally, for the query service classes, one can get an understanding of query completion rates and resource usage distribution across the multiple periods. As pointed out earlier, the query workload was built to resemble the processing resource profile typical of our customers. Notice there are many more query completions in the first period, yet the predominant processing consumption is in the later periods, 2, 3, and 4.

In our base scenario all service classes and service class periods are making their goals (PIs <=1). Prior to obtaining these results, we did go through a couple of iterations, tuning the 3rd and 4th period DWDDF Velocity goals, to what we were actually achieving.

Here we also start to see the value of utilizing DB2 client strings to differentiate the critical knowledge workers from the rest. Note the deltas in response time, throughput, and processing utilization between DWDDFST and DWDDFHI. This is more evident in the higher periods. Actually, in this experiment, we generally had enough processing available to satisfy the higher importance DWDDFHI as well as the DWDDFST service classes. In later experiments, where resources are more in demand, the differences will be more noticeable.

Additionally, the Operational BI processing type queries predominantly fall into period one, where they are being managed to a less then 1-second response time, and again, are meeting that goal.

15.6 Spike in transactional workload

15.6.1 Objective

Understand how WLM responds when an increased demand on the transactional workload arises. Here we rerun the same workload as in our first experiment, with the exception of starting 30% more transactional Browse/Buy users. We observe the WLM trade-offs being made and the impact to the various service classes.

15.6.2 Results

Table 15-11 Spike in transactional workload—processor utilization

zOS Processor Utilization	CP	zAAP	zIIP
Base Transactional + DW Query	*94*	*61*	*89*
Spike in Transactional	*98*	*71*	*84*

Table 15-12 Spike in transactional workload - Key Service class statistics

DWDDFHI					***Base Transactional + DW Query***								***Spike in Transactional***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	2	10K	RT	90%< 1s	98%	0.5	693	0.1	27	27	54	5	98%	0.5	719	0.1	26	29	55	5
2	2	100K	RT	80%< 10s	95%	0.7	70	3.1	39	49	87	12	91%	0.7	68	3.7	43	56	99	12
3	3	1M	Vel	40	56	0.7	11	28.1	49	71	120	17	56	0.7	12	34.2	54	70	124	17
4	4		Vel	10	16	0.6	0.2	330.9	17	30	47	7	11	0.9	0.2	415.4	13	26	39	5
								Totals:	132	176	308	41				**Totals:**	136	181	317	39

DWDDFST					***Base Transactional + DW Query***								***Spike in Transactional***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	3	10K	RT	80%< 1s	98%	0.5	667	0.1	25	27	53	5	97%	0.5	653	0.2	23	29	52	3
2	3	100K	RT	80%< 15s	98%	0.5	68	3.6	39	51	90	12	92%	0.6	68	5.3	33	45	77	7
3	4	1M	Vel	30	43	0.7	12	38.3	48	79	126	16	24	1.2	9	69.2	35	56	91	11
4	5	8M	Vel	5	7	0.7	0.1	717.9	8	14	23	3	4	1.2	0.0	1106.2	4	9	13	2
5			Disc		N/A	0	0	0.0	0	0	0	0	N/A	0	0	0.0	0	0	0	0
								Totals:	120	171	291	36				**Totals:**	94	138	233	23

WASTCHI					***Base Transactional + DW Query***								***Spike in Transactional***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	2		RT	90%< 0.25s	98%	0.5	2033	0.012	174	77	251	10	95%	0.6	2381	0.013	221	90	311	21

WASTCLOW					***Base Transactional + DW Query***								***Spike in Transactional***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	3		RT	90%< 0.9s	98%	0.6	45	0.186	11	12	24	1	96%	0.6	44	0.222	12	13	24	1

15.6.3 Observations and conclusions

As anticipated, we witnessed an increase in throughput and processor usage associated with the WASTCHI service class. The WASTCHI service class continued to exceed its WLM goal

(PI was 0.5 and now 0.6). Prior to the spike in transactional workload, the processor utilization was already fairly high, hence with the spike, WLM had to essentially take processing cycles from the lesser importance work. In this case, the processing cycles came primarily from Importance 4 and 5 work (periods 3 and 4), of the DWDDFST service class. Notice the drop in throughput and increase in response time for those periods. Also note that those periods are no longer making their goals. In both cases, the PIs went from 0.7 to 1.2.

You might observe DWDDFHI period 3, which is Importance 3 (lower than WASTCHI), also appear to witness improved performance. We chalk this up to workload variability. Our workload in general had some variation that we estimate to be +/- 5%.

Our conclusion here is that WLM responded as we had hoped, stealing cycles from lower importance work, to accommodate the increased demand of the higher importance work.

15.7 Spike in data warehouse workload

As discussed previously, data warehouse query workloads will generally tend to have much larger variations in demand for resource. One moment the system might be 60% utilized, and a single new parallel query could drive the system to 100% utilization. During these large changes in resource demand, it is important that the system be able to protect the cherished higher importance workloads by continuing to provide them the service required.

15.7.1 Objective

For this experiment, we have the following test scenarios:

- Test A (Base): Rerun our base workload using our base policy.
- Test B (Spike DW query): Run our base workload along with seven large resource consumption IBM Cognos BI reports. Combined, these seven reports could easily consume all of the z/OS system resources. Our service policy was unchanged.

15.7.2 Results

Table 15-13 Spike in data warehouse workload—processor utilization

zOS Processor Utilization	CP	zAAP	zIIP
Base Transactional + DW Query	*98*	*81*	*84*
Spike in DW Query	*100*	*86*	*91*

Table 15-14 Spike in data warehouse workload - Key service class statistics

DWDDFHI					Base Transactional + DW Query								Spike in DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	2	10K	RT	90%< 1s	98%	0.5	705	0.10	25	28	53	4	98%	0.5	693	0.12	27	26	54	6
2	2	100K	RT	80%< 10s	82%	1	67	5	39	51	90	12	83%	1	67	5	40	47	87	13
3	3	1M	Vel	40	55	0.7	12	34	50	78	128	17	58	0.7	11	32	50	68	119	18
4	4		Vel	10	13	0.8	0.1	513	13	26	39	5	10	1	0.1	669	8	16	24	3
								Totals:	127	183	309	38				**Totals:**	126	158	284	40

DWDDFST					Base Transactional + DW Query								Spike in DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	3	10K	RT	80%< 1s	98%	0.5	662.4	0.11	23	28	52	4	98	0.5	672.6	0.12	25	28	53	5
2	3	100K	RT	80%< 15s	98%	0.5	68.0	4	39	51	90	12	92%	0.6	68.0	5	33	45	77	7
3	4	1M	Vel	30	37	0.8	12.0	49	40	67	107	14	29	1	11.0	53	46	72	118	16
4	5	8M	Vel	5	6	0.9	0.1	1285	2	5	7	1	2	2.2	0.0	2293	21	49	70	7
5			Disc		N/A	0	0.0	0.0	0	0	0	0	8	0	0.0	0.0	0.4	1.5	1.9	0
								Totals:	105	151	256	30				**Totals:**	125	194	319	36

WASTCHI					Base Transactional + DW Query								Spike in DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	2		RT	90%< 0.25s	95%	0.6	1948	0.015	201	114	315	35	92%	0.8	1825	0.017	176	105	281	25

WASTCLOW					Base Transactional + DW Query								Spike in DW Query							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	3		RT	90%< 0.9s	96%	0.5	91	0.148	26	18	44	3	92%	0.9	85	0.183	23	19	42	3

15.7.3 Observations and conclusions

The additional seven large queries quickly fall into DWDDFST periods 4 and 5 (Importance 5 and Discretionary), hence we notice the increase in processing utilization in those periods, and the fact we are no longer reaching our goal (PI is now 2.2) for period 4. Additionally, the DWDDFST period 4 average response time has elongated and transactions/minute is so low, it shows as 0 in our table (it is actually 0.03).

Now let us look for any impact to the higher importance, cherished workloads. All the DWDDFHI work is still reaching their goals and their throughput and average response time have remained fairly consistent. Examining the WASTCHI work, we notice we are still achieving our goal (PI = 0.8), though we did witness a drop in processing utilization, with an associated drop in throughput and a slight elongation in response time. WLM is behaving as we hoped it would. It essentially stole some of the previous WASTCHI processing cycles, and is now providing them to DWDDST period 4, while still maintaining all of the higher importance service class goals.

Though WLM is behaving as expected, the drop in throughput for the transactional workload might not be welcome. There might be some work or service class(es) that we never want lower importance work taking processing cycles from. This leads to our next experiment.

15.8 Spike in data warehouse workload - protect transactional

15.8.1 Objective

In this experiment, we're looking to repeat the same experiment as in 15.7, though in this case, we are trying to provide additional protection of work being serviced by the WASTCHI service class. As preparation, we create another WLM policy, CL_WCRIT, in which we override the WASTCHI service class definition. The only definition change we made, was to set CPU Critical =YES. Everything else remained the same. Setting CPU Critical=YES for a given service class will never allow a lower importance piece of work (work classified to a service class with lower importance) to have a higher dispatch priority.

Preparation

Set CPU Critical = YES for service class WASTCHI, as shown in Figure 15-3.

```
Service Policy Name  . . . . : CL_WCRIT
Service Class Name . . . . . : WASTCHI

Override the following information:
Resource Group . . . . . . . . ____      (name or ?)
Cpu Critical . . . . . . . . . YES       (YES or NO)

Action Codes: I=Insert new period, E=Edit period, D=Delete period.

        ---Period---  --------------------Goal---------------------
Action  #  Duration   Imp.  Description

  _     1            2     90% complete within 00:00:00.250
```

Figure 15-3 Setting CPU Critical = YES for WASTCHI service class

15.8.2 Results

Table 15-15 WASTCHI: CPU Critical = YES - processor utilization

zOS Processor Utilization	CP	zAAP	zIIP
WASTCHI CPU Critical = NO	*100*	*86*	*91*
WASTCHI CPU Critical = YES	*100*	*90*	*88*

Archived

Table 15-16 Key service classes for WASTCHI CPU Critical experiment

DWDDFHI					*WASTCHI CPU Critical = No*								*WASTCHI CPU Critical = Yes*							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	2	10K	RT	90%< 1s	98%	0.5	693	0.12	27	26	54	6	98%	0.5	704	0.12	26	29	55	5
2	2	100K	RT	80%< 10s	83%	1	67	5	40	47	87	13	86%	0.9	69	4	40	52	91	12
3	3	1M	Vel	40	58	0.7	11	32	50	68	119	18	49	0.8	12	38	47	73	120	15
4	4		Vel	10	10	1	0.1	669	8	16	24	3	10	0.9	0.1	540	9	18	26	3
								Totals:	126	158	284	40				**Totals:**	122	171	293	34

DWDDFST					*WASTCHI CPU Critical = No*								*WASTCHI CPU Critical = Yes*							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	3	10K	RT	80%< 1s	98%	0.5	673	0.12	25	28	53	5	97%	0.5	666	0.16	23	31	53	3
2	3	100K	RT	80%< 15s	92%	0.6	68	5	33	45	77	7	90%	0.8	69	6	34	50	84	7
3	4	1M	Vel	30	29	1	11	53	46	72	118	16	23	1.3	9	68	39	67	106	11
4	5	8M	Vel	5	2	2.2	0	2293	21	49	70	7	2	2.4	0	0	10	26	36	2
5			Disc		8	0	0.0	0.0	0.4	1.5	1.9	0	N/A	0	0	0	0	1	1	0
								Totals:	125	194	319	36				**Totals:**	94	138	279	24

WASTCHI					*WASTCHI CPU Critical = No*								*WASTCHI CPU Critical = Yes*							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	2		RT	90%< 0.25s	92%	0.8	1825	0.017	176	105	281	25	99%	0.5	2110	0.012	209	131	340	31

WASTCLOW					*WASTCHI CPU Critical = No*								*Spike in Transactional*							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	3		RT	90%< 0.9s	92%	0.9	85	0.183	23	19	42	3	89%	1.1	76	0.307	22	15	37	3

15.8.3 Observations

The setting of WASTCHI to CPU Critical = YES resulted in directing more resources toward WASTCHI, hence improved throughput, response time, and PI. This came at the expense of the lower importance DWDDFST service class periods—their PI, throughput and response time degraded. To repeat what we earlier mentioned, the setting of CPU Critical=YES for a given service class ensures that all its work requests will always run at a higher dispatch priority than less important (WLM *Importance*) work requests. In this experiment, all DWDDFST service class periods have a lower importance than the WASTCHI service class. We also observe that due to this same protection of WASTCHI work, the WASTCLOW service class no longer achieved its goal.

While this might be the desired performance trade-off, this limits WLM's ability to manage toward service class *goals*, which it was designed to do. In this case, WLM is managing dispatch priority of WASTCHI work requests, purely on relative importance, not goals. The general WLM recommendation is to utilize CPU Critical = YES only in cases where it is required.

For a graphical view of the experiment just discussed, see Figure 15-4 on page 364, which is an excerpt from an RMF spreadsheet reporter Workload Activity Trend report.

Note the impact on WASTCHI PI and throughput, once the CL_WCRIT policy was deployed.

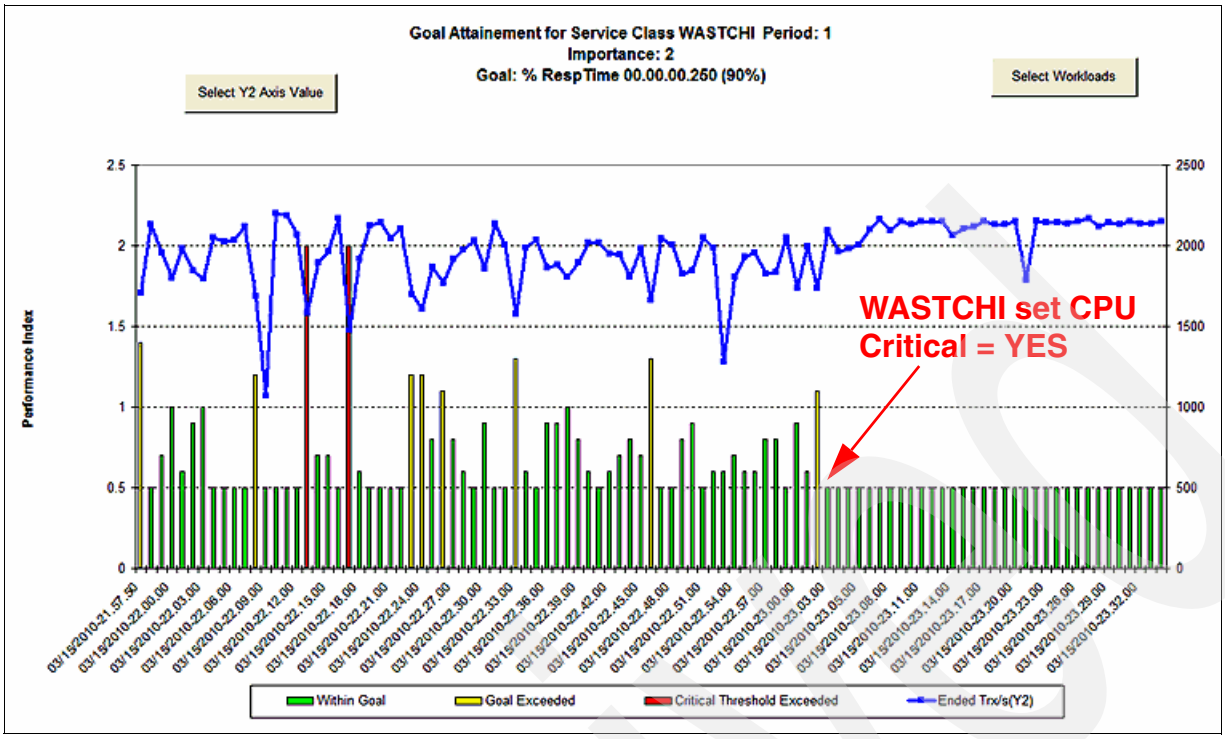


Figure 15-4 WASTCHI PI and throughput with CPU Critical=YES

15.9 zIIP-eligible work to only run on zIIP processors

15.9.1 Objective

In this experiment the objective was to evaluate the impact and value of preventing zIIP-eligible work from running on general purpose processors. The key question here is: will the higher importance data warehouse work be able to tolerate the latency associated with waiting on an available zIIP processor as opposed to running more immediately on a general purpose processor.

15.9.2 Preparation

The z/OS IEAOPTxx default is IIPHONORPRIORITY=YES. This signifies general purpose processors to run both zIIP-eligible and non-zIIP-eligible work in priority order, when the zIIP processors indicate the need for help. Setting the IEAOPTxx IIPHONORPRIORITY=NO signifies general purpose processors to not examine zIIP-eligible work, regardless of the demand for zIIP processors.

For this base experiment, like our previous base experiments, we ran with the default of IIPHONORPRIORITY=YES. Additionally, for this set of experiments we wanted to ensure a high level of demand on the zIIP processors, hence we included the same 7 large queries run in the spike data warehouse experiment. We then set IIPHONORPRIORITY=NO and reran the same workload, including the 7 large queries. This was dynamically accomplished via the MVS command SET OPT=DW (our SYS1.PARMLIB(IEAOPTDW) member contained the IIPHONORPRIORITY=NO statement).

15.9.3 Results

Table 15-17 Processor utilizations

zOS Processor Utilization	CP	zAAP	zIIP
IIPHONORPRIORITY = YES	*100*	*67*	*94*
IIPHONORPRIORITY=NO	*92*	*70*	*100*

Table 15-18 Key service class results

DWDDFHI					***IIPHONORPRIORITY=YES***								***IIPHONORPRIORITY=NO***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	2	10K	RT	90%< 1s	98%	0.5	701	0.12	28	27	55	9	96%	0.5	693	0.19	19	35	54	1
2	2	100K	RT	80%< 10s	90%	0.8	68	4	42	46	88	15	88%	0.8	68	4	30	58	88	3
3	3	1M	Vel	40	61	0.7	11	28	53	67	120	20	56	0.7	11	32	38	84	122	7
4	4		Vel	10	13	0.8	0.1	476	13	19	32	6	12	0.7	0.1	454	10	31	41	0
								Totals:	135	160	295	49				**Totals:**	97	207	304	10

DWDDFST					***IIPHONORPRIORITY=YES***								***IIPHONORPRIORITY=NO***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP	Actual	PI	Trans per min	Avg resp time	CP	IIP	Total	IIPCP
1	3	10K	RT	80%< 1s	98%	0.5	673	0.13	26	28	54	8	95%	0.5	675	0.25	19	35	54	1
2	3	100K	RT	80%< 15s	94%	0.6	70	5	41	48	88	14	92%	0.7	69	6	30	56	85	3
3	4	1M	Vel	30	30	1	11	45	52	74	126	22	26	1.2	9	65	30	68	98	6
4	5	8M	Vel	5	3	1.5	0	1723	31	59	89	13	2	2.4	0	2200	6	20	25	0
5			Disc			0	0.0	0.0	0	0	0	0	N/A	0	0	0	1	4	4.8	0.1
								Totals:	149	208	358	56				**Totals:**	85	182	267	10

WASTCHI					***IIPHONORPRIORITY=YES***								***IIPHONORPRIORITY=NO***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	2		RT	90%< 0.25s	93%	0.7	1843	0.019	153	75	227	5	98%	0.5	2078	0.013	183	78	261	11

WASTCLOW					***IIPHONORPRIORITY=YES***								***IIPHONORPRIORITY=NO***							
Service Class Definition					Goal Results		KPIs		Processor Utilization				Goal Results		KPIs		Processor Utilization			
Per	Imp	Durat (SUs)	Goal type	Goal	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP	Actual	PI	Trans per min	Avg resp time	CP	AAP	Total	AAP CP
1	3		RT	90%< 0.9s	94%	0.7	88	0.188	23	20	43	1	100%	0.5	98	0.096	26	22	48	2

15.9.4 Observations and conclusions

Observing processor utilization for the base experiment we notice the general purpose processors (CPs) are saturated at 100% utilization, and our zIIP processors are also highly utilized at 94%. Since we included the 7 large queries, this was anticipated. For the IIPHONORPRIORITY=NO experiment, we notice a drop in CP utilization along with an increase in zIIP utilization. Because we are no longer allowing zIIP-eligible work to run on CPs, this again is consistent with our expectations.

Observing the DWDDFHI processor utilization totals, the total processor utilization remained consistent (295 vs 304), yet the breakdown of CP vs IIP utilization changed, essentially moving the processing cycles from the CPs to the IIPs. We also notice a drop in IIPCP utilization. Examining the DWDDFHI service class, there is a slight decrease in throughput, and an associated increase in average response time (going from 0.12 to 0.19 seconds).

Even though there was some increase in latency, WLM goals are still being met for all periods of higher importance DWDDFHI service class.

Observing the DWDDFST processor utilization totals, there is a substantial drop (almost one processor) in total utilization (358 vs 267). This is predominantly based on a drop in CP utilization in the lower importance periods (3 and 4), due to the fact we are out of zIIP capacity and are no longer allowing the zIIP work to run on CPs. Related to this drop in CP utilization, there is degradation in the PIs, average response time, and throughput.

Observing the transactional workload service classes, WASTCHI and WASTCLOW, there is an increase in processor utilization, primarily coming from CP. We also witness an increase in throughput, reduction in average elapsed time, and improved PIs. This was an expected outcome, given the reduction in CP utilization coming from the data warehouse workloads.

Note: One might ask why there remains some IIPCP utilization, even with IIPHONORPRIORITY=NO. The documentation states that this is due to potential contention on a z/OS suspend lock, which would still run on a general purpose processor, even if it was zIIP-eligible.

From these results, we conclude that it is quite reasonable to consider utilizing IIPHONORPRIORITY=NO for data warehouse workloads, especially when there are no query requirements for subsecond response times, where minor increases to latency can be tolerated.

15.10 Dynamic WLM management of SQW refresh

We conducted a preliminary set of measurements to examine the use of multiple WLM policies for managing critical refresh workloads. We did not have time to run our complete measurement methodology for this, but we are confident in our findings, so we are providing our preliminary results.

At points in the business day or week, there might be times that certain data warehouse refresh activities should take precedence over query activities. Based on this, we designed a pair of experiments where all workloads were running simultaneously. This consisted of the transactional and data warehouse query workload described previously in this chapter, along with InfoSphere Change Data Capture (CDC) and InfoSphere for z SQW refresh workloads discussed in Chapter 8, “Our test configuration and infrastructure” on page 207.

In experiment 1 we ran with a WLM policy (CL_OPTML) that managed SQW refresh at a WLM importance lower than all of the queries, accept the last two periods in the DWDDFST service class. In experiment 2 we ran with a WLM policy (CL_RFRSH) that managed SQW refresh at equal importance of the 1st and 2nd periods of the DWDDFHI service class, hence higher than all the rest of the query work. Figure 15-19 illustrates the policy definition differences for the DWDDFREF service class, responsible for the SQW refresh workload.

Table 15-19 SQW refresh service class definitions for different WLM policies

	Service Class DWDDFREF			
Service Policy	Period	Importance	Type	Goal
CL_OPTML	1	4	Velocity	30
CL_RFRSH	1	2	Velocity	70

In experiment 1, using the CL_OPTML policy, the SQW refresh workload completed in 1 hour and 25 minutes. In experiment 2, using the CL_RFRSH policy, the SQW refresh workload

completed in 30 minutes. The measurements exhibited the value and ability to use WLM dynamically for adjusting the level of service for refresh workloads, concurrent with query.

15.11 Summary

In our project, we executed a set of experiments to understand WLM's ability to effectively manage system resources when co-locating transactional and data warehouse query workloads in a single LPAR. These were our focus areas:

- Ability to respond to dynamic large increases in demand (spikes) for processing resources from either transactional or data warehouse query workloads.
- Differentiate and provide a higher level of service to the business-critical data warehouse knowledge workers over the intermediate and novice knowledge workers.
- Managing the shorter-running small consumption queries to WLM response time goals.
- Dynamically manage data warehouse refreshes concurrently with query workloads.

The results of our measurements boosted our confidence in using WLM to effectively manage system resources for a mix of transactional and data warehouse query workloads within a single LPAR. In order to obtain satisfactory performance for both workloads, our experience suggests that the single LPAR environment will likely take more workload analysis and policy iterations than running the two workloads in separate LPARs. Additionally, in the single LPAR environment, there is more likely a need to go beyond the use of traditional WLM goals, and make use of WLM's "protection of work" functions, such as CPU Critical.

By the nature of their work, data warehouse and business intelligence workloads tend to require a lot of processor, I/O, and potentially memory resources. Additionally, growth of users and applications in this arena is dynamic. Hence, if deciding to co-locate this work with transactional workloads on the same LPAR, you must ensure that your WLM service definitions are properly aligned with the business priorities. (For tips on doing this, see Chapter 11, "Resource management of data warehouse mixed workloads" on page 263.) Ensuring that there are enough resources available to satisfy the important transactional and data warehouse cherished workloads will result in satisfied clients.

Archived

Archived

16

Multiple z/OS LPAR experiments

In this chapter we describe the results of our project experiments using multiple LPARs.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

16.1 Introduction

Running transactional workloads in multiple LPARs offers higher availability. The failure of one LPAR does not lead to complete outage of the applications. The same configuration also applies to data warehouse applications. As data warehouse applications become more pervasive in an enterprise, their availability becomes more important to deliver competitiveness in their business. With the advent of operational BI, an outage of a data warehouse application could lead to loss of customer loyalty. Thus it is becoming more critical to run the applications in multiple LPARs so that failure of one LPAR still allows business operations to continue.

Chapter 15 provides insight to workload performance in a single LPAR environment. But how about running the mixed workloads in multiple LPARs? Would it make a difference? The measurements described in this chapter were designed to answer this question.

16.2 Topology

The multiple LPAR measurements utilized a 4-LPAR configuration. There was a DB2 member in each LPAR forming a 4-way data sharing group. The operational workloads ran in two of the LPARs, while the business intelligence workloads ran in the other two LPARs. See Figure 16-1.

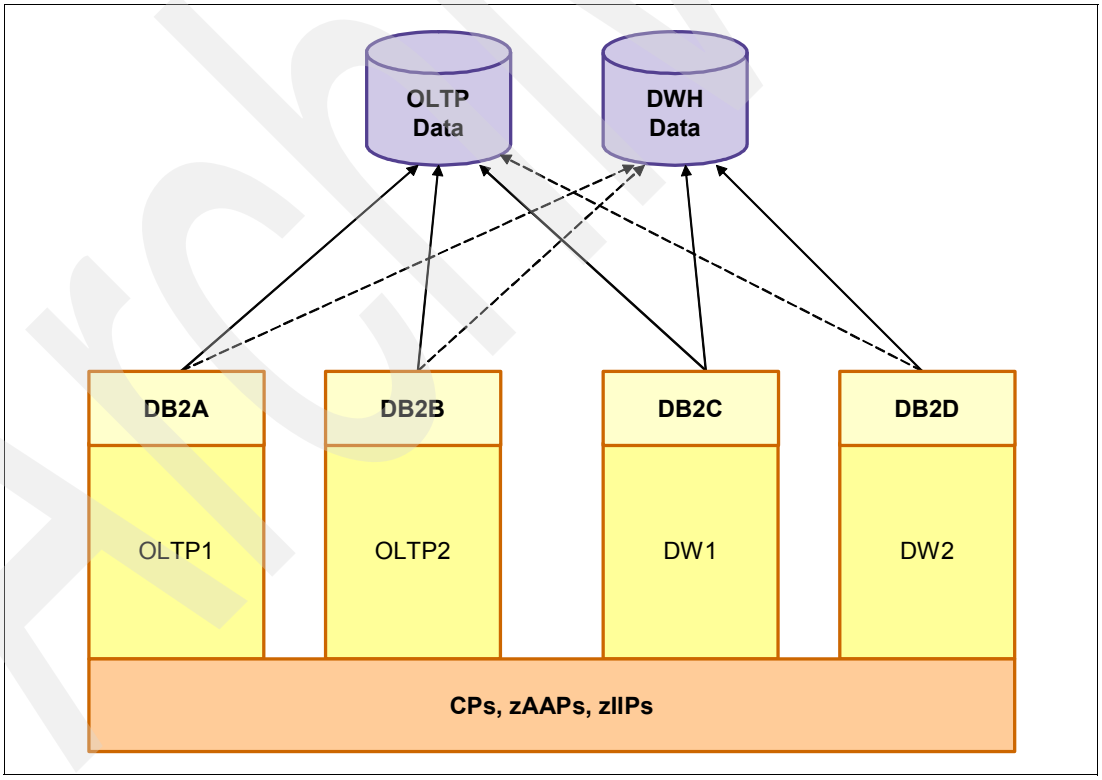


Figure 16-1 Measurement configuration

In theory, each LPAR was capable of accessing all the data maintained in the data sharing group. However, the workloads were segregated so that the OLTP LPARs accessed the OLTP database only. The queries running on the data warehouse LPARs accessed all their data

from the data warehouse only. The solid arrows show the data accesses in the measurements while the dashed arrows show the additional possible data accesses.

There is a solid arrow from the first data warehouse LPAR to the operational databases. This stems from two ETL processes that were active during the measurements. CDC was used to capture updates of the inventory table from the operational side. Although there was not any direct access to the inventory table, reading the DB2 logs still constituted an access to the operational data. A second ETL process, driven by a batch program running on DW1, read data directly from the order and book order tables to populate the fact table in the data warehouse.

The Linux configuration supporting IBM Cognos BI was identical to what we used in the single LPAR measurements. Descriptions about that configuration can be found in Chapter 8, “Our test configuration and infrastructure” on page 207.

16.3 Key performance metrics

We used several key metrics to demonstrate the performance characteristics of a workload, and they are defined here.

16.3.1 TPS

TPS stands for transactions per second, measuring the throughput of a workload. It is commonly used for transactional workloads. Two service classes were set up to capture performance data of the operational workload. High priority transactions were managed in one service class while the low priority transactions were captured in another. The TPS metric is available in the Workload Activity section of an RMF Monitor I report.

By monitoring changes of TPS between two measurements, one can determine if the changes introduced in the second measurement produce a positive or negative effect. Other metrics such as response times also paint a picture of the effect, but TPS generally gives a first indication of the performance change.

16.3.2 Queries per minute (QPM)

Since business intelligence queries typically take seconds or longer to run while transactions complete in sub-seconds, TPS is not appropriate for business intelligence workloads. Instead, a metric, such as queries per minute, is more suitable. Similar to TPS, changes in this metric between measurements indicate the effectiveness of the tuning effort.

Obtaining the metric values is straightforward. Two service classes were used to distinguish between high and low priority business intelligence queries. In addition, four service periods were defined in each service class. Short queries completed during period 1, while long running queries dropped to period 4. In all there were eight categories of business intelligence queries based on the periods they were in at completion time. Queries per second data is directly available in the Workload Manager section of an RMF Monitor I report. This can easily be converted to queries per minute.

16.3.3 Response time

Response times for both the operational and business intelligence workloads can be obtained from the Workload Activity section of an RMF Monitor I report. In the case of the operational

workload, queuing delay data is also available. Since WebSphere was used as the application server, delays due to all worker threads in the servant regions that were busy could be pinpointed easily.

16.3.4 Processor utilizations by workloads

Processor consumption by a workload is a good indicator of the amount of work processed. To the extent that there are no scalability issues, a higher level of processor consumption suggests a higher throughput. When the input workload level is increased, not only is throughput expected to be increased, but so is processor consumption. An increase in throughput without an accompanying increase in processor consumption implies better than linear scalability for an application. This is rather uncommon and should be investigated to rule out measurement errors. Likewise, an increase in processor consumption without a corresponding increase in throughput indicates scalability issues, which should also be examined.

Values of this metric are readily available in the Workload Activity section of an RMF Monitor I report. There is one value for each service period in each service class.

16.3.5 Total processor utilization

This metric is simply an extension to the processor utilization by workload values. Aggregating processor consumption across all workloads, including system component activities, gives an overall picture of system behavior. An increase in the workload level should be accompanied by an increase in total processor consumption. All the ramifications listed in the processor utilization by workload metric section apply here as well.

Values of this metric can be obtained from the processor section of an RMF Monitor I report.

16.4 Measurements

Three sets of measurements were performed for the multiple LPAR configuration. They consisted of (1) running the OLTP workloads only, (2) running OLTP and business intelligence workloads simultaneously, and (3) running OLTP and business intelligence workloads but with an increased workload level in OLTP.

16.4.1 OLTP only

Processor configuration

There were 18 physical processors, including 10 CPs, four zIIPs, and four zAAPs. The number of virtual processors for each LPAR is shown in Table 16-1.

Table 16-1 LPAR weight assignment

	General processor		zAAP		zIIP	
LPAR	CP	CP weight	zAAP	zAAP weight	zIIP	zIIP weight
OLTP1	3	180	2	40		
OLTP2	3	180	2	40		
DW1	2	80			2	40

DW2	2	80			2	40

The number of virtual processors was the same as the number of physical processors. This made the LPAR weight assignment irrelevant.

Only the OLTP1 and OLTP2 LPARs were active during this measurement. The two data warehouse LPARs were online. However, no work was sent to these LPARs.

Analysis

Table 16-2 shows the performance results of this measurement.

Table 16-2 OLTP workloads performance metrics - OLTP only

		OLTP1	**OLTP2**
	CPU (3)	54.51%	56.43%
	zAAP (2)	38.94%	39.58%
High priority OLTP			
RMF	TPS	1,271	1,271
	Response time	9	10
	CPU utilization	85.75%	88.79%
	zAAP utilization	45.49%	47.00%
	Total utilization	131.24%	135.79%
DB2	Getpages / trx	44	44
Low priority OLTP			
RMF	TPS	50	50
	Response time	71	74
	Execution	67	70
	CPU utilization	9.57%	9.59%
	zAAP utilization	10.63%	10.91%
	Total utilization	20.20%	20.50%

TPS was at 1,271 for each LPAR. Response times were very good, hovering around 9 to 10 milliseconds. Processor utilization for the high-priority OLTP workload was between 130% and 135%, with a majority of the cycles consumed by the general processors. This implies SQL processing takes up a larger share of the processor cycles while business logic written in Java takes a smaller slice. Average getpages per transaction came out to 44, indicating DB2 resource consumption was not on the heavy side.

Each transaction of the low-priority OLTP workload placed a heavier load on the system, but at the workload level it consumed fewer processor cycles than its high-priority cousin. Each transaction performed a purchase of a book, requiring updates to the database. On the other hand, the high-priority workload performed browsing only, which made it a read-only workload. TPS was at 50 only due to a smaller percentage in the input workload mix. Response time was noticeably higher because of the complexity of the transactions. Although not shown in the figure, processor cost per transaction was also higher than its counterparts in the high priority OLTP workload.

Overall general processor utilization was at the mid-50s and zAAP processor utilization was around 40%. This is far from saturation level, suggesting that a higher volume of transactions can be handled with ease.

16.4.2 OLTP with data warehouse

It used the same configuration as the OLTP-only measurement. The number of virtual processors in each data warehouse LPAR was slightly less than the OLTP LPARs. The intention was to simulate a production environment where the OLTP workloads have a higher level of significance and therefore more resources are allocated for their consumption. Again the LPAR weights were not a factor since the number of virtual and physical processors was identical.

OLTP workload analysis

Table 16-3 shows the results of this measurement.

Table 16-3 OLTP workloads performance metrics - OLTP with data warehouse

		OLTP1	OLTP2
	CPU (3)	75.04%	63.76%
	zAAP (2)	42.70%	45.17%
High priority OLTP			
RMF	TPS	1,235	1,246
	Response time	22	18
	CPU utilization	93.12%	104.15%
	zAAP utilization	51.05%	53.90%
	Total utilization	144.17%	158.05%
DB2	Getpages / trx	44	44
Low priority OLTP			

RMF	TPS	47	48
	Response time	152	129
	CPU utilization	9.36%	10.54%
	zAAP utilization	12.50%	14.47%
	Total utilization	21.86%	25.01%

TPS drops slightly from 1,271 to 1,240, but the difference is small enough that this could be due to measurement fluctuations. In this sense, the change to TPS is insignificant.

A bigger difference is observed for the processor utilization of the high-priority transaction workload. It was 133.5% during the OLTP-only run but it went up to 151.1% when the data warehouse LPARs were added to the configuration. For the OLTP-only run, there were 5 engines (3 general processors + 2 zAAPs) in each LPAR plus the 8 engines assigned for the z/VM LPAR, yielding a total of 18 engines. As a result, each engine is running at a speed equivalent to an 18-way server. When the data warehouse LPARs were activated, 8 more engines were added to the server, yielding a total of 26 engines. With more engines in the server, the MIPS rating of each engine drops slightly. Based on the processor ratings for a z10 EC server, an engine in a 26-way machine is rated at 7% slower than the same engine in an 18-way machine.

There is a change of 13% ((151.1 – 133.5) / 151.1) in processor utilization. This explains that the increase in processor utilization was most likely due to the higher number of engines in the OLTP + DW run.

Transaction response time increases from 10 ms to 20 ms. Although it looks large from a percentage standpoint, the increment is only 10 ms, which is not significant or noticeable for the online users. In addition, with slower engines in the OLTP + DW run, the increase in processor time per transaction accounts for a portion of the response time increase.

The lower-priority transactions exhibit a similar pattern. There are no significant changes in the TPS or processor utilization. There is a noticeable jump in transaction response times. Part of that is due to the slower engines in the OLTP + DW run, and part of it is due to the combination of higher system processor utilization and lower priority of the transactions.

Data warehouse workload analysis

The data warehouse workload consists of two sets of queries: a higher priority set and a lower priority set. Each set is further divided into four categories based on their time of completion. One way to categorize these queries is to label them as small, medium, large, and extra large queries. Small queries take the least time to execute and they complete in period 1. Medium queries take slightly longer and they end in period 2. Extending this concept, large queries complete in period 3 and extra large queries end in period 4. As observed in production workloads, the small category constitutes the largest percentage of the queries, followed by medium, large, and extra large. The mix of these query categories in this study is based on production client workload profiles; see Table 16-4 on page 376.

Table 16-4 Data warehouse workloads performance metrics - OLTP with data warehouse

		DW1	DW2
	CPU (4)	37.26%	37.33%
	zIIP (4)	34.67%	36.11%
High priority Data Warehouse			
Period 1	QPM	422	356
	Response time	320	335
	CPU	95	79
	zIIP	90	73
	CPU utilization	7.72%	6.40%
	zIIP utilization	7.28%	5.91%
	Total utilization	15.00%	12.31%
Period 2	QPM	34	28
	Response time	2,528	2,802
	CPU	42	39
	zIIP	58	52
	CPU utilization	3.39%	3.16%
	zIIP utilization	4.70%	4.22%
	Total utilization	8.09%	7.38%
Period 3	QPM	5	5
	Response time	14,390	17,329
	CPU	386	397
	zIIP	464	460
	CPU utilization	31.25%	32.17%
	zIIP utilization	37.60%	37.26%
	Total utilization	68.85%	69.43%

Period 4	QPM	0	0
	Response time	136,366	115,882
	CPU	19	29
	zIIP	17	23
	CPU utilization	1.58%	2.33%
	zIIP utilization	1.38%	1.85%
	Total utilization	2.96%	4.18%
Low priority Data Warehouse			
Period 1	QPM	364	297
	Response time	850	659
	CPU	114	94
	zIIP	113	89
	CPU utilization	9.22%	7.63%
	zIIP utilization	9.14%	7.20%
	Total utilization	18.36%	14.83%
Period 2	QPM	46	38
	Response time	5,828	4,316
	CPU	104	94
	zIIP	182	146
	CPU utilization	8.44%	7.64%
	zIIP utilization	14.72%	11.80%
	Total utilization	23.16%	19.44%
Period 3	QPM	8	8
	Response time	39,467	50,912
	CPU	499	430
	zIIP	572	483
	CPU utilization	40.42%	34.86%
	zIIP utilization	46.34%	39.12%
	Total utilization	86.76%	73.98%

Archived

Period 4	QPM	1	1
	Response time	243,608	235,869
	CPU	120	238
	zIIP	182	418
	CPU utilization	9.69%	19.25%
	zIIP utilization	14.78%	33.84%
	Total utilization	24.47%	53.09%

System processor utilization was not high in this measurement, slightly less than 40% for both the general purpose and the zIIP processors. More work could be processed in the systems if the input arrival rates were higher.

The combined throughput of the high-priority small queries from the two LPARs was quite high, coming out to 778 per minute, or about 47,000 per hour. This demonstrates that the systems were capable of handling a large load of data warehouse queries, especially since the systems were running at 40% busy only. It may come as a surprise to some readers, but analysis of customer data indicates a majority of their production data warehouse queries fall into this category. As such, a system designed to support high concurrency workloads will be a good fit for data warehousing.

Response times of the small queries were good, coming in at about 0.3 seconds, with about 48% of the processing offloaded to the zIIP processors. Beyond the zIIP offload due to distributed requests, parallel queries enjoyed additional offload opportunities. In general, the range of offloads was between 45% and 70% based on observations from customer data and internal studies. The higher percentage of time that a query runs in parallel, the larger is the offload.

Moving down the categories, queries take longer to complete and their throughputs are smaller.

The lower-priority set of queries exhibits behaviors similar to their high-priority counterparts, except that they are more likely to experience performance degradation if the systems run into processor constraint conditions.

16.4.3 Spike OLTP

This set of measurements used the same configuration as 16.4.2, “OLTP with data warehouse” on page 374. The objective of the test was to determine whether running the operational transactions and data warehouse queries in a mixed workload environment would present problems to the transactions if there is a spike in their arrival rates. Since these two workloads shared a DB2 data sharing group, it was important to find out whether resource contention might lead to a difficulty of the transactional systems to take on more work. Resource contention could come in two forms, either from database resources or from processor capacity, or both.

The data warehouse workload only read from the operational database. This was coming from an ETL program extracting data from the operational database and writing it to the data warehouse. A second stream of ETL processing, implemented via Change Data Capture (CDC), read the DB2 operational system logs and replicated the data in the data warehouse systems. Neither of these ETL processes were expected to cause database resource contention, but it was important to have this point proven using measurement results.

Similarly, the two workloads ran on different LPARs so we could avoid processor capacity interference.

OLTP workload analysis

Table 16-5 shows the baseline performance data for the transactions.

Table 16-5 OLTP workloads performance metrics - baseline

		OLTP1	OLTP2
	CPU (3)	52.87%	40.71%
	zAAP (2)	25.75%	22.77%
High priority OLTP			
RMF	TPS	1,240	1,249
	Response time	22	17
	CPU utilization	137.02%	137.06%
	zAAP utilization	55.16%	51.04%
	Total utilization	192.18%	188.10%
DB2	Getpages / trx	44	44
Low priority OLTP			
RMF	TPS	45	47
	Response time	212	145
	CPU utilization	13.04%	13.02%
	zAAP utilization	18.57%	15.56%
	Total utilization	31.61%	28.58%

Table 16-6 on page 380 shows the performance data for the transactions with a higher arrival rate.

Archived

Table 16-6 OLTP workloads performance metrics - spike OLTP

		OLTP1	OLTP2
	CPU (3)	57.03%	53.55%
	zAAP (2)	27.61%	28.93%
High Priority OLTP			
RMF	TPS	1,543	1,724
	Response time	91	45
	CPU utilization	166.37%	183.95%
	zAAP utilization	72.27%	71.05%
	Total utilization	238.64%	255.00%
DB2	Getpages / trx	44	44
Low priority OLTP			
RMF	TPS	26	42
	Response time	1,204	308
	CPU utilization	7.49%	12.34%
	zAAP utilization	7.14%	12.47%
	Total utilization	14.63%	24.81%

Higher priority transaction throughput increased from 2,489 to 3,268 TPS, indicating a 31% improvement. Workload level for the data warehouse queries remained constant. This shows that running OLTP and DW workloads in a data sharing group does not lead to resource contention preventing additional OLTP transactions to be processed.

Average response time for the transactions did shoot up from 20 to 68 ms. However, most of the increase came from queuing time. As more transactions came into the systems, additional WebSphere servant threads were needed to handle the higher load. Due to the short time window allocated for these measurements, it was not possible to do further tuning. Queuing time will be dropped significantly if additional threads are allocated.

Total processor utilization went up from 308.28% to 496.64% to process more work. To determine whether any scalability issues were present in the systems, simply divide total processor utilization by TPS to compute the processor to process one TPS. For the baseline case, this came out to 0.001528, and for the spike OLTP case it came out to 0.001511. They are virtually identical, indicating higher levels of throughput require linearly higher levels of processor utilization. No scalability issues were found with these measurements.

Since the intention was to increase the throughput of the high-priority transactional workload, the lower-priority transactions did show some performance impact. These two levels of transactions shared the same WebSphere servant regions. Because most of the threads were allocated to the high-priority transactions, the lower-priority transactions experienced

queuing delays, leading to longer response times and lower throughput. The number of users driving the lower-priority transactions did not change for the “Spike OLTP” measurement, but a larger number of users were used to drive the higher-priority transactions to simulate a higher arrival rate.

Data warehouse workload analysis

Because there were eight categories of queries in the data warehouse workload, it was easier to show query performance in bar charts instead of a table.

Figure 16-2 shows the throughput of the data warehouse queries. For completeness, throughput data for the OLTP workloads is also included. The left bars come from the baseline measurement and are set with a value of 1. Results from the “Spike OLTP” measurement are normalized against the baseline data and are shown by the right bars. A value larger than 1, as indicated by a bar longer than the baseline, depicts an increase in throughput. As stated previously, there was an increase in the high-priority transaction throughput.

Throughputs across the four periods for the high-priority data warehouse queries stayed relatively constant. Given that the differences between the baseline and the “Spike OLTP” measurements strictly come from the high-priority transactions arrival rates, there is no reason to expect the performance characteristics of the high-priority data warehouse queries to exhibit any significant difference.

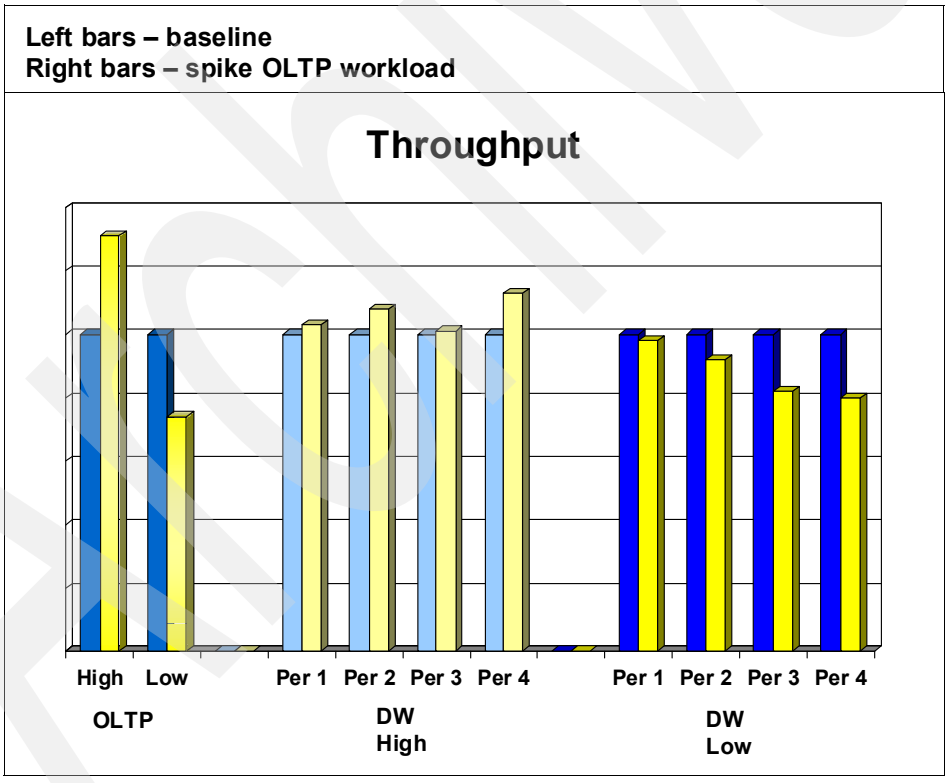


Figure 16-2 Throughput

The higher response times for the transactional workloads were explained in the previous section. For the high-priority data warehouse workloads, there is no significant difference in response times; see Figure 16-3 on page 382.

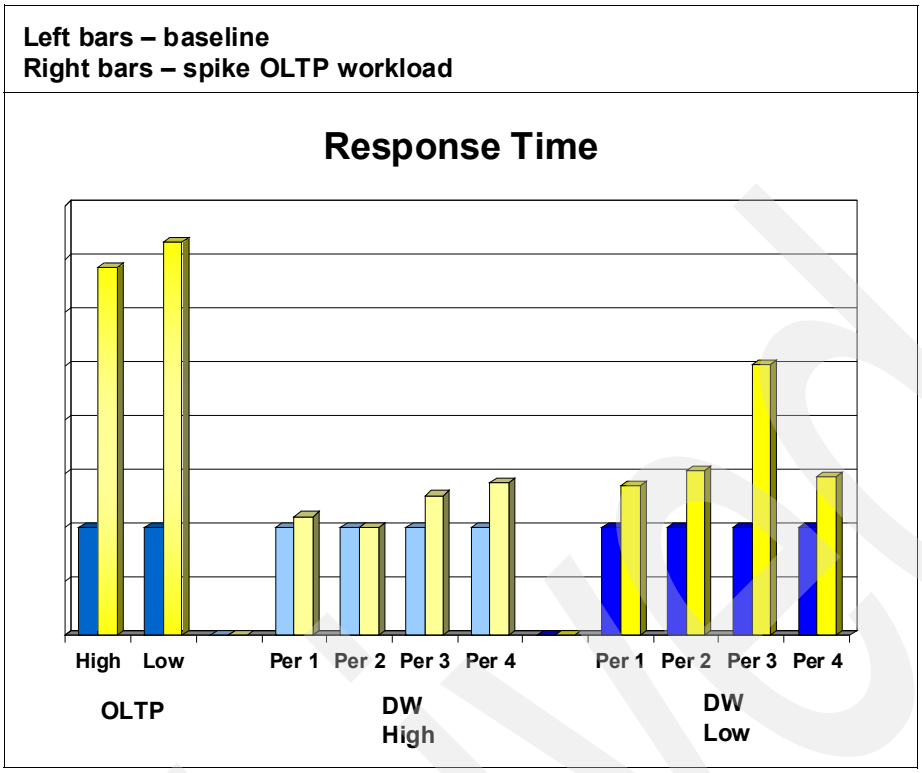


Figure 16-3 Response time

As expected, there is not much difference between the two sets of measurements regarding processor consumption by the high-priority data warehouse queries; see Figure 16-4.

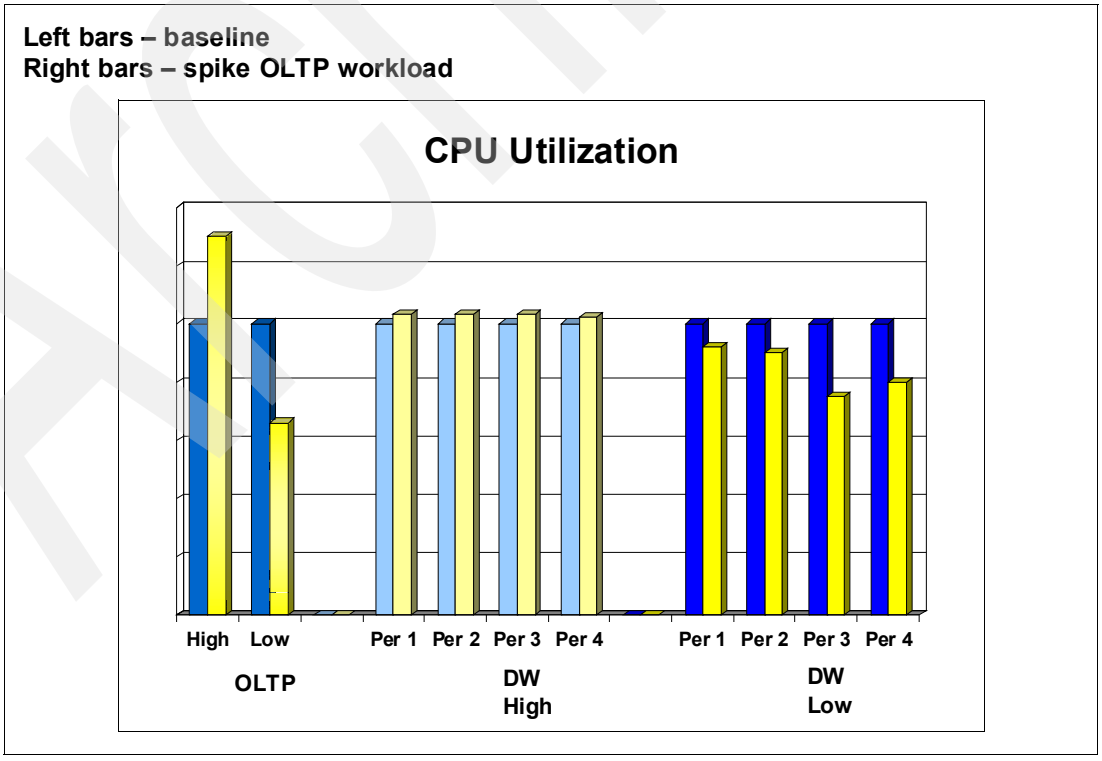


Figure 16-4 Processor utilization

Archive

Overall, the lower-priority data warehouse queries experience some degree of performance degradation, probably due to the combination of CPU contention and their lower priority.

16.5 Summary

The landscape of the business intelligence world is shifting, and the applications are becoming more mission-critical. A growing number of clients are starting to consider running their DW applications on multiple LPARs. Some would even insist on using at least three LPARs to minimize the possibility of an outage.

Chapter 15, “Single z/OS LPAR topology experiments” on page 347 provides details of the performance characteristics of the OLTP and DW applications in a single LPAR configuration. However, to increase availability of the applications, multiple LPARs are recommended. Measurement results from this study show very similar performance when deploying the applications in a single or multiple LPARs.

In addition, adding DW LPARs to the existing OLTP LPARs and forming a large data sharing group does not lead to any performance interference with the OLTP applications. In return, co-locating the OLTP and DW applications in one data sharing group opens up opportunities for more efficient ETL processing and additional query data access capabilities. It does require crafting a Workload Manager policy to handle mixed workloads. The technique of designing such a policy is described in this book. Armed with this knowledge, installations can take full advantage of co-location in a multi-LPAR environment

Archive

Archived

Part 5

Appendixes

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Archived

A

InfoSphere Change Data Capture DDL and binds

In this appendix we list the DDL that was used to create the InfoSphere Change Data Capture metadata tables; see Example A-1. We also list the job that was used to bind the plans used by InfoSphere Change Data Capture; see Example A-2 on page 408.

Example A-1 DDL for InfoSphere Change Data Capture metadata tables

```
CREATE DATABASE CDCDB
    STOGROUP BOOKDW1
    BUFFERPOOL BP0;

  CREATE TABLESPACE CDCTS32K
    IN CDCDB
    SEGSIZE 4
    BUFFERPOOL BP32K
    LOCKSIZE ROW
    LOCKMAX SYSTEM
    CLOSE NO
    USING STOGROUP BOOKDW1
      PRIQTY 240
      SECQTY 20
      ERASE NO
    FREEPAGE 1
    PCTFREE 20
    CCSID UNICODE;

  CREATE TABLE CHC01.DMMD_DMCONFAUD
  (
    CNFTIME            TIMESTAMP              NOT NULL,
    SRCTIME            TIMESTAMP              NOT NULL,
    SRCSYSID           CHAR(8)                NOT NULL,
    SRCSCHEMA          VARCHAR(137)           NOT NULL,
    SRCNAME            VARCHAR(128)           NOT NULL,
    SRCMEMBER          CHAR(10)               NOT NULL,
```

© Copyright IBM Corp. 2010. All rights reserved.

```
  TGTSCHEMA          VARCHAR(128)            NOT NULL,
  TGTNAME            VARCHAR(128)            NOT NULL,
  OPTYPE             INTEGER                 NOT NULL,
  CNFTYPE            INTEGER                 NOT NULL,
  RESMTD             INTEGER                 NOT NULL,
  CNFRES             CHAR                    NOT NULL,
  BEFORETRNC         CHAR                    NOT NULL,
  BEFOREIMG          VARCHAR(8000)           NOT NULL,
  AFTERTRNC          CHAR                    NOT NULL,
  AFTERIMG           VARCHAR(8000)           NOT NULL,
  TGTTRNC            CHAR                    NOT NULL,
  TGTIMG             VARCHAR(8000)           NOT NULL,
  WINTRNC            CHAR                    NOT NULL,
  WINIMG             VARCHAR(8000)           NOT NULL,
  CHECK(OPTYPE       BETWEEN 1 AND 3),
  CHECK(CNFTYPE      BETWEEN 1 AND 4),
  CHECK(RESMTD       BETWEEN 1 AND 5),
  CHECK(CNFRES       IN ('Y', 'N')),
  CHECK(BEFORETRNC   IN ('Y', 'N')),
  CHECK(AFTERTRNC    IN ('Y', 'N')),
  CHECK(TGTTRNC      IN ('Y', 'N')),
  CHECK(WINTRNC      IN ('Y', 'N'))
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS32K;
  COMMIT;

  CREATE TABLESPACE CDCTS4K
  IN CDCDB
  SEGSIZE 4
  BUFFERPOOL BP0
  LOCKSIZE ROW
  LOCKMAX SYSTEM
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 240
    SECQTY 20
    ERASE NO
  FREEPAGE 1
  PCTFREE 20
  CCSID UNICODE;
COMMIT;

CREATE TABLE CHC01.DMMD_AUTH_DB
(
  USER_ID            CHAR(8)                  NOT NULL,
  DBNAME             VARCHAR(24)              NOT NULL,
  DBREAD_AUTH        SMALLINT                 NOT NULL,
  DBWRITE_AUTH       SMALLINT                 NOT NULL,
  DBCREATE_AUTH      SMALLINT                 NOT NULL,
  UPDATE_TS          TIMESTAMP         DEFAULT NOT NULL,
  UNIQUE (USER_ID, DBNAME)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;
```

Archived

```
CREATE UNIQUE INDEX CHCO1.DMMD_UC_AUTH_DB_PIDX
  ON CHCO1.DMMD_AUTH_DB
    (USER_ID ASC,
     DBNAME  ASC)
  BUFFERPOOL BPO
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE TABLE CHCO1.DMMD_AUTH_SUB
(
  USER_ID            CHAR(8)                  NOT NULL,
  TYPE               SMALLINT                 NOT NULL,
  NAME               CHAR(10)                 NOT NULL,
  AUTHORIZED         SMALLINT                 NOT NULL,
  GROUP_ID           CHAR(8)                  NOT NULL,
  UPDATE_TS          TIMESTAMP         DEFAULT NOT NULL,
  UNIQUE (USER_ID, TYPE, NAME)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_AUTH_SUB_PIDX
  ON CHCO1.DMMD_AUTH_SUB
    (USER_ID ASC,
     TYPE    ASC,
     NAME    ASC)
  BUFFERPOOL BPO
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE INDEX CHCO1.DMMD_UC_AUTH_SUB_IDX2
  ON CHCO1.DMMD_AUTH_SUB
    (TYPE    ASC,
     NAME    ASC)
  BUFFERPOOL BPO
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE TABLE CHCO1.DMMD_AUTH_USER
(
  USER_ID            CHAR(8)                  NOT NULL,
  USER_TYPE          SMALLINT                 NOT NULL,
  SYSREAD_AUTH       SMALLINT                 NOT NULL,
  SYSWRITE_AUTH      SMALLINT                 NOT NULL,
  SYSCREATE_AUTH     SMALLINT                 NOT NULL,
  LAST_LOGON_TS      TIMESTAMP                NOT NULL,
  UPDATE_TS          TIMESTAMP         DEFAULT NOT NULL,
  UNIQUE (USER_ID)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_AUTH_USER_PIDX
  ON CHCO1.DMMD_AUTH_USER
```

Archived

```
    (USER_ID ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE TABLE CHC01.DMMD_USER_GROUP
(
  USER_ID            CHAR(8)                   NOT NULL,
  GROUP_ID           CHAR(8)                   NOT NULL,
  ADDED_TS           TIMESTAMP         DEFAULT NOT NULL,
  UNIQUE (USER_ID, GROUP_ID)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_USER_GROUP_PIDX
  ON CHC01.DMMD_USER_GROUP
    (USER_ID  ASC,
     GROUP_ID ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE INDEX CHC01.DMMD_UC_USER_GROUP_IDX2
  ON CHC01.DMMD_USER_GROUP
    (GROUP_ID ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE TABLE CHC01.DMMD_JOURNAL_STATS
(
  TGTNAME            CHAR(10)                  NOT NULL,
  JOURNALID          VARCHAR(90)               NOT NULL,
  BOOKMARK           VARCHAR(90)  FOR BIT DATA NOT NULL,
  BMKTIMESTAMP       TIMESTAMP                 NOT NULL,
  RESETBMK           SMALLINT                  NOT NULL,
  UNIQUE (TGTNAME, JOURNALID),
  CHECK (RESETBMK BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_JOURNAL_STATS_IDX1
  ON CHC01.DMMD_JOURNAL_STATS
    (TGTNAME       ASC,
     JOURNALID     ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;
```

Archived

```
CREATE TABLE CHCO1.DMMD_SCHEMA
(
  TBLSCHEMA          VARCHAR(137)       UNIQUE NOT NULL,
  SCHEMAALIAS        VARCHAR(30)        UNIQUE NOT NULL,
  SEND_ALIAS         SMALLINT                  NOT NULL DEFAULT 0,
  CHECK (SEND_ALIAS BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_SCHEMA_IDX1
  ON CHCO1.DMMD_SCHEMA
    (TBLSCHEMA     ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_SCHEMA_IDX2
  ON CHCO1.DMMD_SCHEMA
    (SCHEMAALIAS     ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHCO1.DMMD_TARGET_ID
(
  TGTNAME            CHAR(10)           UNIQUE  NOT NULL,
  TGTSYSDESC         VARCHAR(50)        DEFAULT NOT NULL,
  SRCSYSID           CHAR(8)                    NOT NULL,
  SRCSYSDESC         VARCHAR(50)        DEFAULT NOT NULL,
  PENDINGSYSID       CHAR(8)            DEFAULT NOT NULL,
  PENDINGSYSDESC     VARCHAR(50)        DEFAULT NOT NULL,
  TGTSYSTYP          CHAR(10)                   NOT NULL,
  TGTDBTYPE          SMALLINT                   NOT NULL,
  TGTDBID            VARCHAR(128)               NOT NULL,
  TGTDBOWN           VARCHAR(128)               NOT NULL,
  TGTDBPWD           VARCHAR(128) FOR BIT DATA NOT NULL,
  TGTLOCNAME         VARCHAR(100)               NOT NULL,
  TGTTCPPORT         INTEGER                    NOT NULL,
  LCLTCPPORT         INTEGER                    NOT NULL,
  LCLLOCNAME         VARCHAR(100)               NOT NULL,
  PERSISTENCY        CHAR         FOR BIT DATA NOT NULL
                                                DEFAULT X'00',
  RLEVEL             SMALLINT                   NOT NULL DEFAULT 1
) DATA CAPTURE CHANGES
```

Archived

```
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_TARGET_ID_IDX1
  ON CHC01.DMMD_TARGET_ID
    (TGTNAME ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_USER_LOG
(
  LOGREC             SMALLINT   UNIQUE DEFAULT NOT NULL,
  LOGTRANID          SMALLINT          DEFAULT NOT NULL,
  LOGRECORD          VARCHAR(4000)     DEFAULT NOT NULL
                                       FOR BIT DATA
) IN CDCDB.CDCTS4K
  DATA CAPTURE CHANGES;

CREATE UNIQUE INDEX CHC01.DMMD_UC_USER_LOG_IDX1
  ON CHC01.DMMD_USER_LOG
    (LOGREC ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

COMMIT;

INSERT INTO CHC01.DMMD_USER_LOG
  VALUES (0, 0, ' ');

COMMIT;

CREATE TABLE CHC01.DMMD_SOURCE_SERIAL
(
  SRCSYSID           CHAR(8)            UNIQUE NOT NULL,
  SRCSERIAL          CHAR(10)                  NOT NULL,
  SRCSYSDESC         VARCHAR(50)               NOT NULL,
  SRCSYSTYP          CHAR(10)                  NOT NULL,
  USELOADER          SMALLINT          DEFAULT NOT NULL,
  LOADERPARMS        CHAR(2)           DEFAULT NOT NULL,
  SRCDBTYPE          SMALLINT                  NOT NULL,
  SRCSYSNAME         VARCHAR(128)              NOT NULL,
  SRCINSTALLID       VARCHAR(60)               NOT NULL,
  LATENCY_ENABLED    SMALLINT          DEFAULT NOT NULL,
```

Archived

```
  LATENCY_WARNING    SMALLINT          DEFAULT NOT NULL,
  LATENCY_PROBLEM    SMALLINT          DEFAULT NOT NULL,
  CHECK (USELOADER       BETWEEN 0 AND 2),
  CHECK (LATENCY_ENABLED BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_SOURCE_SERIAL_IDX1
  ON CHCO1.DMMD_SOURCE_SERIAL
    (SRCSYSID  ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
  ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHCO1.DMMD_JRNL_STAT_TGT
(
  SRCSYSID           CHAR(8)                    NOT NULL,
  JOURNALID          VARCHAR(90)                NOT NULL,
  BOOKMARK           VARCHAR(90)  FOR BIT DATA NOT NULL,
  BMK_TIMESTP        TIMESTAMP         DEFAULT NOT NULL,
  UNIQUE (SRCSYSID, JOURNALID)
) DATA CAPTURE CHANGES
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_JRNL_STAT_TGT_PIDX
  ON CHCO1.DMMD_JRNL_STAT_TGT
    (SRCSYSID   ASC,
     JOURNALID  ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHCO1.PLAN_TABLE
(
  QUERYNO             INTEGER        NOT NULL,
  QBLOCKNO            SMALLINT       NOT NULL,
  APPLNAME            CHAR(8)        NOT NULL,
  PROGNAME            VARCHAR(128)   NOT NULL,
  PLANNO              SMALLINT       NOT NULL,
  METHOD              SMALLINT       NOT NULL,
  CREATOR             VARCHAR(128)   NOT NULL,
  TNAME               VARCHAR(128)   NOT NULL,
  TABNO               SMALLINT       NOT NULL,
  ACCESSTYPE          CHAR(2)        NOT NULL,
```

Archived

```
  MATCHCOLS           SMALLINT       NOT NULL,
  ACCESSCREATOR       VARCHAR(128)   NOT NULL,
  ACCESSNAME          VARCHAR(128)   NOT NULL,
  INDEXONLY           CHAR(1)        NOT NULL,
  SORTN_UNIQ          CHAR(1)        NOT NULL,
  SORTN_JOIN          CHAR(1)        NOT NULL,
  SORTN_ORDERBY       CHAR(1)        NOT NULL,
  SORTN_GROUPBY       CHAR(1)        NOT NULL,
  SORTC_UNIQ          CHAR(1)        NOT NULL,
  SORTC_JOIN          CHAR(1)        NOT NULL,
  SORTC_ORDERBY       CHAR(1)        NOT NULL,
  SORTC_GROUPBY       CHAR(1)        NOT NULL,
  TSLOCKMODE          CHAR(3)        NOT NULL,
  TIMESTAMP           CHAR(16)       NOT NULL,
  REMARKS             VARCHAR(762)   NOT NULL,
  PREFETCH            CHAR(1)        NOT NULL WITH DEFAULT,
  COLUMN_FN_EVAL      CHAR(1)        NOT NULL WITH DEFAULT,
  MIXOPSEQ            SMALLINT       NOT NULL WITH DEFAULT,
  VERSION             VARCHAR(64)    NOT NULL WITH DEFAULT,
  COLLID              VARCHAR(128)   NOT NULL WITH DEFAULT,
  ACCESS_DEGREE       SMALLINT       ,
  ACCESS_PGROUP_ID    SMALLINT       ,
  JOIN_DEGREE         SMALLINT       ,
  JOIN_PGROUP_ID      SMALLINT       ,
  SORTC_PGROUP_ID     SMALLINT       ,
  SORTN_PGROUP_ID     SMALLINT       ,
  PARALLELISM_MODE    CHAR(1)        ,
  MERGE_JOIN_COLS     SMALLINT       ,
  CORRELATION_NAME    VARCHAR(128)   ,
  PAGE_RANGE          CHAR(1)        NOT NULL WITH DEFAULT,
  JOIN_TYPE           CHAR(1)        NOT NULL WITH DEFAULT,
  GROUP_MEMBER        CHAR(8)        NOT NULL WITH DEFAULT,
  IBM_SERVICE_DATA    VARCHAR(254)   FOR BIT DATA NOT NULL
                                     WITH DEFAULT,
  WHEN_OPTIMIZE       CHAR(1)        NOT NULL WITH DEFAULT,
  QBLOCK_TYPE         CHAR(6)        NOT NULL WITH DEFAULT,
  BIND_TIME           TIMESTAMP      NOT NULL WITH DEFAULT,
  OPTHINT             VARCHAR(128)   NOT NULL WITH DEFAULT,
  HINT_USED           VARCHAR(128)   NOT NULL WITH DEFAULT,
  PRIMARY_ACCESSTYPE  CHAR(1)        NOT NULL WITH DEFAULT,
  PARENT_QBLOCKNO     SMALLINT       NOT NULL WITH DEFAULT,
  TABLE_TYPE          CHAR(1)        ,
  TABLE_ENCODE        CHAR(1)        NOT NULL WITH DEFAULT,
  TABLE_SCCSID        SMALLINT       NOT NULL WITH DEFAULT,
  TABLE_MCCSID        SMALLINT       NOT NULL WITH DEFAULT,
  TABLE_DCCSID        SMALLINT       NOT NULL WITH DEFAULT,
  ROUTINE_ID          INTEGER        NOT NULL WITH DEFAULT,
  CTEREF              SMALLINT       NOT NULL WITH DEFAULT,
  STMTTOKEN           VARCHAR(240)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

COMMIT;
CREATE TABLE CHC01.DMMD_AUTH_TBL
```

Archived

```
(
  USER_ID           CHAR(8)                  NOT NULL,
  TBOWNER           VARCHAR(128)             NOT NULL,
  TBNAME            VARCHAR(128)             NOT NULL,
  SELECT_AUTH       SMALLINT                 NOT NULL,
  INSERT_AUTH       SMALLINT                 NOT NULL,
  UPDATE_AUTH       SMALLINT                 NOT NULL,
  DELETE_AUTH       SMALLINT                 NOT NULL,
  UPDATE_TS         TIMESTAMP        DEFAULT NOT NULL,
  UNIQUE (USER_ID, TBOWNER, TBNAME)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_AUTH_TABLE_PIDX
  ON CHCO1.DMMD_AUTH_TBL
    (USER_ID ASC,
     TBOWNER ASC,
     TBNAME  ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE INDEX CHCO1.DMMD_UC_AUTH_TABLE_IDX2
  ON CHCO1.DMMD_AUTH_TBL
    (TBOWNER ASC,
     TBNAME  ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE TABLE CHCO1.DMMD_CATLG_TBL
(
  OWNER             VARCHAR(128)             NOT NULL,
  NAME              VARCHAR(128)             NOT NULL,
  DBNAME            VARCHAR(24)              NOT NULL,
  ADDED_TS          TIMESTAMP        DEFAULT NOT NULL,
  UNIQUE (OWNER, NAME)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_CATLG_TBL_PIDX
  ON CHCO1.DMMD_CATLG_TBL
    (OWNER   ASC,
     NAME    ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP   BOOKDW1;

CREATE INDEX CHCO1.DMMD_UC_CATLG_TBL_IDX2
  ON CHCO1.DMMD_CATLG_TBL
    (DBNAME  ASC)
  BUFFERPOOL BP0
  CLOSE NO
```

Archived

```
  USING STOGROUP   BOOKDW1;

COMMIT;

CREATE TABLE CHCO1.DMMD_SRC_DERIV_DEF
(
  TGTNAME            CHAR(10)                  NOT NULL,
  TBLSCHEMA          VARCHAR(137)              NOT NULL,
  TBLNAME            VARCHAR(128)              NOT NULL,
  COLUMN_NAME        VARCHAR(30)               NOT NULL,
  COLUMN_DESC        VARCHAR(40)               NOT NULL,
  DATA_TYPE          CHAR(8)                   NOT NULL,
  DRV_ID             INTEGER                   NOT NULL,
  COLUMN_ID          INTEGER                   NOT NULL,
  DATA_LENGTH        INTEGER                   NOT NULL,
  DATA_PRECISION     INTEGER                   NOT NULL,
  DATA_SCALE         INTEGER                   NOT NULL,
  FLDSELIND          CHAR                      NOT NULL,
  NULLABLE           CHAR                      NOT NULL,
  EVAL_FREQ          CHAR(4)                   NOT NULL,
  EXPRSN             VARCHAR(1000)             NOT NULL,
  BASE_DATABASE      VARCHAR(24)               NOT NULL,
  BASE_OWNER         VARCHAR(128)              NOT NULL,
  BASE_TBLNAME       VARCHAR(128)              NOT NULL,
  BASE_COLNAME       VARCHAR(30)               NOT NULL,
  DRVALT_TS          TIMESTAMP         DEFAULT NOT NULL,
  NET_TYPE           SMALLINT                  NOT NULL,
  LOC_TYPE           SMALLINT                  NOT NULL,
  UNIQUE (TGTNAME, TBLSCHEMA, TBLNAME, COLUMN_NAME),
  CHECK (FLDSELIND IN ('0', '1', '2')),
  CHECK (NULLABLE IN ('Y', 'N')),
  CHECK (EVAL_FREQ IN ('*AFT', '*BTH'))
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_SRC_DERIV_DEF_PIDX
  ON CHCO1.DMMD_SRC_DERIV_DEF
    (TGTNAME       ASC,
     TBLSCHEMA     ASC,
     TBLNAME       ASC,
  COLUMN_NAME   ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE UNIQUE INDEX CHCO1.DMMD_UC_SRC_DERIV_DEF_IDX2
  ON CHCO1.DMMD_SRC_DERIV_DEF
    (TGTNAME       ASC,
     TBLSCHEMA     ASC,
```

```
     TBLNAME       ASC,
     COLUMN_ID     ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE UNIQUE INDEX CHC01.DMMD_UC_SRC_DERIV_DEF_IDX3
   ON CHC01.DMMD_SRC_DERIV_DEF
     (TGTNAME       ASC,
      TBLSCHEMA     ASC,
      TBLNAME       ASC,
      DRV_ID        ASC)
   BUFFERPOOL BP0
   CLOSE NO
   USING STOGROUP BOOKDW1
     PRIQTY 48
     ERASE NO
   FREEPAGE 0
   PCTFREE 20;
CREATE TABLE CHC01.DMMD_TABLE
(
  TBLSCHEMA          VARCHAR(137)              NOT NULL,
  TBLNAME            VARCHAR(128)              NOT NULL,
  NAMEALIAS          VARCHAR(30)               NOT NULL,
  LAST_DDL_TIME      TIMESTAMP                 NOT NULL,
  DBID               SMALLINT               DEFAULT NULL,
  OBID               SMALLINT               DEFAULT NULL,
  DATACAPTURE        CHAR                      NOT NULL,
  COLCOUNT           SMALLINT                  NOT NULL,
  PSETID             SMALLINT                  NOT NULL,
  TYPE               CHAR                      NOT NULL DEFAULT 'T',
  ENCODING_SCHEME    CHAR                      NOT NULL DEFAULT 'E',
  SEND_ALIAS         SMALLINT                  NOT NULL DEFAULT 0,
  UNIQUE (TBLSCHEMA, TBLNAME),
  CHECK (DATACAPTURE IN (' ', 'Y')),
  CHECK (TYPE IN ('T', 'M', 'V')),
  CHECK (ENCODING_SCHEME IN ('A', 'E', 'U')),
  CHECK (SEND_ALIAS BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_TABLE_PIDX
  ON CHC01.DMMD_TABLE
    (TBLSCHEMA     ASC,
     TBLNAME       ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
```

Archived

```
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE UNIQUE INDEX CHC01.DMMD_UC_TABLE_IDX2
  ON CHC01.DMMD_TABLE
  (NAMEALIAS     ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE UNIQUE WHERE NOT NULL INDEX CHC01.DMMD_UC_TABLE_IDX3
  ON CHC01.DMMD_TABLE
  (DBID            ASC,
   PSETID          ASC,
   OBID            ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_TABLE_FIELDS
(
  TBLSCHEMA          VARCHAR(137)              NOT NULL,
  TBLNAME            VARCHAR(128)              NOT NULL,
  COLUMN_ID          INTEGER                   NOT NULL,
  COLUMN_NAME        VARCHAR(30)               NOT NULL,
  COLUMN_ALIAS       VARCHAR(30)               NOT NULL,
  DATA_TYPE          CHAR(8)                   NOT NULL,
  DATA_LENGTH        INTEGER                   NOT NULL,
  DATA_PRECISION     INTEGER                   NOT NULL,
  DATA_SCALE         INTEGER                   NOT NULL,
  PKEY_SEQ           SMALLINT                  NOT NULL,
  NULLABLE           CHAR                      NOT NULL,
  CHAR_SUBTYPE       CHAR          DEFAULT ' ' NOT NULL,
  NET_TYPE           SMALLINT                  NOT NULL,
  LOC_TYPE           SMALLINT                  NOT NULL,
  ORIG_DATATYPE      VARCHAR(30)               NOT NULL,
  COLUMN_TEXT        VARCHAR(50)   DEFAULT     NOT NULL,
  SEND_ALIAS         SMALLINT                  NOT NULL DEFAULT 0,
  UNIQUE (TBLSCHEMA, TBLNAME, COLUMN_ID),
  CHECK (NULLABLE IN ('Y', 'N')),
  CHECK (CHAR_SUBTYPE IN (' ', 'B', 'M', 'S')),
  CHECK (SEND_ALIAS BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;
```

Archived

```
CREATE UNIQUE INDEX CHC01.DMMD_UC_TABLE_FIELDS_PIDX
  ON CHC01.DMMD_TABLE_FIELDS
    (TBLSCHEMA     ASC,
     TBLNAME       ASC,
     COLUMN_ID     ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_TARGET_FIELDS
(
  TGTNAME            CHAR(10)                NOT NULL,
  TBLSCHEMA          VARCHAR(137)            NOT NULL,
  TBLNAME            VARCHAR(128)            NOT NULL,
  COLUMN_ID          INTEGER                 NOT NULL,
  FLDSELIND          CHAR                    NOT NULL,
  SRCENCMTD          SMALLINT                NOT NULL DEFAULT 1,
  SRCENCCCSID        VARCHAR(20)       DEFAULT NOT NULL,
  SRCENCNAME         VARCHAR(20)       DEFAULT NOT NULL,
  SRCENCLEN          SMALLINT                NOT NULL DEFAULT 0,
  TGTENCMTD          SMALLINT                NOT NULL DEFAULT 1,
  TGTENCCCSID        VARCHAR(20)       DEFAULT NOT NULL,
  TGTENCNAME         VARCHAR(20)       DEFAULT NOT NULL,
  TGTENCLEN          SMALLINT                NOT NULL DEFAULT 0,
  NET_TYPE           SMALLINT                NOT NULL DEFAULT 0,
  DATA_PRECISION     INTEGER                 NOT NULL DEFAULT 0,
  DATA_LENGTH        INTEGER                 NOT NULL DEFAULT 0,
  UNIQUE (TGTNAME, TBLSCHEMA, TBLNAME, COLUMN_ID),
  CHECK (FLDSELIND IN ('0', '1', '2')),
  CHECK (SRCENCMTD BETWEEN 1 AND 3),
  CHECK (SRCENCLEN BETWEEN 0 AND 3),
  CHECK (TGTENCMTD BETWEEN 1 AND 3),
  CHECK (TGTENCLEN BETWEEN 0 AND 3)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_TARGET_FIELDS_PIDX
  ON CHC01.DMMD_TARGET_FIELDS
    (TGTNAME       ASC,
     TBLSCHEMA     ASC,
     TBLNAME       ASC,
     COLUMN_ID     ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
```

Archived

```
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_TARGET_TABLE
(
  TGTNAME            CHAR(10)                  NOT NULL,
  TBLPRIORITY        INTEGER           DEFAULT NOT NULL,
  REFSEQUENCE        INTEGER           DEFAULT NOT NULL,
  TBLSCHEMA          VARCHAR(137)              NOT NULL,
  TBLNAME            VARCHAR(128)              NOT NULL,
  CHARGEBACKID       VARCHAR(30)       DEFAULT NOT NULL,
  REFDATE            TIMESTAMP         DEFAULT NOT NULL,
  DFNLOADSTATUS      CHAR                      NOT NULL,
  UPDMETHOD          CHAR                      NOT NULL,
  UPDSTATUS          CHAR                      NOT NULL,
  ROWSELTYPE         CHAR              DEFAULT NOT NULL,
  ROWSELEXPR         VARCHAR(1000)     DEFAULT NOT NULL,
  HEADRBA            CHAR(8)      FOR BIT DATA NOT NULL
                                       DEFAULT X'0000000000000000',
  RBAADDCODE         CHAR(1)           DEFAULT NOT NULL,
  CASCADE            CHAR(1)       DEFAULT 'S' NOT NULL,
  REFISOLATION       SMALLINT          DEFAULT NOT NULL,
  DRVALT_TS          TIMESTAMP         DEFAULT NOT NULL,
  UNIQUE (TGTNAME, TBLSCHEMA, TBLNAME),
  CHECK (DFNLOADSTATUS IN ('1', '2')),
  CHECK (UPDMETHOD IN ('J', 'P')),
  CHECK (UPDSTATUS IN ('0', '1', '2', '3', '4')),
  CHECK (ROWSELTYPE IN ('N', '0', 'S')),
  CHECK (RBAADDCODE IN (' ', 'A', 'R', 'S')),
  CHECK (CASCADE IN ('S', 'Y', 'N')),
  CHECK (REFISOLATION BETWEEN 0 AND 3)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_TARGET_TABLE_PIDX
  ON CHC01.DMMD_TARGET_TABLE
    (TGTNAME       ASC,
     TBLSCHEMA     ASC,
     TBLNAME       ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;
CREATE INDEX CHC01.DMMD_UC_TARGET_TABLE_IDX2
  ON CHC01.DMMD_TARGET_TABLE
    (TBLSCHEMA     ASC,
     TBLNAME       ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
```

```
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_DERIV_DEF
(
  SRCSYSID           CHAR(8)                  NOT NULL,
  SRCFILNME          VARCHAR(128)             NOT NULL,
  SRCFILLIB          VARCHAR(137)             NOT NULL,
  TGTTBLNME          VARCHAR(128)      DEFAULT NOT NULL,
  TGTTBLOWN          VARCHAR(128)      DEFAULT NOT NULL,
  ATRNAME            VARCHAR(30)       DEFAULT NOT NULL,
  ATREXPRSN          VARCHAR(1000)     DEFAULT NOT NULL,
  DRVALT_TS          TIMESTAMP         DEFAULT NOT NULL,
  UNIQUE (SRCSYSID,  SRCFILNME, SRCFILLIB,
          TGTTBLNME, TGTTBLOWN, ATRNAME)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_DERIV_DEF_PIDX
  ON CHC01.DMMD_DERIV_DEF
    (SRCSYSID  ASC,
     SRCFILNME ASC,
     SRCFILLIB ASC,
     TGTTBLNME ASC,
     TGTTBLOWN ASC,
     ATRNAME   ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_SOURCE_FIELD
(
  SRCSYSID           CHAR(8)                  NOT NULL,
  SRCFILNME          VARCHAR(128)             NOT NULL,
  SRCFILLIB          VARCHAR(137)             NOT NULL,
  FLDNME             VARCHAR(30)              NOT NULL,
  FLDTXT             VARCHAR(40)              NOT NULL,
  FLDDATLEN          INTEGER                  NOT NULL,
  FLDSEQNO           SMALLINT                 NOT NULL,
  FLDDGT             SMALLINT                 NOT NULL,
  FLDDECPSN          SMALLINT                 NOT NULL,
  FLDDATTYPE         SMALLINT                 NOT NULL,
  FLDDTMSEP          CHAR                     NOT NULL,
  FLDNULL            CHAR                     NOT NULL,
  FLDCCSID           VARCHAR(20)              NOT NULL,
  SRCENCNAME         VARCHAR(20)       DEFAULT NOT NULL,
  FLDLENKNOWN        SMALLINT                 NOT NULL,
  ORIG_DATATYPE      VARCHAR(30)              NOT NULL,
```

Archived

```
  UNIQUE (SRCSYSID, SRCFILNME, SRCFILLIB, FLDNME),
  CHECK (FLDNULL IN ('Y', 'N')),
  CHECK (FLDLENKNOWN BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_SOURCE_FIELD_PIDX
  ON CHC01.DMMD_SOURCE_FIELD
    (SRCSYSID   ASC,
     SRCFILNME  ASC,
     SRCFILLIB  ASC,
     FLDNME     ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_SOURCE_ID
(
  SRCSYSID          CHAR(8)                  NOT NULL,
  SRCFILNME         VARCHAR(128)             NOT NULL,
  SRCFILLIB         VARCHAR(137)             NOT NULL,
  TGTTBLOWN         VARCHAR(128)     DEFAULT NOT NULL,
  TGTTBLNME         VARCHAR(128)     DEFAULT NOT NULL,
  SRCFILTXT         VARCHAR(255)             NOT NULL,
  UPDMTD            SMALLINT                 NOT NULL,
  TGTDBTYPE         CHAR                     NOT NULL,
  DETECT_COLL       SMALLINT                 NOT NULL,
  USELOADER         SMALLINT                 NOT NULL,
  LOADERPARMS       CHAR(2)                  NOT NULL,
  TGTTBLIDXNME      VARCHAR(128)     DEFAULT NOT NULL,
  TGTTBLIDXOWN      VARCHAR(128)     DEFAULT NOT NULL,
  IDX_ALTEREDTS     TIMESTAMP                NOT NULL,
  TBL_ALTEREDTS     TIMESTAMP                NOT NULL,
  SRCTBLVERSION     TIMESTAMP                NOT NULL,
  BEFIMG            SMALLINT                 NOT NULL,
  MULTIMBR          SMALLINT                 NOT NULL,
  ASSIGN_TYPE       SMALLINT         DEFAULT NOT NULL,
  IDENT_COL         INTEGER                  UNIQUE
                          GENERATED ALWAYS AS IDENTITY,
  CRMETHOD          SMALLINT        DEFAULT 6 NOT NULL,
  CRVCCOLUMN        VARCHAR(30)       DEFAULT NOT NULL,
  CRUSEREXIT        CHAR(8)           DEFAULT NOT NULL,
  ENCODING_SCHEME   CHAR                     NOT NULL DEFAULT 'E',
  UNIQUE (SRCSYSID, SRCFILNME, SRCFILLIB),
  CHECK (UPDMTD     BETWEEN 0 AND 4),
  CHECK (TGTDBTYPE  IN (' ', 'S')),
  CHECK (USELOADER  BETWEEN 0 AND 2),
  CHECK (BEFIMG     BETWEEN 0 AND 1),
  CHECK (MULTIMBR   BETWEEN 0 AND 1),
```

Archived

```
  CHECK (ASSIGN_TYPE BETWEEN 0 AND 5),
  CHECK (CRMETHOD    BETWEEN 1 AND 6),
  CHECK (ENCODING_SCHEME IN ('A', 'E', 'U'))
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_SOURCE_ID_PIDX
  ON CHC01.DMMD_SOURCE_ID
    (SRCSYSID  ASC,
     SRCFILNME ASC,
     SRCFILLIB ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE UNIQUE INDEX CHC01.DMMD_UC_SOURCE_ID_IDX2
  ON CHC01.DMMD_SOURCE_ID
    (IDENT_COL  ASC)
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_SRC_ROW_ID
(
  SRCSYSID          CHAR(8)                 NOT NULL,
  SRCFILNME         VARCHAR(128)            NOT NULL,
  SRCFILLIB         VARCHAR(137)            NOT NULL,
  ROWSELEXP         VARCHAR(375)            NOT NULL,
  UNIQUE (SRCSYSID, SRCFILNME, SRCFILLIB)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_SRC_ROW_ID_PIDX
  ON CHC01.DMMD_SRC_ROW_ID
    (SRCSYSID  ASC,
     SRCFILNME ASC,
     SRCFILLIB ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
```

Archived

```
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_TGT_TBL_ATTR
(
  TGTTBLNME          VARCHAR(128)              NOT NULL,
  TGTTBLOWN          VARCHAR(137)              NOT NULL,
  ATRNAME            VARCHAR(30)               NOT NULL,
  ATRDATTYPE         CHAR(8)                   NOT NULL,
  ATRSEQNUM          SMALLINT                  NOT NULL,
  ATRLEN             INTEGER                   NOT NULL,
  ATRDGT             SMALLINT                  NOT NULL,
  ATRDECPSN          SMALLINT                  NOT NULL,
  ATRKEYSEQ          SMALLINT                  NOT NULL,
  ATRNULL            CHAR                      NOT NULL,
  ATRINFO            VARCHAR(128) FOR BIT DATA NOT NULL,
  LOC_TYPE           SMALLINT                  NOT NULL,
  UNIQUE (TGTTBLNME, TGTTBLOWN, ATRNAME),
  CHECK (ATRNULL IN ('Y', 'N'))
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_TGT_TBL_ATTR_PIDX
  ON CHC01.DMMD_TGT_TBL_ATTR
    (TGTTBLNME ASC,
     TGTTBLOWN ASC,
     ATRNAME   ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_TGT_TBL_MAP
(
  SRCSYSID           CHAR(8)                   NOT NULL,
  TGTTBLNME          VARCHAR(128)              NOT NULL,
  TGTTBLOWN          VARCHAR(128)              NOT NULL,
  SRCFILNME          VARCHAR(128)              NOT NULL,
  SRCFILLIB          VARCHAR(137)              NOT NULL,
  ATRNME             VARCHAR(30)               NOT NULL,
  ATRDFTTYP          CHAR                      NOT NULL,
  ATRDFTOTH          VARCHAR(30)               NOT NULL,
  FLDNME             VARCHAR(30)               NOT NULL,
  SUMM_TYPE          SMALLINT          DEFAULT NOT NULL,
  SUMM_COL_KEY       SMALLINT          DEFAULT NOT NULL,
  CRSELECTED         SMALLINT          DEFAULT NOT NULL,
  CHECK(CRSELECTED   IN (0, 1)),
  UNIQUE (SRCSYSID, TGTTBLNME, TGTTBLOWN,
          SRCFILNME, SRCFILLIB, ATRNME),
  CHECK (ATRDFTTYP  IN ('*', 'B', 'C', 'N', 'O', 'Z')),
  CHECK (SUMM_TYPE  BETWEEN 0 AND 2),
```

```
  CHECK (CRSELECTED BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_TGT_TBL_MAP_PIDX
  ON CHC01.DMMD_TGT_TBL_MAP
    (SRCSYSID  ASC,
     TGTTBLNME ASC,
     TGTTBLOWN ASC,
     SRCFILNME ASC,
     SRCFILLIB ASC,
     ATRNME    ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_USER_EXIT_TGT
(
  SRCSYSID          CHAR(8)                 NOT NULL,
  SRCFILNME         VARCHAR(128)            NOT NULL,
  SRCFILLIB         VARCHAR(137)            NOT NULL,
  STDCLR            CHAR            DEFAULT NOT NULL,
  USRACPGM          CHAR(8)         DEFAULT NOT NULL,
  USRBCPGM          CHAR(8)         DEFAULT NOT NULL,
  STDDLT            CHAR            DEFAULT NOT NULL,
  USRADPGM          CHAR(8)         DEFAULT NOT NULL,
  USRBDPGM          CHAR(8)         DEFAULT NOT NULL,
  STDINS            CHAR            DEFAULT NOT NULL,
  USRAIPGM          CHAR(8)         DEFAULT NOT NULL,
  USRBIPGM          CHAR(8)         DEFAULT NOT NULL,
  STDUPD            CHAR            DEFAULT NOT NULL,
  USRAUPGM          CHAR(8)         DEFAULT NOT NULL,
  USRBUPGM          CHAR(8)         DEFAULT NOT NULL,
  USRARPGM          CHAR(8)         DEFAULT NOT NULL,
  USRBRPGM          CHAR(8)         DEFAULT NOT NULL,
  INSMSG            SMALLINT        DEFAULT NOT NULL,
  UPDMSG            SMALLINT        DEFAULT NOT NULL,
  DLTMSG            SMALLINT        DEFAULT NOT NULL,
  UNIQUE (SRCSYSID, SRCFILNME, SRCFILLIB),
  CHECK (STDCLR IN ('A', 'N', 'Y')),
  CHECK (STDDLT IN ('A', 'D', 'I', 'N', 'U', 'Y')),
  CHECK (STDINS IN ('A', 'D', 'I', 'N', 'U', 'Y')),
  CHECK (STDUPD IN ('A', 'B', 'D', 'I', 'N', 'U', 'Y')),
  CHECK (INSMSG BETWEEN 0 AND 1),
  CHECK (UPDMSG BETWEEN 0 AND 1),
  CHECK (DLTMSG BETWEEN 0 AND 1)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;
```

Archived

```
CREATE UNIQUE INDEX CHC01.DMMD_UC_USER_EXIT_TGT_PIDX
  ON CHC01.DMMD_USER_EXIT_TGT
    (SRCSYSID  ASC,
     SRCFILNME ASC,
     SRCFILLIB ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_VAL_TRANS
(
  SRCSYSID           CHAR(8)                  NOT NULL,
  SRCFILNME          VARCHAR(128)             NOT NULL,
  SRCFILLIB          VARCHAR(137)             NOT NULL,
  TGTTBLNME          VARCHAR(128)             NOT NULL,
  TGTTBLOWN          VARCHAR(128)             NOT NULL,
  ATRNAME            VARCHAR(30)              NOT NULL,
  ATROLDVAL          VARCHAR(26)              NOT NULL,
  ATROLDNBR          VARCHAR(18)              NOT NULL,
  ATROLDNUL          CHAR                     NOT NULL,
  ATRNEWVAL          VARCHAR(26)       DEFAULT NOT NULL,
  ATRNEWNBR          VARCHAR(18)       DEFAULT NOT NULL,
  ATRNEWNUL          CHAR              DEFAULT NOT NULL,
  UNIQUE (SRCSYSID, SRCFILNME, SRCFILLIB, TGTTBLNME, TGTTBLOWN,
          ATRNAME, ATROLDVAL, ATROLDNBR, ATROLDNUL),
  CHECK (ATROLDNUL IN ('Y', ' ')),
  CHECK (ATRNEWNUL IN ('Y', ' '))
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_VAL_TRANS_PIDX
  ON CHC01.DMMD_VAL_TRANS
    (SRCSYSID   ASC,
     SRCFILNME  ASC,
     SRCFILLIB  ASC,
     TGTTBLNME  ASC,
     TGTTBLOWN  ASC,
     ATRNAME    ASC,
     ATROLDVAL  ASC,
     ATROLDNBR  ASC,
     ATROLDNUL  ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
```

Archived

```
  PCTFREE 20;

CREATE TABLE CHC01.DMMD_SQL_STMT
(
  IDENT_COL          INTEGER                  NOT NULL,
  STMT_TYPE          SMALLINT                 NOT NULL,
  STATEMENT          VARCHAR(4000)            NOT NULL,
  UNIQUE (IDENT_COL, STMT_TYPE),
  FOREIGN KEY(IDENT_COL)
    REFERENCES CHC01.DMMD_SOURCE_ID(IDENT_COL)
    ON DELETE CASCADE,
  CHECK (STMT_TYPE BETWEEN 1 AND 2)
) DATA CAPTURE CHANGES
  CCSID UNICODE
  IN CDCDB.CDCTS4K;

CREATE UNIQUE INDEX CHC01.DMMD_UC_SQL_STMT_PIDX
  ON CHC01.DMMD_SQL_STMT
    (IDENT_COL  ASC,
     STMT_TYPE  ASC)
  CLUSTER
  BUFFERPOOL BP0
  CLOSE NO
  USING STOGROUP BOOKDW1
    PRIQTY 48
    ERASE NO
  FREEPAGE 0
  PCTFREE 20;

COMMIT;
CREATE INDEX CHC01.DMMD_SYSTABATH_IDX
  ON SYSIBM.SYSTABAUTH
  (TCREATOR        ASC,
   TTNAME          ASC)
  BUFFERPOOL       BP0
  CLOSE            NO
  USING STOGROUP   BOOKDW1
  PRIQTY           48
  ERASE            NO
  FREEPAGE         0
  PCTFREE          20;

COMMIT;

  CREATE INDEX CHC01.DMMD_SYSTABLES_IDX
  ON SYSIBM.SYSTABLES
  (DBNAME          ASC)
  BUFFERPOOL       BP0
  CLOSE            NO
  USING STOGROUP   BOOKDW1
  PRIQTY           48
  ERASE            NO
  FREEPAGE         0
  PCTFREE          20;
```

Archived

```
  COMMIT;
```

Example A-2 shows the job that was used to bind the InfoSphere Change Data Capture plans. We chose to use a plan suffix of '01' and a user ID of CHC01.

Example A-2 Bind for InfoSphere Change Data Capture plans

```
//CHCBNDPL JOB MSGLEVEL=(1,1),CLASS=A,MSGCLASS=X,REGION=0M,
//            TIME=300,NOTIFY=&SYSUID
//*
//DB2BIND  EXEC PGM=IKJEFT01,
//            DYNAMNBR=20
//STEPLIB  DD DSN=DB2910.ZDW.SDSNLOAD,DISP=SHR
//SYSTSIN  DD *
  DSN SYSTEM(DDW1)
  BIND -
    PLAN(CHCDAL01) -
    MEMBER(CHCDALDB CHCDB2DL) -
    ACTION(REPLACE) RETAIN -
    ISOLATION(CS) -
    CACHESIZE(256) -
    RELEASE(COMMIT) -
    CURRENTDATA(NO) -
    ENCODING(UNICODE) -
    OWNER(CHC01) -
    QUALIFIER(CHC01)
  BIND -
    PLAN(CHCDDR01) -
    MEMBER(CHCDDRDB CHCDB2DL CHCMDRDL) -
    ACTION(REPLACE) RETAIN -
    ISOLATION(CS) -
    CACHESIZE(256) -
    RELEASE(COMMIT) -
    CURRENTDATA(NO) -
    ENCODING(UNICODE) -
    OWNER(CHC01) -
    QUALIFIER(CHC01)
  BIND -
    PLAN(CHCDDT01) -
    MEMBER(CHCDDTDB CHCDB2DL) -
    ACTION(REPLACE) RETAIN -
    ISOLATION(CS) -
    CACHESIZE(256) -
    RELEASE(COMMIT) -
    CURRENTDATA(NO) -
    ENCODING(UNICODE) -
    OWNER(CHC01) -
    QUALIFIER(CHC01)
  BIND -
    PLAN(CHCDLP01) -
    MEMBER(CHCDLPMD CHCDB2DL) -
    ACTION(REPLACE) RETAIN -
    ISOLATION(CS) -
    CACHESIZE(256) -
    RELEASE(COMMIT) -
```

Archived

```
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCDLRO1) -
  MEMBER(CHCDLRDB CHCDB2DL) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCDSCO1) -
  MEMBER(CHCDSCDB CHCDSCPG CHCDB2DL) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCDSLO1) -
  MEMBER(CHCDSLDB CHCDB2DL) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCDTCO1) -
  MEMBER(CHCDTCDB CHCDTCMD CHCDB2DL CHCMIDB -
    CHCDTCFR                                                    -
    ) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  KEEPDYNAMIC(YES) -
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCLAUO1) -
  MEMBER(CHCLAUDB) -
  ACTION(REPLACE) RETAIN -
```

Archived

```
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCMDPO1) -
  MEMBER(CHCMDPDB CHCMDPMD) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(EBCDIC) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCMTAO1) -
  MEMBER(CHCMTAMN) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(EBCDIC) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCPAAO1) -
  MEMBER(CHCPAADB CHCMIDB CHCSECDL CHCDB2DL CHCMDRDL) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCSCTO1) -
  MEMBER(CHCSCTDB CHCDB2DL) -
  ACTION(REPLACE) RETAIN -
  ISOLATION(CS) -
  CACHESIZE(256) -
  RELEASE(COMMIT) -
  CURRENTDATA(NO) -
  ENCODING(UNICODE) -
  OWNER(CHCO1) -
  QUALIFIER(CHCO1)
BIND -
  PLAN(CHCSDTO1) -
  MEMBER(CHCSDTDB CHCDTRDB CHCDLSDB CHCMIDB CHCDB2DL  -
    CHCDTRD8                          /* <DB2-V8.1-UP>     <== */ -
```

```
      ) -
    ACTION(REPLACE) RETAIN -
    ISOLATION(CS) -
    CACHESIZE(256) -
    RELEASE(COMMIT) -
    CURRENTDATA(NO) -
    ENCODING(UNICODE) -
    OWNER(CHC01) -
    QUALIFIER(CHC01)
  BIND -
    PLAN(CHCTCT01) -
    MEMBER(CHCTCTDB CHCDB2DL) -
    ACTION(REPLACE) RETAIN -
    ISOLATION(CS) -
    CACHESIZE(256) -
    RELEASE(COMMIT) -
    CURRENTDATA(NO) -
    ENCODING(UNICODE) -
    OWNER(CHC01) -
    QUALIFIER(CHC01)
  BIND -
    PLAN(CHCV6M01) -
    MEMBER(CHCV6MIG) -
    ACTION(REPLACE) RETAIN -
    ISOLATION(CS) -
    CACHESIZE(256) -
    RELEASE(COMMIT) -
    CURRENTDATA(NO) -
    ENCODING(UNICODE) -
    OWNER(CHC01) -
    QUALIFIER(CHC01)
  END
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//DBRMLIB  DD DSNAME=ICDC.ZDW.SCHCDBRM,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//*
//
```

Archived

Archived

B

Additional performance data

This appendix provides details on how to collect performance statistics on Linux. We also include high-level Linux performance metrics collected during one of our experiments.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

B.1 Linux monitoring

A well-known way to monitor Linux usage is on z/VM, by means of the Performance Toolkit, which gives a good overview at the LPAR level. In our project we did not want to be constrained by Linux horsepower, because we used Linux and IBM Cognos BI as a DB2 workload injector.

SYSSTAT

Another possibility to monitor Linux performance is to use the SYSSTAT package. SYSSTAT is available in SLES 10 in version 8.0.4. It contains these commands:

- **iostat** - Reports processor statistics and input/output statistics for devices, partitions, and network file systems.
- **mpstat** - Reports individual or combined processor related statistics.
- **pidstat** - Reports statistics for Linux tasks (processes): I/O, processor, memory, and so on.
- **sar** - Collects, reports and saves system activity information (processor, memory, disks, interrupts, network interfaces, TTY, kernel tables, and so on.)
- **sadc** - Is the system activity data collector, used as a back end for sar.
- **sa1** - Collects and stores binary data in the system activity daily data file. It is a front end to **sadc** designed to be run from cron.
- **sa2** - Writes a summarized daily activity report. It is a front end to **sar** designed to be run from cron.
- **sadf** - Displays data collected by **sar** in multiple formats (CSV, XML, and so on.) This is useful for loading performance data into a database, or import it in a spreadsheet to make graphs.

In our methodology we used only two of these commands, **sar** and **sadf**. The first one could be compared to SMF because it gathers raw system-wide data in binary format, and the second could be compared to RMF because it is used to extract reports from that data.

A small example of a processor report created with **sadf**.

```
sadf -d 15 [sar_bin_file]
18:05:45        CPU     %user     %nice   %system   %iowait    %steal     %idle
18:06:00        all      0.08      0.00      0.06      0.01      0.00     99.86
18:06:00          0      0.07      0.00      0.07      0.00      0.00     99.87
18:06:00          1      0.13      0.00      0.07      0.00      0.00     99.80
18:06:00          2      0.07      0.00      0.00      0.00      0.00     99.93
18:06:00          3      0.07      0.00      0.00      0.00      0.00     99.93
18:06:00          4      0.13      0.00      0.07      0.00      0.00     99.80
18:06:00          5      0.07      0.00      0.07      0.00      0.00     99.87
18:06:00          6      0.20      0.00      0.07      0.00      0.00     99.73
18:06:00          7      0.00      0.00      0.07      0.00      0.00     99.93
18:06:15        all      0.07      0.00      0.04      0.01      0.00     99.88
18:06:15          0      0.07      0.00      0.07      0.00      0.00     99.87
18:06:15          1      0.00      0.00      0.00      0.00      0.00    100.00
18:06:15          2      0.07      0.00      0.07      0.00      0.00     99.87
18:06:15          3      0.00      0.00      0.07      0.00      0.00     99.93
18:06:15          4      0.00      0.00      0.00      0.00      0.00    100.00
18:06:15          5      0.13      0.00      0.07      0.00      0.00     99.80
18:06:15          6      0.07      0.00      0.07      0.00      0.00     99.87
18:06:15          7      0.07      0.00      0.07      0.00      0.00     99.87
```

For this project, we had several Linux operating systems running for different purposes. With the use of a shell script we wanted to start the data gatherer on all of them at the same time. Prior the running the script, we had to cross-exchange ssh keys between them. Here is the methodology.

SSH key exchange

To simplify the process of starting data gathering (with the **sar** command) on all platforms and to have consistent data, started at the same time on all machines, we enabled ssh logins without password through the exchanges of ssh_keys. This is quite standard these days. Here is how to do it.

First log in to one machine. Choose this machine wisely because it will be the “master”, the one from which you are going to do everything. For us it was zdwl2.

Now, create an ssh key by entering this command. When you are asked for a passphrase just press the Enter key. If you were to create a passphrase, you would have to enter it each time you want to use the key.

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
96:31:95:e0:01:62:86:81:67:bf:d2:7e:04:03:9a:fa root@zdwl2
```

Now you need to copy your identity (the public key) to all the Linux systems you want to access.

```
ssh-copy-id -i /root/.ssh/id_rsa.pub zdwl4
root@zdwl4's password:
Now try logging into the machine, with "ssh zdwl4", and check in:
.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

This command automatically concatenates the file `id_rsa.pub` to `/root/.ssh/authorized_keys` over ssh to the Linux machine zdwl4.

Double-check that it is working nicely by connecting from a console with ssh zdwl4. You might have to type the password once. If you access from multiple paths such as several IP addresses or host names, you may need to redo this process for all of them.

Start data gatherer and create text reports through a shell script

After having exchanged ssh keys, it is possible to run a command on (for example) zdwl2 from zdwl1. Just prefix the command with `ssh zdwl2` and the command will run remotely:

```
   ssh zdwl2 sar -o /root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID" $INTVAL
$SAMPLE > /dev/null 2>&1 &
```

This command will run the command **ssh -o** on zdwl2. In the original shell script, several variables are positioned: $DATE, $JULIEN, $RUNTYPE, $RUNID, $INTVAL, and $SAMPLE. We generalized the behavior for all three Linux servers and took care of the exits in case we canceled (Ctrl+C) it so that the data would still be stored and reported on.

Archived

Here are all the reports that are created from the newly gathered system data:

```
sar -f /root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID" -n DEV  60 >
/root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID".NET
   sar -f /root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID" -P ALL  60 >
/root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID".CPU
   sar -f /root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID" -B   60 >
/root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID".MEM
   sar -f /root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID" -d -p  60 >
/root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID".DASD
   sar -f /root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID" -r 60 >
/root/Monitoring/$DATE/C9"$JULIEN$RUNTYPE"L"$RUNID".RAM
```

This creates five files named after the type of performance data; see Example B-1, Example B-2, Example B-3, Example B-4, and Example B-5.

-n DEV: Network data for all network devices.

Example B-1 Example of a network data report from the sar command

14:08:59	IFACE	rxpck/s	txpck/s	rxkB/s	txkB/s	rxcmp/s	txcmp/s	rxmcst/s
14:09:59	lo	2855.64	2855.64	6892.02	6892.02	0.00	0.00	0.00
14:09:59	sit0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14:09:59	hsi1	100.50	130.32	752.38	27.75	0.00	0.00	0.00
14:09:59	hsi3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14:09:59	hsi0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14:09:59	hsi2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14:09:59	eth1	1012.94	1324.19	537.28	834.32	0.00	0.00	0.08
14:09:59	eth2	8.85	0.88	2.07	0.05	0.00	0.00	7.97
14:09:59	vlan302	1012.92	1324.19	537.28	834.32	0.00	0.00	0.00
14:09:59	vlan537	0.88	0.88	0.13	0.05	0.00	0.00	0.00

-P ALL: CPUs

Example B-2 Example of a CPU data report from the sar command

14:08:59	CPU	%user	%nice	%system	%iowait	%steal	%idle
14:09:59	all	65.63	0.00	4.07	0.30	1.45	28.55
14:09:59	0	65.59	0.00	4.15	0.33	1.43	28.49
14:09:59	1	65.90	0.00	4.08	0.52	1.38	28.12
14:09:59	2	66.07	0.00	4.15	0.22	1.40	28.17
14:09:59	3	64.86	0.00	4.05	0.23	1.50	29.36
14:09:59	4	66.50	0.00	3.90	0.23	1.52	27.85
14:09:59	5	65.53	0.00	4.13	0.22	1.42	28.70
14:09:59	6	65.53	0.00	4.05	0.30	1.50	28.62
14:09:59	7	65.10	0.00	4.02	0.30	1.45	29.13

-d -p: DASD utilization (read and write /s)

Example B-3 Example of a DASD data report from the sar command

14:08:59	DEV	tps	rd_sec/s	wr_sec/s	avgrq-sz	avgqu-sz	await	svctm	%util
14:09:59	dasda	0.87	0.00	28.54	32.92	0.00	0.38	0.38	0.03
14:09:59	dasdb	0.67	0.00	15.34	23.00	0.00	0.25	0.25	0.02
14:09:59	dasdc	13.07	0.00	1062.04	81.27	0.01	0.42	0.41	0.53
14:09:59	dasdd	0.02	0.00	0.13	8.00	0.00	10.00	10.00	0.02
14:09:59	dasde	0.02	0.00	0.13	8.00	0.00	10.00	10.00	0.02
14:09:59	dasdf	0.15	0.00	1.47	9.78	0.00	0.00	0.00	0.00
14:09:59	dasdg	0.37	0.00	3.20	8.73	0.00	0.91	0.45	0.02
14:09:59	dasdh	0.52	0.00	10.40	20.13	0.00	0.65	0.65	0.03
14:09:59	dasdi	32.17	0.00	3314.55	103.03	0.33	10.37	3.62	11.64

Archive

-B: Memory management in pages (pages in /s, pages out /s, pages free /s ...)

Example B-4 Example of a memory management data report from the sar command

14:08:59	pgpgin/s	pgpgout/s	fault/s	majflt/s	pgfree/s	pgscank/s	pgscand/s	pgsteal/s	%vmeff
14:09:59	0.00	2190.03	65605.97	0.00	21958.41	0.00	0.00	0.00	0.00
14:10:59	0.00	2807.73	68065.84	0.00	22378.11	0.00	0.00	0.00	0.00
14:11:59	33.33	1943.40	61953.83	0.75	20431.93	0.00	0.00	0.00	0.00
14:12:59	24.33	2139.80	64215.12	0.28	21145.12	0.00	0.00	0.00	0.00
14:13:59	89.97	1768.94	65006.40	3.52	21078.91	0.00	0.00	0.00	0.00

-r: RAM utilization (KB free, KB used, KB free, KB swapped ...)

Example B-5 Example of a memory usage data report from the sar command

14:08:59	kbmemfree	kbmemused	%memused	kbbuffers	kbcached	kbswpfree	kbswpused	%swpused	kbswpcad
14:09:59	3552428	4669896	56.80	112096	926096	0	0	0.00	0
14:10:59	3379708	4842616	58.90	115108	930308	0	0	0.00	0
14:11:59	3318056	4904268	59.65	118016	934108	0	0	0.00	0
14:12:59	3287052	4935272	60.02	120936	931188	0	0	0.00	0
14:13:59	3277756	4944568	60.14	122856	937008	0	0	0.00	0

B.2 Linux performance results

We did little or no tuning of IBM Cognos BI and Linux. Our objective was to use IBM Cognos BI as an injector to create realistic DB2 BI workloads. We wanted to inject as much as possible with a manageable workload on Linux; that is, we wanted to stay below 80% processor utilization.

Figure B-1 on page 418 shows an example of one measurement where we reached 270 concurrent RPT users. A few observations are called for. First, in RPT we did not implement web browsing in the injection, which reduces the amount of work the application has to perform concurrently. It is a little bit like what we called *think time*. Also, by a rule of thumb, the number of named users was between 10 and 20 times the number of concurrent users. So we could serve between 3000 and 6000 users.

Archived

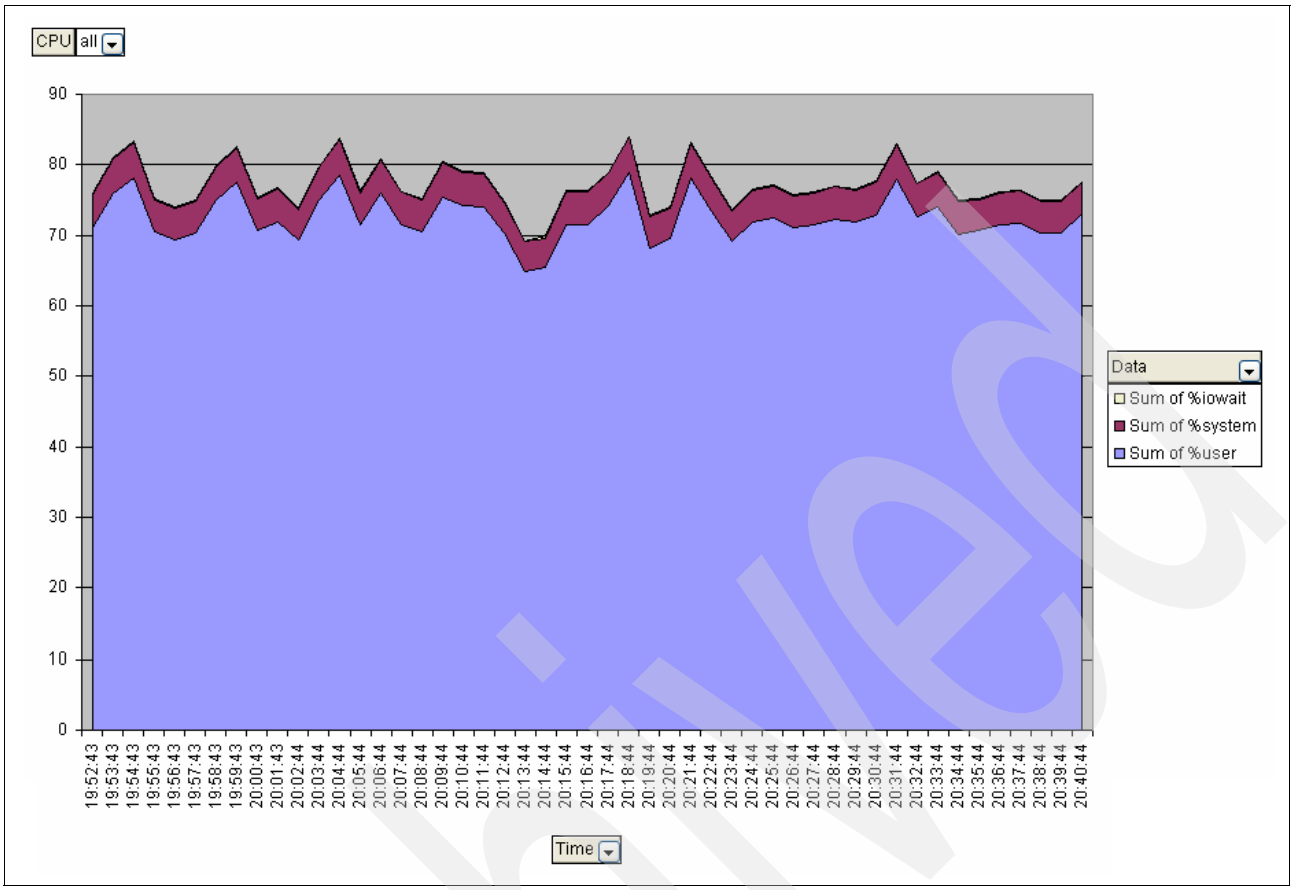


Figure B-1 Linux/IBM Cognos BI overall processor utilization for 270 concurrent users

Our monitoring showed some improvement possibilities, especially on the disk subsystem. The way the logical volumes have been created in Linux is not optimal. Indeed, we saw that the LVM was built without striping. Of course, there is a lot of virtualization along the way, and it is obviously striped in the DS8000. However, Linux has internals that also need to be stripe-aware.

In Figure B-2 on page 419, there is the utilization rate per type of activity. The pink color represents the LVM mounted on /opt, and the other colors represent the DASD's physical definition in z/VM. We can see in this sample that almost all the volume group utilization is only on dasdi, rather than dispatched across all physical disks. We did not have enough time to recreate an LVM with the striping parameters.

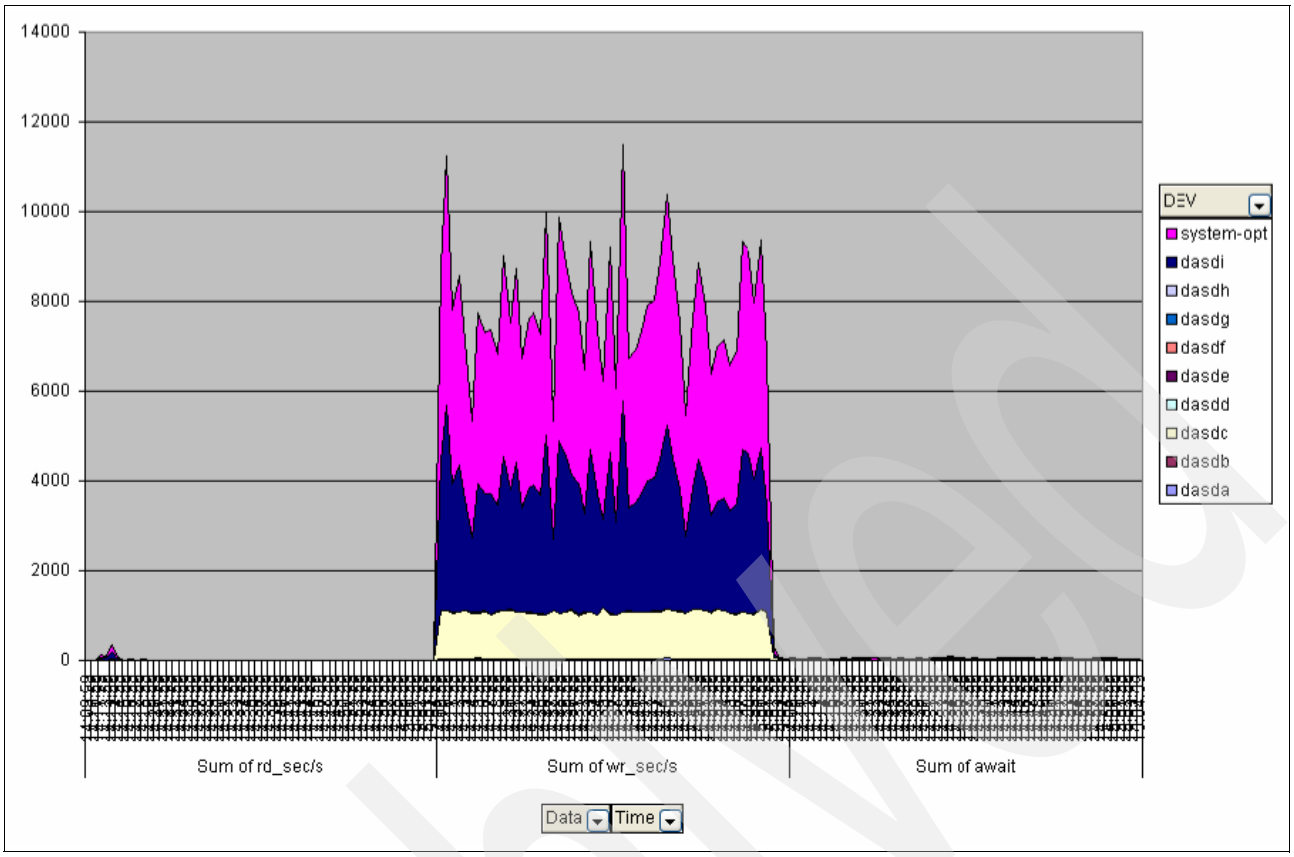


Figure B-2 Disk subsystem utilization

Archived

Archived

C

Using DB2 accounting data for setting WLM period durations

In this appendix we provide a technique for determining WLM period durations, using DB2 Class 1 accounting data as input.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

C.1 Sizing SC periods using DB2 accounting information

Here we provide an alternative technique for sizing the periods of DDF service classes, based on the observation and analysis of the DB2 accounting. The DB2 accounting information is normally readily available and is often used by DBAs for performance analysis, so hopefully this approach will help a database administrator to size or provide input for the sizing of the periods. Another advantage of this approach is fostering communication between the DBAs and z/OS performance administrators.

By observation and analysis of the DB2 accounting data we can help determine reasonable size service class periods and an initial take of response time goals for the early periods.

Before going further, it is important to point out some prerequisites for using this method:

- DB2 zparms need to have CMTSTAT=INACTIVE. This is the DB2 default and is already highly advised for any level of sophisticated WLM resource management of data warehouse queries.
- For DRDA parallel queries, APAR PM06953 must be applied. This APAR provides single enclave management, hence a single transaction and single overall elapsed time, for each DB2 parallel query. Without this APAR, a given DB2 parallel query would be managed as multiple enclaves and reported as multiple transactions within WLM.
- No accounting roll-up used during the collection of DB2 accounting data.
- The DB2 application does not hold resources beyond a commit. This is the general recommended practice and should not be happening anyway.

Figure C-1 on page 423 depicts the DB2 accounting class times for a given DDF unit of work. Assuming the previously mentioned prerequisites to be true, then the DB2 CLASS 1 Elapsed time will be equal to the elapsed time WLM captures and utilizes for resource management and monitoring. This is also known as the elapsed time of the WLM independent enclave associated with the DDF unit of work. Taking it one step further, the enclave is created by DDF for an incoming connection at the first SQL statement. The enclave is deleted (and a DB2 accounting record is cut) when the DDF thread Commits.

Archived

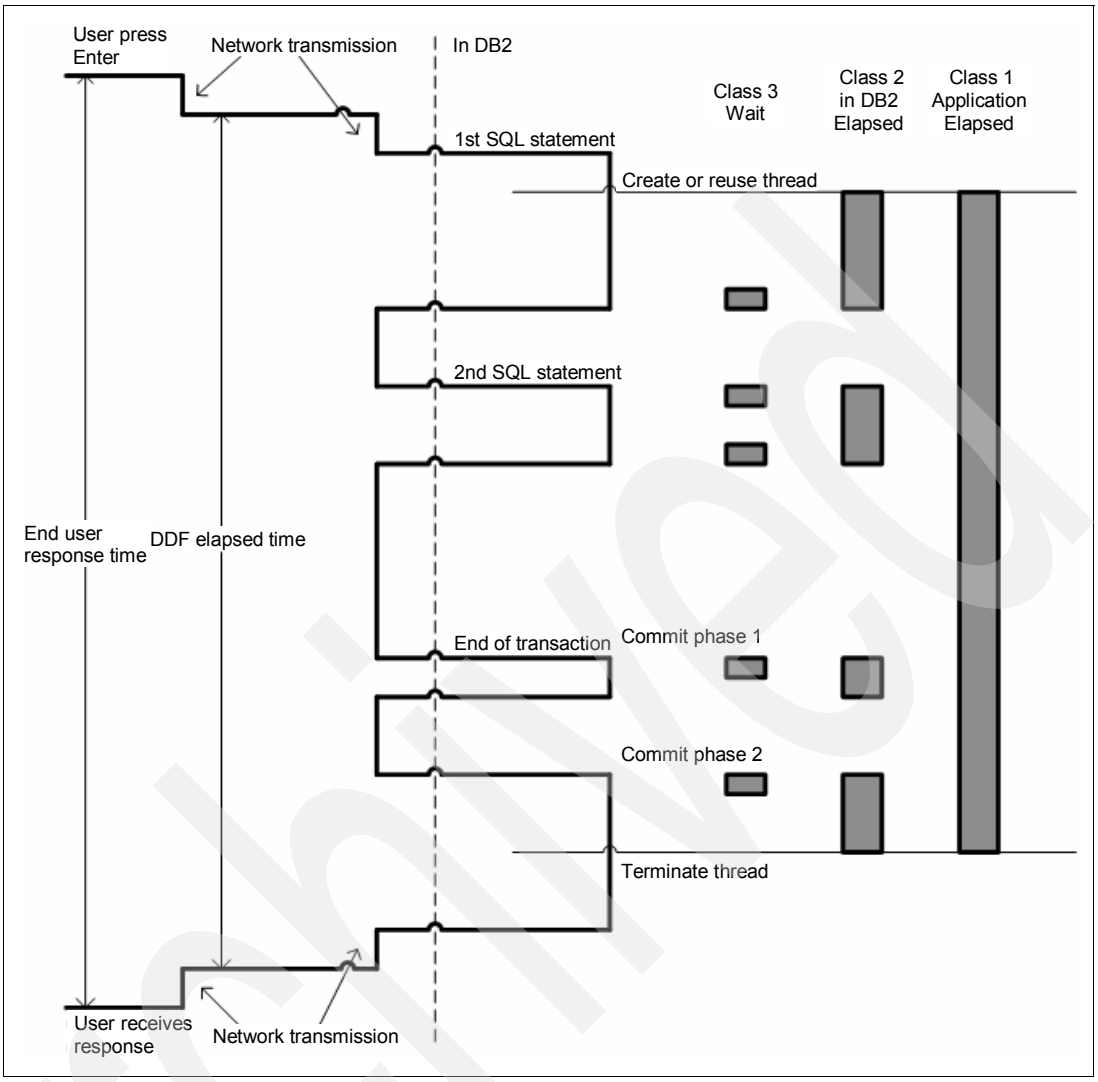


Figure C-1 DB2 accounting classes

Preparing DB2 accounting data for service class period sizing

Table C-1 depicts a sample of the type of DB2 accounting data necessary for this technique. This is standard DB2 accounting data, which is captured when running with DB2 Class 1 accounting trace, a standard for most DB2 customers.

Refer to “Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS documentation” for more details at the IBM website:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe_db2.doc/ko2ccb12198.htm

Table C-1 Preparing accounting data for Service Class period sizing

CLIENT_TRANSACTION	CLASS1_SU_CPU	CLASS1_SU_IIP_CPU	Total_Class1_SU	CLASS1_ELAPSED
Report11M	5759	6716	12475	0.610031
Report11M	6885	5590	12475	0.591942
Report6M	6200	6276	12476	1.017922

Archived

CLIENT_TRANSACTION	CLASS1_SU_CPU	CLASS1_SU_IIP_CPU	Total_Class1_SU	CLASS1_ELAPSED
Report11M	6427			

Table C-1 contains the following information:

- CLIENT_TRANSACTION - User transaction name, accounting field QWHCEUTX. For our project we utilized WLM set client information to pass into DB2 the actual IBM Cognos BI report name, for example Report11M.
- CLASS1_ELAPSED - The time covered by the accounting record and calculated from the DB2 field QWACESC - QWACBSC. Special considerations apply when accounting roll-up is active.
- CLASS1_SU_CPU: Indicates:
  - The TCB service units.
  - The accumulated TCB service units for processing stored procedures if stored procedures are present.
  - The accumulated CPU service units for processing parallel tasks. This is valid for query CP parallelism, sysplex query parallelism, and parallel tasks generated by utilities.
  - These CPU service units do not include the service units that were consumed on an IBM specialty engine such as a zIIP.
- CLASS1_SU_IIP_CPU: The sum of several accumulated processor service units consumed while running on an IBM specialty engine in all environments. These service units are consumed when:
  - Running stored procedure requests and triggers on the main application execution unit
  - Satisfying stored procedure requests processed in a DB2 stored procedure or WLM address space
  - Satisfying UDF requests processed in a DB2 stored procedure or WLM address space
  - Running triggers on a nested task

Figure C-2 on page 425 shows a representation of the distribution of the queries by CLASS1_ELAPSED time.

Archive

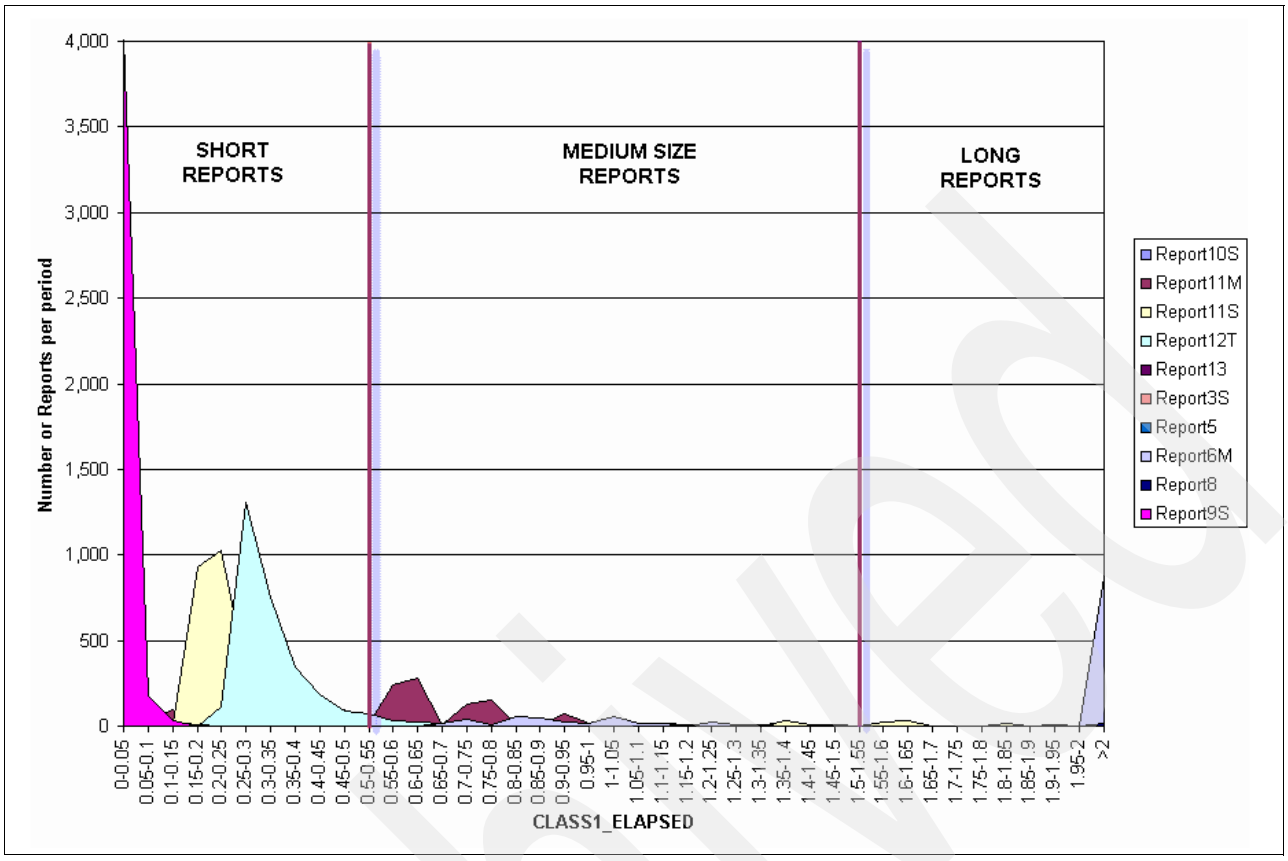


Figure C-2 Distribution of transactions per group of CLASS1_ELAPSED time (RT)

This chart allows us to understand how the queries are grouped by elapsed time or response time (RT). The values on the X axis are CLASS1_ELPASED times of RT grouped by intervals of 0.05 seconds and the Y axis shows the number of reports or queries that were ended within that RT.

The last period indicated with >2 contains the count of reports for which RT is larger than 2 seconds.

This chart can be used to understand how the reports are distributed by RT. Notice that different reports are represented using different colors for a more detailed analysis. If necessary this chart could be limited to a single report type by using the field CLIENT_TRANSACTION for selecting records.

Creating the report

The transaction distribution reports in this section were created using Pivot table functions of a spreadsheet software. Create a Pivot report by selecting the data to be analyzed including at least DB2 CLASS 1 ELAPSED TIME and a Service Unit field such as Total Service Units. An example is shown in Figure C-3 on page 426.

Archive

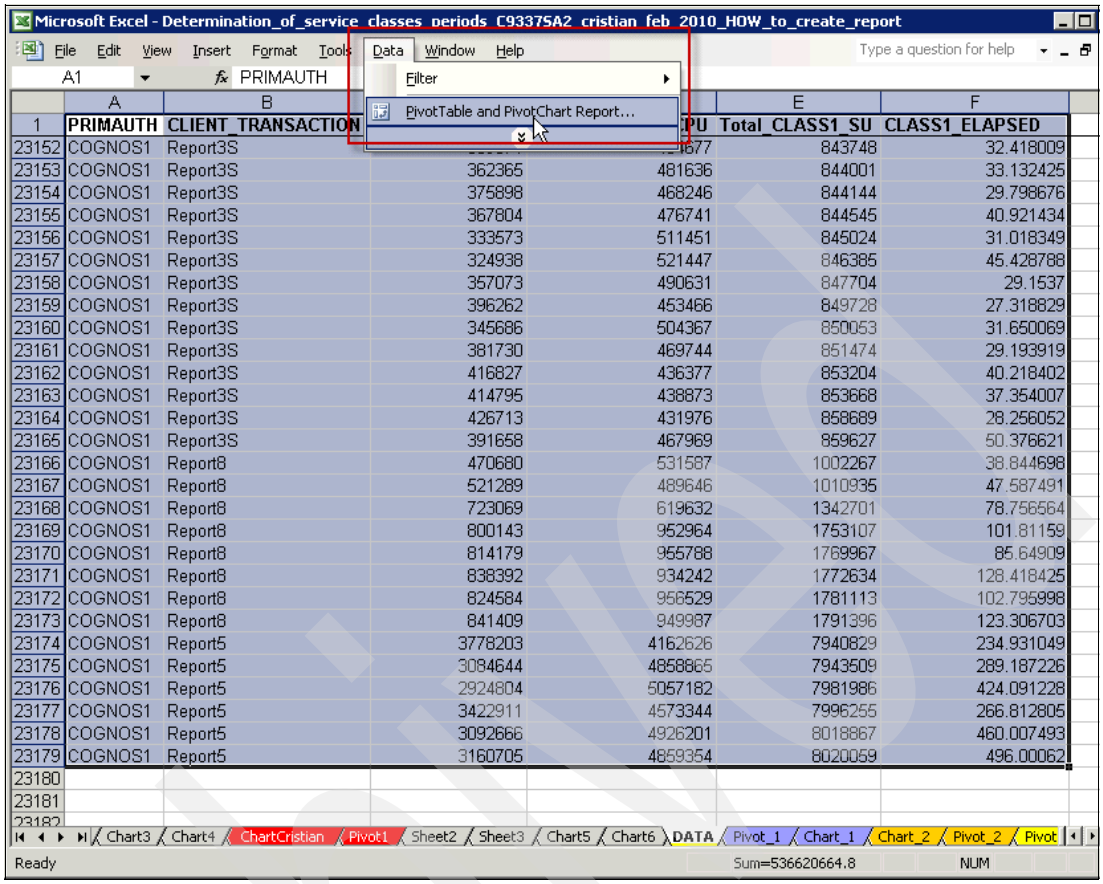


Figure C-3 Creating a workload distribution report: pivot table

Once the pivot table is created, distribute the fields as shown in Figure C-4 on page 427. Make sure that you change the default SUM to COUNT field properties for the filed in the data section as shown in the figure. By doing so, we are able to count the number of transactions that were finished within a particular CLASS1 ELAPSED TIME.

Archive

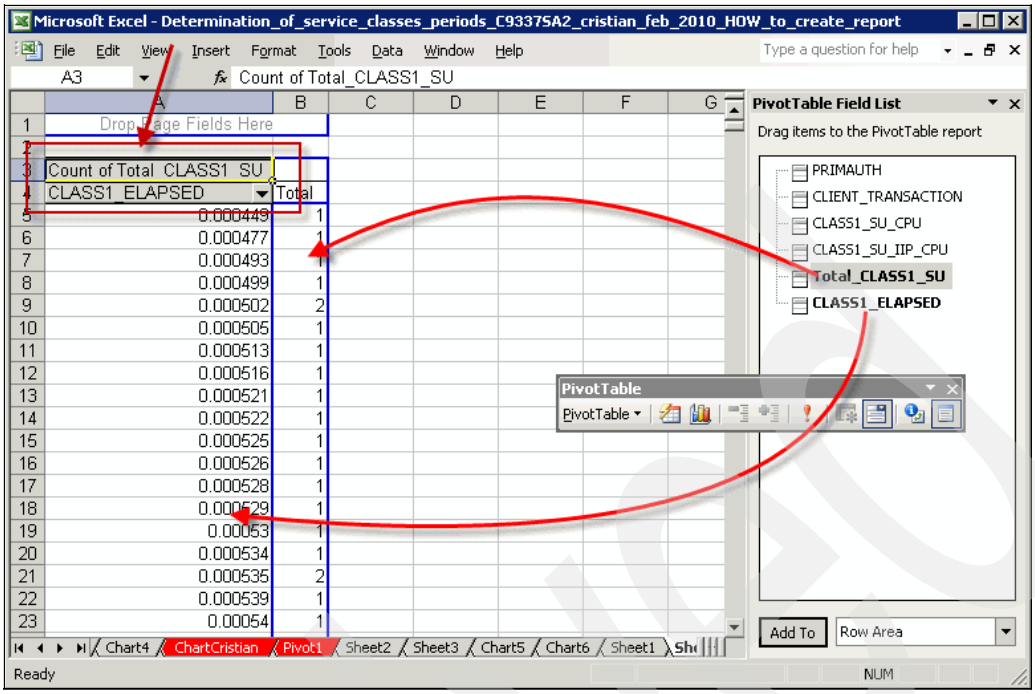


Figure C-4 Creating a workload distribution report: selecting fields

With a right-click on the CLASS 1 column drill down to the **Group and Show Detail** option and then select **Group**; refer to Figure C-5 for details.

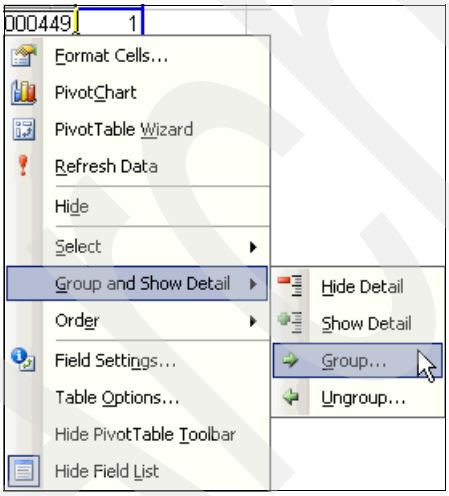


Figure C-5 Creating a workload distribution report: group option

Figure C-6 on page 428 shows the Grouping panel where we define the CLASS 1 ELAPSE time boundaries and grouping size. Boundaries are defined by the **Starting at** option, 0 in our case, and the **Ending at** option, which limit the data being plotted. All the transactions that exceeded this CLASS 1 ELAPSED TIME upper boundary will be grouped in a single last interval. We used an interval of 0.05 seconds for the grouping as defined by the option **By**. The size of the last option and the **Ending at** limit define the number of periods and its granularity in terms of how many transactions each one will contain. Our figures may not fit your data and you may need to test several combinations of these parameters in an iterative process.

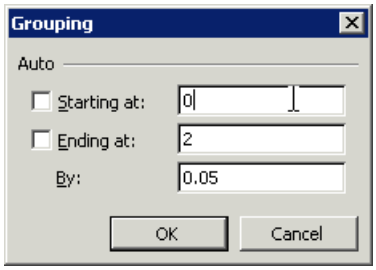


Figure C-6 Creating a workload distribution report: grouping data

Figure C-7 shows an example of how the data looks before creating the chart using the chart options of the Pivot table function.

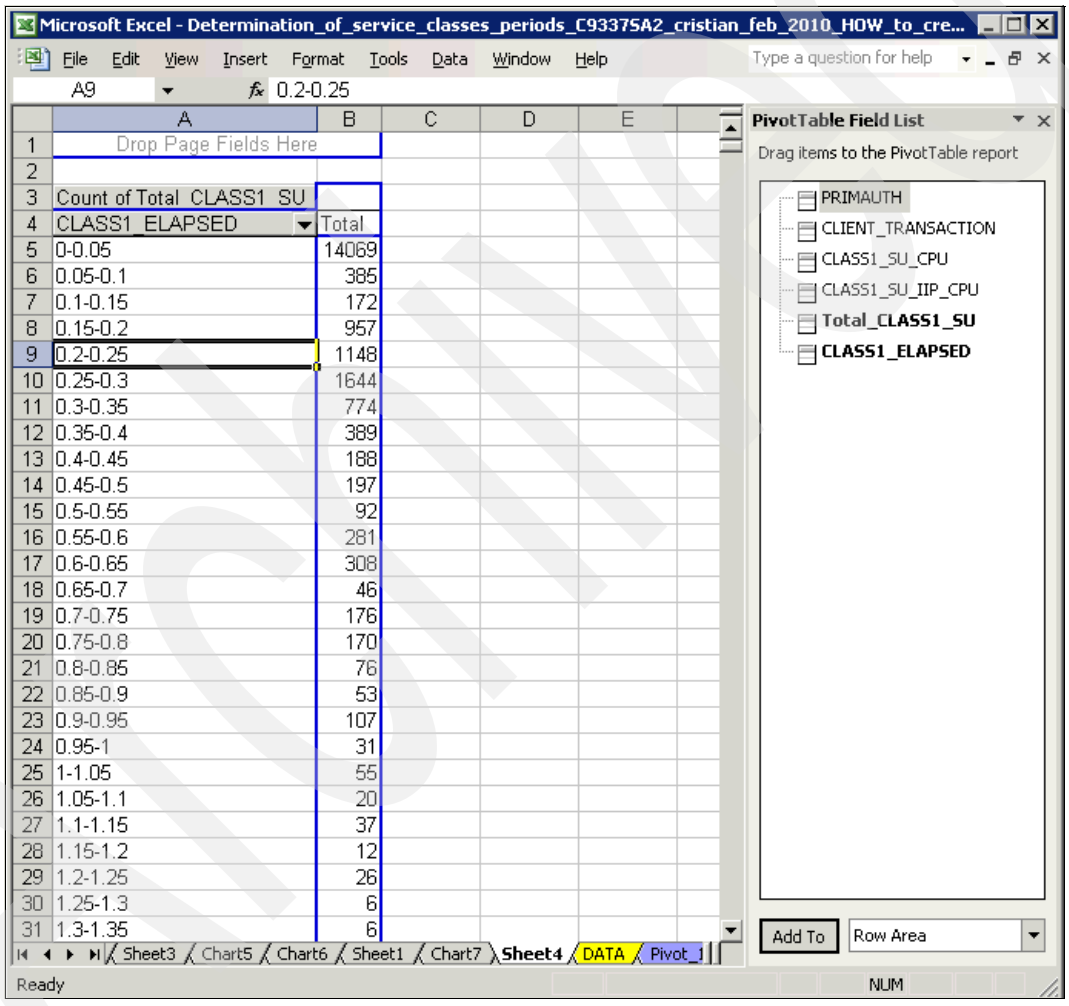


Figure C-7 Creating a workload distribution report: grouping data example

Analyzing the report

In Figure C-8 on page 429 we can identify three sections:

- SHORT REPORTS - We can see that a large number of reports end in less than 0.5 seconds. These are short reports executed often.
- MEDUIM SIZE REPORTS - A less important number of queries ends between 0.5 and 1.5 seconds.

- LONG REPORTS - We classified as long reports that require more than 1.5 seconds to be finished.

The chart in Figure C-8 shows how the queries are distributed using this classification in our test scenario.

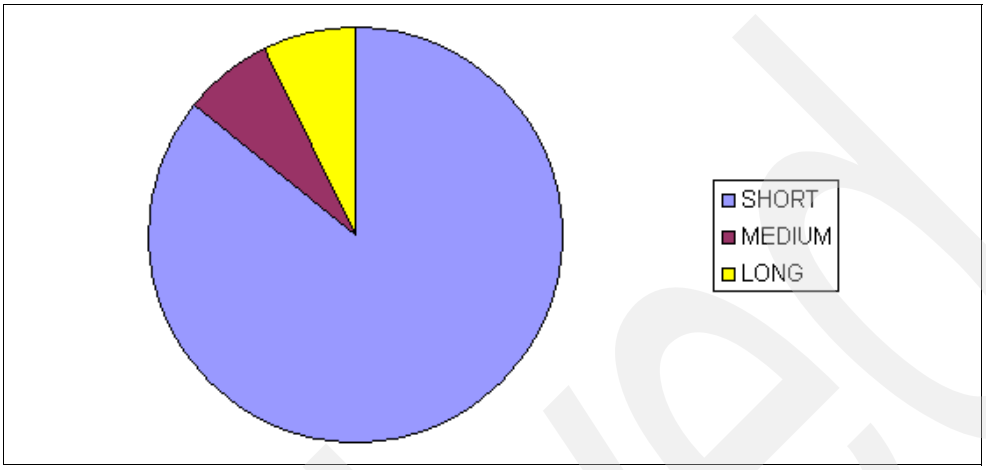


Figure C-8 Classification of queries according to its RT

The range used for the CLASS1_ELAPSED axis will vary depending on the workload distribution for each installation and/or application.

Keeping the same RT distribution, Figure C-9 shows the accumulated Service Units (SU) consumed per period for all the queries. SU includes the CP and zIIP service times.

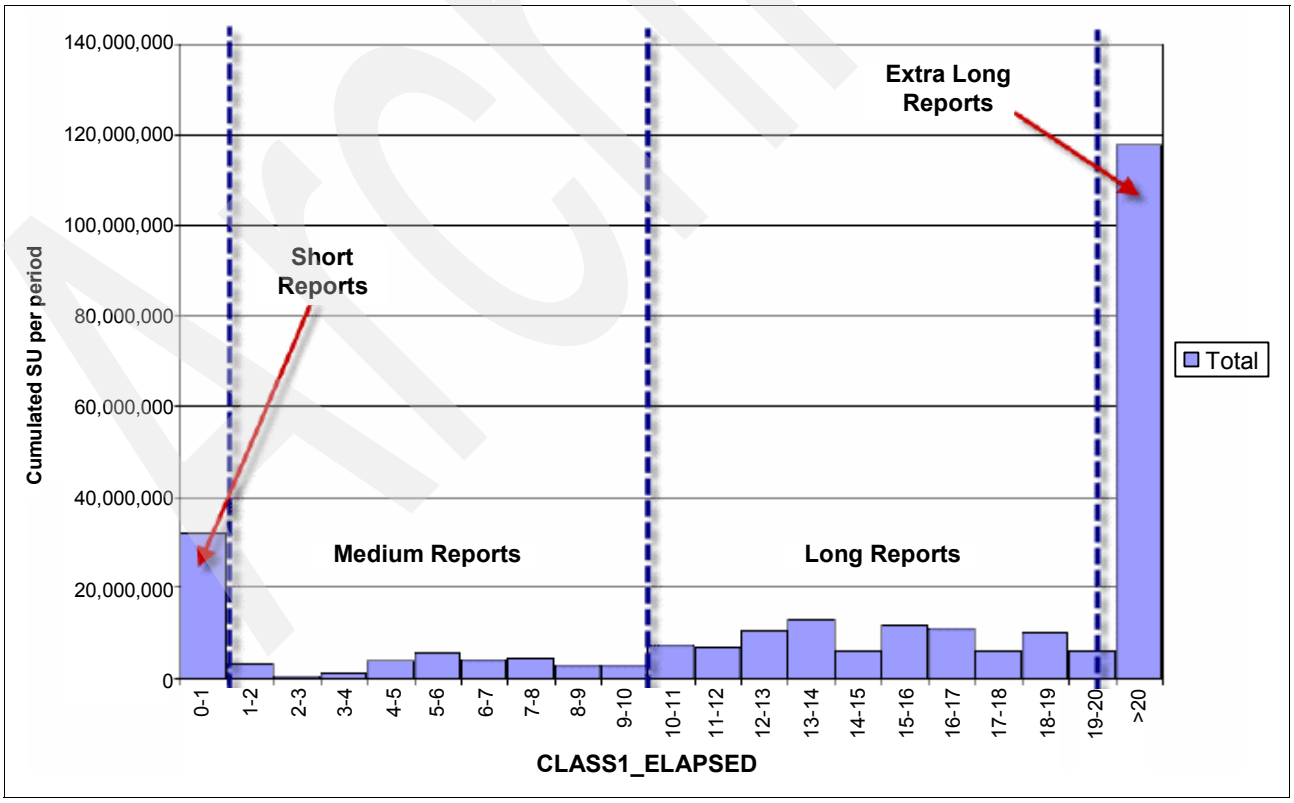


Figure C-9 Distribution of Service Units per group of CLASS1_ELAPSED time (RT)

Archived

This chart shows us that the SUs used by the numerous SHORT REPORTS are actually quite small compared with the SUs used by the few LONG REPORTS. This distribution reflects what we were expecting from our workload and may represent a typical environment where long-running queries may monopolize resources for a long time.

Figure C-9 on page 429 allows us to understand how SUs are used in our environment, and it is possible to identify four distinct workload profiles:

- Short reports with an elapsed time of 1 second or less
- Medium reports with a duration between 2 and 10 seconds
- Long reports with elapsed times between 11 and 20 seconds
- Extra long reports with a high resource consumption and long elapsed time

The challenge is to size the periods of a service class in order to reflect this classification. Remember that the ultimate goal is to protect the important work. In our example, we are merely providing more consistent response times for the shorter queries. We need to now build a chart that would help us define the size, in SUs, of each period.

Figure C-10 shows the cumulative SUs per transaction (average) per RT period.

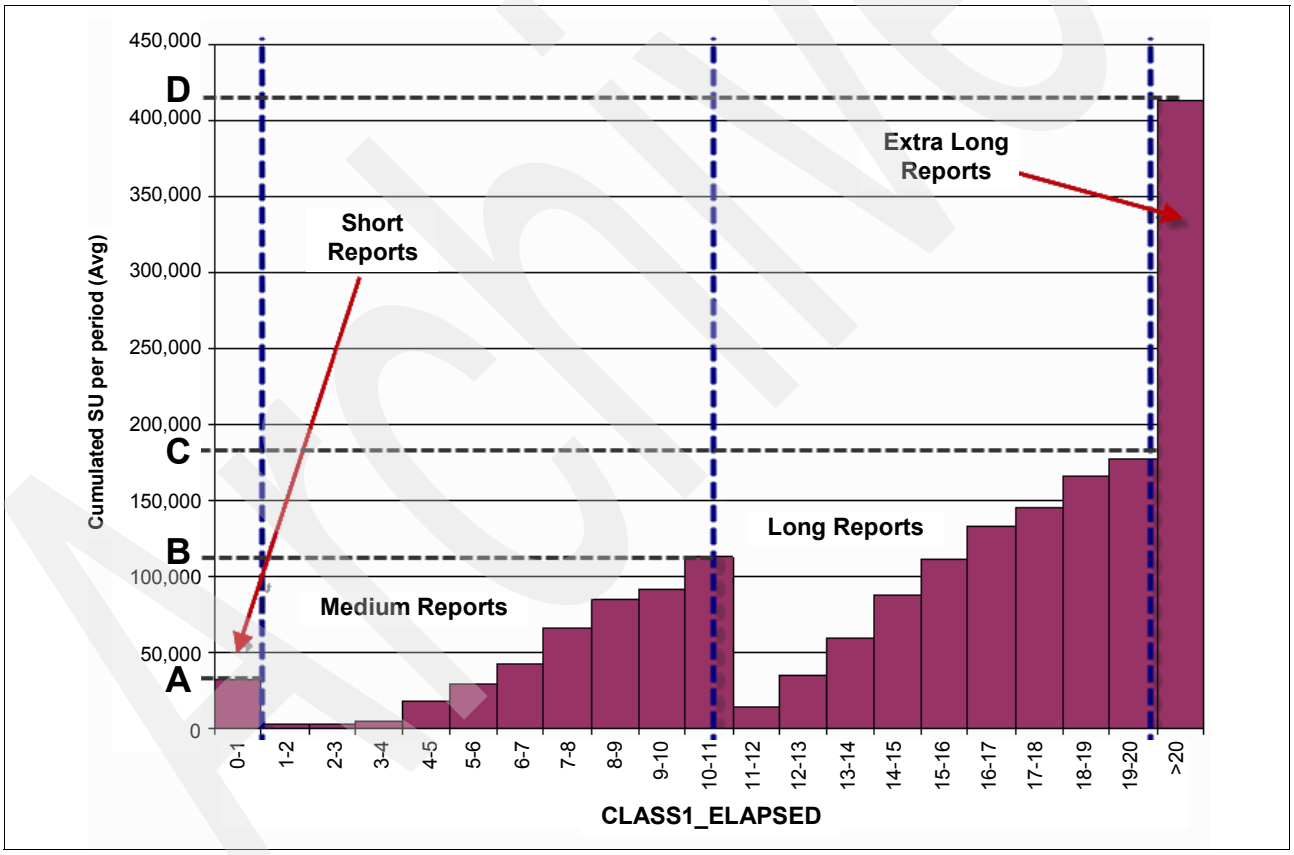


Figure C-10 Service units per response time bucket

We can identify the report classification as represented in Figure C-10. Because we are looking for the correct size in SUs for each period, we accumulate the SUs that are being consumed on each one of the 4 periods. When another period starts, the SU counter is reset to 0 and this explains the shape of the figure. The Y-axis shows the SUs per period for a single transaction. Each of the period limits is identified by the letters A to D, which represent the period duration.

Figure C-10 on page 430 shows us that under the circumstances in which the accounting data was collected, a transaction showing an RT of 1 second would have used about 30,000 SU (CPU + zIIP) time; point A in the chart.

We can use this point for the sizing of the first period, as shown in Table C-2. The Importance used in this table is not derived from the charts but is dictated by the business priorities. As discussed above, we use a response time goal for the first period with an objective derived from the chart.

Table C-2 Sizing the first SC period

Period	Duration	Importance	Goal
1	30,000	2	RT = 1 secs

The second period starts at point A and ends at point B, and as shown in the chart, a transaction that runs across all of this period would use an approximate total of 120,000 SUs. We used this information to size the second period, with less importance than the previous period. Table C-3 summarizes this information. Table C-4 summarizes this information for the third period.

Table C-3 Sizing the second SC period

Period	Duration	Importance	Goal
2	120,000	3	RT = 10 secs

Table C-4 Sizing the third SC period

Period	Duration	Importance	Goal
3	400,000	4	Velocity 30

Table C-5 summarizes the final shape of the Service Class.

Table C-5 Periods sizing summary

Period	Duration	Importance	Goal
1	30,000	2	RT = 0.5 secs
2	120,000	3	RT = 10 secs
3	400,000	4	Velocity 30
4		5	Velocity 10

Effects of changing the service class definitions

After introducing changes into the service class definition, the shape of the workload distribution as shown in Figure C-2 on page 425 will probably evolve. Depending on the definitions adopted for each period, the observed RT will change. For instance, the request falling into the first period would see its RT reduced as a consequence of getting more resources, queries in the second period may see the response time increased to some degree, and it is to be expected that the queries falling into the last period will be strongly delayed depending on the system workload. While this is a normal consequence of distributing resources, it also indicates that the exercise of sizing the service class periods can be an iterative process depending on how these changes follow the business requirements of your installation.

You need at least to verify the impacts on the queries' execution. One way of doing this is to repeat the process described here.

Recall that the use of durations is essentially what is going to differentiate our trivial, small, medium, large, and very large resource consumers.

Archived

D

Sample WLM service definition

In Chapter 11, “Resource management of data warehouse mixed workloads” on page 263 we described a process for building a WLM service definition to support data warehouse workloads. In this appendix we provide a sample service definition.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

Sample service definition

```
* Service Definition CL_WSCI - Colocation w WLM Set Client info

    12 workloads, with 23 service classes
    0 resource groups
    4 service policies
    1 classification group
    15 subsystem types
    83 report classes
    22 application environments
    0 scheduling environments
    0 resources

    CPU coefficient of 1.0
    IOC coefficient of 0.5
    MSO coefficient of 0.0001
    SRB coefficient of 1.0

    I/O priority management           Yes
    Dynamic alias tuning management   Yes
```

D.1 Service policies

D.1.1 Service Policy CL_OPTML - Optimal classification for management

D.1.1.1 Workload DWMAINT - data warehouse maintenance

```
          2 service classes are defined in this workload.

           * Service Class DWBATHI - High Priority DW Batch

             Base goal:
             CPU Critical flag: NO

               #  Duration   Imp  Goal description
               -  ---------  -    ---------------------------------------
               1             1    Execution velocity of 70


           * Service Class DWBATLO - Low Priority DW Batch (default)

             Base goal:
             CPU Critical flag: NO

               #  Duration   Imp  Goal description
               -  ---------  -    ---------------------------------------
               1                  Discretionary

1         ============================================================
```

Archived

D.1.1.2 Workload DWQUERY - Data warehouse query workload

```
2 service classes are defined in this workload.

  * Service Class DWDDFHI - DW OLAP DDF High Importance

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -   ---------------------------------------
      1  25000      2    90% complete within 00:00:03.000
      2  200000     2    80% complete within 00:00:15.000
      3  1000000    3    Execution velocity of 30
      4             4    Execution velocity of 20

  * Service Class DWDDFMD - DW OLAP DDF Med to Low Importnce

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -   ---------------------------------------
      1  25000      3    90% complete within 00:00:03.000
      2  500000     4    Execution velocity of 5
      3               Discretionary

============================================================
```

D.1.1.3 Workload DWREFRSH - Data warehouse refresh workload

```
1 service class is defined in this workload.

  * Service Class DWDDFREF - DW IFW Refresh

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -   ---------------------------------------
      1             3    Execution velocity of 30

============================================================
```

D.1.1.4 Workload DWSCHREP - DW scheduled reporting

```
2 service classes are defined in this workload.

  * Service Class DWSCHDHI - DW Scheduled Reports High Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
```

Archived

```
    -  ---------  -    ----------------------------------------
    1             3    Execution velocity of 70

  * Service Class DWSCHDMD - DW Scheduled Reports Med Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ----------------------------------------
      1             4    Execution velocity of 30

================================================================
```

D.1.1.5 Workload DWSTC - Data warehouse STC workload

```
2 service classes are defined in this workload.

  * Service Class DWDB2SYS - DataWarehouse DB2 System @spaces

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ----------------------------------------
      1             1    Execution velocity of 45

  * Service Class DWSTCHI - High Priority DW STC

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ----------------------------------------
      1             2    Execution velocity of 65

================================================================
```

D.1.1.6 Workload NOWKLD - Work needs to be classified

```
1 service class is defined in this workload.

  * Service Class DDFDFLT - DDF Default Service Class

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ----------------------------------------
      1                  Discretionary

================================================================
```

Archived

D.1.1.7 Workload OPERATNL - OLTP online workload

```
          3 service classes are defined in this workload.

            * Service Class STCOHIGH - High Priority OLTP STC

              Base goal:
              CPU Critical flag: NO
1
                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1             1    Execution velocity of 30

             * Service Class WASTCHI-Bkstor TC High

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1             2    90% complete within 00:00:00.250
            * Service Class WASTCLOW-Bkstor TC Low

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1             3    90% complete within 00:00:00.900

          ============================================================
```

D.1.1.8 Workload OPERBAT - Batch workload in uspport of OLTP

```
          3 service classes are defined in this workload.

            * Service Class BATCHHI - High Priority Batch

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1             1    Execution velocity of 70

            * Service Class BATCHLOW - Low Priority Batch (Default)

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1                  Discretionary

            * Service Class BATCHMED - Medium Priority Batch
```

Archived

```
              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1             4    Execution velocity of 40

============================================================
```

D.1.1.9 Workload OPERSTC - OLTP STC workload

```
          1 service class is defined in this workload.

            * Service Class OPDB2SYS - Opertnl DB2  System @spaces

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1             1    Execution velocity of 70

          ============================================================
```

D.1.1.10 Workload SERVICES - Additional support services

```
          4 service classes are defined in this workload.

            * Service Class OMVSHI - High Priority OMVS

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1  500        3    80% complete within 00:00:00.500
                2             4    Execution velocity of 50

            * Service Class PERFLO - Low  Priority work

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1               Discretionary

            * Service Class STCHIGH - High Priority STC

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
```

```
    -  ---------  -    ---------------------------------------
    1             1    Execution velocity of 65

  * Service Class STCLOW - Low  Priority STC (Default)

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

=============================================================
```

D.1.1.11 Workload TSO

```
2 service classes are defined in this workload.

  * Service Class TSOHI - TSO High  System Admin

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 90

  * Service Class TSOUSER - TSO user

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 90
*************************************************************
```

D.1.2 Service Policy CL_POWR - Optimal policy for power users

D.1.2.1 Workload DWMAINT - Data warehouse maintenance

```
2 service classes are defined in this workload.

  * Service Class DWBATHI - High Priority DW Batch

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 70

  * Service Class DWBATLO - Low Priority DW Batch (default)
```

Archived

```
    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

=============================================================
```

D.1.2.2 Workload DWQUERY - Data warehouse query workload

```
2 service classes are defined in this workload.

  * Service Class DWDDFHI - DW OLAP DDF High Importance

    Overridden goal in policy CL_POWR:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1  25000      2    90% complete within 00:00:03.000
      2  200000     2    80% complete within 00:00:15.000
      3  1000000    3    Execution velocity of 40
      4             4    Execution velocity of 10


  * Service Class DWDDFMD - DW OLAP DDF Med to Low Importnce

    Overridden goal in policy CL_POWR:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1  25000      4    90% complete within 00:00:15.000
      2  8000000    5    Execution velocity of 5
      3                  Discretionary

=============================================================
```

D.1.2.3 Workload DWREFRSH - Data warehouse refresh workload

```
1 service class is defined in this workload.

  * Service Class DWDDFREF - DW IFW Refresh

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             3    Execution velocity of 30

=============================================================
```

D.1.2.4 Workload DWSCHREP - DW scheduled reporting

```
2 service classes are defined in this workload.

  * Service Class DWSCHDHI - DW Scheduled Reports High Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            3    Execution velocity of 70


  * Service Class DWSCHDMD - DW Scheduled Reports Med Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            4    Execution velocity of 30

============================================================
```

D.1.2.5 Workload DWSTC - Data warehouse STC workload

```
2 service classes are defined in this workload.

  * Service Class DWDB2SYS - DataWarehouse DB2 System @spaces

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            1    Execution velocity of 45

  * Service Class DWSTCHI - High Priority DW STC

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            2    Execution velocity of 65

============================================================
```

D.1.2.6 Workload NOWKLD - Work needs to be classified

```
1 service class is defined in this workload.

  * Service Class DDFDFLT - DDF Default Service Class
```

Archived

```
                Base goal:
                CPU Critical flag: NO

                  #  Duration   Imp  Goal description
                  -  ---------  -    ---------------------------------------
                  1                  Discretionary

              ============================================================
```

D.1.2.7 Workload OPERATNL - OLTP online workload

```
              3 service classes are defined in this workload.

                * Service Class STCOHIGH - High Priority OLTP STC

                  Base goal:
1                 CPU Critical flag: NO

                    #  Duration   Imp  Goal description
                    -  ---------  -    ---------------------------------------
                    1             1    Execution velocity of 30

                * Service Class WASTCHI - Bkstor TC High

                  Overridden goal in policy CL_POWR:
                  CPU Critical flag: YES

                    #  Duration   Imp  Goal description
                    -  ---------  -    ---------------------------------------
                    1             2    90% complete within 00:00:00.250

                * Service Class WASTCLOW - Bkstor TC Low

                  Base goal:
                  CPU Critical flag: NO

                    #  Duration   Imp  Goal description
                    -  ---------  -    ---------------------------------------
                    1             3    90% complete within 00:00:00.900

              ============================================================
```

D.1.2.8 Workload OPERBAT - Batch workload in support of OLTP

```
              3 service classes are defined in this workload.

                * Service Class BATCHHI - High Priority Batch

                  Base goal:
                  CPU Critical flag: NO

                    #  Duration   Imp  Goal description
                    -  ---------  -    ---------------------------------------
```

```
     1            1    Execution velocity of 70

  * Service Class BATCHLOW - Low Priority Batch (Default)

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

  * Service Class BATCHMED - Medium Priority Batch

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1            4    Execution velocity of 40

=============================================================
```

D.1.2.9 Workload OPERSTC - OLTP STC workload

```
1 service class is defined in this workload.

  * Service Class OPDB2SYS - Opertnl DB2  System @spaces

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1            1    Execution velocity of 70

=============================================================
```

D.1.2.10 Workload SERVICES - Additional support services

```
4 service classes are defined in this workload.

  * Service Class OMVSHI - High Priority OMVS

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1  500       3    80% complete within 00:00:00.500
      2            4    Execution velocity of 50

  * Service Class PERFLO - Low  Priority work

    Base goal:
```

Archived

```
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

  * Service Class STCHIGH - High Priority STC

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 65

  * Service Class STCLOW - Low  Priority STC (Default)

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

=============================================================
```

D.1.2.11 Workload TSO

```
2 service classes are defined in this workload.

  * Service Class TSOHI - TSO High  System Admin

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 90

  * Service Class TSOUSER - TSO user

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 90

************************************************************
```

D.1.3 Service Policy CL_RFRSH - Policy to favor SQW refresh

D.1.3.1 Workload DWMAINT - Data warehouse maintenance

```
2 service classes are defined in this workload.

            * Service Class DWBATHI - High Priority DW Batch

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1             1    Execution velocity of 70

            * Service Class DWBATLO - Low Priority DW Batch (default)

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                 1                 Discretionary

          ============================================================
```

D.1.3.2 Workload DWQUERY - Data warehouse query workload

```
          2 service classes are defined in this workload.

            * Service Class DWDDFHI - DW OLAP DDF High Importance

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1  25000      2    90% complete within 00:00:03.000
                2  200000     2    80% complete within 00:00:15.000
                3  1000000    3    Execution velocity of 30
                4             4    Execution velocity of 20

            * Service Class DWDDFMD - DW OLAP DDF Med to Low Importnce

              Base goal:
              CPU Critical flag: NO

                #  Duration   Imp  Goal description
                -  ---------  -    ---------------------------------------
                1  25000      3    90% complete within 00:00:03.000
                2  500000     4    Execution velocity of 5
                3               Discretionary

          ============================================================
```

D.1.3.3 Workload DWREFRSH - Data warehouse refresh workload

```
          1 service class is defined in this workload.
```

```
  * Service Class DWDDFREF - DW IFW Refresh

    Overridden goal in policy CL_RFRSH:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            2    Execution velocity of 80

============================================================
```

D.1.3.4 Workload DWSCHREP - DW scheduled reporting

```
2 service classes are defined in this workload.

  * Service Class DWSCHDHI - DW Scheduled Reports High Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      3    Execution velocity of 70

  * Service Class DWSCHDMD - DW Scheduled Reports Med Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            4    Execution velocity of 30

============================================================
```

D.1.3.5 Workload DWSTC - Data warehouse STC workload

```
2 service classes are defined in this workload.

  * Service Class DWDB2SYS - DataWarehouse DB2 System @spaces

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            1    Execution velocity of 45

  * Service Class DWSTCHI - High Priority DW STC

    Base goal:
    CPU Critical flag: NO
```

Archived

```
      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             2    Execution velocity of 65

=============================================================
```

D.1.3.6 Workload NOWKLD - Work needs to be classified

```
1 service class is defined in this workload.

  * Service Class DDFDFLT - DDF Default Service Class

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

=============================================================
```

D.1.3.7 Workload OPERATNL - OLTP online workload

```
3 service classes are defined in this workload.

  * Service Class STCOHIGH - High Priority OLTP STC

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 30

  * Service Class WASTCHI - Bkstor TC High

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             2    90% complete within 00:00:00.250

  * Service Class WASTCLOW - Bkstor TC Low

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             3    90% complete within 00:00:00.900

=============================================================
```

Archived

D.1.3.8 Workload OPERBAT - Batch workload in support of OLTP

```
3 service classes are defined in this workload.

  * Service Class BATCHHI - High Priority Batch

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            1    Execution velocity of 70

  * Service Class BATCHLOW - Low Priority Batch (Default)

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1               Discretionary

  * Service Class BATCHMED - Medium Priority Batch

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      4    Execution velocity of 40

============================================================
```

D.1.3.9 Workload OPERSTC - OLTP STC workload

```
1 service class is defined in this workload.

  * Service Class OPDB2SYS - Opertnl DB2  System @spaces

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  --------  -    ---------------------------------------
      1            1    Execution velocity of 70

============================================================
```

D.1.3.10 Workload SERVICES - Additional support services

```
4 service classes are defined in this workload.

  * Service Class OMVSHI - High Priority OMVS
```

Archived

```
  Base goal:
  CPU Critical flag: NO

    #  Duration   Imp  Goal description
    -  ---------  -    ---------------------------------------
    1  500        3    80% complete within 00:00:00.500
    2             4    Execution velocity of 50

* Service Class PERFLO - Low  Priority work

  Base goal:
  CPU Critical flag: NO

    #  Duration   Imp  Goal description
    -  ---------  -    ---------------------------------------
    1               Discretionary


* Service Class STCHIGH - High Priority STC

  Base goal:
  CPU Critical flag: NO

    #  Duration   Imp  Goal description
    -  ---------  -    ---------------------------------------
    1             1    Execution velocity of 65

* Service Class STCLOW - Low  Priority STC (Default)

  Base goal:
  CPU Critical flag: NO

    #  Duration   Imp  Goal description
    -  ---------  -    ---------------------------------------
    1               Discretionary

============================================================
```

D.1.3.11 Workload TSO

```
2 service classes are defined in this workload.

 * Service Class TSOHI - TSO High  System Admin

  Base goal:
  CPU Critical flag: NO

    #  Duration   Imp  Goal description
    -  ---------  -    ---------------------------------------
    1             1    Execution velocity of 90


* Service Class TSOUSER - TSO user

  Base goal:
```

Archived

```
   CPU Critical flag: NO

     #  Duration   Imp  Goal description
     -  ---------  -    ---------------------------------------
     1             1    Execution velocity of 90

************************************************************
```

D.1.4 Service Policy CL_WCRIT - WASTCHI is Critical CPU = YES

D.1.4.1 Workload DWMAINT - Data warehouse maintenance

```
2 service classes are defined in this workload.

  * Service Class DWBATHI - High Priority DW Batch

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 70

  * Service Class DWBATLO - Low Priority DW Batch (default)

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

============================================================
```

D.1.4.2 Workload DWQUERY - Data warehouse query workload

```
2 service classes are defined in this workload.

  * Service Class DWDDFHI - DW OLAP DDF High Importance

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1  25000      2    90% complete within 00:00:03.000
      2  200000     2    80% complete within 00:00:15.000
      3  1000000    3    Execution velocity of 30
      4             4    Execution velocity of 20

  * Service Class DWDDFMD - DW OLAP DDF Med to Low Importnce
```


Archived

```
    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1  25000      3    90% complete within 00:00:03.000
      2  500000     4    Execution velocity of 5
      3               3    Discretionary

==============================================================
```

D.1.4.3 Workload DWREFRSH - Data warehouse refresh workload

```
1 service class is defined in this workload.

  * Service Class DWDDFREF - DW IFW Refresh

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             3    Execution velocity of 30

==============================================================
```

D.1.4.4 Workload DWSCHREP - DW scheduled reporting

```
2 service classes are defined in this workload.

  * Service Class DWSCHDHI - DW Scheduled Reports High Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             3    Execution velocity of 70

  * Service Class DWSCHDMD - DW Scheduled Reports Med Imp

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             4    Execution velocity of 30

==============================================================
```

D.1.4.5 Workload DWSTC - Data warehouse STC workload

```
2 service classes are defined in this workload.
```

```
  * Service Class DWDB2SYS - DataWarehouse DB2 System @spaces

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 45

  * Service Class DWSTCHI - High Priority DW STC

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             2    Execution velocity of 65

=============================================================
```

D.1.4.6 Workload NOWKLD - Work needs to be classified

```
1 service class is defined in this workload.

  * Service Class DDFDFLT - DDF Default Service Class

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             -    Discretionary

=============================================================
```

D.1.4.7 Workload OPERATNL - OLTP online workload

```
3 service classes are defined in this workload.

  * Service Class STCOHIGH - High Priority OLTP STC

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 30

  * Service Class WASTCHI - Bkstor TC High

    Overridden goal in policy CL_WCRIT:
    CPU Critical flag: YES

      #  Duration   Imp  Goal description
```


Archived

```
      -  ---------  -    ---------------------------------------
      1             2    90% complete within 00:00:00.250

  * Service Class WASTCLOW - Bkstor TC Low

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             3    90% complete within 00:00:00.900

=============================================================
```

D.1.4.8 Workload OPERBAT - Batch workload in support of OLTP

```
3 service classes are defined in this workload.

  * Service Class BATCHHI - High Priority Batch

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 70

  * Service Class BATCHLOW - Low Priority Batch (Default)

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1                  Discretionary

  * Service Class BATCHMED - Medium Priority Batch

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
       -  ---------  -    ---------------------------------------
      1             4    Execution velocity of 40

=============================================================
```

D.1.4.9 Workload OPERSTC - OLTP STC workload

```
1 service class is defined in this workload.

  * Service Class OPDB2SYS - Opertnl DB2  System @spaces

    Base goal:
```

```
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 70

================================================================
```

D.1.4.10 Workload SERVICES - Additional support services

```
4 service classes are defined in this workload.

  * Service Class OMVSHI - High Priority OMVS

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1  500        3    80% complete within 00:00:00.500
      2             4    Execution velocity of 50

  * Service Class PERFLO - Low  Priority work

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1               Discretionary

  * Service Class STCHIGH - High Priority STC

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1             1    Execution velocity of 65

  * Service Class STCLOW - Low  Priority STC (Default)

    Base goal:
    CPU Critical flag: NO

      #  Duration   Imp  Goal description
      -  ---------  -    ---------------------------------------
      1               Discretionary

================================================================
```

D.1.4.11 Workload TSO

```
2 service classes are defined in this workload.
```

```
    * Service Class TSOHI - TSO High  System Admin

      Base goal:
      CPU Critical flag: NO

        #  Duration   Imp  Goal description
        -  ---------  -    ---------------------------------------
        1             1    Execution velocity of 90

    * Service Class TSOUSER - TSO user

      Base goal:
      CPU Critical flag: NO

        #  Duration   Imp  Goal description
        -  ---------  -    ---------------------------------------
        1             1    Execution velocity of 90

************************************************************************
```

D.2 Classification by subsystem type

D.2.1 Subsystem Type ASCH - Use Modify to enter *your* rules

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.

================================================================
```

D.2.2 Subsystem Type CB - Rules for WebSphere Application Server (transaction only)

```
Classification:

  Default service class is WASTCLOW
  Default report class is RWASTCDF

    Qualifier  Qualifier      Starting       Service  Report
  # type       name           position       Class    Class
  - ---------- -------------- ---------      -------- --------
  1 TC         WASTCLOW                      WASTCLOW RWASTCLO
  1 TC         WASTCHI                       WASTCHI  RWASTCHI

Descriptions:
```

Archived

```
    Qualifier  Qualifier      Description
  # type       name
  - ---------- -------------- --------------------------------
  1 TC         WASTCLOW
  1 TC         WASTCHI

Descriptions:

    Qualifier  Qualifier      Storage   Manage Region
  # type       name           Critical  Using Goals Of
  - ---------- -------------- --------- --------------
  1 TC         WASTCLOW       NO        N/A
  1 TC         WASTCHI        NO        N/A

================================================================
```

D.2.3 Subsystem Type CICS - Rules for CTS (transaction only)

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.

================================================================
```

D.2.4 Subsystem Type DB2 - Use Modify to enter *your* rules

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.

================================================================
```

D.2.5 Subsystem Type DDF - Rules for DDF (transaction only)

```
Classification:
   Default service class is DDFDFLT
  Default report class is RDDFDFLT

    Qualifier  Qualifier      Starting       Service  Report
  # type       name           position       Class    Class
  - ---------- -------------- ---------      -------- --------
  1 SI         DWPR                          DWDDFMD  RDWPDFLT
  2 . UI       . Cognos1                     DWDDFMD  RCOGPDFL
  3 . . PC     . . PH3DW          25         DWDDFMD  RCBKPKG
```


Archived

```
  4 . . . SPM  . . . userK*                 DWDDFHI  RCBKCRIT
  4 . . . SPM  . . . userS*                 DWDDFHI  RCBKSERV
  4 . . . SPM  . . . userM*                 DWDDFMD  RCBKMED
  4 . . . SPM  . . . SCHEDHI                DWSCHDHI RCSCHDHI
  4 . . . SPM  . . . SCHEDMD                DWSCHDMD RCSCHDMD
  2 . UI       . SQWREF                     DWDDFREF RSQWREF
  1 SI         DWTS                         DWDDFMD  RDWTSTDF

Descriptions:

    Qualifier  Qualifier      Description
  # type       name
  - ---------- -------------- --------------------------------
  1 SI         DWPR
  2 . UI       . Cognos1
  3 . . PC     . . PH3DW
  4 . . . SPM  . . . userK*
  4 . . . SPM  . . . userS*
  4 . . . SPM  . . . userM*
  4 . . . SPM  . . . SCHEDHI
  4 . . . SPM  . . . SCHEDMD
  2 . UI       . SQWREF
  1 SI         DWTS

Descriptions:

    Qualifier  Qualifier      Storage   Manage Region
  # type       name           Critical  Using Goals Of
  - ---------- -------------- --------- --------------
  1 SI         DWPR           NO        N/A
  2 . UI       . Cognos1      NO        N/A
  3 . . PC     . . PH3DW      NO        N/A
  4 . . . SPM  . . . userK*   NO        N/A
  4 . . . SPM  . . . userS*   NO        N/A
  4 . . . SPM  . . . userM*   NO        N/A
  4 . . . SPM  . . . SCHEDHI  NO        N/A
  4 . . . SPM  . . . SCHEDMD  NO        N/A
  2 . UI       . SQWREF       NO        N/A
  1 SI         DWTS           NO        N/A

1 ================================================================
```

D.2.6 Subsystem Type EWLM - Use Modify to enter *your* rules

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.

================================================================
```

Archived

D.2.7 Subsystem Type IMS - Rules for IMS (transaction only)

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.

================================================================
```

D.2.8 Subsystem Type IWEB - Use Modify to enter *your* rules

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.

================================================================
```

D.2.9 Subsystem Type JES - Rules for batch jobs

```
Classification:

  Default service class is BATCHLOW
  There is no default report class.

    Qualifier  Qualifier      Starting       Service  Report
  # type       name           position       Class    Class
  - ---------- -------------- ---------      -------- --------

  1 TN         MDB*                          BATCHMED
  1 TN         HI*                           BATCHHI


================================================================
```

D.2.10 Subsystem Type MQ - Use Modify to enter *your* rules

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.
```

Archived

```
============================================================
```

D.2.11 Subsystem Type OMVS - Rules for forked OMVS work

```
Classification:

  Default service class is PERFLO
  There is no default report class.

 There are no classification rules.
```

D.2.12 Subsystem Type SOM - Use Modify to enter *your* rules

```
Classification:

  There is no default service class.
  There is no default report class.

  There are no classification rules.

============================================================
```

D.2.13 Subsystem Type STC - Rules for started tasks

```
Classification:

  Default service class is STCLOW
  There is no default report class.

    Qualifier  Qualifier      Starting       Service  Report
  # type       name           position       Class    Class
  - ---------- -------------- ---------      -------- --------
  1 TN         DBDMDBM1                      DWDB2SYS RDBDMDBM
  1 TN         DDW1DBM1                      OPDB2SYS RDDW1DBM
  1 TN         DDW2DBM1                      OPDB2SYS RDDW2DBM
  1 TN         DDW3DBM1                      DWDB2SYS RDDW3DBM
  1 TN         DDW4DBM1                      DWDB2SYS RDDW4DBM
  1 TN         DBDMDIST                      DWDB2SYS RDBDMDST
  1 TN         DDW1DIST                      OPDB2SYS RDDW1DST
  1 TN         DDW2DIST                      OPDB2SYS RDDW2DST
  1 TN         DDW3DIST                      DWDB2SYS RDDW3DST
  1 TN         DDW4DIST                      DWDB2SYS RDDW4DST
  1 TN         DBDMMSTR                      DWDB2SYS RDBDMMST
  1 TN         DDW1MSTR                      OPDB2SYS RDDW1MST
  1 TN         DDW2MSTR                      OPDB2SYS RDDW2MST
  1 TN         DDW3MSTR                      DWDB2SYS RDDW3MST
  1 TN         DDW4MSTR                      DWDB2SYS RDDW4MST
  1 TN         DBDMIRLM                      SYSSTC   RDBDMIRL
  1 TN         DDW1IRLM                      SYSSTC   RDDW1IRL
  1 TN         DDW2IRLM                      SYSSTC   RDDW2IRL
```

Archived

```
1 TN         DDW3IRLM                    SYSSTC   RDDW3IRL
1 TN         DDW4IRLM                    SYSSTC   RDDW4IRL
1 TN         DB2WLM*                     STCHIGH  RDDWXWLM
1 TN         DDMWLM*                     STCHIGH  RDBDMWLM
1 TN         BBODMGR*                    SYSSTC   RWASDMGR
1 TN         BBODMN*                     SYSSTC   RWASDMN
1 TN         BBONO%%                     SYSSTC   RWASNODE
1 TN         BBOSO%%                     SYSSTC   RWASCR
1 TN         BBOSO%%A                    STCOHIGH RWASADJ
1 TN         BBOSO%%S                    STCOHIGH RWASSRV
1 TN         CANS*                       STCHIGH  ROMEGAMN
1 TN         %MASTER*                             RMASTER
1 TN         ALLOCAS                              RALLOCAS
1 TN         ANTMAIN                              RANTMAIN
1 TN         CATALOG                              RCATALOG
1 TN         CONSOLE                              RCONSOLE
1 TN         DUMPSRV                              RDUMPSRV
1 TN         GPMSERVE                             RRMF
1 TN         GRS                                  RGRS
1 TN         IOSAS                                RIOSAS
1 TN         IWM*                        STCHIGH  RIWMWEB
1 TN         IXGLOGR                              RIXGLOGR
1 TN         JESXCF                               RJESXCF
1 TN         JES%AUX                              RJESAUX
1 TN         JES*                                 RJES
1 TN         LLA                                  RLLA
1 TN         MVSNFS*                     STCHIGH  RNFS
1 TN         OMVS                                 ROMVS
1 TN         PCAUTH                               RPCAUTH
1 TN         RACF                                 RRACF
1 TN         RASP                                 RRASP
1 TN         RMFGAT*                     SYSSTC   RRMFGAT
1 TN         RMF*                        SYSSTC   RRMF
1 TN         RRS                         SYSSTC   RRRS
1 TN         SMF                                  RSMF
1 TN         TCPIP*                      SYSSTC   RTCPIP
1 TN         TRACE                                RTRACE
1 TN         VLF                                  RVLF
1 TN         VTAM*                       SYSSTC   RVTAM
1 TN         XCFAS                                RXCFAS
1 TN         WLM                                  RWLM
1 TN         ZFS                                  RZFS

1 TN         CHCPROC%                    DWSTCHI  RCDC
1 SPM        SYSTEM                      SYSTEM
1 SPM        SYSSTC                      SYSSTC

Descriptions:

  Qualifier  Qualifier      Storage   Manage Region
# type       name           Critical  Using Goals Of
- ---------- -------------- --------- --------------
1 TN         DBDMDBM1       NO        TRANSACTION
1 TN         DDW1DBM1       NO        TRANSACTION
1 TN         DDW2DBM1       NO        TRANSACTION
```

```
 1 TN         DDW3DBM1       NO        TRANSACTION
 1 TN         DDW4DBM1       NO        TRANSACTION
 1 TN         DBDMDIST       NO        TRANSACTION
 1 TN         DDW1DIST       NO        TRANSACTION
 1 TN         DDW2DIST       NO        TRANSACTION
 1 TN         DDW3DIST       NO        TRANSACTION
 1 TN         DDW4DIST       NO        TRANSACTION
 1 TN         DBDMMSTR       NO        TRANSACTION
 1 TN         DDW1MSTR       NO        TRANSACTION
 1 TN         DDW2MSTR       NO        TRANSACTION
 1 TN         DDW3MSTR       NO        TRANSACTION
 1 TN         DDW4MSTR       NO        TRANSACTION
 1 TN         DBDMIRLM       NO        TRANSACTION
 1 TN         DDW1IRLM       NO        TRANSACTION
 1 TN         DDW2IRLM       NO        TRANSACTION
 1 TN         DDW3IRLM       NO        TRANSACTION
 1 TN         DDW4IRLM       NO        TRANSACTION
 1 TN         DB2WLM*        NO        TRANSACTION
 1 TN         DDMWLM*        NO        TRANSACTION
 1 TN         BBODMGR*       NO        TRANSACTION
 1 TN         BBODMN*        NO        TRANSACTION
 1 TN         BBONO%%        NO        TRANSACTION
 1 TN         BBOSO%%        NO        TRANSACTION
 1 TN         BBOSO%%A       NO        TRANSACTION
 1 TN         BBOSO%%S       NO        TRANSACTION
 1 TN         CANS*          NO        TRANSACTION
 1 TN         %MASTER*       NO        TRANSACTION
 1 TN         ALLOCAS        NO        TRANSACTION
 1 TN         ANTMAIN        NO        TRANSACTION
 1 TN         CATALOG        NO        TRANSACTION
 1 TN         CONSOLE        NO        TRANSACTION
 1 TN         DUMPSRV        NO        TRANSACTION
 1 TN         GPMSERVE       NO        TRANSACTION
 1 TN         GRS            NO        TRANSACTION
 1 TN         IOSAS          NO        TRANSACTION
 1 TN         IWM*           NO        TRANSACTION
 1 TN         IXGLOGR        NO        TRANSACTION
 1 TN         JESXCF         NO        TRANSACTION
 1 TN         JES%AUX        NO        TRANSACTION
 1 TN         JES*           NO        TRANSACTION
 1 TN         LLA            NO        TRANSACTION
 1 TN         MVSNFS*        NO        TRANSACTION
 1 TN         OMVS           NO        TRANSACTION
 1 TN         PCAUTH         NO        TRANSACTION
 1 TN         RACF           NO        TRANSACTION
 1 TN         RASP           NO        TRANSACTION
 1 TN         RMFGAT*        NO        TRANSACTION
 1 TN         RMF*           NO        TRANSACTION
 1 TN         RRS            NO        TRANSACTION
 1 TN         SMF            NO        TRANSACTION
 1 TN         TCPIP*         NO        TRANSACTION
 1 TN         TRACE          NO        TRANSACTION
 1 TN         VLF            NO        TRANSACTION
 1 TN         VTAM*          NO        TRANSACTION
 1 TN         XCFAS          NO        TRANSACTION
```

Archive

```
  1 TN         WLM            NO        TRANSACTION
  1 TN         ZFS            NO        TRANSACTION

  1 TN         CHCPROC%       NO        TRANSACTION
  1 SPM        SYSTEM         NO        TRANSACTION
  1 SPM        SYSSTC         NO        TRANSACTION

 ------------------------------------------------------------------
```

D.2.14 Subsystem Type TCP - Rules for TCP (transaction only)

```
 Classification:

   Default service class is STCHIGH
   Default report class is RTCPIP

   There are no classification rules.

 ------------------------------------------------------------------
```

D.2.15 Subsystem Type TSO - Rules for TSO usage

```
 Classification:

   Default service class is TSOHI
   There is no default report class.

   There are no classification rules.
1********************************************************************
```

D.3 Report classes

```
* Report Class RALLOCAS - ALLOC Address Space
* Report Class RANTMAIN - Concurrent copy support
* Report Class RBKWASAL -
* Report Class RCATALOG - CATALOG Address Space
* Report Class RCDC - InfoSphere Change Data Capture
* Report Class RCOGNOV - RClass IBM Cognos Novice users
* Report Class RCOGOPBI - RClass Operational BI Report
* Report Class RCOGPDFL - RClass IBM Cognos Productn Default
* Report Class RCOGPOWR - RClass IBM Cognos power user
* Report Class RCOGSERV - RClass IBM Cognos Server Center Dflt
* Report Class RCOGUSER - RClass IBM Cognos standard user
* Report Class RCONSOLE - CONSOLE Address Space
* Report Class RCSCHDHI - RClass Scheduled Reports High
* Report Class RCSCHDMD - RClass IBM Cognos Daily Sched Med
* Report Class RDBDMDBM - Rep class DBDM DBM1
* Report Class RDBDMDST - Rep Class DBDM Dist
* Report Class RDBDMIRL - Rep Class DBDM IRLM
* Report Class RDBDMMST - Rep Class DBDM Master
```

Archived

```
* Report Class RDBDMWLM - RC for DBDM Stored proc @spaces
* Report Class RDDFDFLT - Report Class for DDF Default
* Report Class RDDWXWLM - RC for DDWx Stored proc @spaces
* Report Class RDDW1DBM - Rep class for DDW1 DBM1
* Report Class RDDW1DST - Rep Class DDW1 Dist
* Report Class RDDW1IRL - Rep Class DDW1 IRLM
* Report Class RDDW1MST - Rep Class DDW1 Mstr
* Report Class RDDW2DBM -
* Report Class RDDW2DST -
* Report Class RDDW2IRL -
* Report Class RDDW2MST -
* Report Class RDDW3DBM -
* Report Class RDDW3DST -
* Report Class RDDW3IRL -
* Report Class RDDW3MST -
* Report Class RDDW4DBM -
* Report Class RDDW4DST -
* Report Class RDDW4IRL -
* Report Class RDDW4MST -
* Report Class RDUMPSRV - DUMP Services Address Space
* Report Class RDWPDFLT - RClass DW Prod DB2 Default
* Report Class RDWTSTDF - RClass DW DB Test Default
* Report Class RGRS - Global Resource Manager
* Report Class RIOSAS - I/O supervisor      Report Class
* Report Class RIRLM - IMS IRLM Address Space
* Report Class RIWMWEB - IBM HTTP Web Server Report Class
* Report Class RIXGLOGR - System Logger       Report Class
* Report Class RJES - JES2/JES3 Subsystem Report Class
* Report Class RJESAUX - JES2 Additional Support
* Report Class RJESXCF - JES XCF Address Space
* Report Class RLLA - Library-Look-Aside Facility
* Report Class RMASTER - MASTER Address Space
* Report Class RNFS - NFS Server/Client   Report Class
* Report Class ROMEGAMN - Report class for Omegamon
* Report Class ROMVS - OMVS Kernel         Report Class
* Report Class RPCAUTH - PCAUTH Address Space
* Report Class RRACF - Resource Access Mgmt Facility
* Report Class RRASP - Real Storage Mgr Address Space
* Report Class RRMF - Resource Measurement Facility
* Report Class RRMFGAT - RMF MonIII Gatherer Report Class
* Report Class RRRS - Resource Recovery Services
* Report Class RSMF - System Management Facility
* Report Class RSQWREF -
* Report Class RTCPIP - TCP/IP Report Class
* Report Class RTRACE - TRACE Address Space
* Report Class RVLF - Virtual Look-Aside Facility
* Report Class RVTAM - VTAM Report Class
* Report Class RWASADJ - Report Class WAS Adjunct Region
* Report Class RWASCR - Report Class WAS Control Region
* Report Class RWASDFLT - RC Default for WAS transactions
* Report Class RWASDMGR - Report Class WAS Deployment Mgr
* Report Class RWASDMN - Report Class WAS Daemon
* Report Class RWASNODE - Report Class WAS Node Agent
* Report Class RWASSRV - Report Class WAS Servant Regions
* Report Class RWASTCDF - WAS Operatnl transactn Default
```

Archived

```
  * Report Class RWASTCHI - Bookstore OLTP High Imp
  * Report Class RWASTCLO - Bookstore OLTP Medium Imp
  * Report Class RWLM - Workload Manager    Report Class
  * Report Class RXCFAS - Cross System Coupling Facility
  * Report Class RZFS - ZFS Address Space   Report Class
  * Report Class SQWSOURC - SQW Refresh Source Non-Dshare
  * Report Class SQWTARGT - SQW Target DB Insert Non-Dshare
```

D.4 Application environments

```
Appl Environment Name . . DB2WLMJU
    Description . . . . . . . DB2 Stored Procedure for OSC
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DB2WLMJU
    Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENVJU

    Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ------------------------------------------------------------------

    Appl Environment Name . . DDMENV
    Description . . . . . . . DB2 Stored Procedures
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DDMWLM
    Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENV

    Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ------------------------------------------------------------------

   Appl Environment Name . . DDMENVA
   Description . . . . . . . Stored procedures, authorized
   Subsystem type  . . . . . DB2
   Procedure name  . . . . . DDMWLMA
   Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENVA

   Starting of server address spaces for a subsystem instance:
    Managed by WLM

     ------------------------------------------------------------------

   Appl Environment Name . . DDMENVC
   Description . . . . . . . SYSPROC.ADMIN* STORED PROCEDURES
   Subsystem type  . . . . . DB2
   Procedure name  . . . . . DDMWLMC
   Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENVC

   Starting of server address spaces for a subsystem instance:
    Managed by WLM

     ------------------------------------------------------------------
```

Archived

```
     Appl Environment Name . . DDMENVJ
     Description . . . . . . . DB2 Java Stored Procedures
     Subsystem type  . . . . . DB2
     Procedure name  . . . . . DDMWLMJ
     Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENVJ

     Starting of server address spaces for a subsystem instance:
      Managed by WLM

       ================================================================

     Appl Environment Name . . DDMENVJU
     Description . . . . . . . DB2 Stored Procedure for OSC
     Subsystem type  . . . . . DB2
     Procedure name  . . . . . DDMWLMJU
     Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENVJU

     Starting of server address spaces for a subsystem instance:
      Managed by WLM

       ================================================================

     Appl Environment Name . . DDMENVXP
     Description . . . . . . . Stored Procedure, Explain, QT
     Subsystem type  . . . . . DB2
     Procedure name  . . . . . DDMENVXP
     Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENVXP

Starting of server address spaces for a subsystem instance:
     Limited to a single address space per system

       ================================================================

     Appl Environment Name . . DDMENV1
     Description . . . . . . . DB2 Stored Procedures, utilities
     Subsystem type  . . . . . DB2
     Procedure name  . . . . . DDMWLM1
     Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENV1

     Starting of server address spaces for a subsystem instance:
      Limited to a single address space per system

       ================================================================

     Appl Environment Name . . DDMENV3
     Description . . . . . . . Stored Procedures
     Subsystem type  . . . . . DB2
     Procedure name  . . . . . DDMWLM3
     Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENV3

     Starting of server address spaces for a subsystem instance:
      Managed by WLM

       ================================================================
```

Archived

```
    Appl Environment Name . . DDMENV5
    Description . . . . . . . DB2 Stored Procedures, utilities
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DDMWLM5
    Start parameters  . . . . DB2SSN=&IWMSSNM,NUMTCB=8

    Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ================================================================

    Appl Environment Name . . DDMENV6
    Description . . . . . . . Stored Procedures
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DDMWLM6
    Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENV6

Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ================================================================

    Appl Environment Name . . DDMENV7
    Description . . . . . . . Stored Procedures
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DDMWLM7
    Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=DDMENV7

Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ================================================================

   Appl Environment Name . . WLMENV
   Description . . . . . . . DB2 Stored Procedures
   Subsystem type  . . . . . DB2
   Procedure name  . . . . . DB2WLM
   Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENV

   Starting of server address spaces for a subsystem instance:
    Managed by WLM

      ================================================================

    Appl Environment Name . . WLMENVA
    Description . . . . . . . Stored procedures, authorized
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DB2WLMA
    Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENVA

    Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ================================================================
```

Archived

```
Appl Environment Name . . WLMENVC
Description . . . . . . . SYSPROC.ADMIN* STORED PROCEDURES
Subsystem type  . . . . . DB2
Procedure name  . . . . . DB2WLMC
Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENVC

Starting of server address spaces for a subsystem instance:
 Managed by WLM

  ================================================================

Appl Environment Name . . WLMENVJ
Description . . . . . . . DB2 Java Stored Procedures
Subsystem type  . . . . . DB2
Procedure name  . . . . . DB2WLMJ
Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENVJ

Starting of server address spaces for a subsystem instance:
 Managed by WLM

  ================================================================

  Appl Environment Name . . WLMENVXP
Description . . . . . . . Stored Procedure, Explain, QT
Subsystem type  . . . . . DB2
Procedure name  . . . . . DB2WLMEN
Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENVXP

Starting of server address spaces for a subsystem instance:
 Limited to a single address space per system

  ================================================================

Appl Environment Name . . WLMENV1
Description . . . . . . . DB2 Stored Procedures, utilities
Subsystem type  . . . . . DB2
Procedure name  . . . . . DB2WLM1
Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENV1

Starting of server address spaces for a subsystem instance:
 Limited to a single address space per system

  ================================================================

Appl Environment Name . . WLMENV3
Description . . . . . . . Stored Procedures
Subsystem type  . . . . . DB2
Procedure name  . . . . . DB2WLM3
Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENV3

Starting of server address spaces for a subsystem instance:
 Managed by WLM

  ================================================================
```

Archived

```
    Appl Environment Name . . WLMENV5
    Description . . . . . . . DB2 SYSIBM Procedures
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DB2WLM5
    Start parameters  . . . . DB2SSN=&IWMSSNM,NUMTCB=8

     Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ------------------------------------------------------------------

Appl Environment Name . . WLMENV6
    Description . . . . . . . Stored Procedures
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DB2WLM6
    Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENV6

    Starting of server address spaces for a subsystem instance:
     Managed by WLM

      ------------------------------------------------------------------

    Appl Environment Name . . WLMENV7
    Description . . . . . . . Stored Procedures
    Subsystem type  . . . . . DB2
    Procedure name  . . . . . DB2WLM7
    Start parameters  . . . . DB2SSN=&IWMSSNM,APPLENV=WLMENV7

    Starting of server address spaces for a subsystem instance:
     Managed by WLM

********************** End Of Service Definition **********************
```

Archived

E

Setting up the transactional workload

In this appendix we discuss how to set up the Online Transaction Processing (OLTP) workload used in this project, including the following topics:

- EJB Mapping using Rational Application Developer (RAD)
- Application deployment on WebSphere Application Server
- JDBC Connections configuration
- WebSphere setup topology

Archived

© Copyright IBM Corp. 2010. All rights reserved.

E.1 Build BookStore EJB mapping

In this section, we outline the steps to map the EJB to database tables and columns. The BookStore application is a J2EE application. In compliance with the J2EE specifications, different components of code were built into an Enterprise Archive Repository (EAR) file. The EAR file then was deployed onto a WebSphere Application Server. For detailed information and steps required for developing an Enterprise Java Application, refer to IBM Rational Application Developer (RAD) documentation.

EJB mapping using Rational Application Developer (RAD)

There are three types of EJBs: session beans, entity beans, and Message Driven Beans (MDB). An entity bean represents persistent data stored in one row of a relational database table. The entity bean data is saved, either by container management or bean management. Bean-managed persistence is under control of the application to save the bean's state. BookStore uses container-managed persistence (CMP) entity beans; the persistence is managed by the container services. The bean's state is stored in a relational database according to an implicit mapping (CMP 2.0). During deployment, the EJB container is informed about whether the bean data requires persistence, and the container is responsible for implementing the persistence. The container implements the persistence based on the EJB deployment descriptor. We need to perform the EJB mapping that maps the EJB to DB2 backend tables. The following tables in the BookStore OLTP database need to be mapped:

- TITLE
- BOOK_ORDER
- ORDER_DETAIL
- CUSTOMER
- CUSTOMER_ACCOUNT
- STORE

Here are the steps to do the EJB mapping for BookStore:

1. From RAD 7, open the project at the J2EE perspective.
2. Select the **BookStoreModelEJBDB2** project in the Project Explorer panel.
   a. Highlight **BookStoreModelEJBDB2**. Right-click **EJB to RDB Mapping** → **Generate Map,** then select.
   b. Create a new backend folder.
   c. Meet-In-The-Middle.
   d. Create a new connection (uncheck "Retrieve objects created by this user only").
   e. In the Connection Identification GUI panel, specify the information for the connection. See the sample in Figure E-1. For example, in the Specify Class location, insert:

```
C:\Documents and Settings\All Users\Application Data\IBM\Installation
Manager\plugins\com.ibm.datatools.db2_1.0.4.v200807241724\driver\db2jcc.jar;
C:\Documents and Settings\All Users\Application Data\IBM\Installation
Manager\plugins\com.ibm.datatools.db2_1.0.4.v200807241724\driver\db2jcc_lice
nse_cisuz.jar
```

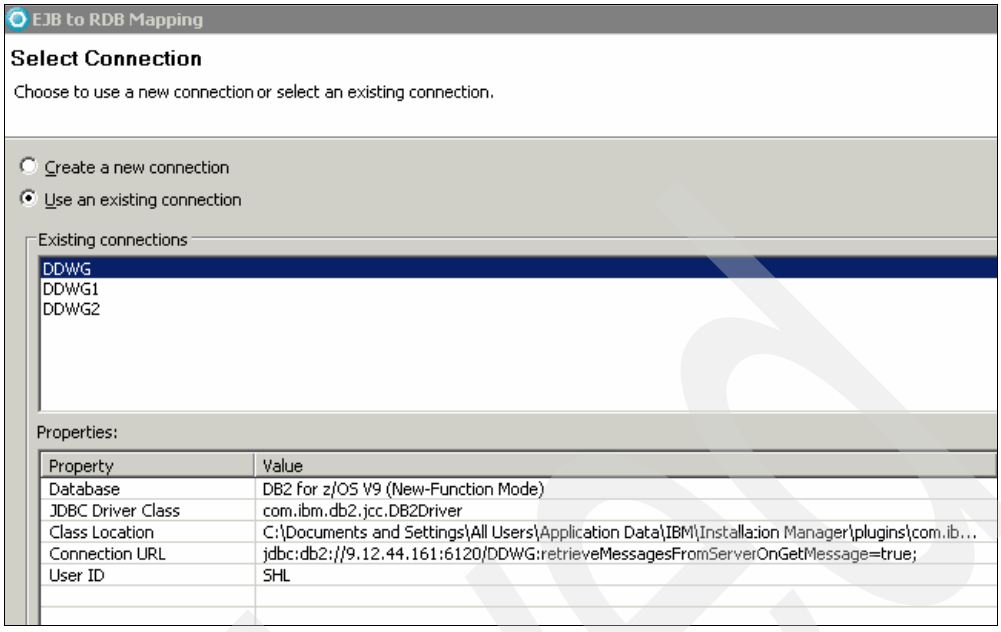


Figure E-1 Sample content of an existing connection definition

f. In the Filter panel, specify the DB schema name, that is, PH3OLTP.

g. Select the tables that need to be mapped.

h. Match by Name.

i. Click **Finish**.

3. The `Map.mapxmi` file should be displayed. The red color in the picture means the mapping is incomplete. Since we did not start mapping yet, we can ignore the warning. See Figure E-2 on page 472.

4. Each enterprise bean needs to be mapped to a table and each of its attributes needs to be mapped to a column.

   Since we selected **Match by Name** for the Meet-in-the-Middle option, the EJB will automatically map to the table of the same name. To complete the mapping for the EJBs that did not match any of the TABLE names, drag and drop items from the Enterprise Beans panels to the corresponding items in the Tables panel. After the EJB is mapped, there should be a blue triangle next to the EJB and an orange triangle next to the table.

   For example: Select the **Customer Account** bean and drag-and-drop it onto the CUSTOMER_ACCOUNT table, as we show in Figure E-2 on page 472.

Archived

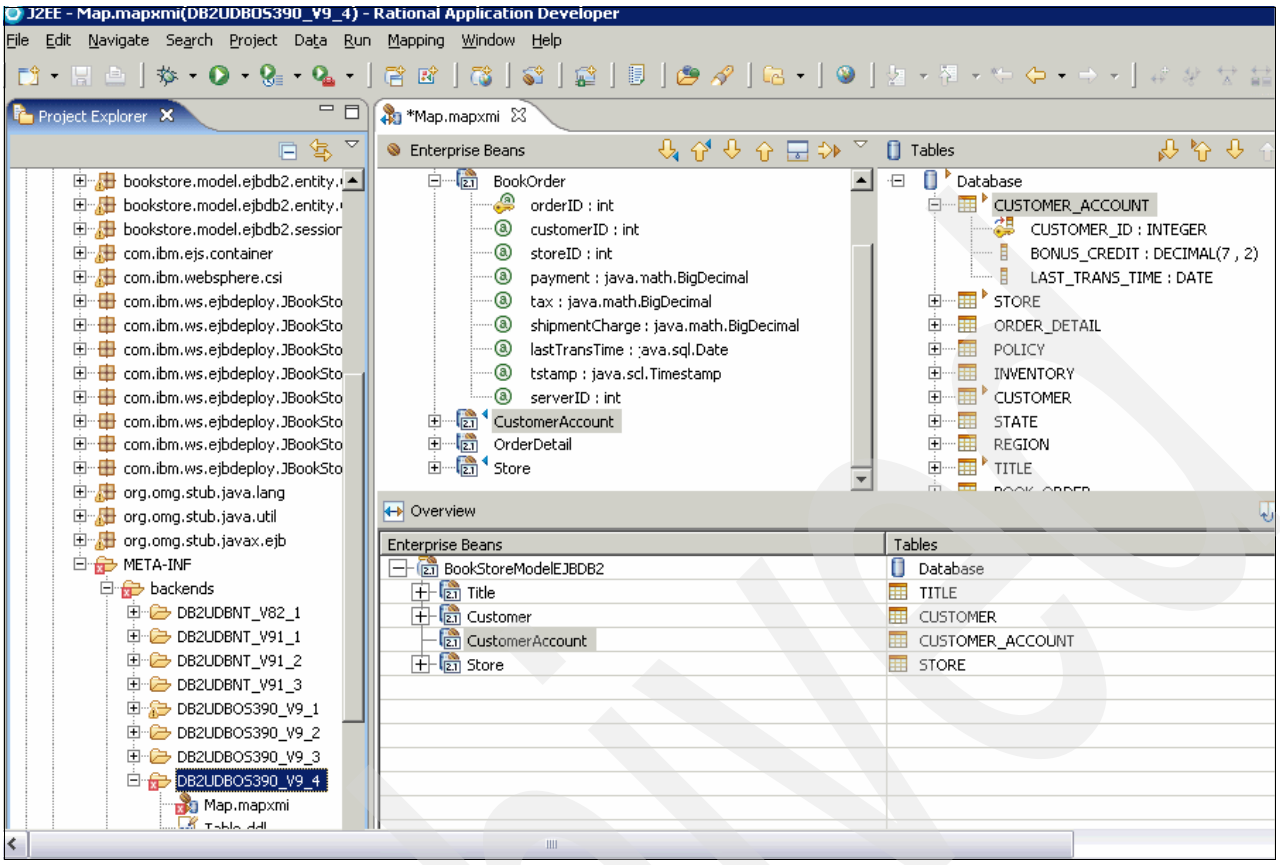


Figure E-2 Perform EJB to RDB mapping from RAD 7 to bring up Map.mapxmi for editing

5. A blue triangle next to the EJB is not necessary to indicate that all the attributes have been mapped. Figure E-3 on page 473 shows that not all the attributes of EJB OrderDetail have been mapped.

   To finish the mapping process:

   a. Expand the EJB OrderDetail in the Enterprise Beans panel.

   b. Drag-and-drop each attribute to the corresponding column of the table ORDER_DETAIL at the Tables panel.

6. When complete, the EJB OrderDetail will show the expanding sign(+) in the Overview panel. See Figure E-3 on page 473.

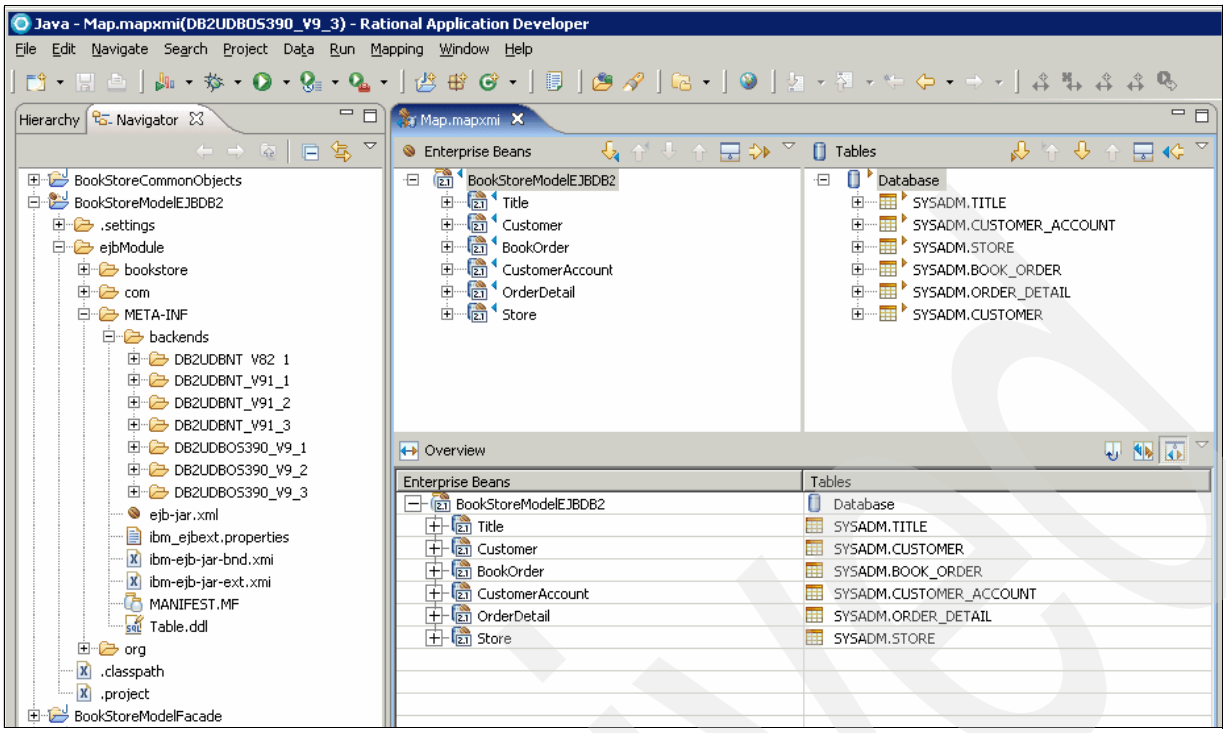


Figure E-3 Map.mapxmi when mapping is complete

Repeat the above process for each unmapped Enterprise Bean and all of its attributes. Mapping is complete when all Enterprise Beans and their attributes are mapped; however, not all tables or table columns need to be mapped.

7. If you get errors that need a composer or converter to be defined, Using “Title” as an example. Refer to Figure E-4 on page 474.
   a. Navigate to the Enterprise Bean “Title” and expand it.
   b. Select the mapping in error, attribute “status” in this case.
   c. In the lower right panel, select the Properties tab (if not already displayed).
   d. Click **None** in the Value column for Transformation; it will reveal a drop-down list.
   e. From the drop-down menu, select an appropriate converter (com,ibm.vap.converters, VapIntegerToStringConverter in this case). The name of the converter indicates the types that are converted, with the database field listed first in the name followed by the EJB attribute.

Archive

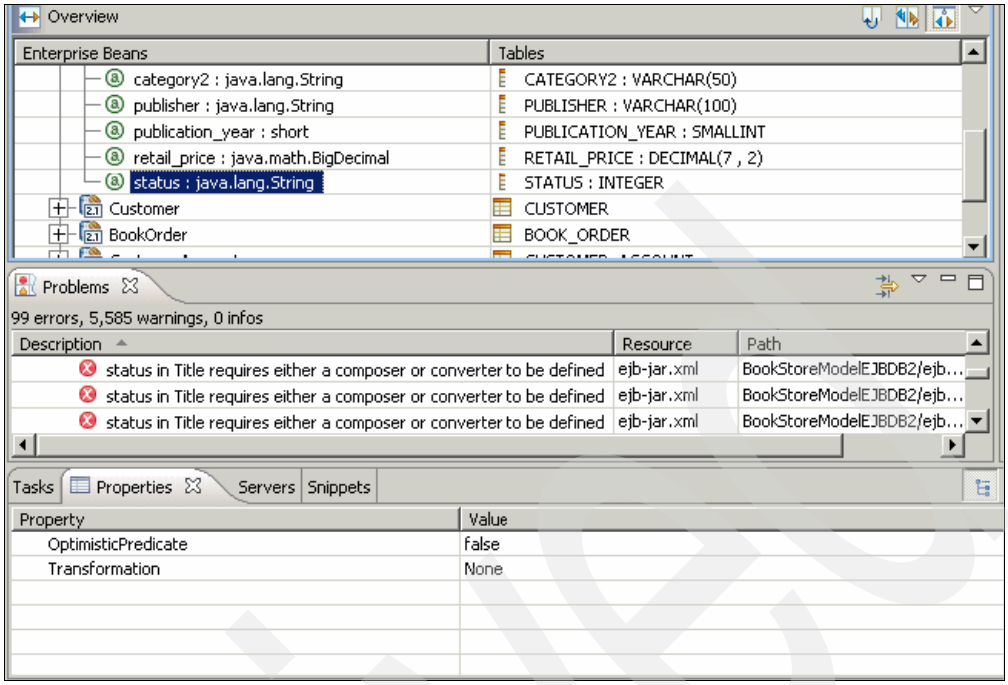


Figure E-4 Fix data type conversion after EJB mapping

8. Generate EJB deployment code.
   a. Select the **BookStoreModelEJBDB2** project in the Project Explorer panel.
   b. Right-click **Prepare for Deployment**.
   c. Then double-click **Deployment Descriptor: BookStoreModelEJBDB2**.
   d. Verify that backend ID is using the new backend that you just created.

See Figure E-5 on page 475.

Archive

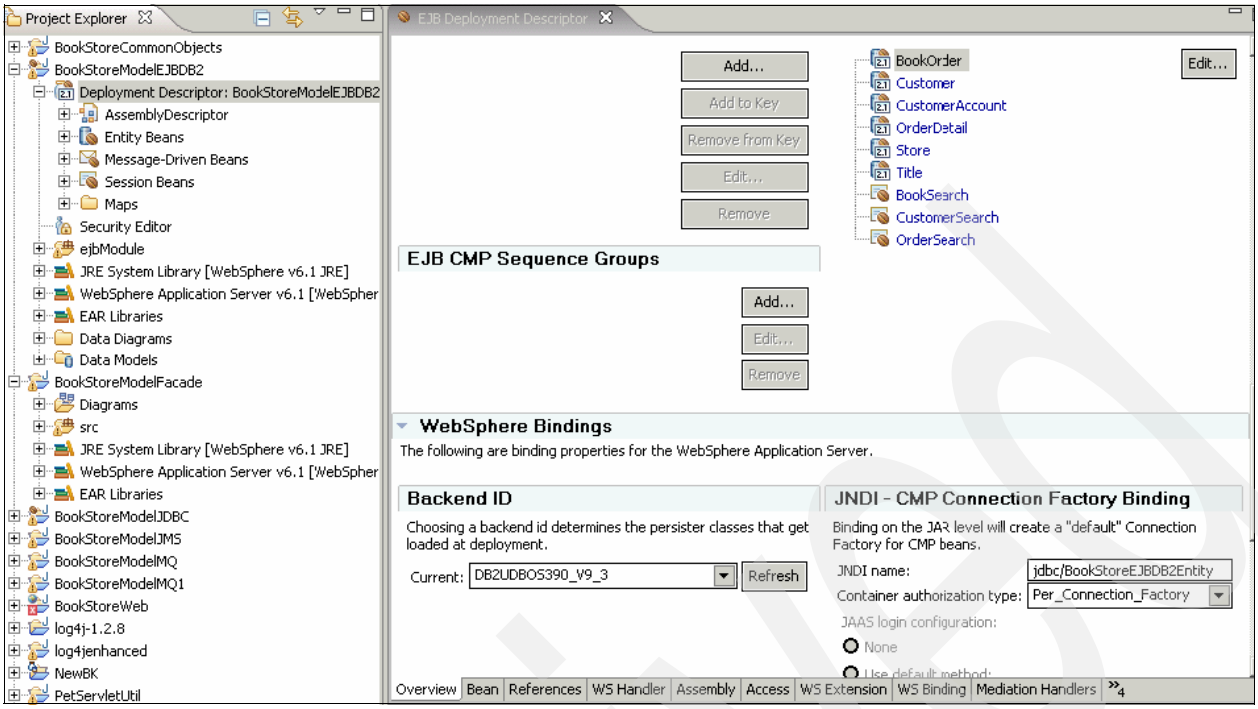


Figure E-5 Make Deployment Descriptor point to the new backend ID

E.2 Application deployment

To install an ear file, follow these steps:

1. Go to the WebSphere admin console: **Applications** → **New Application** → **Enterprise Application** → **Specify ear file location** → **Fast Path**.

   Then go step-by-step through the panels of the installation.

2. In the Data Source configuration step, you can enter either a component or a container-managed auth alias, but only one will be used. Per the EJB resource authentication specification, an application can be either container-managed or component-managed.

3. The authentication method is specified in the application EJB deployment descriptor. From the WebSphere admin console, view the EJB deployment descriptor:

   Enterprise Applications → **zipSeriesStore** → **Manage Modules** → **BookStoreModelJDBC.jar** → **Deployment Descriptor**

4. You will see <res-auth>Container</res-auth> or <res-auth>Application</res-auth>. See Figure E-6 on page 476.

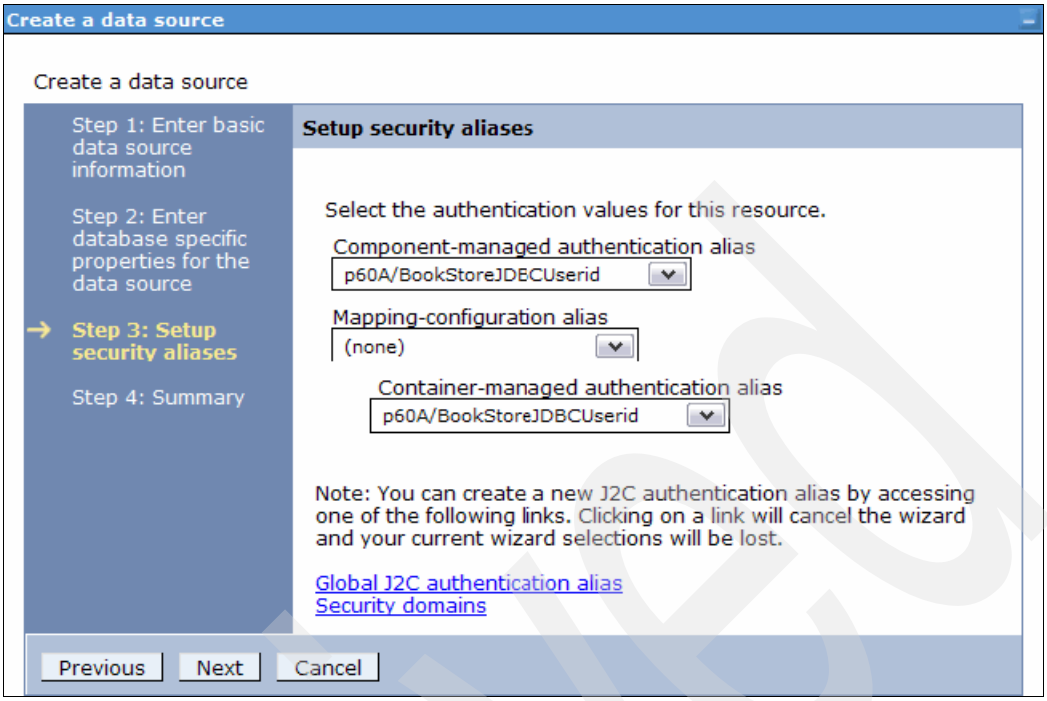


Figure E-6 Specify the authentication method during the ear file deployment process

After the ear file is installed successfully, go to the WebSphere admin console to finish configuring the modules and resources:

- **Enterprise Applications → zipSeriesStore → Manage Modules** (see Figure E-7)
- **Enterprise Applications → zipSeriesStore → Resource references**

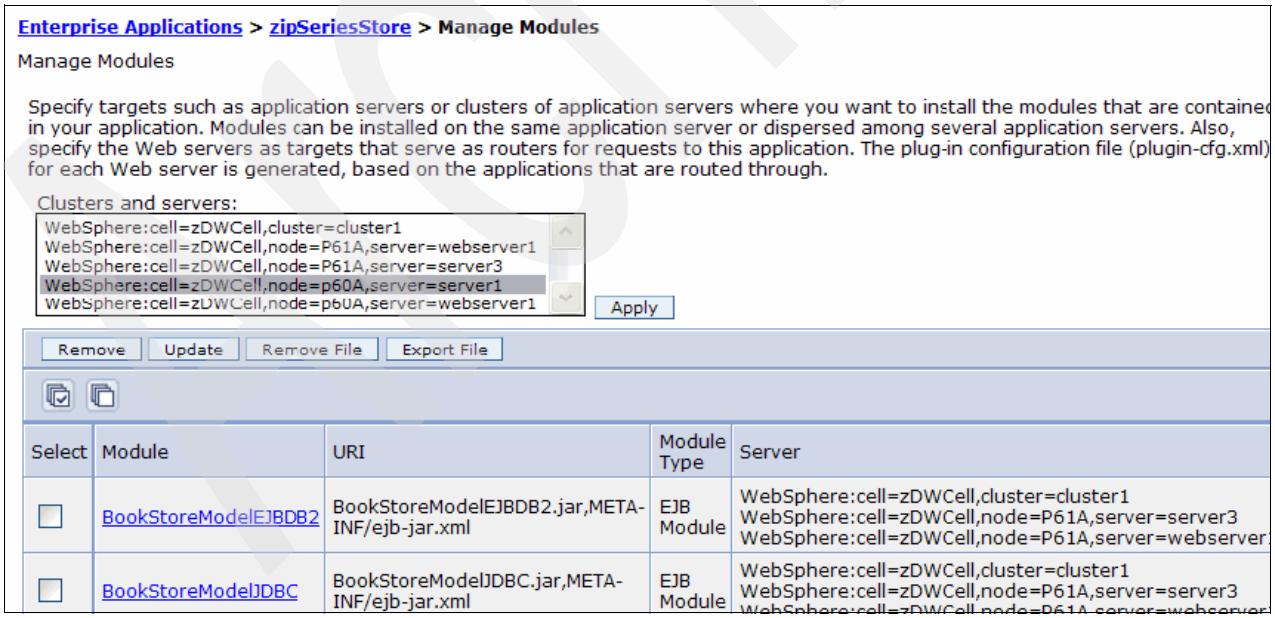


Figure E-7 Specify target server for modules to be installed

If container-managed auth alias is used, you will see the Application Resource references panel on the Admin console showing Resource authorization: Container after the application is deployed, as shown in Figure E-8 on page 477.

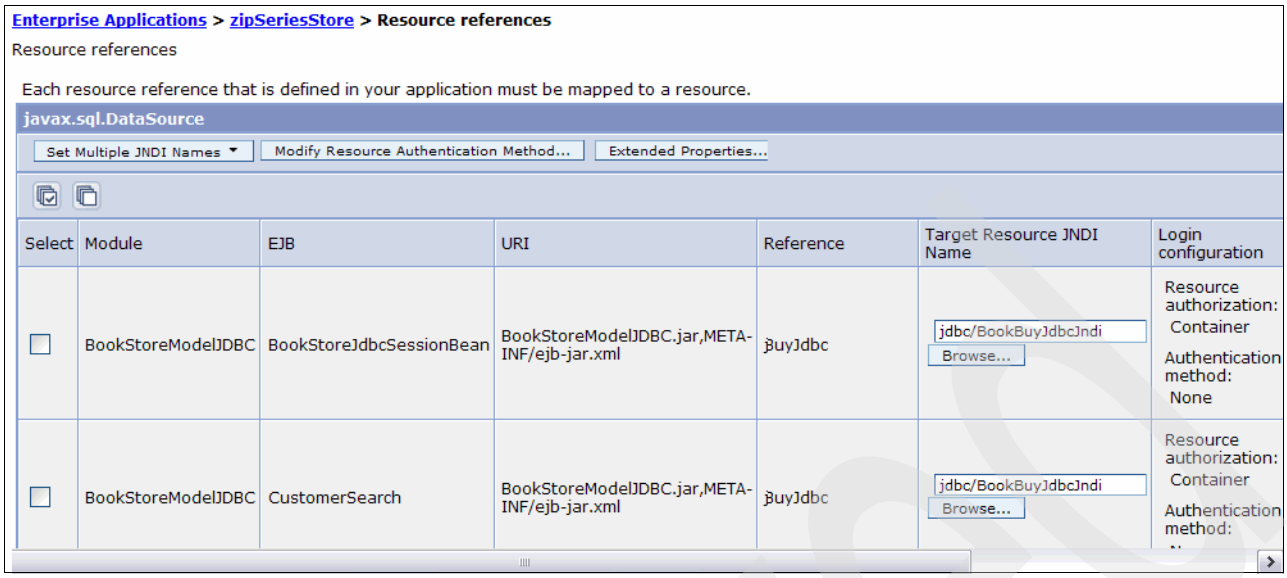


Figure E-8 Display authentication information

E.3 JDBC configuration

The persistent EJB preserved the data persistence in the relational database. It is accessed through the database by the connection that the data source provided. A data source is an object that provides the connectivity to the database through the JDBC interface, which is defined by the JDBC provider.

After the application is deployed onto the WebSphere server, we need to configure the data sources that are associated with the JDBC provider you selected; these are the steps:

- Define JDBC Connection Types.
- Define DB2_jcc property file (Type 2 only).
- Define WebSphere environment variable for JDBC.
- Define J2C authentication user ID.
- Configure JDBC data source.

E.3.1 JDBC connection types

There are four types of JDBC connections:

- Type 1 - JDBC-ODBC Bridge driver

  Convert a JDBC into an Open Database Connection (ODBC).
- Type 2 - Partial Java/Native-API driver

  Convert JDBC calls into database-specific native API calls.
- Type 3 - Pure Java/Net-protocol driver

  Pass a client request through the network to a middleware server, then communicate with the database through JDBC type 1, 2, or 4.

Archive

- Type 4 - Pure Java/Native-protocol driver

  Use Java network libraries to communicate with the database. Convert JDBC calls into database-specific native API calls. A client needs different drivers for different databases.

In this project, the JDBC provider was the IBM Universal JDBC Driver for JDBC and SQLJ. The IBM DB2 Driver for JDBC and SQLJ provides JDBC 2.0 and most JDBC 3.0 functionality For this project, we were limited to type 2 and type 4 JDBC drivers. Here are the definitions of the two driver types from the DB2 manuals:

Type 2 Drivers that are written partly in the Java programming language and partly in native code. The drivers use a native client library specific to the data source to which they connect. Because of the native code, their portability is limited.

Type 4 Drivers that are pure Java and implement the network protocol for a specific data source. The client connects directly to the data source.

The choice of using JDBC connection type 2 for the OLTP transactions was based on the consideration that both WebSphere and OLTP DB were on the same z/OS LPAR. Because of the specially designed code used to interact with DB2 with type 2, type 2 should perform better, whereas the type 4 driver relies on network protocols that flow through the TCP/IP stack. In general, the generic network protocols and the TCP/IP overhead associated with type 4 causes the type 4 driver to not perform as well. Based on this we chose to use the type 2 driver.

Figure E-9 illustrates the type 2 and type 4 connections that are the two JDBC connection types on z/OS.

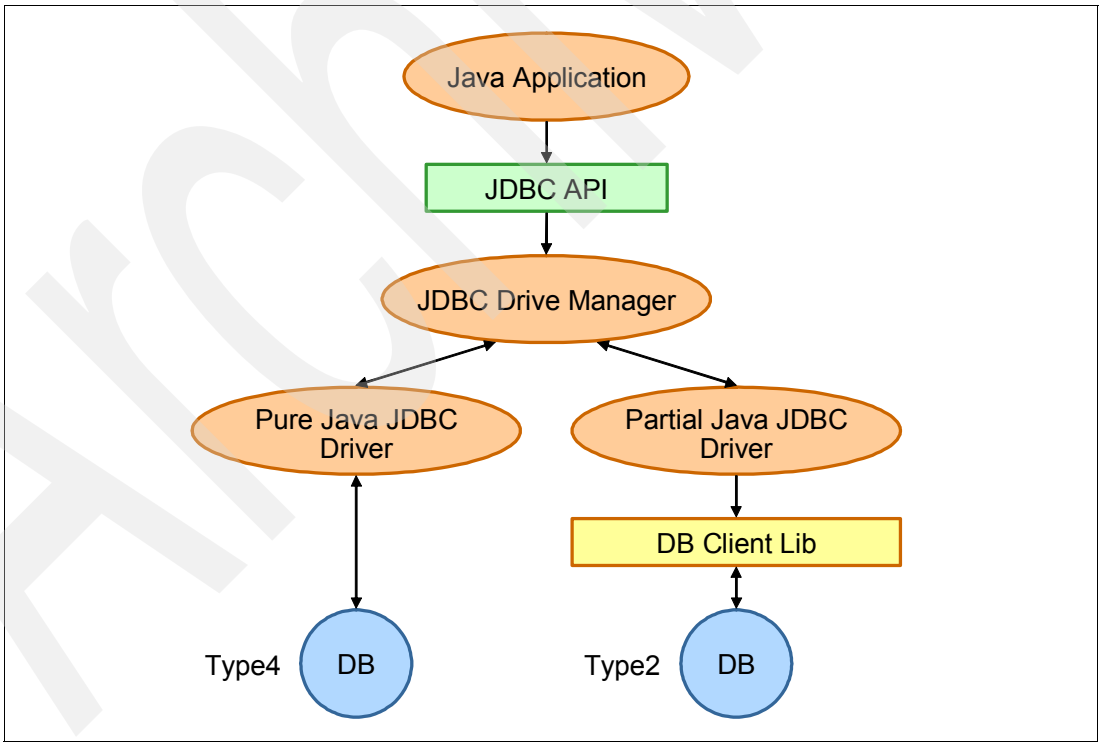


Figure E-9 JDBC connection types 2 and 4

E.3.2 DB2_jcc property file

During the WebSphere JDBC type 2 configuration, you need to specify the jcc property file in the WebSphere JVM custom properties panel. See Figure E-10.

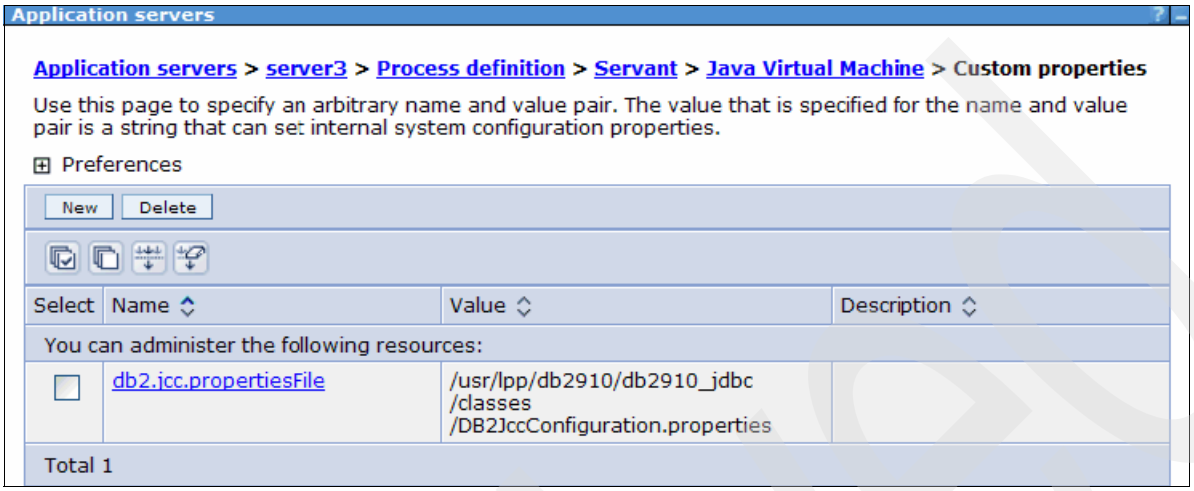


Figure E-10 DB2 jcc property file location

Example E-1 shows the content of our property file. DDWG is the data sharing DB2 group ID. Remove the line `db2.jcc.traceFile=/tmp/jdbctrace` to turn off the trace.

Example E-1 Content of jcc properties file

```
db2.jcc.ssid=DDWG
db2.jcc.traceFile=/tmp/jdbctrace
```

WebSphere environment variable for JDBC

Next, you need to define two WebSphere environment variables for JDBC:

```
DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH
DB2UNIVERSAL_JDBC_DRIVER_PATH
```

See 5.5, “WebSphere environment variables” on page 79 for detailed setup information.

Define J2C authentication user ID

Configure the J2C authentication user ID and password and assign an alias. This alias is then referred to in the JDBC data resource authentication panel; see Figure E-11 on page 480.

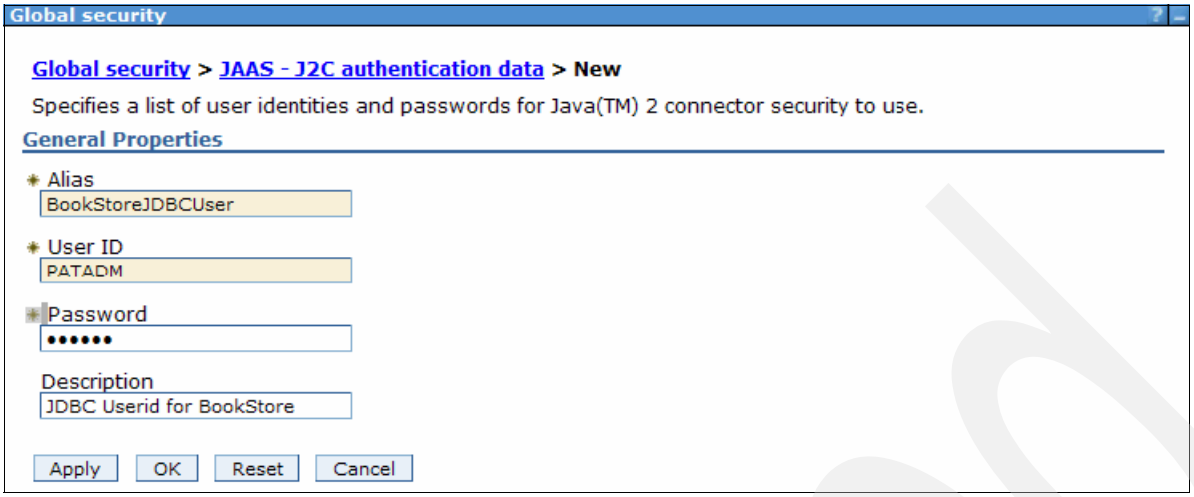


Figure E-11 Define J2C authentication user ID from the WebSphere administrator panel

Configure JDBC data source

Here are the steps to configure the data sources that are associated with the JDBC provider that you selected:

- Choose the data source definition visible scope, that is, node or server.
- Choose the JDBC provider from the drop-down list.
- Specify the data source definition name and JNDI name.
- Specify the resource authentication alias.
- Specify the data source properties. The database name is the LOCATION and the Port name is the TCPPORT that were shown on the MVS SYSLOG when the DB2 subsystem came up. See Figure E-12 on page 481 for an example.

Archived

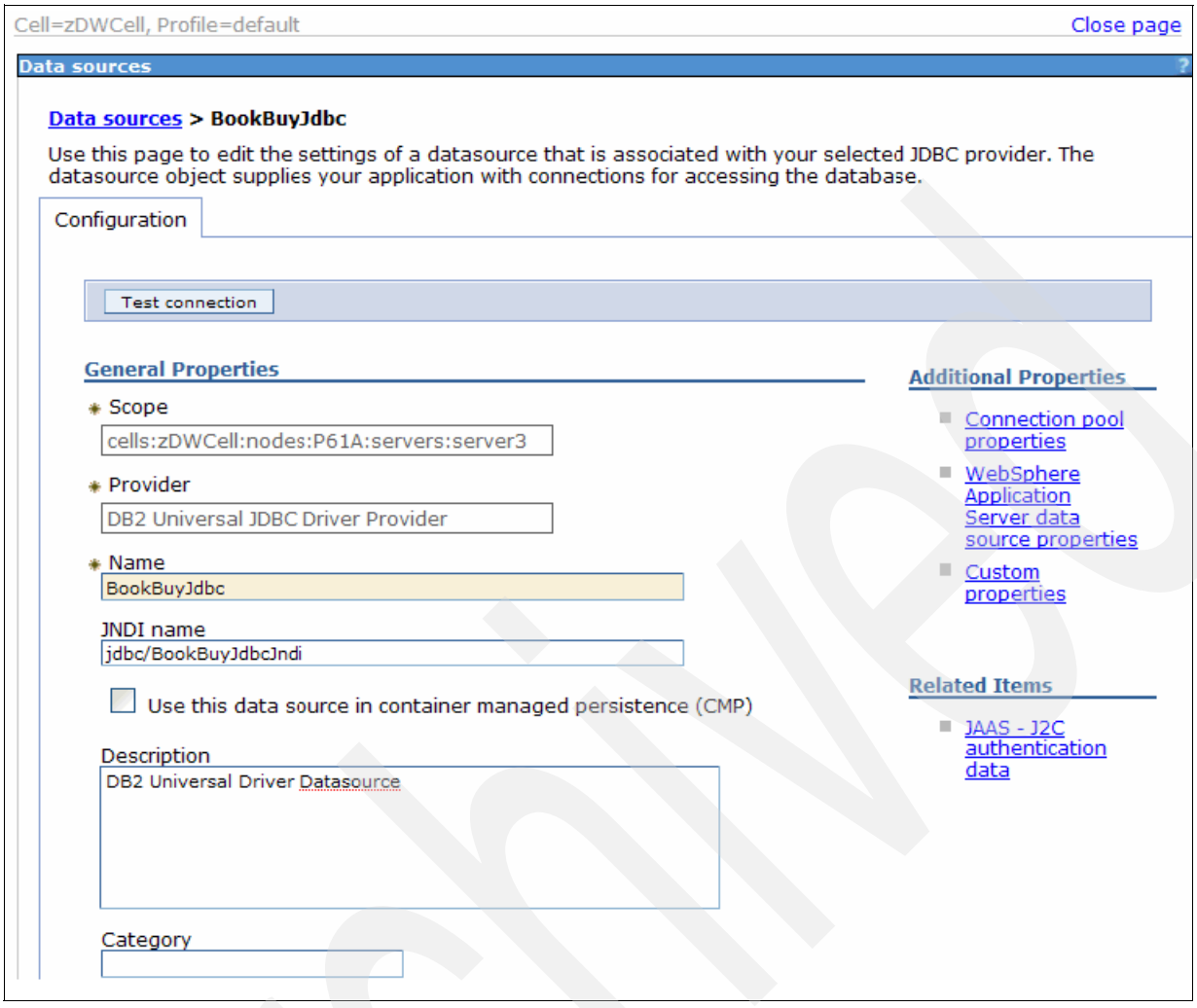


Figure E-12 Define new data source - Part 1

Figure E-13 on page 482 shows the continuation of this panel.

Archive

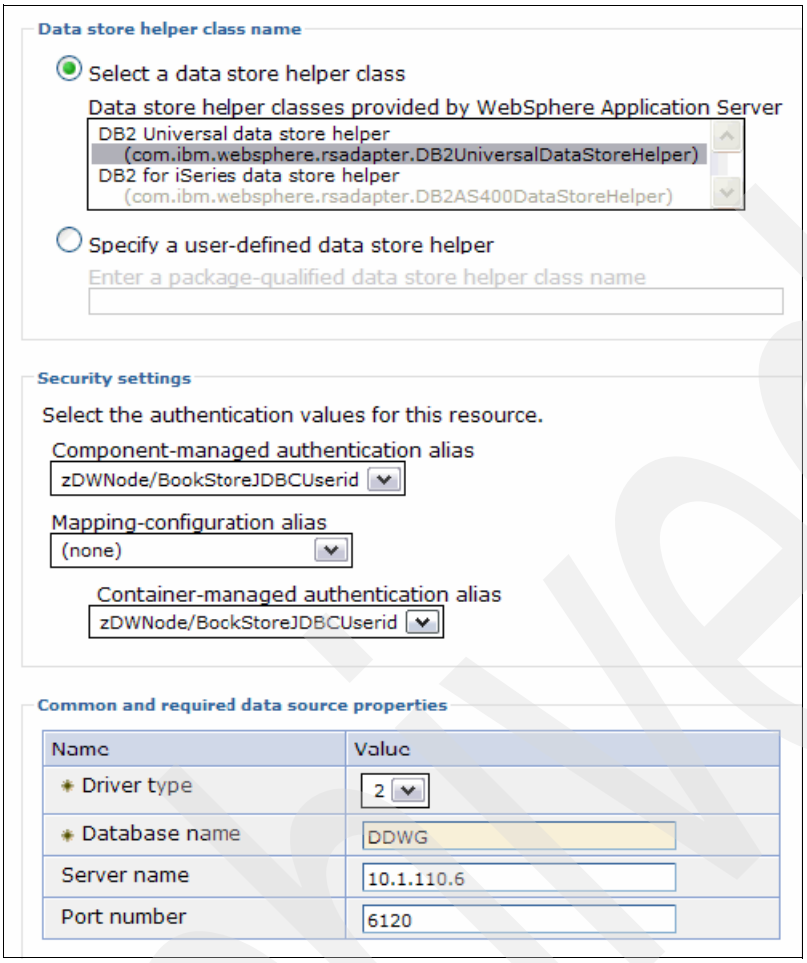


Figure E-13 Define new data source - Part 2

There are two ways to specify the JDBC DB2 SSID:

1. Set custom properties `db2.jcc.propertiesFile` on the Servant Java Virtual Machine of the application server or servers.

   a. In the WebSphere Application Server admin console, click **Application servers** → **<server_name** → **Process Definition** → **Servant** → **Java Virtual Machine** → **Custom Properties**.

   b. Click **New** to create the `db2.jcc.propertiesFile` property and set the value to the location of your DB2 JCC Properties file, for example `/usr/lpp/db2910/db2.jcc.properties`. This file must contain the name of the local DB2 subsystem or a group attachment name. For example `db2.jcc.ssid=DDWG. The` Group name makes it more versatile, set at the JVM level.

2. Set property variable SSID

   With JCC V3.6+, JCC V3.50+, and JCC v4.1+, for type 2 you can set the SSID at the data source level.

   To use the local SSID keyword, navigate to the desired data source:

   Select **[-] Resources** → **[-] JDBC** → o **Data sources**, then select the desired data source.

   Under **Additional Properties** → **Custom Properties**, add the new SSID keyword with the subsystem ID or Data Sharing Group name; See Figure E-14 on page 483.

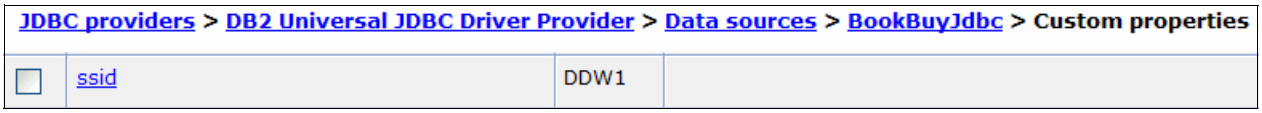


Figure E-14 Specify SSID in the data sharing environment CI

WebSphere Application Server setup topology

There are multiple topologies for the co-location project. The first was a *single DB2 instance* that contained the transactional data and data warehouse data. The second environment was a *data sharing group* with multiple DB2 instances able to access the transactional data and data warehouse data.

All of the WebSphere topologies were Network Deployment topologies, with a deployment manager, node agent and one or more WebSphere servers. The decision to run this way was made because of the flexibility it allowed when switching between topologies. Nothing had to be reconfigured, reinstalled or even changed when switching between environments. If runs were needed to be made in the single DB2 instance environment, the system was IPLed and the deployment manager, node agent, and application server were started. If runs were needed to be made in the data sharing environment, the systems were IPLed and the deployment manager, node agent, and (in this case) two application servers were started.

In the *single DB2 instance* environment, there was one WebSphere Application Server that was used for the two BookStore transactions. This WebSphere Application Server had two servant instances configured. The decision to run with two servants was made for a couple of reasons:

- To double the number of application threads.
- There were two BookStore transactions, each classified to its own WLM service class. WLM will try to queue work for a service class to its own server if available. With the two transactions, the incoming work for each transaction would be queued to its own server.

In the *data sharing environment*, the WebSphere Deployment Manager managed the cell across two System z LPARs, with each LPAR containing a node agent and an application server. Each application server also had two servant instances running and for the same reasons listed for the single DB2 instance environment.

During the OLTP testing, the goal was to achieve 90% processor utilization and maintain that utilization for the duration of the run. If running in the data sharing environment, each of the servers in the two LPARs should produce equal transaction rates and also achieve similar processor utilizations. In either the single DB2 instance environment or the data sharing environment, the decision was made to have the systems run at roughly the same transaction rates.

As far as the WebSphere configuration, except for configuring the application servers with two servants each, a standard WebSphere install, with no modifications to thread counts, heap storage, or special setting for WebSphere for z/OS was used.

There were several ways to configure WebSphere servers to service the work from the application clients. Since these application clients used the HTTP protocol, these topologies needed a web server to handle the HTTP protocol. Three web server topologies are described for reference. Since the main topic discussed in this book is WLM resource management rather than OLTP performance tuning, for ease of the configuration effort the decision was made to send requests directly to WebSphere Application Servers running with

the HTTP transport. The configuration is described in 8.1, “Hardware and software” on page 208.

Distributing incoming work to application servers

What really drove the WebSphere configuration for this project? In the data sharing environment there were two DB2 instances in a data sharing group that handled the incoming HTTP clients and the requests for transactional data. Each of the two DB2 instances resided on its own System z LPAR configured into a sysplex. The one requirement we had for the WebSphere configuration was that it had to be able to drive those DB2 instances to a high processor utilization and with equal client distribution.

The decision was made to go with the simple approach for the web server configuration. Rather than drive the two DB2 instances from a single WebSphere application server, two WebSphere Application Servers were used, each connected to a DB2 instance. Since the DB2 instances were on separate systems, driving from a single WebSphere application server would have meant one or possibly both DB2 instances would have been connected to WebSphere with a type 4 JDBC driver. In most cases, type 2 JDBC will perform better than type 4 JDBC when the WebSphere application server and DB2 instance are in the same z/OS LPAR, Since this was our configuration and many customers run with type 2, we chose to run with type 2 JDBC.

If one WebSphere Application Server was connected with type 2 and the other connected with type 4, the possibility of not driving equal transaction rates existed. Getting similar behavior out of both systems was important, and if the systems are configured differently, getting to the root cause of performance differences could be difficult. Taking this into account, each System z LPAR would have its own WebSphere application server connected to the DB2 instance on that same system with a type 2 JDBC driver.

How to handle the incoming HTTP work and distribute that work to the WebSphere Application Servers was the next issue that needed to be resolved. Three possible alternatives were explored. These were:

- Use the HTTP transport built into the WebSphere Application Server. Each application server would be a standalone entity that would handle the incoming HTTP protocol and execute the request.
- Use a standalone web server with a WebSphere plug-in to route the requests to the WebSphere Application Servers.
- Use the WebSphere Proxy server and route the requests to the application servers.

Both the standalone web server and the WebSphere Proxy server seemed more applicable to IBM's clients. Each has affinity capability, which is the capability to route incoming work based on affinity to an application server caused by previous associated work that was routed to that application server. Routing of new work to the application servers for the standalone web server was a round-robin approach that could be weighted. The routing of new work for the Proxy Server is based on metrics from WLM, which will give better distribution of work based on current load on the application servers.

The decision was to go with the HTTP transport that is built into the WebSphere application server. Clients would be set up to send HTTP requests directly to each of the WebSphere application servers. Each of the web servers would have an equal distribution of the two BookStore transactions routed to it from these clients. This simplified the overall approach and would be more than adequate for what was needed.

Send requests to the HTTP Server

Figure E-15 depicts one of the possible topologies for the co-location project in a data-sharing environment.

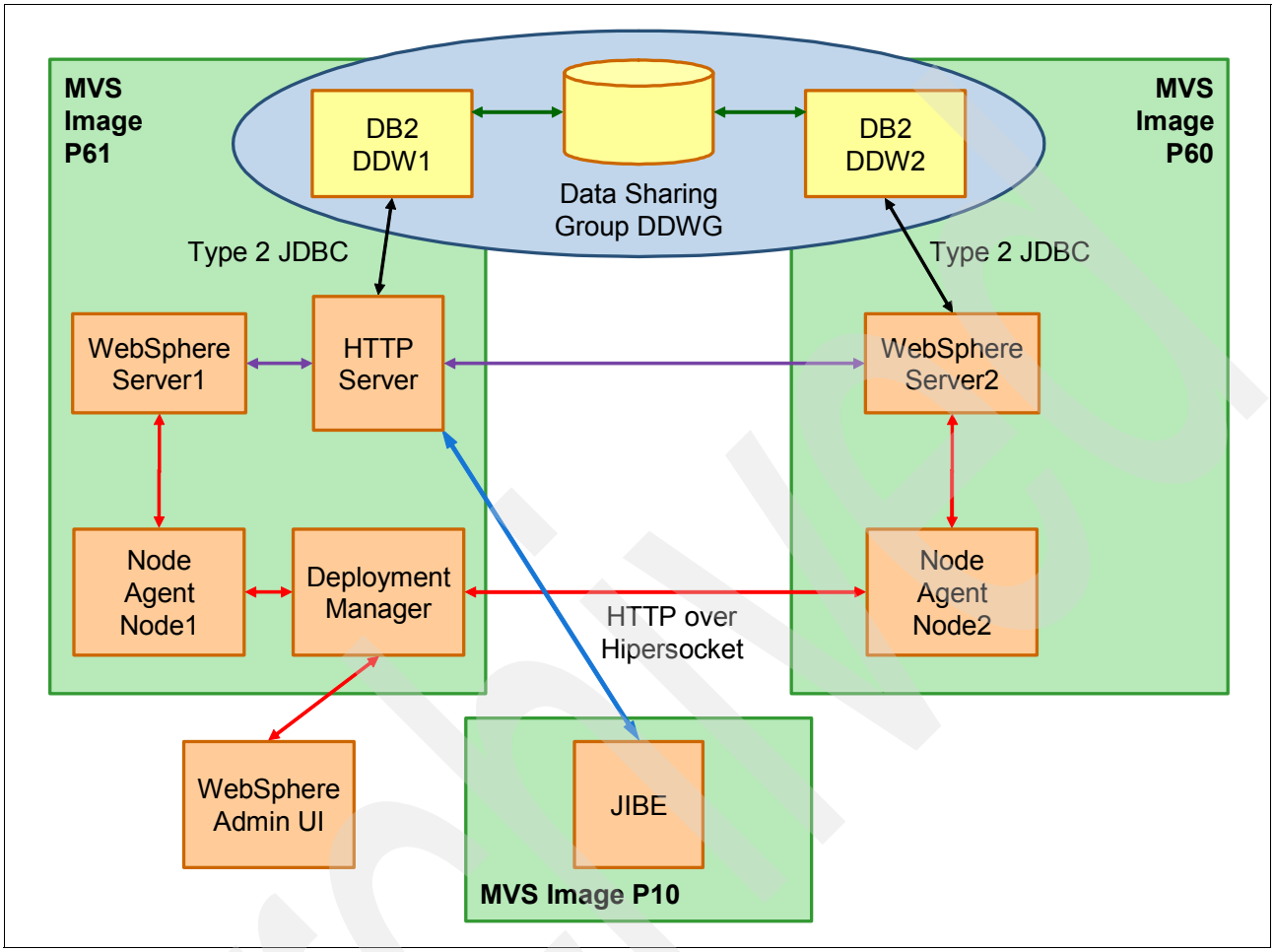


Figure E-15 Transactional workload topology using the standalone HTTP server

In this case, client scripts, running on system P10, route HTTP requests to a standalone web server. The web server plug-in directs those requests to one of two WebSphere Application Servers on systems P60 and P61. There are two DB2 instances, one on system P60, the other on system P61. These two database instances are in a data-sharing group, which also included DB2 instances that were used to access the data warehouse databases. The DB2 instances for the data warehouse data are not shown in Figure E-15.

Send requests to proxy server

Figure E-16 on page 486 shows the same environment with a proxy server receiving the HTTP requests from the clients. In this case, client scripts running on system P10 are sending HTTP requests to a proxy server on system P61.

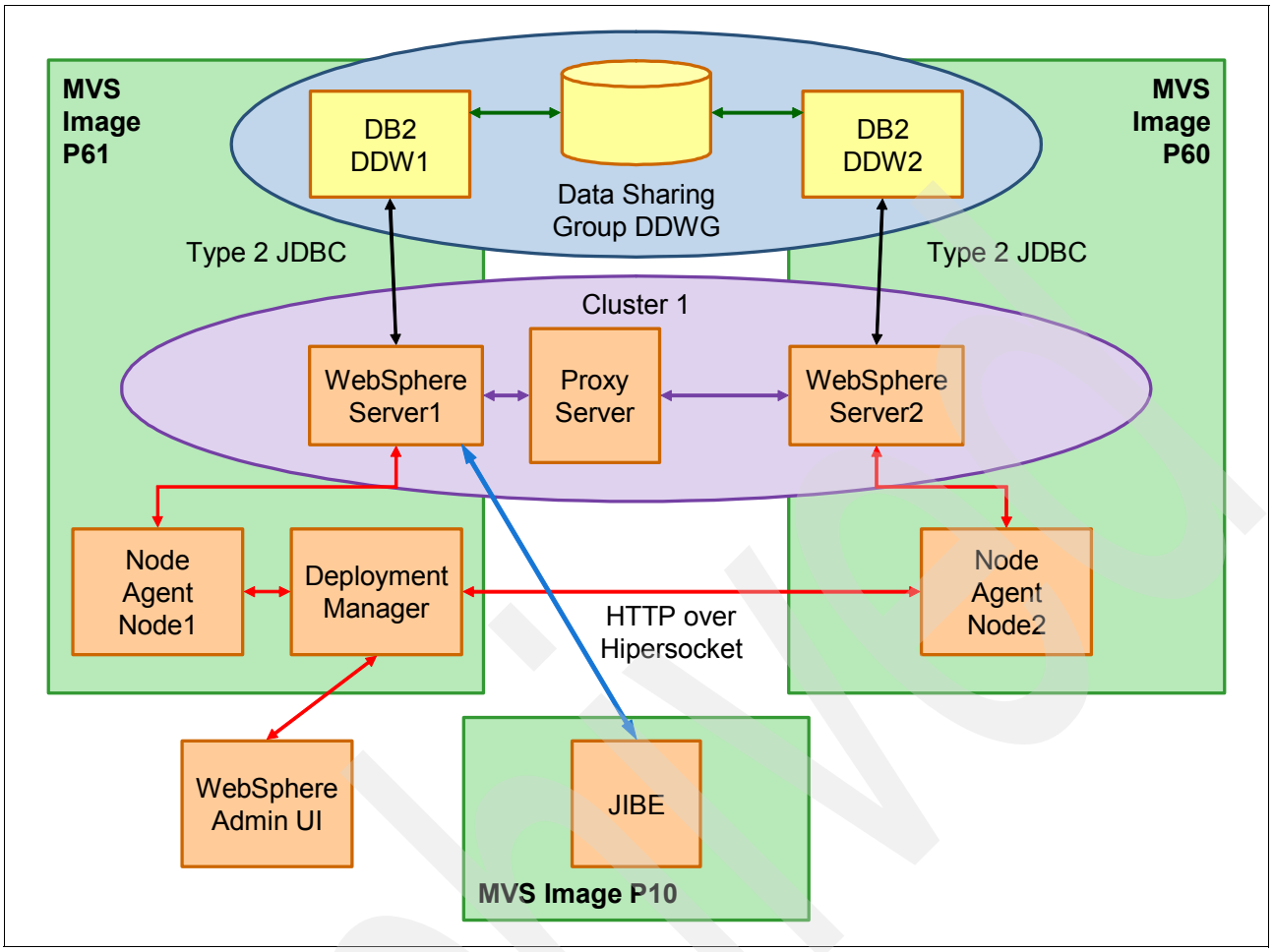


Figure E-16 Transactional workload topology using the proxy server

The proxy server uses information about activity in the two WebSphere Application Servers to route the HTTP requests to an application server. There is a DB2 instance running on systems P60 and P61, that are in a data-sharing group. The application servers have a JDBC type 2 connection to a DB2 instance on that system. The DB2 data-sharing group also includes DB2 instances that were used exclusively for access to the data warehouse. The DB2 instances used for the data warehouse are not shown in these diagrams. The WebSphere Application Servers are in a cluster sharing the BookStore application. This cluster is known by the proxy server.

Send requests to a WebSphere Application Server

Figure E-17 on page 487 shows clients running on system P10, sending HTTP requests to WebSphere Application Servers running on systems P60 and P61. There are two sets of clients, one set sending HTTP requests to the application server on system P60, and the other set sending HTTP requests to the application servers on system P61. The application servers are using the HTTP transport that is built into the WebSphere Application Server. There are two DB2 instances running in a data-sharing group. One of these instances is running on system P61, and the other is running on system P60. The data-sharing group also includes DB2 instances that access the data warehouse data. The DB2 instances for the data warehouse data are not shown in this diagram.

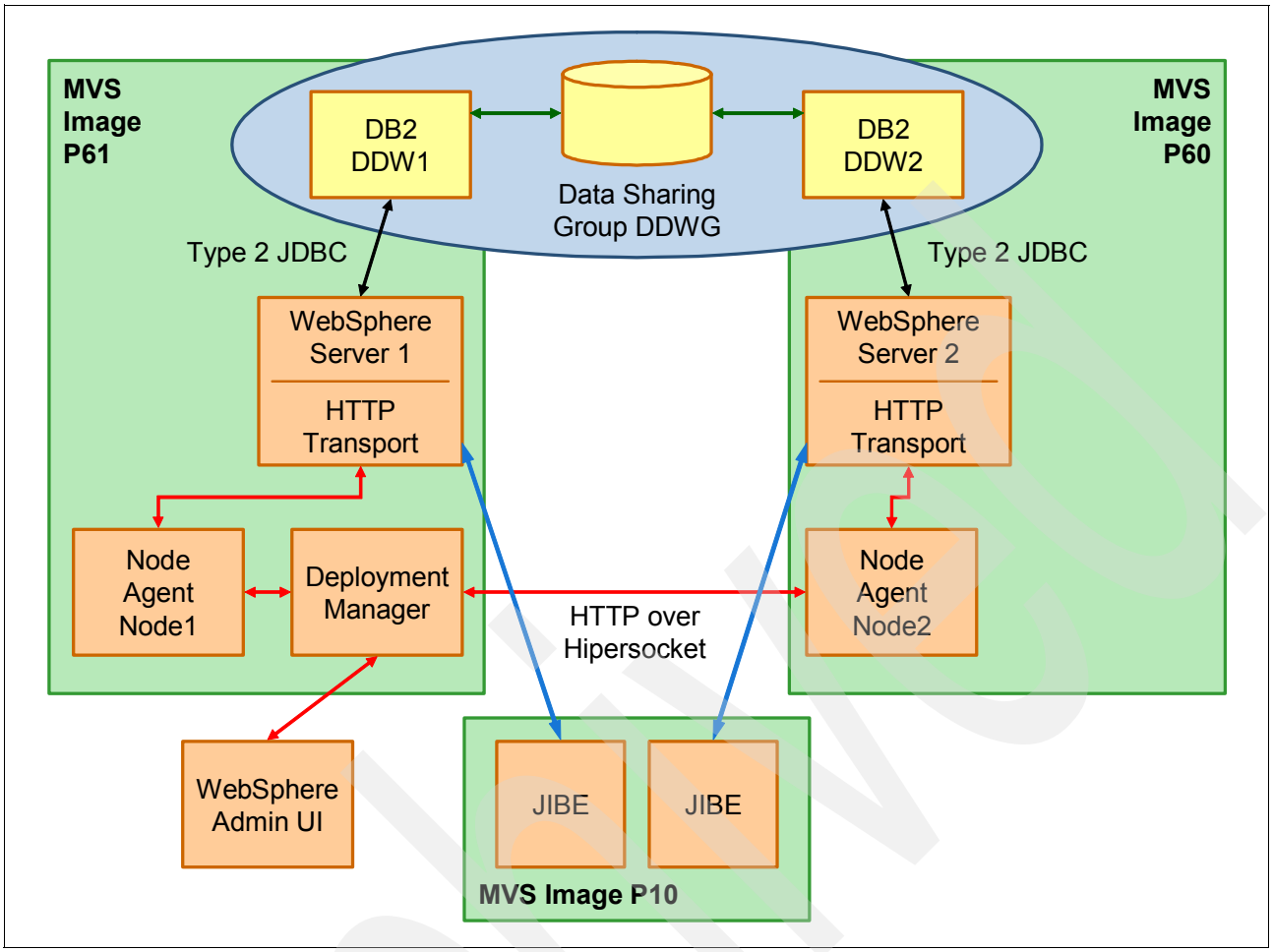


Figure E-17 Transactional workload topology using the HTTP transport

WebSphere clusters

When working through the process of figuring out what the configuration should be for a data-sharing environment, using the WebSphere proxy server was considered. There were two systems in the data-sharing environment allocated to running the transactional workload and a WebSphere Application Server on each system. A natural progression of setting up the proxy server to handle the inbound HTTP and having two WebSphere Application Servers was to make a cluster with those two servers. See Figures E-18 and E-19 on page 488.

There are a couple of advantages in using a cluster:

- The application would be installed on the cluster and by adding the servers to the cluster, the application was then installed on them.
- Using the proxy server would give workload distribution to the application servers based on the proxy server and its knowledge about the current activity in the cluster and on those application servers.

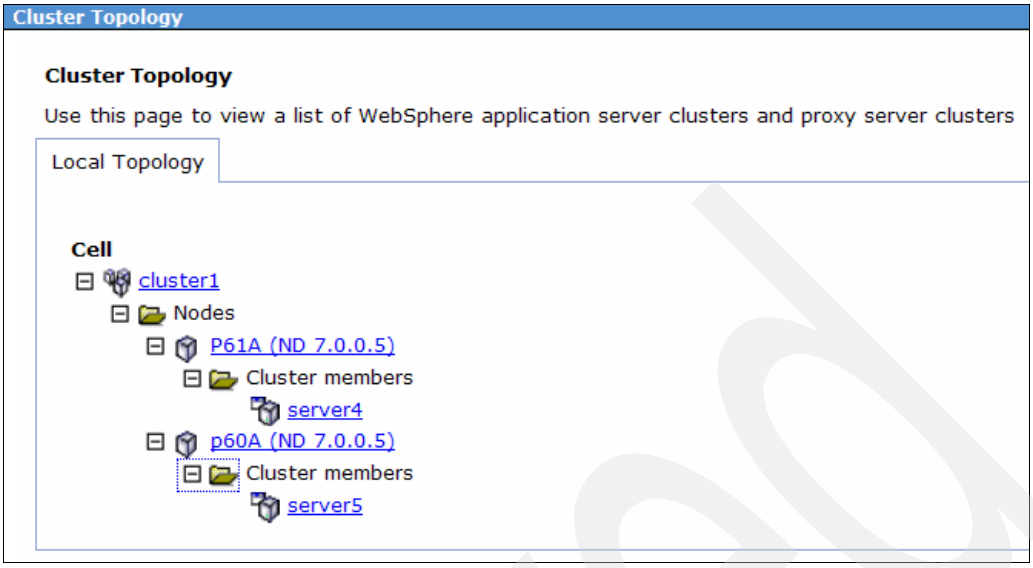


Figure E-18 Proxy server and server cluster topology

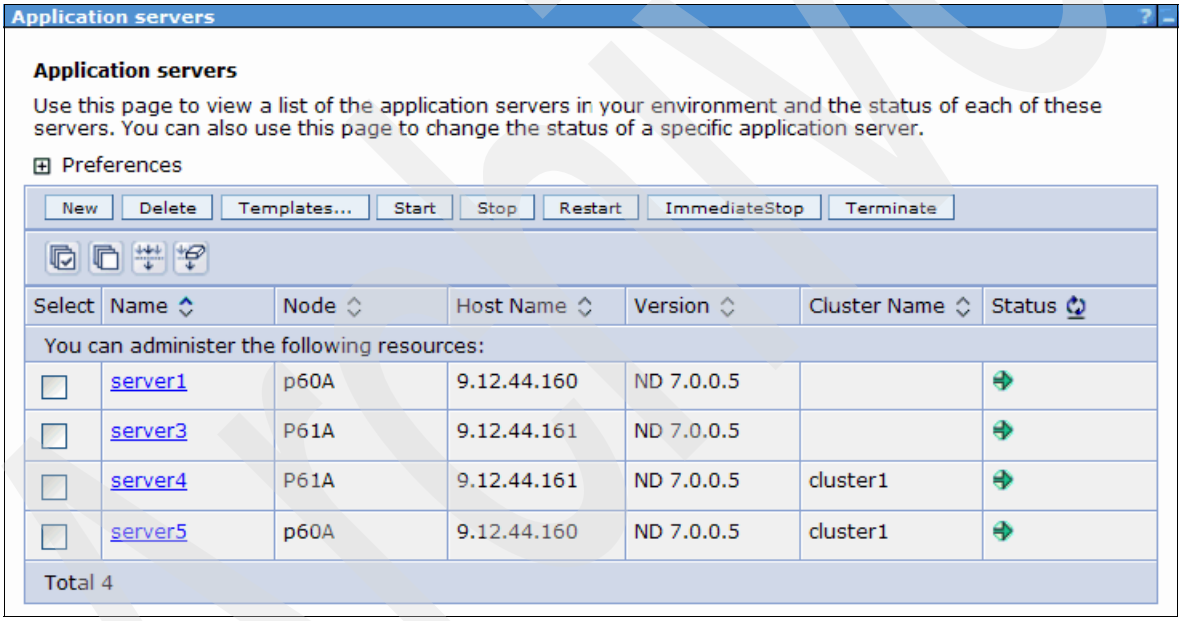


Figure E-19 WebSphere Application Servers at a glance

Issue when running two application servers

An interesting issue that we experienced when running in the data-sharing environment with two WebSphere Application Servers was that, when reviewing the RMF data for the runs, there was a big difference in the transaction times between the transactions running on the two servers. After some research and trying a number of things, the issue was discovered. It was the interaction between the Nagle algorithm on the client side and code meant to enhance performance on the server side. The Nagle algorithm's intent is to decrease network traffic by buffering small sends, specifically when there is a previous send that has not been acknowledged, the next send will be buffered until the previous send is acknowledged. This interacts poorly with the receiver side option that tries to buffer acknowledgements until another send is received or a timer expires and the receiver can then acknowledge the

previous send. Both have their purpose. The sender side buffering sends saves on network traffic. The server side buffering of acknowledgements saves on TCP/IP path length by sending multiple acknowledgements all at once if possible.

The difference in transaction times was caused by the client side buffering a second send request because it had not received acknowledgement from the last send and the server side not acknowledging the first send. This interaction caused an approximate 200 ms gap in the middle of some of the BookStore transactions and must have occurred after the transaction had started in WebSphere and before it had completed.

There are two ways to approach and modify the behavior of this problem:

- Disable the Nagle algorithm on the client side with the TCP_NODELAY setsockopt.
- Turn off using a timer on the server side and acknowledge every request. This can be done by specifying NODELAYACKS on the port definition in the TCP/IP profile.

WebSphere tooling does generate the NODELAYACKS keyword on the TCP/IP profile port definitions for secure ports and did not seem to generate NODELAYACKS for the non-secure ports.

This issue is apparently well known with the SSL handshake and is also common with short HTTP responses.

Setting TCP_NODELAY with the setsockopt on the client side was one of the recommended solutions to this problem. The project team was unsure of how to do this with JIBE and the HTTP scripts that were set up. To get around the problem, NODELAYACKS was specified on the TCP/IP profile port definitions.

Why did this problem arise? For some reason when the TCP/IP profile was set up, the port definition for one of the application servers had NODELAYACKS specified, yet the other did not, even though both were listening on non-secure ports.

Further investigation needs to happen with this issue, but the fix was to add NODELAYACKS on the port definition. See Example E-2. The transaction times were as expected, with this change, and essentially equal.

Example E-2 Port definitions in the TCP profile

```
9080 TCP BBOS001 NODELAYACKS ; HTTP port
9081 TCP BBOS003 NODELAYACKS ; HTTP port
```

16.5.1 WebSphere administrative command examples

The WebSphere administrative commands in Examples E-3 and E-4 show the power of the command language and reinforce the use of the capability. Rather than always administering the WebSphere environment from the user interface (UI), scripts can be set up that perform the same actions. This gives the flexibility to issue a set of commands, modify the scripts and reissue the commands with a new set of parameters and lessen the risk of errors. This also allows a script to be set up with commands that are frequently entered. Once the script has run successfully, rerunning it should be less prone to errors.

Example E-3 Sample command to federate a node into a WebSphere cell

```
> cd /p60/tmp
> touch p60portprops092209
```

```
> addNode.sh TEST1.POK.IBM.COM 18110 -conntype SOAP -nodeagentshortname SVTTEST1
-portprops /tmp/p60/portprops092209 -username WSADMIN -password wsadmin
-localusername WSADMIN -localpassword wsadmin
```

Example E-4 Sample command to configure resources using Python scripts

```
/P60/WebSphere/V7R0/AppServer/profiles/default/bin/wsadmin.sh -conntype SOAP -port
18310 -lang jython -profile 70_functions.py -f 70setupERWW.py -user WSADMIN
-password WSADMIN
```

Put the two .py files in a directory, then issue the above command from the directory.

Archived

F

WLM refresher

A key focus of this book is System z resource management, hence we decided to include a an introduction to z/OS Workload Manager (WLM), and some of the key WLM for DB2 concepts and considerations.

Archived

© Copyright IBM Corp. 2010. All rights reserved.

F.1 How WLM works

The idea behind WLM is to make a contract between the installation (performance administrator) and the operating system. The installation classifies the work running on the z/OS operating system in distinct service classes. The installation defines business importance and goals for the service classes. WLM uses these definitions to manage the work across all systems of a sysplex environment. WLM adjusts dispatch priorities and resource allocations to meet the goals of the service class definitions. It does this in order of the importance specified, highest first. Resources include processors, memory, and I/O processing.

Refer to the following documentation for further details:

- IBM WLM Internet site:
  http://www.ibm.com/servers/eserver/zseries/zos/wlm/
- IBM Redbooks publication *System Programmer's Guide to: Workload Manager*, SG24-6472
- WLM presentations
  http://www.ibm.com/servers/eserver/zseries/zos/wlm/presentations/
- WLM tools
  http://www.ibm.com/servers/eserver/zseries/zos/wlm/tools/

F.1.1 WLM components

All the business performance requirements of an installation are stored in a *service definition*. There is only one service definition for the entire sysplex. The definition is given a name and stored in the WLM couple data set accessible by all the z/OS images in the sysplex. In addition, there is a work data set used for backup and for policy changes.

The service definition contains the elements that WLM uses to manage the workloads:

- Service policies
- Workloads
- Service classes
- Report classes
- Performance goals
- Classification rules and classification groups
- Resource groups

Figure F-1 on page 493 shows the hierarchical relation between the WLM components that are described in the sections that follow the figure.

Archived

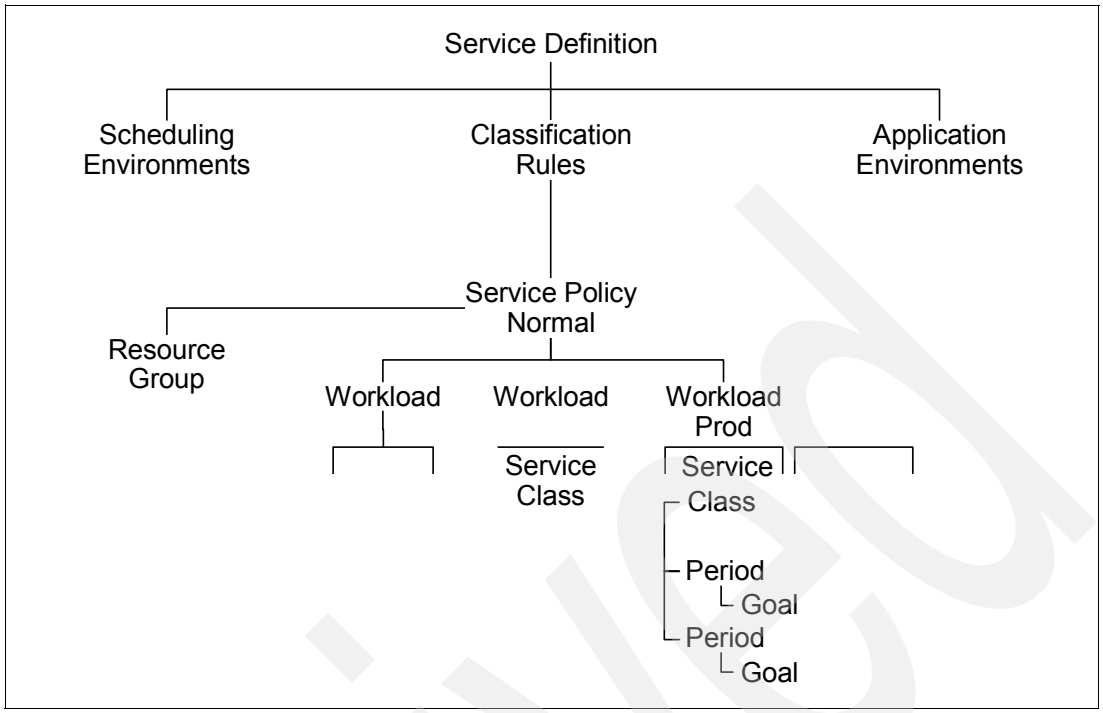


Figure F-1 WLM component relationship

Service policies

The service definition consists of one or more service policies. There is only one active service policy at a time in the sysplex. A service policy is a named collection of performance goals and processing capacity bounds. It is composed of workloads, which consist of service classes and resource groups. Two different service policies can share the same set of service classes, yet the performance goals can be different.

Workloads

Workloads are arbitrary names used to group various service classes together for reporting and accounting purposes. At least one workload is required. A workload is a named collection of service classes to be tracked and reported as a unit. It does not affect the management of work. You can arrange workloads by subsystem (such as CICS or IMS), by major workload (for example, production, batch, or office), or by line of business (ATM, inventory, or department). The Resource Measurement Facility (RMF) Workload Activity Report groups performance data by workload and by service class periods within workloads.

Service classes

A service class is a key construct for WLM. Each service class has at least one period, and each period has one goal. Address spaces and transactions are assigned to service classes using classification rules. A group of work, within a workload, with similar performance requirements can share the same service class. *Service class* describes a group of work within a workload with similar performance characteristics. A service class is associated with only one workload, and it can consist of one or more periods.

There are three system-provided service classes:

SYSTEM This is used as the default service class for certain system address spaces. It does not have a goal, and it is assigned the highest fixed dispatch priority of x'FF' and a fixed I/O priority of 255.

SYSSTC This is the default service class for system tasks and privileged address spaces. It does not have a goal, and it is assigned a fixed dispatch priority of x'FE' and IOP of 254.

S**YSOTHER** This functions as the default service class for non-STC address spaces when no classification rules exist for the subsystem type. It is assigned a discretionary goal.

Report classes

This is an aggregate set of work for reporting purposes. You can use report classes to analyze the performances of individual workloads running in the same or different service classes. Work is classified into report classes using the same classification rules that are used for classification into service classes. A good way to contrast report classes to service classes is that report classes are used for monitoring work, while service classes should primarily be used for managing work.

Goal types

Goals can be of four different types:

- Performance goals

  Each service class period has a goal and importance associated with it. WLM manages work requests to the specified goal. WLM does sampling to determine how well each service class is meeting its goal. There are four types of performance goals:

  - Average response time
  - Percentile response time
  - Execution velocity
  - Discretionary

- Response time goals

  Response-time goals take multiple forms. An average response-time goal is managed at the average response time. As one can imagine, this can be influenced by a few long-running transactions. Percentile response-time goals suggest that a percentage of the requests need to complete within a desired response time. Percentile response-time goals reduce the impact of outliers and are generally recommended for consistency. In either case, response-time reflects both queue time and execution time within z/OS. It is not end-to-end response time. As a general rule, work should have at least one completion every 10 seconds to provide adequate samples for WLM to manage response time goals effectively.

- Velocity goals

  Velocity signifies the percentage of time workload is ready and able to run, and is not delayed for lack of resources. For example, a defined velocity of 60 signifies that resources should be available for work to run 60% of the time. Velocity is a goal that defines the amount of acceptable delay. This percentage is calculated after unaccounted-for delays (that WLM is not aware of) are factored out of the equation. Velocity goals are good for always running or long-running work, such as subsystem address spaces like DB2 or transactions and queries that have widely fluctuating resource requirements and less-frequent completions. Velocity goals are more sensitive to system or application changes and require revisiting after these types of changes. Response time goals are generally less sensitive to these types of changes, requiring less revisiting.

- Discretionary goals

  A discretionary goal means "do the best that you can." Discretionary goals are for work that is processed using resources that other work does not require to meet the other work's goals. Discretionary work has no specific business goals attached to it.

Discretionary work can make it easier to manage an extremely busy system. A z/OS system can run at 100% processor busy without problems, as long as there is work that can wait during the peaks of more important work. Defining discretionary work allows WLM to know immediately which work must donate resources when an important workload spikes without having to wait for a WLM interval and going through the normal donor/receiver logic.

Business importance

When there is not sufficient capacity for all work in the system to meet its goals, business importance is used to determine which work should give up resources and which work should receive more. You assign an importance to a service class period, which indicates how important it is that the goal be met relative to other goals. Importance plays a role only when a service class period is not meeting its goal. There are five levels of importance: lowest (5), low, medium, high, and highest (1); se also Table F-1. In addition, discretionary goals are associated with an importance of 6 and are always considered as donors. Workload in a discretionary goal receives resources only when all the other service classes are achieving their goals.

Table F-1 Business importance levels

Importance	Description
1	Highest
2	High
3	Medium
4	Low
5	Lowest
6	Discretionary

Classification rules and classification groups

Classification rules and classification groups are used to assign the incoming work to a service class and, if needed, to a report class.

Classification rules

z/OS can manage many types of workloads, each one with different business importance and processing characteristics. To accomplish this, the installation must categorize the incoming work to the system using the classification rules. Classification rules are the filters that WLM uses to associate a transaction's external properties (also called work qualifiers, such as LU name or user ID) with a goal.

Archived

Figure F-2 represents how incoming workload is classified into a specific service class by the classification rules.

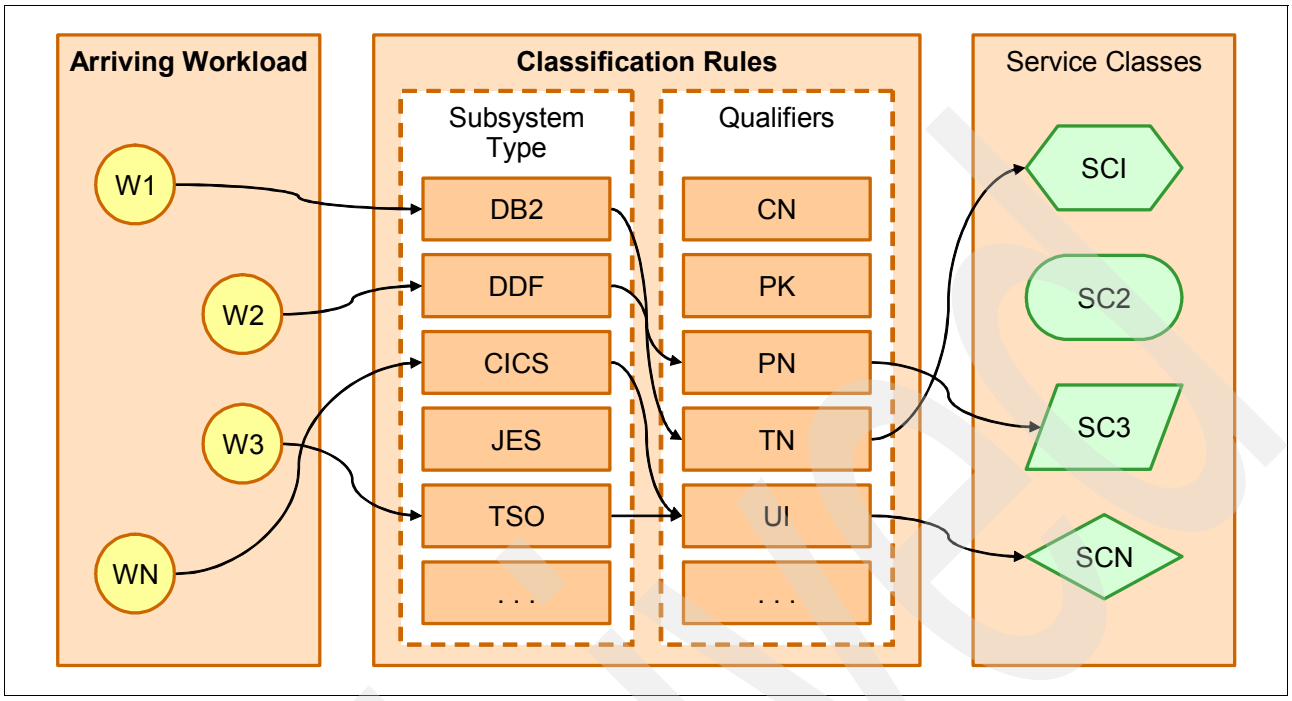


Figure F-2 WLM workload classification

For DB2-specific performance information for WLM, see the DB2 Administration Guide for your version of DB2.

Table F-2 shows an extract of the IBM-defined subsystem and Table F-3 on page 497 shows an extract of the work qualifier that can be used in the WLM classification rules. Not all work qualifiers are valid for every subsystem type. They are subsystem dependent. For information about which qualifiers are valid for which subsystems, and a full list of qualifiers, see *z/OS V1R10.0 MVS Planning: Workload Management,* SA22-7602.

Table F-2 Extract of IBM-defined subsystem types

Subsystem type	Work description
CICS	The work requests include all transactions processed by CICS Version 4 and higher.
DB2	The work requests include only the queries that DB2 has created by splitting a single, larger query and distributed to remote systems in a Sysplex. The local piece of a split query, and any other DB2 work, is classified according to the subsystem type of the originator of the request (for example, DDF, TSO, or JES).
DDF	The work requests include all DB2 distributed data facility work requests.
JES	The work requests include all jobs that JES2 or JES3 initiates.
STC	The work requests include all work initiated by the START and MOUNT commands. STC also includes system component address spaces such as the TRACE and PC/AUTH address spaces.
TSO	The work requests include all transactions processed by CICS Version 4 and higher.

Table F-3 Extract of work qualifiers and their abbreviations

Qualifier	Details
AI	Accounting information
CI	Correlation information
CN	Collection name
PK	Package name
PN	Plan name
PR	Procedure name
SI	Subsystem instance
SY	System name
TN	Transaction name/job name
UI	User ID

Using classification groups

If you have a long list of work that you want to use in a classification rule, you can create a group. You can create groups for the following qualifier types:

- Connection Type
- LU Name
- Net ID
- Package Name
- Plan Name
- Perform
- Subsystem Instance
- System Name
- Transaction Class
- Transaction Name
- User ID Groups

You can use classification groups for easy classification and their posterior maintenance. It could be a good idea to create a group that contains the critical user IDs, for instance.

Resource groups

Resource groups are used to assign a minimum and a maximum amount of processor service for one or more service classes.

F.1.2 The performance index

WLM maintains a performance index (PI) for each service class period to measuring how the actual performance varies from the goal.

Important: The following list explains the meaning of the PI values:

- PI = 1: SC period is exactly meeting its goal.
- PI > 1: SC period is missing its goal.
- PI < 1: SC period is beating its goal.

Archived

Performance index calculation

The way the PI is calculated differs according to the type of goal. The following paragraphs explain how the PI is calculated.

- Average response type goal PI calculation

  The average response time value is calculated by dividing the sum of response times by the number of ended transactions. This gives the average response time field, which corresponds to the achieved performance of the service class. This field is divided by the goal response time to give the performance index of this service class period, as shown in Example F-1.

  Example: F-1 PI calculation for RT goals

```
PI = Avg RT / Goal Avg RT
```

- Velocity goal type PI calculation

  The execution velocity percentage is calculated by dividing the number of the using sample by the sum of the using and delay samples, as shown in Example F-2.

  Example: F-2 PI calculation for velocity goals

```
PI = Goal Execution Velocity / Actual Execution Velocity
```

- Average response time with percentile goal type PI calculation

  The calculation of the achieved response time is more complex for this type of goal. You can use RMF reports for a quick view of the PI for this type of goal. WLM keeps response time distribution data in buckets for service class periods that have a response time goal specified. These buckets are counters that keep track of the number of transactions ended within a certain response time range. The response times are stored in a structure that contains 14 buckets. These buckets exist per service class period. Example F-3 shows how this PI is calculated.

  Example: F-3 PI calculation for RT goals with percentile

```
PI = Percentile Actual / Percentile Goal
```

Figure F-3 on page 499 shows an example of RMF response time distribution trend chart. This chart provides information about the percentage of transactions ended within a time bucket and is useful for analyzing response time goal achievement.

Archived

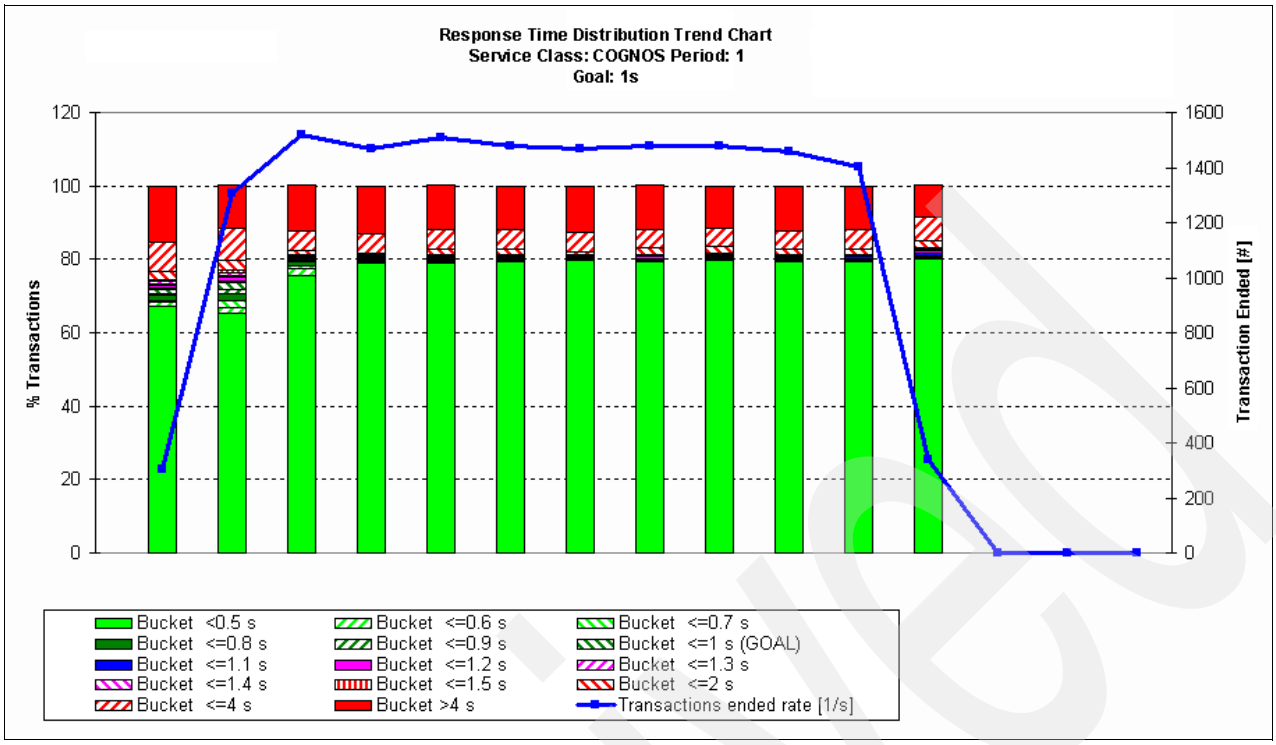


Figure F-3 RMF response time distribution trend chart

Performance index of discretionary goal type

A discretionary service class period is defined to have a PI equal to 0.81. A discretionary service class period is also called a universal donor, because its PI indicates that it is always beating its goal.

Important: A discretionary goal is a universal donor with a fixed PI=0.81. It receives resources only when all other service classes are achieving their goals.

F.2 Considerations for WLM and DB2

Here we discuss how to use WLM to set goals for the DB2 workload.

DB2 address spaces velocity goals

Use the following service classes for non-DBMS address spaces:

- SYSSTC service class for:
  - VTAM® and TCP/IP address spaces
  - IRLM address space (IRLMPROC)
- IRLM must be eligible for the SYSSTC service class. To make IRLM eligible for SYSSTC, you do not need to classify IRLM to one of your own service classes.
- An installation-defined service class with a high velocity goal for DB2 (all address spaces, except for the DB2-established stored procedures address space):
  - %%%%MSTR
  - %%%%DBM1
  - %%%%DIST (DDF address space)

When you set response time goals for Distributed Data Facility (DDF) threads or for stored procedures in a WLM-established address space, the only work that is controlled by the DDF or stored procedures velocity goals are the DB2 service tasks (work performed for DB2 that cannot be attributed to a single user). The user work runs under separate goals for the enclave.

For the DB2-established stored procedures address space, use a velocity goal that reflects the requirements of the stored procedures in comparison to other application work. Usually, it is lower than the goal for DB2 address spaces, but it might be equal to the DB2 address space depending on what type of distributed work your installation does.

DB2 distributed requests

DDF receives requests from different clients: DB2 and other vendors' software; each request becomes an *enclave*. An enclave is an independent dispatchable unit of work, which is basically a business transaction that can span multiple address spaces, and can include multiple SRBs and TCBs. DB2 uses enclaves for work coming into the system through DDF. (DB2 also uses enclaves for other purposes, such as processor parallelism.) The enclave is created by DDF for an incoming connection when the first SQL statement starts to execute.

To better understand best practices for DDF workload classification, it is important to know the DB2 DDF thread and enclave relationship. Database access threads have two modes of processing:

- ACTIVE MODE

  A database access thread is always active from initial creation to termination.

- INACTIVE MODE

  A database access thread can be active or pooled. A database access thread that is not currently processing a unit of work is called a pooled thread, and it is disconnected. When a database access thread in INACTIVE MODE is active, it processes requests from client connections within units of work. When a database access thread is pooled, it waits for the next request from a client to start a new unit of work.

The CMTSTAT subsystem parameter (DDF THREADS installation field) specifies whether to make a thread active or inactive after it successfully commits or rolls back and holds no cursors. Refer to the IBM publication *DB2 Version 9.1 for z/OS, Installation Guide,* GC18-9846 for more details.

Because WLM assigns the performance goals to the enclaves, it is the lifetime of the enclave that WLM takes as the duration of the work. Therefore, when you run with CMTSTAT=INACTIVE, DDF creates one enclave per transaction, and response time goals and multiple time periods can be used. However, if you have CMTSTAT=ACTIVE or DDF creates one enclave for the life of the thread, response time goals and multiple periods should not be used.

You can have multiple filters to classify DDF threads (for example, their authorization ID, package name, user ID, or plan name). For a complete list, refer to *z/OS V1R8.0 MVS Planning Workload Management*, SA22-7602.

Tip: We suggest that you use INACTIVE MODE to take advantage of WLM's ability to manage each individual unit of work according to its business goals.

JDBC requests

JDBC sends its requests to DB2 using Driver Type-2 or Driver Type-4.

The work of Type-2 comes from WebSphere Application Server to DB2 and is classified under CB subsystem type. You are not required to classify it under DDF subsystem type.

For the Type-4/DDF workload, the DDF rules allow us to do classification. Not all of the DDF qualifiers are usable. For example, all JDBC applications use the same packages, so it is not a useful filter. If you want to base prioritization on application behavior, you can classify it based upon the first stored procedure called in a transaction, but it is only usable if you call a stored procedure.

A simple approach is to use a different data source within WebSphere Application Server for each classification and programmatically use a specific data source. Each data source has either a different AUTHID associated with it or a JDBC driver collection. You can classify based upon either of those.

The only other approach is to use the DB2 client strings. The client strings are text attributes associated with the connection, which we can use for classification of the workload. The client strings can be set as a part of the data source definition, or you can set them programmatically within the application so that every transaction can have a different value. We can use the client strings for workload classification, and we can also use them for end-to-end auditing (the fields are written in DB2 accounting reports). The recommended client strings are application name and user ID.

DB2 stored procedures

WLM is involved with DB2 stored procedures in three ways:

- It enforces goals to the DB2 stored procedures enclaves. These enclaves also support TCBs.
- It controls the number of DB2 stored procedure address spaces automatically, based on your service class goals.
- It controls the distribution of the stored procedures across the server address spaces. You can use WLM to dedicate address spaces to stored procedures of the highest business priority to attempt to ensure that there is always an address space available. You can do this by assigning work to different application environments.

Keep group-related stored procedures in the same address space to isolate them from others associated with different business functions. Isolate stored procedures from each other by configuring an address space to run only one procedure at a time by using the DB2 parameter NUMTCB=1.

When you access a stored procedure from a local client application, you do not need to define a specific performance requirement. All such calls inherit the performance requirements from the calling address space, because there is no independent enclave associated with the execution of the stored procedure. The stored procedure call is a continuation of the original transaction.

Only define specific performance requirements for the independent enclave under the DDF subsystem type when you access the stored procedure remotely through the DDF address space.

WLM definition for DB2 stored procedures

If you use WLM-established stored procedures address spaces, you must define a WLM application environment. WLM uses these definitions for its server address space management. Figure F-4 shows the relation between stored procedures, WLM application environments, and procedures definitions.

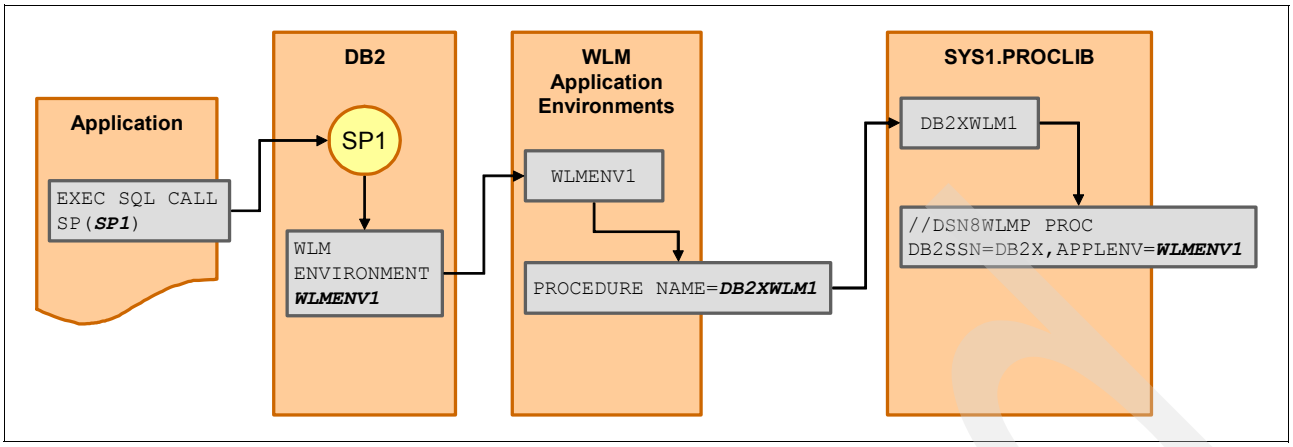


Figure F-4 Stored procedures definitions relationship

Table F-4 lists the recommended basic application environments that you typically define in your environment. There could be more, depending on your environment's requirements.

Table F-4 Basic WLM application environments

WLM Env	Description	APF Authorized	Notes
1	DB2-supplied stored procedures	Yes	Must be authorized for WLM refresh
2	REXX stored procedures	No	
3	User SQL stored procedures	No	
4	User JAVA stored procedures	No	
5	Execute DB2 utilities via DSNUTILS	Yes	NUMTCB=1

Important: To avoid creating unnecessary address spaces, create only a relatively small number of WLM application environments and service classes.

WLM routes work to stored procedure address spaces based on the application environment name and service class associated with the stored procedure. The service class is assigned using the WLM classification rules. Stored procedures inherit the service class of the caller. There is no separate set of classification rules for stored procedures.

Defining classification rules for stored procedures

You can define your own performance goals for a stored procedure only if you access the stored procedure remotely, because DDF creates an independent enclave for the incoming requests. All local calls inherit the performance attribute of the calling address space and are continuations of existing address space transactions (dependent enclave).

You have to define classification rules for the incoming DDF work.

Important: If you do not define any classification rules for DDF requests, all enclaves get the default service class SYSOTHER. This is a default service class for low priority work.

Define classification rules for SUBSYS=DDF using the possible work qualifiers (there are 11 applicable to DDF) to classify the DDF requests and assign them a service class.

Important: You can classify DDF threads by, among other things, stored procedure name. But the stored procedure name is only used as a work qualifier if the first statement issued by the client after the CONNECT is an SQL CALL statement.

Other classification attributes are, for instance: account ID, user ID, and subsystem identifier of the DB2 subsystem instance, or LU name of the client application.

For a detailed description and samples of how to classify DB2 stored procedures, refer to *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083.

Archived

Archived

G

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

ftp://www.redbooks.ibm.com/redbooks/SG247726

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247726.

Using the Web material

The additional web material that accompanies this book includes the following files:

File name	*Description*
ISWCDC.doc	Code Samples

How to use the web material

Copy and paste the DDL into your program, customizing it to fit your environment. Run the job with the JCL provided, after customizing it.

© Copyright IBM Corp. 2010. All rights reserved.

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 508. Note that some of the documents referenced here may be available in softcopy only.

- *50 TB Data Warehouse Benchmark on IBM System z*, SG24-7674
- *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637
- *50 TB Data Warehouse Benchmark on IBM System z*, SG24-7674
- *Leveraging IBM Cognos BI for Linux on System z*, SG24-7812
- *Using IBM System z as the Foundation for Your Information Management Architecture*, REDP-4606
- *DB2 9 for z/OS Technical Overview*, SG24-7330
- *DB2 for z/OS: Data Sharing in a Nutshell*, SG24-7322
- *Multidimensional Analytics: Delivered with InfoSphere Warehouse Cubing Services,* SG24-7679
- *Best Practices for SAP Business Information Warehouse on DB2 UDB for z/OS V8*, SG24-7644
- *DB2 UDB for z/OS: Application Design for High Performance and Availability*, SG24-7134
- *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370
- *DS8000 Performance Monitoring and Tuning*, SG24-7146
- *IBM Systems Storage DS8000: Architecture and Implementation*, SG24-6786

Other publications

These publications are also relevant as further information sources:

- *IBM InfoSphere Change Data Capture Version 6.2 (DB2 for z/OS) Performance Evaluation and Analysis white paper*
- *DB2 z/OS Utility Guide and Reference,* SC18-9855
- *Access Server and Management Console Installation Guide.* Product documentation
- *z/OS Support for Unicode: Unicode Services,* SA22-7649
- *IBM Cognos BI Installation and Configuration Guide,* http://download.boulder.ibm.com/ibmdl/pub/software/data/cognos/documentation/docs/en/cognos8_bi/8.3/inst_c8zlnx.pdf
- *IBM Cognos BI Framework Manager User Guide*, http://download.boulder.ibm.com/ibmdl/pub/software/data/cognos/documentation/docs/en/cognos8_bi/8.3/ug_fm.pdf

© Copyright IBM Corp. 2010. All rights reserved.

Online resources

These websites are also relevant as further information sources:

- IBM InfoSphere Change Data Capture for z/OS o6.2 delivers logbased change data capture for real-time data integration

  http://www-01.ibm.com/common/ssi/rep_ca/0/649/ENUSA08-1220/ENUSA08-1220.PDF

- High performance FICON (zHPF)

  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FQ127122

- Modified Indirect Data Address Word (MIDAW)

  http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp4201.html?Open

- IBM Cognos software

  http://www-01.ibm.com/software/data/cognos/customercenter

- IBM Blink project

  http://www.almaden.ibm.com/cs/projects/blink/

- Power consumption

  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101110
  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101265

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Index

Symbols
%SELECT 192

A
Accelerated Query Table (AQT) 36
Access Server 148
Accounting information (AI) 273, 497
Adaptive Apply 187
Address Space 75, 180, 339, 493
APAR 15, 270, 422
APAR OA27495 31, 288
application server 38, 77, 84, 211, 223, 267, 372, 469
  current load 484
  JVM heap size 95
  verbose garbage collection log 96
automated teller machine (ATM) 39, 338
AUTORESTARTINTERVAL 167

B
bandwidth resource 38
base goal 73
batch refresh 123, 266
  environment 123
  process 145
BI administrator 267
BI capabilities 82
BI capability
  broad range 82
  complete set 82
BookStore application 68, 79, 470
  WLMQUE panel 76
BookStore transaction 67, 483
  equal distribution 484
  incoming work 76
Buffer Manager (BM) 33
buffer pool 7, 12, 97, 144, 248, 252, 342, 353
  changed records 342
  multiple pages 248
  parallel sequential threshold 249
  space allocations 255
BUFSIZE 169
BUFTHRESHOLD 170
Business Intelligence
  administrator 325
  application 264
  Competency Center 7
  environment 8, 247
  front end 82
  query 8, 251–252, 371
  query request 32
  report 63
  solution 123
  system 7
  tool 60
  workload 60, 367, 371
  world 3, 383
Business Intelligence (BI) 11, 81, 264, 322, 370
business requirement 50, 264, 431

C
CACHEBLOCKSIZE 170
CACHELEVEL1RESERVED 171
CACHELEVEL1SIZE 171
calling address space
  performance attribute 502
  performance requirements 501
Capacity on Demand (COD) 340
CDC agent 148
  CHCPRINT DD 166
  configuration settings 153
  initial installation verification 154
  load modules 150
  same pair 168
CDC user
  exit 150
  ID 148
  ID access 161
  ID definition 148
  IDs map 148
CDC user ID 148
central processing unit (CPU) 248, 265
central processor complex (CPC) 50
Change Data
  Capture 121, 211, 378
  Capture attempt 203
CHCBNDPL 153
CHCCFGxx 153, 167
CHCCMMxx 154, 168
CHCCRCCH 171
CHCCRMTD 153
CHCDBMxx 154, 169
CHCDFMTD 151
CHCDFPAL 150
CHCGRNTA 153
CHCLDRxx 155, 174
CHCMTAUT 203
CHCPRINT DD 158
CHCUCSxx 155
class 1 processor time accessing a compressed index 17
class period (CP) 356
classification rule 7, 263, 348, 492
  separate set 502
  specific service class 496
client information 99, 273, 321, 352, 424
codepage conversion 155
  CONVERSION statements 155
codepages 155

© Copyright IBM Corp. 2010. All rights reserved.

Cognos report 81, 106, 112, 265, 327, 332, 348–349
development 60
execution 114
name 274, 329, 335, 352, 424
request 350
Cognos. See IBM Cognos
Collaborative Memory Management Assist (CMMA) 21
Collection name (CN) 273, 497
co-location project 1, 24, 72, 75, 483
possible topologies 485
column mapping 189
COMMITFREQ 172, 184
computing resource 271
conflict detection and resolution 196
consolidation one-to-many 187
consolidation one-to-one 187
container-managed persistence (CMP) 470
content store 86, 214
object manipulation functions 86
continuous mirroring 165
continuous update 146, 184
control flow 124
Control Program (CP) 21, 81, 248
CONVERSION SRCCCSID 155
count key data (CKD) 23
Coupling Facility 50, 248
Coupling Facility Control Code (CFCC) 28
CP CPU
time 335
CP time 29, 117
CP utilization 105, 365
CPU cost 250, 374
CPU Critical 269, 362
flag 73
CPU usage 276
significant effect 255
significant impact 170
CPU utilization 34, 180, 372
CTRLQUEUESIZE 168
Customer Relationship Management (CRM) 287

D

DASD Fast Write (DFW) 342
data auditability 42
data flow 124–125
generated code 141
data gatherer 415
Data Out (DO) 182
data source 38, 84, 273, 326, 478, 501
network protocol 478
data type 13, 74, 189, 191
Data Warehouse 1, 4, 9, 46, 68, 113, 121, 223, 228, 248, 263, 313, 321, 347, 422, 433, 483
DWREFRESH 267
DWSCHREP 267
data warehouse
ad-hoc and unpredictable nature 32
data flows 124
high availability 12
history tables 184
OLTP 374
recommended book list table 251
Spike 360
System z/OS resource management 7
understanding resource management 348
workload analysis 375
data warehouse (DW) 11, 47, 124, 247–248, 264, 269, 322, 348, 370, 486
Data Warehouse Maintenance 434
data warehousing
direct affect 247
direct support 11
good fit 378
major, immediate effect 11
System z 11
data warehousing (DW) 9, 47, 144, 247, 263, 322, 378
datapool 115
DATAQUEUESIZE 168
DB2 9 14, 88, 175, 253, 326
Clone table support 16
dynamic prefetch 18
index compression 17
recommended values 256
row format 19
work files 253
DB2 accounting
class time 422
data 8, 271, 273, 421
information 422
record 422
report 273, 334, 501
DB2 Class 1
accounting CPU time 272
accounting data 421
accounting trace 423
DB2 client
connection property 348
info 352
information 8, 99, 273, 321, 352
qualifier 334
string 273, 358, 501
DB2 client information
qualifier 331
DB2 connection ID 265, 349
DB2 Data
Base 264
Warehouse 48
DB2 instance 76, 483
DB2 LOAD utility
documentation 175
support 174
work data set 175
DB2 log
data 168–169
reader process 154
records one 169
DB2 member 49, 248, 254, 370
DB2 PE
table 212
DB2 plan 153

DB2 subsystem 7, 25, 47, 153, 211, 256
  DB2 entities 25
DB2 z/OS
  agent 198
  CDC agent 153
  CPU 205
  data 150
  source table 178
  subsystem 150, 153
  Utility Guide 174
  V8 New Function Mode 153
  workload 178
DB2/LINUXZ64 9.5.4 215
DB2910.ZDW.SDSN Exit 158
DB2910.ZDW.SDSN Load 158
DBA 264, 325
DBAT 328
DBMS Asynchronous Log (DAL) 181
DBMS Log Profile (DLP) 181
DBMS Supervision and Control (DSC) 181
DBMS synchronous log (DSL) 181
DBMS Table Change (DTC) 181
DBMS Table Refresh (DTR) 182
DD DSNAME 150
DDF thread 422, 500
default service class 493
default value 31, 80, 154, 249
defining a CDC agent 159
defining a CDC userid 160
defining an Agent/userid mapping 161
Design Studio 123
  SQW flows 133
discretionary goal 269, 494
Distributed Data Facility (DDF) 496
Distributed Relational Data Architecture (DRDA) 29
DRDA requester 273
DSNDEVT 175
DSNHLQ 175
DSNZPARM keyword
  CDSSRDEF 249
  PARAMDEG 249
DSNZPARM PARAMDEG 249
DTC task 181
DW application 3, 51, 383
DWDDFST service class
  period 363
dynamic prefetch 18
dynamic SQL 11, 322

E

EJB container 71, 470
EJB Mapping 67, 469
Enterprise Archive Repository (EAR) 68, 470
environment variable 80, 327
ESA/390 architecture 19
ETL process 145, 251, 371
  data contention 251
EXEC PGM 150
execution velocity 494
Extract, Transform and Load (ETL) 68, 141

F

fact table 13, 63, 101, 250, 371
Fibre Channel Protocol (FCP) 22
FICON Express4 22
fixed Pi
  universal donor 499
Framework Manager 100
  dimensional views 100
  panel shot 101
Full Disk Encryption (FDE) 340

G

Geographically Dispersed Parallel Sysplex (GDPS) 22
graphical user interface (GUI) 35, 322

H

HEARTBEATTIMEOUT 167
HTTP protocol 72, 483
HTTP request 78, 115, 210, 484
HTTP transport 78, 484

I

I/O Intensity 255
I/O priority
  information 341
I/O processing 27, 248, 492
IBM Cognos 54, 60, 87, 100, 106, 322
  administration tool 273, 330
  batch scheduler 275
  BI version 8.4 214
  Connection 86
  Context 330
  Framework Manager 85
  package 265, 330
    DB2 performance statistics 334
    name 266, 332, 349
    PH3DW 332
  performance 106
  report 112
  report server 89
  report service 221
IBM Cognos BI 82, 124, 214
  adminstrators 326
  analysis functionality 84
  analysis studio 84
  application 83
  architecture 82
  client modeling tool 100
  component 86
  configuration 83
  data 84
  installation 100
  need 86
  platform 82
  Server 86
    components 88
    documentation 87
    product code 91

Archived

version 8.4 88
service 86
IBM Cognos Business Intelligence 82
IBM Cognos Framework Manager 100
IBM Smart Analytics Optimizer 31
brief overview 33
current utilization 34
eligible queries 34
existing MART definition 37
further data storage 33
main memory 33
processing capacity 33
IBM System z
logical partition 27
server 21, 27
ICDC.ZDW.SCHC Load 150
ID mappings 148
IFI 169
Importance High 349
improve the overall (I/O) 10
incoming work 72, 75, 272, 351, 483, 495
independent enclave 272, 354, 501
specific performance requirements 501
index compression 17
InfoSphere CDC 65, 146, 204
address space 194
SELECT workload 204
source table 187
system parameter 160
trickle update 66
workload design 185
InfoSphere CDC, codepage conversion 195
InfoSphere CDC, derived column 191
InfoSphere CDC, effect of %SELECT 205
InfoSphere CDC, event log 200
InfoSphere CDC, expression 191
InfoSphere CDC, preventing recursion 188
InfoSphere CDC, row and column filtering 195
InfoSphere CDC, transformation 194
InfoSphere CDC, user exits 198
InfoSphere Change Data Capture 65, 146–147, 153, 155, 157, 162, 166, 169–170, 174, 183, 278, 366
address space 202–203
attempt 203
authorization ID 175
Configuration 146
CPU usage 180
enterprise architecture 149
installation 161
issue 169
operation 149
OPx buffer 169
performance 178, 184
performance monitoring 167
target library 149
task 166, 180
InfoSphere Warehouse
component 123
refresh scenario 144
SQL Warehousing Tool 65
SQW 121
infrastructure that (IT) 9
infrequently observed (I/O) 248, 272
initial configuration 149, 153
configuration member 155
INSERT processor consumption 13
Integrated Coupling Facility (ICF) 26
Integrated Facility for Linux (IFL) 26
Intelligent Resource Director (IRD) 11
Internal Coupling Facility (ICF) 28, 210
Internal Resource Lock Manager (IRLM) 33, 144
IP address 25, 415
ISBN 65, 111, 141
issued on distributed (ID) 322

J

J2EE application 70, 470
Java Naming and Directory Interface (JNDI) 70
Java Virtual Machine (JVM) 89, 482
JDBC 67, 469
JDBC connection
pool 259
type 2 478
JDBC driver 478
collection 501
correlation information 324
path 80
JDBC driver, Type 2 78
JDBC driver, Type 4 78
JDBC provider 477
Journal control fields 194

K

KB 257
KEEPDICTIONARY 175
key WLM
construct 7
metric 283

L

Language Environment (LE) 155, 211
latency thresholds 201
LDAP server 25
level 4 275, 352
Linux 10, 207
Linux guest 21, 91, 210, 353
LiveAudit 187
LOAD DATA
control statement 175
statement 176
logical partition 11, 277
central processors 28
entire cluster 11
LOGPOLLINTERVAL 155, 172
LPAR 11, 48, 76, 208, 252, 274, 369, 478
LPARs
I/O devices 19

M

Management Console 148, 159
materialized query table (MQT) 14, 63
MDX 84
Message Driven Beans (MDB) 470
message is queued (MQ) 71
mixed workload 10, 43, 59, 263–264, 341, 347, 370, 378
  considerations 341
  relative importance 265
  traditionally huge amounts 21
Modified Indirect Data Address Word (MIDAW) 22
Multiple Allegiance (MA) 23, 341
multiple Linux
  guest 24
  system 88
multiple LPARs 3, 24, 45, 288, 369, 383
  single System z 40
multiple period 268, 285, 358, 500
multiple z/OS LPAR
  configuration 209
  WebSphere configuration 213

N

New Function Mode (NFM) 15
Node Agent 77, 483
non data 212
  active logs 212

O

OLAP data
  set 41
  source 84
OLTP application 53, 68, 383
OLTP transaction 146, 278, 380, 478
Omegamon XE 332
Online Analytical Processing
  little bit 16
Online Analytical Processing (OLAP) 16, 124, 264
Online Transaction
  Processing 11, 46, 67, 140, 211, 250, 337, 370, 469
  Processing workload 68
Open Database Connectivity (ODBC) 13
operating system 10, 90, 492
Operational BI 10, 264, 278, 329, 348, 370
  activity 104
  query 252
operational data store (ODS) 54
operational database 251, 371, 378
  log records 252
  order table 251
  real time 251
OPTIXOPREF 258
OPx buffer 169
  threshold 169
ORDE R_ID 111

P

Package name (PN) 265, 329, 497
PAL 150
PALCLEANUPTIME 167
PALRETPD 167
Parallel Sysplex
  environment 39
performance index (PI) 280, 355, 497
Persistency 202
PH3DW.BOOK ORDER_DETAIL_FACT 142
  CLIST 65
  Fact 65
PH3OLTP.ORDE R_DETAIL 141
plan name
  various InfoSphere Change Data Capture tasks 180
PLANNAME 324
PLANSUFFIX 154
Prefetch quantity 18
Procedure name 212, 273, 497
processor cost of running a FETCH or INSERT 13
Processor Resource/Systems Manager (PR/SM) 20
processor type 280, 354
processor utilization 41, 77, 280, 343, 354
Product Administration Agent (PAA) 182
product administration log (PAL) 150
Proxy Server 78, 484
  new work 484
  same environment 485
purpose CP 26–27
purpose processor 210, 277, 287, 355–356

Q

Q1035.CUST OMER_ID 141
Q1035.ORDE R_ID 141
Q1035.TSTA MP 141
Q1045.ORDE R_ID 141
Queries per minute (QPM) 371
Queued Direct I/O (QDIO) 21

R

Rational Application Developer (RAD) 469
Rational Performance Tester
  high level concepts 112
Rational Performance Tester (RPT) 68, 112, 210
RationalÆ Application Developer (RAD) 67
real-time monitor 201
RECOVERYRETRYLIMIT 173
Redbooks Web site 508
  Contact us xvii
reliability, availability, and serviceability (RAS) 259
Reordered Row Format (RRF) 18
report class (RC) 272, 331–332, 350, 492, 495
report query 103, 105
REPSTATSINTERVAL 154, 168
resource consumption 32, 90, 265, 322, 358, 430
  extra long reports 430
  high-level view 267
  unpredictable amount 32
resource group 265, 492
  maximum capacity 276
Resource Limit Facility (RLF) 11, 250

Resource Management
Facility 279, 321, 343
Facility Users Guide 279
resource management 1, 4, 19, 99, 225, 263, 347, 422
user groups 350
Resource Measurement Facility (RRMF) 493
Resource Monitor Facility (RMF) 353
response time
detailed characterization 355
significant difference 381
slight elongation 361
response time (RT) 14, 146, 266, 342, 349, 371, 422, 494
response time goal 494
RETRYCACHESIZE 173
right click 130, 427, 470, 474
risk avoidance 42
RMF interval 355
RMF Monitor
I report 371
III Classification Data 281
III data gatherer 281
III Enclave Classification Data 332
RMF monitor
III Classification Data view 281
III interval 281
RPT Schedule
User Group 117
RPT schedule 114
definition 119
RPT test 114
comment 115
Data Pool associate 116
different stages 114
Loop 115
schedule 117
transaction 115
workflow 120
workload distribution 119

S

same data 7, 35, 49, 142, 250–251, 339
co-locating operational and data warehouse 252
Data Warehouse databases 251
different compression ratio 35
multiple, aggregated views 339
same LPAR 48, 348
data warehouse subsystems 49
same System
z server 10
same time 10, 93, 146, 223, 341, 415
data warehousing 146
multiple workloads 10
same volume 23
sar command 415
scalability issue 372
Scalable Architecture for Financial Reporting (SAFR) 44
SCHCDATA library 150
first member 167
SE20 DSNVMON 260
SEGSIZE 253
selected service class
dispatching priority 276
separate LPAR 39
servant instance 73, 483
server address space 269, 501
stored procedures 501
Service Class
DWDDFHI 275, 281, 350
final shape 431
named collection 493
only queue work 73
queue work 483
relative importance 278–279
same set 493
service period 372
WASTCHI 73, 349
WASTCLOW 73
service class 7, 10, 72, 263, 331, 348, 371, 422, 430, 483, 492–493
achieved performance 498
available processor resources 276
Service Class period 355
service policy 284, 354, 492–493
Service Request Block (SRB) 250
service unit 268, 354
SERVICENAME 154
Set Client Information (SCI) 321
shell script 415
create text reports 415
single LPAR 24, 47, 210, 345, 367
configuration 383
environment 5, 252, 367
measurement 371
transactional and data warehouse query workloads 5
SMF record 74
SMF type 120 subtype 9 classification data section 75
SMF type 120
subtype 9 data 75
subtype 9 record 74
SORTDEVT 176
SORTKEYS 176
SORTNUM 176
SORTOUTSIZE 176
source table 141, 148
column names 190
row insert operation 187
specialty engine
offloading 355
today 31, 288
specialty engine (SE) 8, 11, 26, 210, 250, 287, 335
Spike OLTP 378
SQL 84
SQL command 215
SQL statement 16, 133, 216, 248, 335, 338
target table 174
SQLWHEREROWSEL 155
standard table mapping type 187
static SQL 249, 322

Storage Pool Striping (SPS) 22, 342
Subscriptions, flag for refresh 200
Subscriptions, parking 200
Subscriptions, starting and stopping 198
subsystem instance (SI) 274, 497
subsystem type 494
summarization table mapping type 187
SYSDISCSIZE 176
SYSERRSIZE 176
SYSMAPSIZE 176
sysplex 77, 248, 267, 484, 492
  service definition 492
sysplex PATPLX61 211
System Assist Processor (SAP) 26
system managed storage (SMS) 41
system P61 79, 213, 485
  application servers 486
  proxy server 485
system resource 11, 32, 72–73, 339, 360
System z 3, 9–10, 54, 82, 86, 123, 207, 247, 263, 340
  common monitoring and management tools 7
  data warehouse co-location 38
  IBM InfoSphere Warehouse 123
  InfoSphere Warehouse 123
  Linux 213
  many strengths 24
  performance 340
  processor scalability 41
  running Cognos 82
  transactional systems 6
  well-known benefits 82
System z9
  channel enhancement 22
systems P60 79, 210, 485
  WebSphere application servers 486
SYSUT1SIZE 176

T

table mapping 162, 186
table mapping types 187
table mappings
  large number 162
table space 11, 97, 132, 253
  NOSHARE attribute 144
Tablespace Id 217
target CDC agent 157, 169
Target table 132, 155, 174
  column names 190
  full refreshes 186
  initial load 165
  multiple rows 187
  new row 197
  refresh or initial load 183
  row insert operation 187
  SQL statements 174
TB 12, 338
timestamp 61, 194
TITLE_DIM 101
total utilization 366, 373
Transaction Class 351
transactional workload 9, 60, 68, 210, 266–267, 348, 370, 469
  increased demand 359
  response times 381
  similar view 267
  spike 359
transactions per second (TPS) 371

U

URL 116, 267
USELOADER 174, 183
user
  ID 326
  workstation name 326
USER cognos 215
User group 100, 114, 117
user ID (UI) 107, 148, 266, 322, 495
user interface
  WebSphere environment 489
user interface (UI) 489
users 264, 329

V

Value Added Edition (VUE) 49
VARCHAR column 18
VDWQT 342
Velocity goal 269, 494
  installation-defined service class 499
  Service classes 284
virtual machine
  Java classes 94
Virtual Machine (VM) 19, 89
Virtual Machine Resource Manager (VMRM) 21
virtual pool size (VPSIZE) 249
VSAM data
  set 213

W

WASTCHI service class 73, 75, 278, 359
website 76, 283, 313, 423
WebSphere Application Server 67–68, 87, 91, 221, 232, 470, 482, 501
  administration 94
  code 91
  different data source 501
  maps incoming work 72
  profile 91
  setup topology 72
  web container 95
websphere environment variable
  displays part 80
WebSphere setup topology
  detail 67
  introduction 67
WebSphere Studio Workload Simulator (WSWS) 68, 210
WLM classification
  file 73
  qualifier 331

Archived

rule 274, 331, 496
WLM DDF
qualifier 272
subsystem 273
WLM definition 265, 501
WLM goal 277, 355, 359
WLM policy 20, 106, 263, 284, 324, 354
other components 286
WLM report
class 329, 351
WLM report class
RCOGSERV 329
RCOGUSER 329
WLM SCI 327
WLM service
class 72–73, 329
class definition 268
definition 7, 264, 367
definition formatter 283
definition modification 267
period 270
policy 72–73
WLM service definition 263
WLM_SET_CLIENT_INFO 323
work file 13, 248, 253
insufficient number 253
work request 272–273, 332, 351, 494
Monitor views 332
Workload Manager 10, 263, 491

Z

z/OS 10, 81, 123, 347
z/OS DB2
monitoring tool 332
query workload 355
z/OS performance
administrator 422
analysis 355
enhancement 285
z/OS subsystem 31, 141, 348
data warehouse DB2 142
dedicated data warehouse DB2 348
different DB2 141
DWH DB2 142
local target DB2 141
remote OLTP DB2 141
remote source DB2 141
same local DB2 142
source DB2 142
z/OS WLM
class 72
performance data 353
z/OS Workload
Management 105
Manager 20, 23, 341
z/VM 5.4 21
zAAP 26, 28, 288, 354
zAAP processor
eligible work 31
utilization 374
zAAP utilization 373
zIIP capability 31
zAAP 31
zIIP eligible work 30, 364
zIIP processor 29, 250, 277, 287, 348, 378
zAAP processor eligible work 31
zIIP utilization 365, 376
zSeries Integrated Information Processor (ZIIP) 287, 348

Archived

Redbooks

Co-locating Transactional and Data Warehouse Workloads on System z

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM.COM/REDBOOKS

Archived

Archived

Co-locating Transactional and Data Warehouse Workloads on System z

Reaping the benefits of data-sharing your OLTP and DW databases

Facilitating resource management for co-located workloads

Exploiting DB2 client attributes with IBM Cognos

As business cycles speed up, many clients gain significant competitive advantage from quicker and more accurate business decision-making by using real data. For many, choosing the path to co-locate their transactional and analytical workloads on System z better leverages their existing investment in hardware, software, and skills. We created a project to address a number of best practice questions on how to manage these newer, analytical type workloads, especially when co-located with traditional transactional workloads.

The goal of this IBM Redbooks publication is to provide technical guidance and performance trade-offs associated with resource management and potentially DB2 data-sharing in a variety of mixed transactional and data warehouse System z topologies. The term co-location used here is specifically defined as the practice of housing both transactional (OLTP) and data warehouse (analytical) workloads within the same System z configuration. We also assumed that key portions of the transactional and data warehouse databases would reside on DB2 for z/OS. The databases may or may not reside in a DB2 data-sharing environment; we discuss pros and cons in this book.

The intended audience includes DB2 data warehouse architects and practitioners who are facing choices in resource management and system topologies in the data warehouse arena. This specifically includes Business Intelligence (BI) administrators, DB2 database administrators (DBAs) and z/OS performance administrators and systems programmers. In addition, decision makers and architects can utilize this book to assist in making platform and database topology decisions.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7726-00 ISBN 0738434787