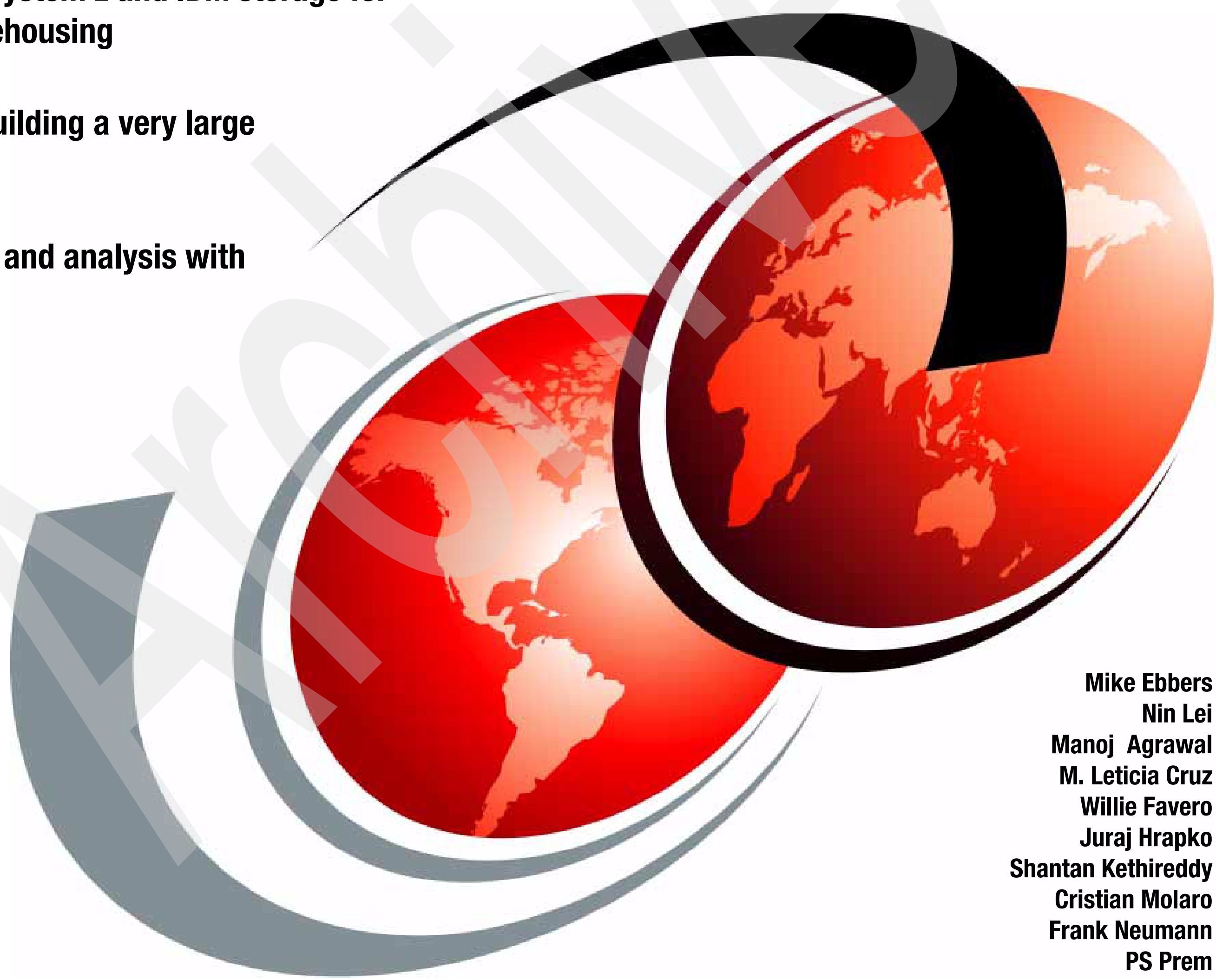
IBM

50 TB Data Warehouse Benchmark on IBM System z

- Why use System z and IBM storage for Data Warehousing
- Tips for building a very large database
- Reporting and analysis with Cognos



Mike Ebbers
Nin Lei
Manoj Agrawal
M. Leticia Cruz
Willie Favero
Juraj Hrapko
Shantan Kethireddy
Cristian Molaro
Frank Neumann
PS Prem

Redbooks

ibm.com/redbooks

International Technical Support Organization

50 TB Data Warehouse Benchmark on IBM System z

May 2009

SG24-7674-00

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

First Edition (May 2009)

This edition applies to Version 1 Release 9 of z/OS and Version 9 of DB2.

Contents

Figures

Tables

Examples

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.

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®
CICS®
DataStage®
DB2 Connect™
DB2®
DFSMS™
DFSMSdfp™
DFSMShsm™
DFSORT™
DRDA®
DS6000™
DS8000™
ECKD™
Enterprise Storage Server®
ESCON®
eServer™
FICON®
FlashCopy®
GDPS®
Geographically Dispersed Parallel Sysplex™
HACMP™
HiperSockets™
Hiperspace™
HyperSwap™
IBM®
IMS™
Informix®
InfoSphere™
Language Environment®
MVS™
OMEGAMON®
Optim™
OS/390®
Parallel Sysplex®
pSeries®
pureXML™
QMF™
RACF®
RAMAC®
Rational®
Redbooks®
Redbooks (logo) ®
REXX™
RMF™
Sysplex Timer®
System p®
System Storage™
System x™
System z10™
System z9®
System z®
Tivoli®
TotalStorage®
VTAM®
WebSphere®
z/OS®
z/VM®
z10™
z10 EC™
z9®
zSeries®

The following terms are trademarks of other companies:

Cognos, and the Cognos logo are trademarks or registered trademarks of Cognos Incorporated, an IBM Company, in the United States and/or other countries.

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

ACS, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

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

Microsoft, SQL Server, 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, other countries, or both.

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.

Preface

Customers have a strong interest in the scalability of data warehousing solutions on the IBM® System z® platform using DB2® for z/OS®. Because business intelligence environments continue to grow, this IBM Redbooks® publication explores the scaling and management of data in sizes beyond 50 TB. We explore the architectural software components that enable us to manage a data warehouse of this magnitude, including the Cognos® product set.

The objective of our 50 TB study was to test System z scalability in a business intelligence environment and to develop the best practices of managing large data warehouses. The IBM System z was proven to scale to larger volumes, manage a mixed workload, lower the cost of a BI solution using zIIPs, and increase performance by using hardware compression.

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 at the International Technical Support Organization, Poughkeepsie Center. He has been with IBM for 35 years, supporting and teaching about mainframe computers. He has been with the ITSO for 14 years.

Nin Lei is a recognized expert in DB2 performance and database design with interests in Very Large Databases (VLDB). He is a frequent speaker at conferences presenting data warehouse topics. He is a Certified Executive IT Specialist at the IBM World Wide Mainframe Benchmark Center conducting customer studies. He drives performance analysis work for all 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.

Manoj Agrawal is a Senior IT Specialist with IBM Global Business Services. He has 7 years of experience at IBM working on Application Integration with clients in the Computer Services and Banking industries. His areas of expertise are database technologies, application integration and the payments domain. He holds a bachelors degree in Computer Science and Engineering from Berhampur University, India.

M. Leticia Cruz is an Accredited Senior IT Specialist and a Certified DB2 Solutions expert at the IBM Worldwide Mainframe Benchmark Center. She has more than 10 years experience in customer engagements involving benchmark, proof-of-concept, and scalability studies. She has written various technical papers, presentations, and customer demos, and has participated in various ITSO residency programs. She is a technical resource for VLDB build and data warehousing projects.

Willie Favero is an IBM Senior Certified IT Software Specialist and DB2 SME with IBM's Silicon Valley Lab Data Warehouse on System z Swat Team. He has over 30 years of experience working with with databases with more than 24 years of that working with DB2. He is a sought after speaker for international conferences and user groups, publishes articles and white papers, and has a top technical blog on the Internet.

Juraj Hrapko is an IT specialist in IBM Slovakia working at Global Technology Services for four years, mostly on data warehouse and business intelligence projects. Juraj has a Master of Science degree from the University of Economics in Bratislava and is an IBM certified solution designer in Business Intelligence.

Shantan Kethireddy joined IBM in 2001. Shantan is a DB2 for System z Field Technical Sales Specialist supporting information integration, data warehousing, and business intelligence products and solutions. He spent six years working in DB2 development with a focus on SQL query optimization and data warehousing. He received his Master of Science degree from the University of Iowa in Electrical and Computer Engineering.

Cristian Molaro is an independent consultant based in Belgium working in Europe for costumers with some of the largest DB2 installations. His main activity is linked to DB2 for z/OS administration and performance. He is an IBM Certified DBA and Application Developer for DB2 for z/OS V7, V8 and 9. He is co-author of the IBM Redbooks publication *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637. He holds a Chemical Engineer degree and a Master in Management Sciences.

Frank Neumann is a Software Architect with the Datawarehouse on System z Center of Excellence, based in the IBM development lab in Boeblingen, Germany. During his 11 years with IBM, Frank worked as a developer, team lead and architect for software components and solutions for WebSphere® and Information Management products. He holds a Master degree in Computer Science from the University of Karlsruhe, Germany.

PS Prem is a Senior IT Specialist with IBM AP SWG TechWorks. He is the brand technical specialist on IBM Information Management for System z products in the Asia Pacific region. Prem has been with IBM for five years, primarily focusing on Technical Sales. Prior to joining IBM, he spent 10 years in application development on z/OS and four years in pre-sales for application development tools on z/OS.

Figure 1 Row 1, seated: Shantan Kethireddy, Willie Favero, Nin Lei, Mike Ebbers. Row 2, standing: Frank Neumann, Manoj Agrawal, Juraj Hrapko, PS Prem.

Contributors and reviewers

Thanks to the following for their time and expertise in contributing to this book.

Mike Cain
IBM Rochester

Ann Jackson
BI & DW Solutions Development, IBM

Jonathan Sloan
Data Warehouse for System z SWAT Team Lead, IBM

Edward Stepec
Cognos TES Practice Area Leader, IBM

Thanks to the following for their time and expertise in reviewing this book.

Rick Butler
Bank of Montreal (BMO) Financial Group, Toronto

Gary Crupi
IBM Milwaukee

Roger Miller
IBM Santa Teresa

Thanks to the benchmark team for sharing the results of their hard work with us:

Nin Lei, team lead
M. Leticia Cruz
Andrea Harris
Ron Lounsbury
Susan Lynch
Dino Tonelli
Judy Viccica
Benchmark Center, IBM Poughkeepsie

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

ibm.com/redbooks/residencies.html

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 in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail 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

1

Introduction

Over the last decade, data warehousing and business intelligence (BI) have been undergoing significant evolution and growth. The growth has been phenomenal over the last decade, with increasing customer focus on continuous real-time data being crucial to business functions. Enterprises across the world are increasingly focusing on larger and more efficient warehouses for their extensive data requirements.

In this chapter we give a broad overview of this book on the growing data warehousing landscape, and how enterprises can respond to this phenomenal growth of data with the help of DB2 and System z.

1.1 Introduction

Enterprises across the world are increasing focus on their data warehouse and BI initiatives. It continues to be a strong focus area in the overall strategic plan of most enterprises across the world.

As data warehouses and BI environments continue to grow rapidly, and BI insights become critical components to operational workloads, customers, as well as IBM organizations, have expressed a strong interest in how this growing data can be efficiently stored, processed, and managed.

Customers have expressed a strong interest in understanding how data warehousing solutions built on the System z platform running with DB2 9 for z/OS can be the answer to their growing requirements. The configuration, scalability, and management of data warehousing solutions on System z to create a balanced data warehouse is where enterprises would like to see themselves.

It is thus critical to explore the scaling and management of very large data warehouses with DB2 9 for z/OS and System z. To explore the capabilities of DB2 9 for z/OS, a data warehouse benchmark study was conducted on a 50 TB data warehouse to learn about the operational and performance characteristics of a very large data warehouse. We explore the architectural software components that enable us to manage a data warehouse of this magnitude.

1.2 Objectives

Managing workloads in very large data warehousing environments spans beyond running queries against the database. While response times and query throughput rates are important metrics in a production environment, the management aspect is just as important. Managing mixed workloads intelligently is vital in every organization. Likewise, the feasibility and the ease (or lack of) of using materialized query tables (summary tables) also plays an important role in the day-to-day operations of an organization.

As a result, multiple factors are essential to the optimal operation of a very large data warehouse. The sub-topics below discuss the factors that contribute to the management of a very large data warehouse and are further detailed in the chapters of this book.

1.2.1 VLDB build tips and pointers

Numerous studies have been conducted on data warehouses of various sizes, but the largest of these data warehouses have been in single digits of terabytes in size. These studies have resulted in best practices and presentations used to educate enterprises and customers about these principles. As enterprises grow, they look at their data warehouses being larger and larger, beyond the single digits of terabytes in size. Customers are now looking for an example on how the enterprise will function with their rapidly growing data warehouses.

In a pro-active response, we take the size of data warehouse testing to the next level to build and execute tests against a 50 TB data warehouse using DB2 9 for z/OS and System z. We study the design techniques, data loading times, index build times and sort work file allocation strategies to gain a better understanding of the best techniques for a very large data warehouse using DB2 9 for z/OS.

Chapter 5, “VLDB build tips and pointers” on page 69 discusses our experiences.

1.2.2 Balancing a data warehouse

An offering called Balanced Configuration Unit (BCU) has been available on IBM System p® and System x™ for a number of years. It is based on the concept that it takes a balance of resources, including CPU capacity, memory, storage, and I/O bandwidth, to support a data warehouse of a specific size. System z has many hardware and software features that assist with this balance. However, it does not have official equations that balance these resources.

During our study, we looked at an appropriate balance of resources, based on descriptions of a workload to result in a balanced data warehouse. These studies will help customers understand the components that are part of a balanced data warehouse and determine the right mix for their requirements.

Chapter 6, “Balancing a data warehouse” on page 91 discusses this in detail.

1.2.3 Compression

Compression has been around for a long time and every customer is aware of the significant storage benefits that compression offers to their data warehouse. What is not so well known is the impact of compressed data to query performance. Compressed tables use fewer pages and could lead to performance improvement for certain queries.

Index compression is a new feature in DB2 9 and little performance data is available. Similar to data compression, there is an interest in understanding the effect of storage savings, query performance, and CPU overhead with index compression.

Chapter 7, “Data and index compression” on page 105 discusses this in detail.

1.2.4 Scalability

As businesses grow, data processing needs grow. As rapid growth occurs, companies need a way to scale their business successfully. The ease with which a system is scalable should be one of the major factors in choosing the database that hosts an enterprise data warehouse.

Scalability includes processing and data. By adding more central processor (CPU) capacity, one would expect either faster query response times or higher throughput rates. Data scalability answers the question of what happens to query performance when data size grows. The answer is simple if a query accesses the same amount of data regardless of database sizes. Although such queries exist in production environments, it is likely many queries will access a larger amount of data when the database grows. Measurements have been run with different CPU configurations and varying data sizes to quantify the CPU and data scalability effect.

Chapter 9, “Scalability” on page 167 discusses our findings in detail.

1.2.5 Materialized query tables (MQT)

Materialized query tables (MQT), earlier called summary tables, are an important part of any data warehouse. For large data warehouses, the amount of time and sort work file allocations required for an MQT pose a challenge for administrators.

Chapter 10, “Using materialized query tables” on page 183 discusses how MQTs can be used in the context of very large data warehouses to reduce query response times for strategical and tactical analytical queries.

1.2.6 Reporting and analysis with Cognos

IBM Cognos 8 Business Intelligence is a powerful suite of components that delivers the complete range of BI and performance management capabilities on a single, service-oriented architecture (SOA). IBM Cognos 8 BI helps meet BI objectives in the enterprise, along with the needs of every kind of user from simple report consumers to professional authors and developers.

Chapter 8, “Reporting and analysis with Cognos” on page 123 discusses reporting and analysis with Cognos, with a focus on interaction with a very large database on DB2 9 for z/OS, operational BI, and an implementation on System z.

1.2.7 Workload management

It is important to properly balance the workloads on a server so as to optimize its system resources. Spare capacity is a waste of valuable resources. It is thus economical to place multiple workloads on a server. In such an environment the main problem is to avoid having one workload dominate resource consumption and depriving other workloads from reaching their goals. If sufficient resources are available for all workloads to meet their goals, it is important to put a workload management policy in place to ensure resources are assigned appropriately to the workloads.

Chapter 11, “Workload management” on page 211 discusses this in detail.

1.2.8 zIIP redirection

Queries running in parallel mode are eligible for redirection to run on a zIIP instead of a general processor (CP). Although the percentage of redirection is clearly stated in the DB2 code, the actual amount, as observed by a query, is influenced by the amount of time a query runs in parallel mode. Not all phases of a query run in parallel. Certain functions have to be performed by the coordinator task, and the percentage of time used by this task varies from query to query.

1.3 Summary of objectives

The objective of the 50 TB study was to deliver the proof points of System z scalability in a BI environment and to develop best practices of managing large data warehouses. It showcases the unique value of the IBM System z to perform the following tasks:

- Scale to larger volumes
- Manage a mixed workload, such as operational BI (a large number of users with smaller queries), using Workload manager
- Lower the cost of a BI solution using IBM System z Integrated Information Processors (zIIPs)
- Minimize compression overhead by using hardware compression

2

Data warehousing overview

This chapter provides an overview of data warehouse history, evolution, features, and future perspectives. It shows possible implementation options and the architectures that we used during our study.

2.1 Introduction

To gain a competitive advantage, accurate decisions need to be made. Strategies and decisions need to be built on a solid information base. Organizations have the necessary data in-house, it just needs to be collected and processed. A data warehouse (DW) is a means to gather data in one place to support business decisions. These include not only the strategy of upper management but also the needs of departments all over the organization. A data warehouse supports subjects such as business controls, customer behavior, fraud detection, revenue assurance, and price and product analysis.

2.2 Definitions

Before we discuss our project, we need to define some relevant terms.

Data warehouse

A data warehouse gathers data, actual and historical, from various sources across an organization and the outside world to facilitate reporting and analysis for business users. The term *Business Data Warehouse* was first introduced in IBM System Journal, Vol. 27, No. 1 in 1988. It was defined as follows:

> The BDW is the single logical storehouse of all the information used to report on the business. The BDW is based on the relational concept. In relational terms, the user is presented with a view or number of views that contain the accessed data. The structure of these views is determined solely by the requirements of that user.

Barry Devlin[1] states "A data warehouse is simply a single, complete, and consistent store of data obtained from a variety of sources and made available to users in a way they can understand and use in a business context".

Traditionally, the Kimball Group[2] has referred to the process of providing information to support business decision making as data warehousing. Now they use the phrase *data warehouse/business intelligence* (DW/BI) to mean the complete end-to-end system. They refer to the queryable data in a DW/BI system as the enterprise's *data warehouse*, and *value-add analytics* as BI applications.

A data warehouse is a multisubject-oriented database, historic (point in time) in nature, that typically contains detailed data. A data warehouse is designed to serve all possible decision support processes for an organization and in informational applications. Two types of data subsystems make up the data warehouse environment:

- Enterprise data warehouse (EDW)

  EDWs contain all data with a global scope.

- Data marts

  Data marts contain data for a specific business.

An EDW is corporate wide in scope. with various subject-oriented data about the organization. It contains non-volatile data that is integrated, cleansed, and derived as needed from various source systems and capable of holding historical data.

1 Barry Devlin, "Data Warehouse: from Architecture to Implementation", Addison-Wesley, 1997.
2 http://www.kimballgroup.com/

Data mart

A data mart (or business process dimensional model) is more business-oriented. It has a more restrictive scope. It is not corporate-wide, and tends to be smaller in terms of size and usability. It can have primitive, summarized, and derived data that answers a particular business question and helps a particular business group within an organization. A data mart supports looking at important measures by a variety of contextual dimensions. A data mart is created when the business users have some precise questions in mind that they need consistently answered over a period of time. A data mart commonly supports multi-dimensional analysis and is not necessarily based on a relational database.

Operational data store (ODS)

ODS is a data store to support queries for point in time information for a specific business function (for example, payables, premiums, credit card transactions). The ODS can also be used to feed the EDW and, hence, dependent data marts. An ODS is integrated and subject-oriented, containing current data with direct update paths from applications. This makes it volatile with current valued information that distinguishes it from the data warehouse. For general characteristics and benefits of an ODS, refer to *Building the Operational Data Store on DB2 UDB Using IBM Data Replication, WebSphere MQ Family, and DB2 Warehouse Manager*, SG24-6513.

Extract, transform, and load (ETL)

ETL is a data warehouse function that manages extraction from various sources, transformation of data into desired information and load into data warehouse and data mart databases. ETL consist of multiple processes, that are scheduled and automated mostly using parallelism to fit into designated time-frame. ETL processes are designed to cleanse and combine data to get desired business information. Transformation of data makes use of a staging area, which is a temporary place where data is stored during the transformation process.

Warehouse catalog

A warehouse catalog is the subsystem that stores and manages all the metadata. The metadata refers to information such as data element mapping from source to target. A data element is a piece of information for information systems and business users. Data models, both logical and physical, describe the actual use of the data and temporal information. In a banking environment, for example, if you have a field such as Chargeoff Amount, it would be worthwhile to the information systems and business users if it is net of Recovery amount. An example of the use of metadata would be to inform users how they may be able to distinguish a self-serve safe deposit box from a non-self-serve one, or business and commercial loans from consumer loans. Temporal information refers to the business information where the data spread out over time. This could be loan information such as borrower credit ratings, collateral ratings, and so on.

Query and reporting

A query is defined as a business question that is translated into a computer query. In a query and reporting process, the business question is defined, the data is accessed, the answer set is returned, and the report is presented in a chart or a graph to the end-user. The reports can be canned, where they represent a stable workload, or they can be ad-hoc, where they represent a variable workload.

On-line analytical processing (OLAP)

OLAP is the process of analysis using a multi-dimensional view of data in aggregated form, thus enabling rapid execution time. OLAP usually involves a multi-dimensional data store from which tools (such as those provided by Cognos software) can display measures, use its calculation component along several dimensions, and present data to users. Therefore, a user should know the kind of queries in advance to make use of OLAP. OLAP sometimes requires highly complex cross dimensional calculations, allocations, and time series calculations. In Relational OLAP (ROLAP) tools, the data is sourced from relational structures as opposed to the source being from a prebuilt OLAP cube. Because pure ROLAP tends to have performance issues, IBM also offers a Hybrid OLAP (HOLAP) solution, where the cube is cached in memory but is also persistent in a relational database.

Data mining

Data mining is "exploration and analysis of large quantities of data to discover unknown or hidden, comprehensible and actionable information and use it to make business decisions and to support their implementation, including formulating tactical and strategic initiatives and measuring their success."[3]

You use techniques such as decision trees, scoring, association analysis, neural networks, and so on. Data mining is the process of information discovery, classification, decision making, and predicting behavior. For example, data mining can be based on consumer transactions to find yet unknown product correlations (such as the famous diaper/beer sales relationship).

Business intelligence (BI)

BI is a general term covering all services, techniques, and tools that support business decisions based on information technology. BI is a subsystem layer providing the applications and technologies to accommodate reporting, query, forecasting, dashboarding, and analysis to business users using data from DW. BI assumes a data warehouse, Operational Data Store or Data Mart exists with timely, trusted information. These provides the underlying data foundation required to support BI. The data warehouse layer and the BI application layer together become the data warehouse/ BI or decision support system of the enterprise. A DW/BI system provides the executives and line managers with information to help and support their decisions. This helps to sustain and improve business. For more information refer to *From Multiplatform Operational Data to Data Warehousing and Business Intelligence*, SG24-5174. For more information, visit the IBM BI site:

http://www.ibm.com/BI

Operational BI

For improved decision making and to enable a competitive advantage, the need for more current information continues to grow. Most companies are expending effort to satisfy this need. Operational BI provide real-time information to support decisions of business users. This includes real-time analyzing, reporting, and monitoring of ongoing organization operations.

[3] Berry, J.A.M., and Linoff, G. (1997), Data mining techniques-for marketing, sales and customer support, New York: Wiley

There are two primary, and different, functional areas of any company where BI is required:

- Business strategy and planning

  This area uses traditional BI. It is concerned with more strategic issues (such as what products to make, where to sell them, how and where to distribute them, mergers and acquisitions, and how to maximize profit). To do this they need what has been termed strategic, or informational, BI.

- Business operations

  This area focuses on the tactical day-to-day operations of the company. It is typically focused heavily on developing and executing efficient business processes. The goal is to produce and deliver products quickly at the lowest cost possible, and to maintain good client satisfaction. To do this they need what has been termed *tactical*, or *operational business intelligence*.

Such help in decision making can be useful for frontline managers. But for this to happen there is a need for operational data with relevant history to be available with little delay to the frontline operational managers. Operational data store could be used to supply such data. An example of this information in a bank could be for a front line manager to be better prepared in cross-selling other products to a customer, by knowing about recent transactions of that customer. It could also be to take the lifetime value of a customer and provide free overdraft protection up to a certain dollar value. In large banks in the United States for example, it is mandatory to look for suspicious money laundering activities by looking at transaction level data. This rich repository of transaction data could potentially be used for operational BI enablement. For more information see 2.6.1, “Operational BI” on page 18.

Figure 2-1 shows components of a complete DW/BI solution grouped in layers. Data flows through ETL processes into data warehouses and data marts. From here, it is accessed by users with special BI tools, or with Web browsers or e-mail.

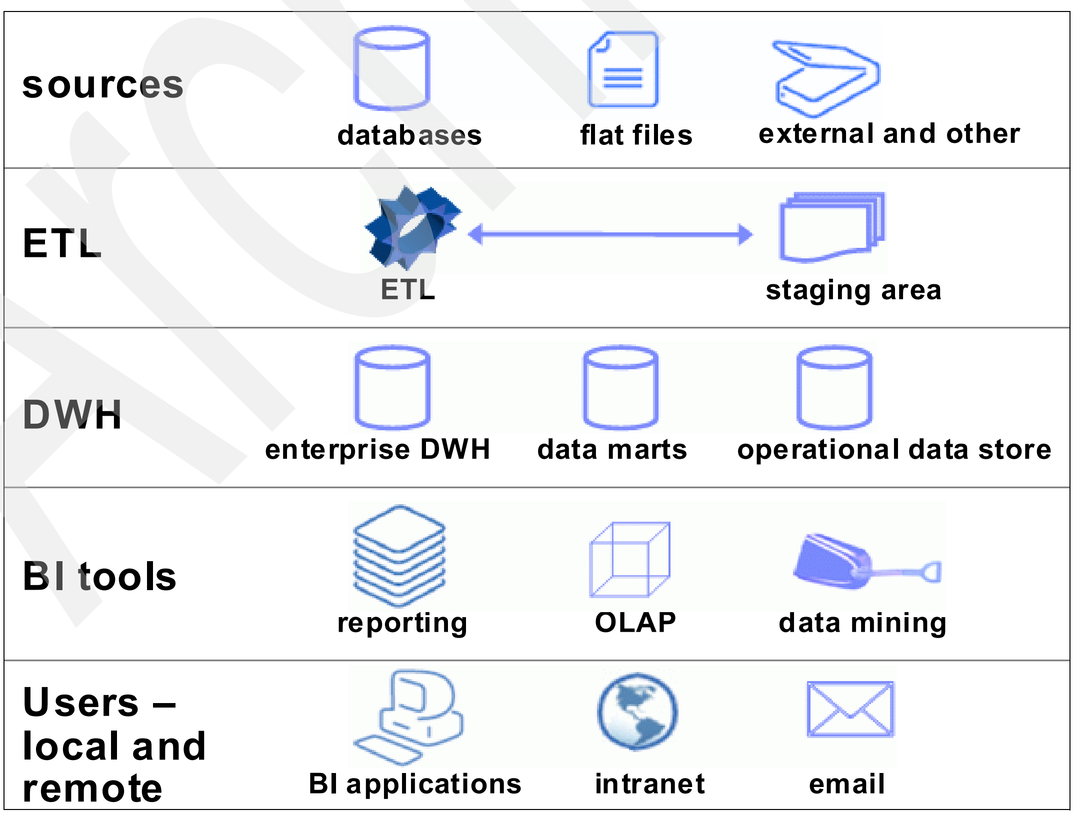


Figure 2-1 DW/BI components in logical layers

2.3 Data modeling in a data warehouse

To look at the data, you need a data model. A good example is the widely used entity relationship (ER) model, which is perhaps the most traditional approach, with subject areas and data granularity and metadata. There are predefined data models that can be purchased to use as a starting point. For banking, for example, IBM offers the the Financial Markets Industry Models, which provides an overall analysis framework for handling risk management, regulatory compliance, and so forth. Within the Financial Markets data models is the Financial Markets Data Warehouse (FMDW), a data management toolkit designed to assist financial institutions in building warehousing solutions, both analytical and operational. As any data warehouse data model should, the FMDW provides the blueprint for a single consistent enterprise view of the data. Such a prepackaged industry model can accelerate your goal for your own data warehouse.

If you have already built a data warehouse, you may be bringing in more source systems with a wider variety of data. For example, you may have loan information in the data warehouse but not lease information. You might also need marketing campaign analysis information. You can use such models for accelerating the next releases of your data warehouse. IBM offers models for various industries, developed by IBM service providers working with customers around the world. The models reflect best practices and the collective experience acquired from those engagements. For IBM industry models, visit the following Web page:

http://www-01.ibm.com/software/data/ips/products/industrymodels/

The data model in a warehouse context can be challenging because of the involvement of the time component, which makes it a temporal-ER model. Section 2.3.1, “Data warehouse data modeling styles” on page 10 and 2.3.2, “Multi-dimensional data model” on page 10 look at the different data modeling styles that are available.

2.3.1 Data warehouse data modeling styles

Over time, relationships between data can change. For instance, if you want only the latest information for a person, you could put their name and marital status in that entity. But if you want a history, you have to enter a time component. There are several different styles for capturing these changes:

- Cumulative snapshots

  Cumulative snapshots are a series of snapshots of the source data.

- Cumulative snapshots with rolling summarization

  This is where older snapshot information can be consolidated into a lower level of detail.

- Continuous temporal model

  The continuous temporal model is where data changes that are important to business are captured without losing changes.

For more information about modeling styles, refer to *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637.

2.3.2 Multi-dimensional data model

To describe multi-dimensional data modeling, the following elements must be defined. Multi-dimensional data modeling can be thought of as the resolution of a many-to-many relationship across all of the contextual dimensions for which the study of the measures is of interest.

Measure

A measure is a data item that information analysts use in their queries to measure business performance. This measures include quantities, amounts, durations, and so on. For instance, the number of policies sold by an insurance company is a measure.

Dimension

Dimensions provide the contextual background for the measures. Usually measures have multiple dimensions to consider. Good examples of dimensions are geography, time, or age. Each dimension usually has its own dimension table.

Fact

A fact is a collection of related measures taken together with their associated dimensions and are represented by the dimension keys. A fact should be of interest to an information analyst. Facts usually resides in fact tables that are connected to corresponding dimension tables.

Grain

A grain is the lowest level of detail available within a dimension.

Granularity

The granularity of a measure is determined by the combination of the grains of its dimensions.

Dimension hierarchy

Dimensions can consist of one or more alternate hierarchies. Each dimension hierarchy can include several aggregation levels.

Multi-dimensional data modeling: Star model schema

This is a representation of a multi-dimensional model design. They are mechanisms to separate facts and dimensions into separate tables. Each table is related to another table with a primary key/foreign key relationship. For example, you may have a Date dimension in which the week, month, and year are all contained. For a year you generally have 365 days. If you have measures that you need to examine for 10-year period, then you would have approximately 3650 rows, with one date for each day in this date dimension table. Each day would have additional detail noting if it is a holiday and which days of the week it is (Monday, Tuesday, and so forth).

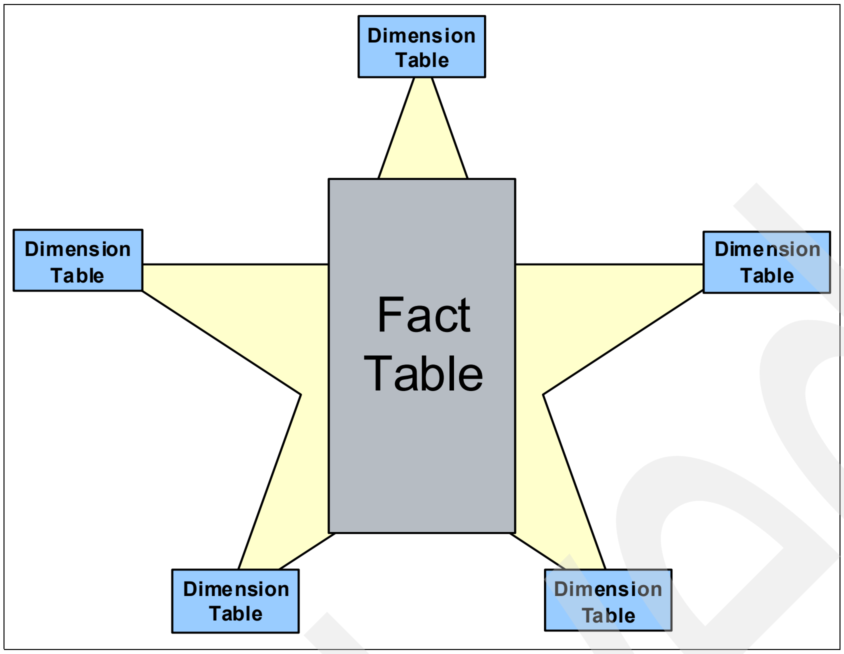


Figure 2-2 Star schema

Multi-dimensional data modeling: Snowflake model

A snowflake schema is a variation on the star schema, in which large dimension tables are normalized into more multiple tables. Dimensions with hierarchies can be decomposed into a snowflake structure when you want to avoid joins to big dimension tables when you are using an aggregate of the fact table. For example, if you have brand information that you want to separate from a product dimension table, you can create a brand snowflake that consists of a single row for each brand and that contains significantly fewer rows than the product dimension table.

Slowly changing dimensions (SCD)

Most dimensions change slowly. For example, if you have a customer dimension that contains name, gender, marital status, and so on, then it will change slowly. When the dimension changes, Ralph Kimball[4] considers the following design choices and implementation techniques:

- SCD Type 1

  Choose SCD Type 1 when you can choose to overwrite the existing information in the dimension with the new information. For example, if the marital status changes, then you just overwrite the marital status losing the older status. There are no key changes to worry about.

- SCD Type 2

  Choose SCD Type 2 when you intend to keep the older customer record for use with data until the day it changes. On that day, you will create a new customer record with the changed information, which will be tied to the fact table from then on. You need surrogate or artificial keys that can be generated for these two same customer records in the customer dimension table.

[4] http://www.kimballgroup.com/

- SCD Type 3

  In this type, we keep the original attribute information and the most current information, losing anything that happens in between. For example, if the marital status changes, then we have the first marital status that we knew of in the Original Marital Status field, and the most recent marital status in the Current Marital Status field. We would not overwrite the Original Marital status field with any future changes, but would overwrite the Current Marital Status value. For more information regarding data definitions, see *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637.

2.4 Market drivers and challenges

Data warehouse systems are becoming increasingly critical in strategic plans of organizations. One of the key areas driving this is the need to invest in performance management and link corporate strategy and initiatives with metrics, queries, and reporting.

Corporate performance management implementations, along with data governance controls, allow an organization to fulfill the following goals:

- Have realistic goals based on real information
- Adapt goals intelligently when required and focus on the outcome
- Align individual’s goals to the strategic goals of the organization
- Provide communication and accountability
- Measure progress against key performance indicators and publish results in a central location where it is available for everyone to provide feedback
- Gain an understanding of the business and what drives it.

New capabilities continue to be added to BI implementations. An example of this is the addition of real-time updates of operational data within an ODS or similar, further allowing users to make use of current information.

Growth within the BI industry is also linked to the challenges organizations are facing:

- Increase employee productivity
- Improve business processes
- Better understand and meet customer expectations
- Manage risk and compliance
- Improve operational efficiency
- Manage ongoing cost pressures
- Promote the use of existing information in the decision-making process

In addition to the business challenges, technical challenges also arise:

- Too much information and not knowing which is most important. The amount of information is growing and new types are being introduced.
- Lack of data integration. Information is scattered throughout the organization sometimes in separate silos.
- Lack of appropriate technical skills available.
- Real-time access to information.
- Query performance optimization.
- The need to integrate structured and unstructured data sources.
- Lack of trust in data sources.

- The need for rapid deployment of new systems.
- Scalability of systems.
- The need to provide extended search capabilities across all enterprise data.
- The need to create master data sets of information important to the business. For example, customers, products, and suppliers.
- Lack of agility in regard to inflexible IT systems. Employees spending too much time on managing change in these systems rather than doing more strategic tasks.
- Lack of self service BI query, reports, and analysis.
- Long update times of scorecards and dashboards from ODS in a traditional BI implementations.

2.5 BI technology and evolution

Let's take a brief look about where BI has come from and where it is going.

2.5.1 DB2 Data Warehousing evolution

DB2 for z/OS has played a role since the beginning of the data warehousing era. Many customers implemented their first warehouse on this platform. DB2 for z/OS was first released in 1985. IMS™ was well proven for transactional systems while DB2 was at first used for decision support systems. After the initial release of DB2 for z/OS, there was significant call to improve its functionality in support of Online Transaction Processing (OLTP). IBM continues to improve DB2 as an OLTP and BI platform, expanding the BI functionality on DB2 for z/OS to make warehousing efforts easier along with OLTP. DB2 9 for z/OS introduced significant data warehouse and BI functionality. The introduction of these features has encouraged further growth. More information about DB2 z/OS data warehouse capabilities is described in Chapter 3, "Why System z and IBM storage for Data Warehousing" on page 21.

The message and concepts of BI have not changed over time. By applying processes and methodologies against corporate data and turning it into useful information, organizations are able to gain insight and support their decision-making process. The maturity of the technology and functionality that can be used to help apply the necessary processes and methodologies has changed, as has the output that BI implementations can deliver to organizations.

Data warehouses extract data from operational storage through batch processes executed within maintenance windows. Today, BI systems incorporate large data warehouses that are consolidated with near real-time operational data, and support message-based integration of data. Real-time data feeds mean that a database is being updated in real time while it is being queried, similar to the transactional database. The new generation of data warehouse can be called Dynamic Warehousing.

Traditional and new breeds of BI applications access the Dynamic Warehouse, using both historical and operational data, to provide information to the user at the time they require while performing their tasks. The time available to provide this information has become shorter, because many organizations need to monitor their operational data and react to certain events as soon as they are triggered. For example, events can be monitored to identify fraud as close to real time as possible.

In addition, the number of users requiring information from the data warehouse environment has increased. BI and data warehousing used to be for the executive officers of an organization. This is no longer true. The technology and functionality available now ensures that BI can be applied at all levels within an organization. This has created pressure on conflicting workloads, so application solutions need to ensure the right balance is found between performance and latency.

Data warehouses now address the following workloads:

- Increasing number of reports being developed throughout an organization
- Increasing number of users wanting to perform ad-hoc queries
- Near real-time data loads
- New functionality being added to operational applications that provide BI capability against OLTP information
- Storage of information to meet the demands of regulatory legislation

The core characteristics of mainframes provide a good platform for processing the heavy workload of BI and performing the role of the key database server. For many organizations that hold their corporate information in VSAM, IMS, or DB2 for z/OS data stores, deciding to put their data warehouse on System z has benefits.

Enterprises that have an existing DB2 for OLTP in place and already use System z as their data serving platform are positioned to consolidate business-relevant data marts on the System z platform. DB2 for z/OS exploits the System z platform strengths that have been there from the beginning. In addition, DB2 for z/OS continues to be enhanced, specifically for data warehousing and BI functionality. This is discussed in Chapter 3, “Why System z and IBM storage for Data Warehousing” on page 21.

2.5.2 Current BI trends

New innovative functionality and technology is being introduced into the BI arena. We have discussed real-time data feeds as one example. Other examples of current trends in BI are as follows:

- Enterprise search

  New search tools allow access to information everywhere, providing enterprise search capability.

- BI Web services

  Web services that allow collaborative and analytic services, aiding incremental application development in a BI environment.

- Embedded analytics

  Prebuilt service-oriented components that can be integrated into operational applications and Web pages to become part of existing business processes. They combine current operational information with historical information from the data warehouse.

- Event driven alerts

  Allows monitoring of key business events. Uses push technology to notify of a change in these events.

- Analysis of unstructured Information

  Organizations have discovered that their unstructured data can contain useful information. Unstructured formats can include voice-mail, images, text, e-mails, and documents. New requirements include integrating structured and unstructured data.

- Virtual BI

  Virtual data marts and operational data stores can provide real-time information across both the current operational data and the historic information held in the data warehouse.

- Dynamic warehousing and operational BI

  Traditional warehousing involved query and reporting of what happened in the past. Long ago, the warehouse would be loaded and then queried for a month. The interval became weekly, then every night, and now business users need data up to the minute or second.

  Dynamic warehousing is an approach to implementing information on demand in regard to real-time information and using information. Operational BI applications react to current operational activities and offer real-time analysis whenever quick decision-making is necessary.

- Data warehouse industry models

  Provide for a faster implementation of data warehouse and BI solutions. IBM provides solutions in areas such as banking, insurance, telecommunications, retail, and health.

2.5.3 BI evolution and maturity

In many organizations the maturity path towards developing an enterprise BI solution may have included a number of stages over time with various outcomes occurring along the way.

Early data warehouse efforts focused on querying and reporting of financial and sales information. The next wave introduced technologies such as Online Analytical Processing (OLAP) and data mining. These implementations were useful for the analysis of the past. As BI has matured, the value that is returned to an organization, and the efficiency and effectiveness of the solutions have also increased. Implementations have become useful for analyzing the present and performing predictive analysis. BI components evolved as follows:

- Static reports and spreadsheets

  The number of reports throughout an organization increased dramatically. Sometimes the same reports were being developed by different people. They may have involved manual effort and had no centrally controlled definition of business rules or metadata, potentially causing multiple versions of the truth.

- Parameter reports

  Parameter reports were able to provide multiple reports within one, reducing the number of reports required to be developed. Traditionally, further analysis was limited to certain select analysts.

- Implementation of OLAP

  The concept of providing slice and dice functionality and summary information became popular. OLAP cubes allowed multi-dimensional analysis of data within a predetermined cube data format. Cubes were created as required and were sometimes uncontrolled, causing lack of integration. Data quality within the source data may not have been addressed sufficiently when building cubes and data quality issues caused problems with maintaining these cubes, such as different versions of hierarchies.

- Traditional warehousing and data integration

  Data integration exercises became important. Traditional data warehouses and data marts were ways to ensure corporate information could be integrated into a single reporting environment. The data warehouse environment proved useful for historical analysis but was not able to provide operational reporting requirements due to the time it took for data to travel from the OLTP source systems into the warehouse environment. (The term *traditional* has been used to distinguish from the term *Dynamic warehousing*)

- Ad-hoc queries

  Self-service technology allowed executives to perform some of their own ad-hoc queries against data within data marts or operational repositories.

- Corporate performance management (CPM)

  Performance management and scorecard initiatives become popular. These allowed the monitoring and management of an organization's performance by linking key indicators to the organization's strategies and goals.

- Data mining and analytics

  Complex data mining initiatives required specialists to build and train appropriate models.

2.6 Information on demand

Information management has traditionally meant centralized data repositories with accurate information that can support high performance and high availability. Although this remains a requirement, for information consumers the contents of these repositories have become harder to access. Some of the reasons why this has occurred are mentioned in 2.4, "Market drivers and challenges" on page 13.

Innovative applications allowing Information on Demand and information as a service capabilities can be applied against selective business processes to move beyond the traditional approaches of integrating and accessing information, allowing information to become a strategic asset that can be used to make better decisions. This is achieved by implementing new technologies such as SOA or master data management. Analyzing historical and real-time information helps unearth insightful relationships, again leading to better business decisions and business operations.

IBM is working for a smarter planet, with new intelligence, the ability to work smarter, to be more efficient and to have a more dynamic infrastructure. See the following articles for more information:

http://www.ibm.com/ibm/ideasfromibm/us/smartplanet/20081106/index.shtml
http://www.ibm.com/ibm/ideasfromibm/us/smartplanet/20081106/sjp_speech.shtml
http://www.ibm.com/innovation/us/think/

The IBM vision for information management is described as Information on Demand and the Information Agenda. Resources are scarce. Costs are high. Information is in silos. The time to react is short. You need a tightly integrated information infrastructure to retain a competitive advantage. The Information Agenda approach accelerates your organization's ability to share and deliver trusted information across all applications and processes, getting you started on your journey to Information on Demand.

The IBM Information Agenda approach has a proven track record of helping companies respond and adapt to unpredictable, up-to-the-minute changes in information, whether it's on a global level, or the next cube over. Using IBM's best-in-class software and consulting services, this approach is designed to help your business develop a customized implementation roadmap in a matter of weeks. This solution can also help you reduce IT spending by using existing investments. See the following Web page for more information:

http://www.ibm.com/software/data/information-agenda/

2.6.1 Operational BI

Operational BI refers to the methodologies, processes, and applications used to report on and analyze operational business performance.

Traditional BI was mostly executives analyzing historical data and performing strategic analysis on general questions. For example, how an organization has performed against certain criteria over a given time period. Operational Intelligence or Operational BI has moved BI into all levels of an organization, allowing not just executives, but everyone within the organization to incorporate proactive analysis into their operational processes and applications. This analysis is more specific and more common than traditional BI analysis, as it covers questions and decisions that are part of the organizations day to day business processes. An example would include “What discount can I offer my customers based on their past ordering history?”

Operational BI is all about optimizing an organization’s performance in the present and not the past, and provides the organization with a near real-time view using dashboards, metrics, and reports. Traditional BI and Operational BI are different, but do complement each other. The traditional data warehouse can become a source of information to provide historical data to Operational BI applications.

Components of an Operational BI implementation should include the following information:

- Historical data

  Self-service dashboards with links to traditional analysis and reports
- Real-time data

  Information should also include near real-time data that is also provided to the dashboards and reports
- Business activity monitoring for complex event and data processing from both operational data sources and the data warehouse environment.

  Key events are distributed to those required using push technology, for example, e-mail notification
- Information on Demand capabilities

  These capabilities are explained further in *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637.

> **Operational Intelligence best practices and lessons learned:** These practices and lessons are helpful when starting an operational intelligence initiative:
>
> - Establish clear and continued executive sponsorship
> - Develop a shared “Point of View”/ organizational perspective on operational intelligence
> - Start with a project that is strategically important
> - Focus on data governance and stewardship across the organization
> - Provide trusted data sources and solution flexibility
> - Use common process models and reference architecture
> - Apply integrated methods and common taxonomy, language
> - Deliver phased, value-driven implementations
> - Deploy scalable, flexible technical infrastructure
> - Capture and share best practices and lessons learned

Data warehouses have been implemented on DB2 for System z servers largely because of the traditional strengths that the System z platform provides, such as unparalleled throughput, continuous availability, and unlimited scalability.

2.6.2 Applying IOD with Dynamic Warehousing

Dynamic warehousing is a new approach that can be used to implement Information on Demand initiatives.

Dynamic warehousing addresses the primary business challenges organizations face today, which requires the ability to deliver the right information to the right people at the right time to use information and enable effective business decisions. It is about Information on Demand to optimize real-time processes.

Where traditional warehousing involved query and reporting of what happened in the past, dynamic warehousing is a approach to implementing Information on Demand in regard to real-time information and using information.

Dynamic warehousing requires the following features:

- Support for real-time access to aggregated, cleansed information, which can be delivered in the context of the activities and processes being performed
- Analytics that can be used as part of a business process
- The ability to incorporate knowledge from unstructured information
- A complete set of integrated capabilities that extend beyond the warehouse to enable Information on Demand

The distinction between data warehousing and online transaction processing is blurring. Data warehousing and analytical applications are accessing operational or near-real-time data. Transactions have become more complex to provide better interaction and productivity for people. Dynamic warehousing has capabilities and strengths on all IBM platforms. The traditional mainframe strengths for consistency with operational data, high security, and continuous availability match well with dynamic warehousing. For more information about Information on Demand, refer to *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637.

3

Why System z and IBM storage for Data Warehousing

This chapter explains why and when implementing a data warehouse on System z is the right choice. Customers already running their transactional system on DB2 for z/OS have multiple options if they want to build their warehouse environment.

Originally, DB2 for z/OS was positioned as a business intelligence (BI) platform, but then in response to customer demand focused more on online transactional processing (OLTP). In the past couple of years however, customers encouraged IBM to expand BI functions on DB2 for z/OS as well, to use System z advantages for their data warehouse, too.

Additionally, new requirements for modern applications, such as using historical data from a data warehouse for tactical decision-making in day-to-day transactions, make System z an interesting and competitive hosting platform for data warehouses.

The close relationship between DB2, z/OS, and the System z platform is unique in the industry. It delivers world-class performance, scalability, and availability.

References

For more information about the features and advantages of IBM System z, refer to *IBM System z Strengths and Values*, SG24-7333.

For prerequisite information about data warehousing on z/OS, refer to *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637.

3.1 New challenges for data warehouse solutions

Enterprises are increasingly harnessing the power of BI to gain a competitive edge. Today's BI systems incorporate large data warehouses that are consolidated with near-real-time operational data stores (ODS) and are continuously updated from multiple sources, such as financial, marketing, and inventory databases. Often, thousands of users access a data warehouse with various BI applications. These applications analyze and synthesize data into real-time information that supports fast and informed decisions.

These applications must be capable of using all types of information, including unstructured content such as call center logs, technical notes, contracts, or call logs. Data warehouses must serve the expanding needs of different types of applications. As a result, data warehouses must also support the varying service level demands of these different applications. There will be a combination of mission critical operational applications combined with analytics insights, that require real-time responsiveness. Also the current traditional back-office reporting and analysis for strategic and tactical planning purposes will still be needed. Both together will lead to increasingly mixed workload environments. And this is further complicated by rising data volumes, continuously expanding amounts of historical data and the growing number of users, which is causing requests for information to become more numerous and sophisticated. Current BI solution proposals need to consider the challenges arising from the mix of the following four workload types:

- Continuous, near-real-time data loading (similar to an OLTP workload)
- A large number of standard reports
- An increasing number of true ad hoc query users
- An increasing level of embedded analytics and BI-oriented functionality in OLTP applications

These four workload types are increasingly top challenges for organizations beyond the growing size of the data warehouse. Mixed workload performance is well on its way to becoming the single most important differentiator issue in data warehousing.

3.2 Data warehousing with System z

The long standing success record of workload management capabilities makes System z an ideal platform to run a data warehouse workload. In this section we outline the main reasons that make System z an ideal platform to run data warehouse workloads to handle emerging BI and Dynamic Warehousing requirements. For an introduction to the unique terms on the System z platform, see the following Web page:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.intro/dsnitk12.pdf

3.2.1 Availability and scalability

Access to the data warehouse in a BI solution is moving from the back office to the front office making the BI solution a business critical application that requires the same kind of availability and security as OLTP systems. Frequently, the data warehouse is large, requiring careful management to ensure that business information is available when needed and that access to that data is responsive but secure. The System z platform and DB2 9 for z/OS provide an excellent infrastructure for these requirements making System z the platform of choice for mission-critical processing.

The IBM System z hardware, z/OS operating system, and DB2 for z/OS are designed with reliability characteristics including self monitoring, redundancy, self healing, dynamic configuration, and management. For example, in DB2 you are able to make database changes, such as adding a partition without an outage.

As businesses grow, their data processing needs also grow. Business ventures, such as mergers, acquisitions, new services, or new government regulations, can accelerate how quickly the data processing needs of the business grow. As rapid growth occurs, companies need a way to scale their business successfully. Parallel Sysplex® clustering technology and DB2 Data Sharing is the answer to availability and scalability. A Sysplex is a group of z/OS systems that communicate and cooperate with one another using specialized hardware and software. They are connected and synchronized through a Sysplex Timer® or System z Server Time Protocol (STP), and enterprise systems connection (ESCON®) or fiber connection (FICON®) channels. A Parallel Sysplex is a Sysplex that uses one or more coupling facilities (CFs), which provide high-speed caching, list processing, and lock processing for any applications on the Sysplex. For information about Parallel Sysplex technology and benefits, see the following Web page:

http://www.ibm.com/systems/z/resiliency/parsys.html

Figure 3-1 illustrates a Parallel Sysplex. A Parallel Sysplex can include CPCs of different generations (for example, an IBM System z10™, a System z9® BC or EC, and an IBM zSeries® System z890 or z990).

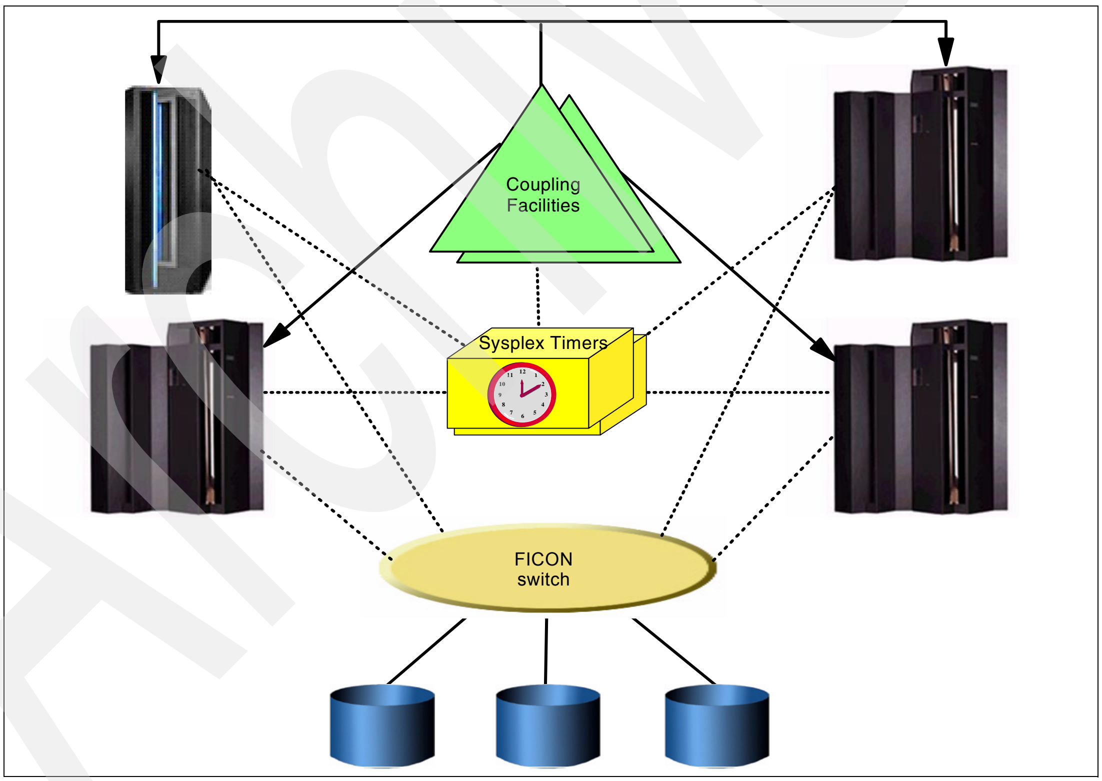


Figure 3-1 Parallel Sysplex Architecture

The Parallel Sysplex can grow incrementally without sacrificing performance. Parallel Sysplex architecture is designed to integrate up to 32 systems in one cluster. In a shared-disk cluster, each system is a member of the cluster and has access to shared data. For more information about this subject, see *The Business Value of DB2 UDB for z/OS*, SG24-6763.

A collection of one or more DB2 subsystems that share DB2 data is called a *data sharing group*. DB2 subsystems that access shared DB2 data must belong to a data sharing group. A DB2 subsystem that belongs to a data sharing group is a member of that group. Each member can belong to one, and only one, data sharing group. All members of a data sharing group share the same DB2 catalog and directory, and all members must reside in the same Parallel Sysplex, Refer to Figure 3-2 for DB2 data sharing architecture. Currently, the maximum number of members in a data sharing group is 32.The DB2 data sharing design gives businesses the ability to add new DB2 subsystems into a data sharing group, or cluster, as the need arises and without disruption. It provides the ability to do rolling upgrades of service or versions of the software stack without any application outage.

For more information about DB2 Data Sharing, refer to *DB2 for z/OS: Data Sharing in a Nutshell*, SG24-7322

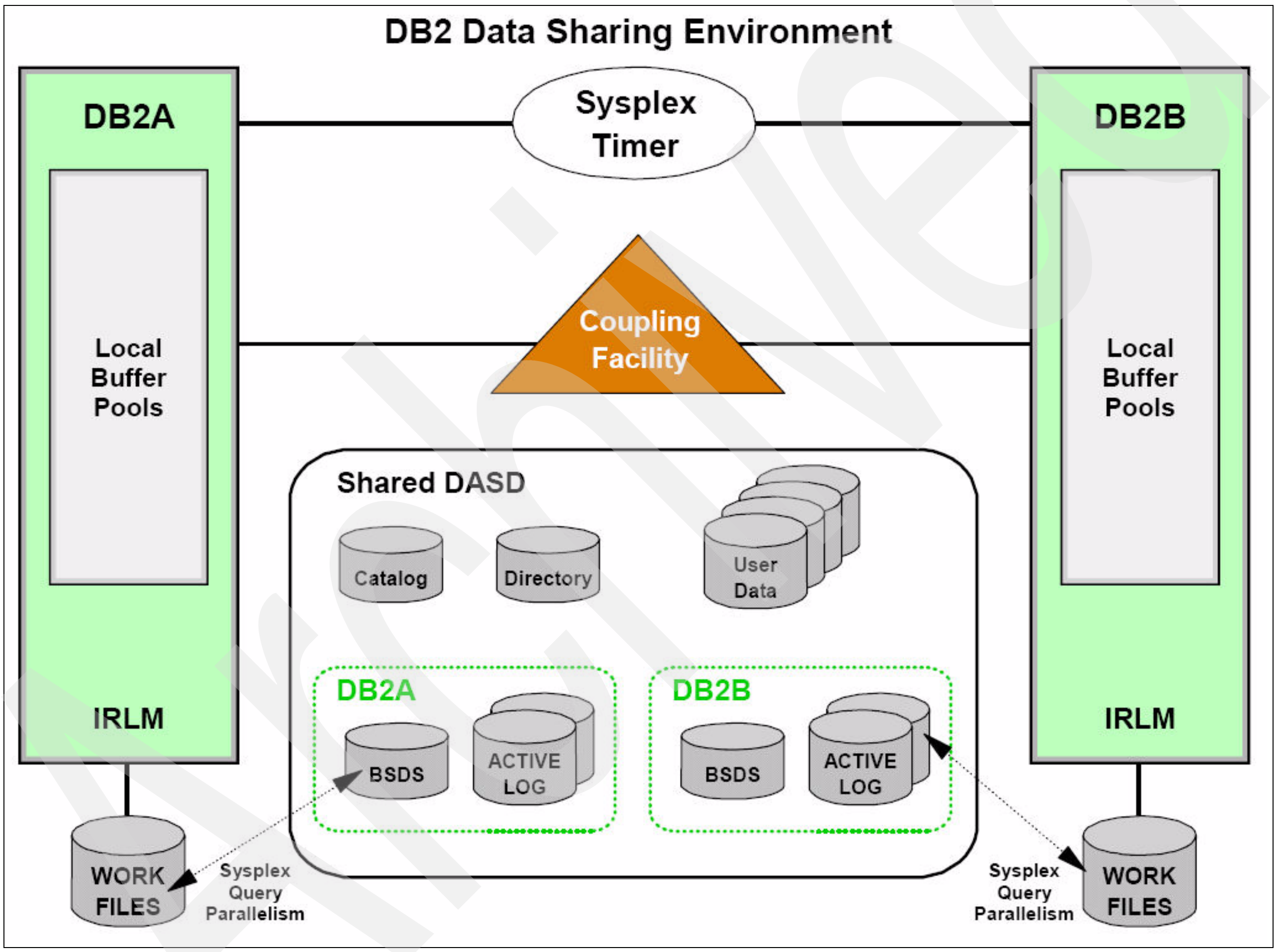


Figure 3-2 DB2 data sharing

Automatic Restart Manager (ARM), a component of z/OS, enables fast recovery of subsystems that might hold critical resources at the time of failure. If other instances of the subsystem in the Parallel Sysplex need any of these critical resources, fast recovery will make these resources available more quickly. Even though automation packages are used today to restart the subsystem to resolve such deadlocks, ARM can be activated closer to the time of failure.

Parallel Sysplex and DB2 data sharing Implementation caters to growing OLTP or BI needs, with the high availability and scalability.

3.2.2 Workload management

The Workload Manager (WLM) component of z/OS has proven its worth in many studies and demonstrations and in every-day work being done in systems around the world with its ability to maximize use of available resources. It manages widely varying workloads efficiently and effectively, allowing you to use the systems resources available. This means you can run your data warehouse workload together with the transactional (OLTP) workload on the same DB2 subsystem or different DB2 subsystems on the same system.

With WLM, a particular element of work can be given an initial priority based on business needs. More importantly, over time, the priority of a given element of work can be altered based on changes in business needs as expressed in the WLM policy. DB2 for z/OS works hand-in-hand with WLM to ensure that these priority alterations take effect immediately with respect to query processing, regardless of how that query has been parallelized. Capacity can be brought to bear whenever it is needed. For more information about data sharing and WLM on z/OS, refer to these IBM Redbooks publications:

- *Workload Management for DB2 Data Warehouse*, REDP-392727
- *System Programmer's Guide to: Workload Manager*, SG24-6472

Table 3-1 illustrates the impact of workload management in a mixed workload environment. The first row, as a baseline, shows times if short running and long running queries are run sequentially. Without being impacted by each other, the short running query takes 1.5 seconds, while the long running query takes 147 seconds. In the second line, both query types are run in parallel. Because both queries now compete against each other and try to get resources from the system, the query response time will suffer. It is important to note that the time for short running queries increases significantly if run concurrently with long running queries. In the example, query times go up from 1.5 seconds to 6.7 seconds. This might become an issue if the short running queries are run as part of a transactional application where users would then experience responses more than four times slower for their requests. The third line illustrates changes in response time with WLM on z/OS. The concept of period aging automatically lowers the priority of long running queries and thereby keeps short running queries running at higher priority. Consequently, long running queries do not impact short running queries too much and we see a minor performance decrease for these.

Table 3-1 Impact of WLM to a mixed workload environment

	Short running queries	**Long running queries**
Sequential processing	1.5	147
Parallel processing with limited workload management	6.7	159
Parallel with z/OS WLM	1.9	158

3.2.3 Hardware data compression

DB2 (since version 3) has provided a hardware-assisted option to compress table spaces so that they occupy less disk space. Compression and decompression of data is accomplished on the System z environment through a hardware instruction that makes it faster than the software-based compression algorithms used by other databases systems on other platforms. As each generation of System z processor gets faster, the compression feature also gets faster.

Counter-intuitively, compressing data can actually reduce elapsed time of most data warehouse type queries. DB2 for z/OS compresses the rows on a page, so that each data page is full of compressed rows. It uses the hardware instruction along with a data dictionary to give the most efficient compression available. The compressed data can also be encrypted, thereby saving space and implementing security requirements at the same time.

With a 50% compression rate, a compressed page contains twice the rows that an uncompressed page would contain. This means that each I/O retrieves twice as much compressed data as it would if the data were uncompressed. The data remains compressed in the buffer pool, which means that DB2 for z/OS can cache twice as much compressed data in its buffer pool as it would if the data was uncompressed. When data is modified in a row that is compressed, the information logged about that data change is also compressed, thus reducing log volume.

Not all data on a compressed page is decompressed, just the rows needed by the application. This, combined with the use of a hardware instruction to perform the decompression, serves to limit the amount of additional CPU needed to access compressed data.

The larger amount of data retrieved in each I/O gets compounded with the DB2 9 for z/OS increased prefetch quantities. This provides significant elapsed time reductions for all types of sequential processes, including the typical BI queries that make use of table scans and index range scans. This includes sequential processes for utility access, providing benefits in terms of faster reorganizations, faster unloads, and faster recovery.

3.2.4 Regulatory compliance

Regulations like the Sarbanes-Oxley Act (SOX), Basel II, Data Protection Act (UK), and the USA Patriot Act, were created to protect investors' interests, to avoid fraud and to improve financial reporting. Companies need to comply with these regulations. Database administrators need to ensure that data is secure, access is controlled, changes are audited, and disaster recovery is in place. Regulations also emphasize the growing need to reproduce versions of data, applications, and even entire business states, which challenges companies to keep a long record of activities. Web sites such as www.commoncriteriaportal.org offer security products and documentation.

System z meets the highest industry security certifications. Encryption support is built-in, even at the hardware level. Authorization functionality is integral part of the operating system. Detection services prevent intrusions and record intrusion attempts. Network communication encryption follows the highest standards .For more information,*Securing DB2 and Implementing MLS on z/OS*, SG24-6480. IBM also offers the DB2 Audit Management Expert, a tool to help with auditing activities. For more information, refer to the following Web page:

http://www.ibm.com/software/data/db2imstools/db2tools/db2ame/

Consolidating warehouse data and operational data on one platform (such as System z) eases the effort to comply with regulation requirements. With fewer data sources, audit management effort is reduced.

3.2.5 Disaster recovery and data replication

With the emergence of BI and dynamic warehousing, the disaster recovery requirements for a DW environment are similar to that of OLTP. Therefore, it is important to consider disaster recovery scenarios before implementing a DW solution.

The following System z and DB2 for z/OS technologies provide some of the best disaster recover solutions in the industry:

- Copy Services

  See page 27.
- GDPS/XRC

  See page 33.
- GDPS/PPRC

  See page 33.
- GDPS/PPRC HyperSwap™

  See page 34.
- Backup and Recover System utility of DB2 for z/OS

  See page 35.

Copy Services

Copy Services is a collection of functions that provide disaster recovery, data migration, and data duplication functions. With the Copy Services functions, for example, you can create backup data with little or no disruption to your application, and you can back up your application data to the remote site for the disaster recovery. Customers using System z along with IBM DS8000™ storage can have following optional licensed copy services:

- IBM System Storage™ FlashCopy®, which is a point-in-time copy function
- Remote mirror and copy functions, which include the following services:
  - IBM System Storage Metro Mirror, previously known as synchronous PPRC
  - IBM System Storage Global Copy, previously known as PPRC Extended Distance
  - IBM System Storage Global Mirror, previously known as asynchronous PPRC
  - IBM System Storage Metro/Global Mirror, a 3-site solution to meet the most rigorous business resiliency needs
- z/OS Global Mirror, previously known as Extended Remote Copy (XRC)
- z/OS Metro/Global Mirror, a 3-site solution that combines z/OS Global Mirror and Metro Mirror

Many design characteristics of the DS8000 and its data copy and mirror capabilities and features contribute to the protection of your data, 24 hours a day and seven days a week.

FlashCopy

FlashCopy provides a point-in-time (PIT) copy of logical volumes with almost instant availability for the application of both the source and target volumes. Only a minimal interruption is required for the FlashCopy relationship to be established, so the copy operation can be initiated. The copy is then created by the IBM TotalStorage® Enterprise Storage Server® (ESS), with minimal impact on other ESS activities.

There are currently two FlashCopy versions available: version 1 and version 2. FlashCopy Version 2 supports all the previous FlashCopy version1 functions, plus several enhancements.

In System z environments, you can use the FlashCopy function to perform data set level copies of your data. You can use the copy with standard backup tools that are available in your environment to create backup copies on tape.

One of the main benefits of the FlashCopy function is that the point-in-time copy is immediately available for creating a backup of production data. The target volume is available for read and write processing so it can be used for testing or backup purposes. Data is physically copied from the source volume to the target volume using a background process. (A FlashCopy operation without a background copy is also possible, which allows only data that is modified on the source to be copied to the target volume.) The amount of time that it takes to complete the background copy depends on the following criteria:

- The amount of data being copied
- The number of background copy processes that are occurring
- The other activities that are occurring on the storage units

It takes only a few seconds to establish the FlashCopy relationships for tens to hundreds or more volume pairs. The copy is immediately available for both read and write access. In a 24x7 environment, the quickness of the FlashCopy operation allows use of FlashCopy in large environments, and to take multiple FlashCopies of the same volume for use with different applications.

A FlashCopy of data can be used to copy data to various systems, including a data warehouse, test system, backup system, or integration system. This avoids a performance impact on the production system.

Space-efficient FlashCopy SE

TPC-R can establish a FlashCopy relationship where the target volume is a DS8000 space-efficient volume. A space-efficient volume does not occupy physical capacity when it is created. Space is allocated when data is actually written to the volume. The amount of space that gets physically allocated is a function of the amount of data changes performed on a volume. The sum of all defined space-efficient volumes can be larger than the physical capacity available.

Because it does not require a target volume that is the exact size of the source volume, the FlashCopy SE feature increases the potential for an effective use of system storage. FlashCopy SE is generally meant for temporary copies. Unless the source data has little write activity, copy duration should not last longer than twenty-four hours. The best use of FlashCopy SE is when less than 20% of the source volume is updated over the life of the relationship. If performance on the source or target volumes is important, standard FlashCopy is strongly suggested.

The space-efficiency attribute for the target volumes is specified during the volume creation process. A space-efficient volume can be created from any extent pool that has space-efficient storage already created in it. Both the source and target volumes of any FlashCopy SE relationship should be on the same server cluster.

Space-efficient volumes are currently supported as FlashCopy target volumes only.

Once a space-efficient volume is specified as a FlashCopy target, the FlashCopy relationship becomes space-efficient. FlashCopy works the same way with a space-efficient volume as it does with a fully provisioned volume. In other words, all existing copy functions work with a space-efficient volume with the exception of the Background Copy function (not permitted with a space-efficient target) and the data set Level FlashCopy function. A miscalculation of the amount of copied data can cause the space-efficient repository to run out of space. If this happens, the FlashCopy relationship fails (that is, reads or writes to the target are prevented) and it is possible to withdraw the FlashCopy relationship to release the space.

As far as the FlashCopy source-target relationship is concerned, the FlashCopy SE target volume acts like a standard (fully provisioned) FlashCopy target volume, but implies that a full background copy is not in effect (the nocopy option is enforced), which makes sense because, if you have requirements for a full background copy, you have no intended need for space efficiency.

FlashCopy SE accomplishes space efficiency by pooling the physical storage requirements of many FlashCopy SE volumes into a common repository. A mapping structure is created to keep track of where the FlashCopy SE volume's data is physically located within the repository.

The FlashCopy SE target volume itself is a virtual volume and has no actual storage allocation, which is why it appears as transparent in the picture. However, FlashCopy SE is not free storage. The repository is an object within an extent pool. The repository is similar to a fully provisioned volume within the extent pool. The repository has a physical size and a logical size. The physical size of the repository is the amount of space that is allocated in the extent pool. The intended overall benefit of FlashCopy SE is that the repository is smaller than the full capacity of the FlashCopy source volumes.

Consider the common practice of creating a FlashCopy relationship (without background copy) to spin data off to tape, for example. In this instance, although the backup application is reading data from the FlashCopy target volume and sending it to tape, no data is actually copied to the FlashCopy target volume. Data is being read from the FlashCopy source volume and is being sent directly to tape. Because the standard FlashCopy target volume is fully provisioned (size is equal to or greater than that of the FlashCopy source volume), it could be argued that the FlashCopy target volume is a waste of valuable storage resources. FlashCopy SE was created to address exactly this kind of situation. FlashCopy SE gives you the capability to create a space efficient volume as the FlashCopy target.

Metro Mirror

Metro Mirror, previously known as Synchronous Peer-to-Peer Remote Copy (PPRC), provides real-time mirroring of logical volumes between two DS8000s that can be located up to 300 km from each other. It is a synchronous copy solution where write operations are completed on both copies (local and remote site) before they are considered complete. It is typically used for applications that cannot suffer any data loss in the event of a failure. As data is synchronously transferred, the distance between primary and secondary disk subsystems will determine the effect on application response time.

When the application performs a write update operation to a primary volume, the following events are completed:

- Write to primary volume (DS8000 cache and NVS).
- Write to secondary volume (DS8000 cache and NVS).
- Signal write complete from the secondary DS8000.
- Post I/O complete to host server.

The Fibre Channel connection between primary and secondary subsystems can be direct, through a switch, or through other supported distance solutions (for example, Dense Wave Division Multiplexor [DWDM]).

During normal Metro Mirror processing, the data on disk at the secondary site is an exact mirror of that at the primary site. During or after an error situation, this depends on the options specified for the pair and the path. Remember that any data still in buffers or processor memory is not yet on disk and so will not be mirrored to the secondary site. A disaster at the primary site will look like a power failure to the secondary site.

Metro Mirror is a hardware mirroring solution. A volume (or logical unit number [LUN]) is paired with a volume (or LUN) in the remote disk subsystem. As the size of the environment grows, so does the complexity of managing it. You must have a means for managing the pairs, ensuring that they are in duplex status, adding volume pairs as required, monitoring for error conditions, and more importantly, for managing data consistency across LSS and across disk subsystems.

Global Copy

Global Copy, previously known as Peer-to-Peer Remote Copy Extended Distance (PPRC-XD), copies data non-synchronously and over longer distances than is possible with Metro Mirror. When operating in Global Copy mode, the source volume sends a periodic, incremental copy of updated tracks to the target volume, instead of sending a constant stream of updates. This causes less impact to application writes for source volumes and less demand for bandwidth resources, while allowing a more flexible use of the available bandwidth and with no limitation in distance between sites.

Global Copy operates in these steps:

1. The host server makes a write I/O to the primary DS8000. The write is staged through cache and non-volatile storage (NVS).
2. The write returns as completed to the host server's application.
3. The primary DS8000 transmits the data in an asynchronous manner so that the updates are reflected on the secondary volumes. The updates are grouped in batches for efficient transmission.
4. The secondary DS8000 returns write completed to the primary DS8000 when the updates are secured in the secondary DS8000 cache and NVS. The primary DS8000 then resets its Global Copy change recording information.

The efficient extended distance mirroring technique of Global Copy is achieved with sophisticated algorithms. For example, if changed data is in the cache, then Global Copy sends only the changed sectors. There are also sophisticated queuing algorithms to schedule the processing to update the tracks for each volume and set the batches of updates to be transmitted.

Global Copy is a suggested solution for remote data copy, data migration, and offsite backup over continental distances without impacting application performance. It can be used for application recovery implementation if application I/O activity can be quiesced and non-zero data loss Recovery Point Objective (RPO) is admissible.

Other features are as follows:

- It can be used over continental distances with excellent application performance.

  The distances are only limited by the network and channel extenders capabilities.
- Application write operations do not have synchronous-like overheads.
- Fuzzy copy of data at the recovery site (sequence of dependent writes may not be respected at the recovery site).
- Recovery data can become a consistent point-in-time copy of the primary data, if appropriate application checkpoints are set to do global catch-ups.

  Pairs are synchronized with application group consistency.
- Synchronizations can be done more frequently, because of short catch-ups.

  RPO is still not zero, but it improves substantially.

Global Mirror

Global Mirror, previously known as Asynchronous PPRC, is a two-site long distance asynchronous remote copy technology. This solution integrates the Global Copy and FlashCopy technologies. With Global Mirror, the data that the host writes to the storage unit at the local site is asynchronously mirrored to the storage unit at the remote site. This way, a consistent copy of the data is automatically maintained on the storage unit at the remote site.

For a given pair of local and remote storage disk subsystems, a time stamp approach leads to consistent data at the remote storage disk subsystem. Using this approach, and by sorting the I/Os by their time stamps, the write I/Os can be applied at the remote disk subsystem in the same sequence as they arrived at the local disk subsystem. But when the application volumes are spread across multiple storage disk subsystems, this time stamp concept alone is not sufficient to replicate data and provide data consistency at the remote site. This additionally requires a Consistency Group concept. z/OS Global Mirror, formerly known as XRC, is an example of such a solution. It uses the concept of Consistency Groups combined with time stamped write I/Os for all involved primary volumes within a Consistency Group or session.

By its own, a non-synchronous technique like that of Global Copy provides no guarantee that the sequence of arrival of the applications write I/Os to the primary volumes is preserved at the secondary site. The order of dependent writes is not preserved at the secondary site.

Global Copy by itself as a non-synchronous data replication method does not provide data consistency at the secondary site.

Metro/Global Mirror

Metro/Global Mirror is a three-site, multi-purpose, replication solution for both System z and open systems data. Local site (site A) to intermediate site (site B) provides high availability replication using Metro Mirror, and intermediate site (site B) to remote site (site C) supports long distance disaster recovery replication with Global Mirror. Both Metro Mirror and Global Mirror are established replication solutions. Metro/Global Mirror combines Metro Mirror and Global Mirror to incorporate the best features of the two solutions.

- Metro Mirror
  - Synchronous operation supports zero data loss.
  - The opportunity to locate the intermediate site disk subsystems close to the local site disk subsystems allows use of intermediate site disk subsystems in a high availability configuration.

> **Note:** Metro Mirror is supported to a distance of 300 km but, when in a Metro/Global Mirror implementation, a shorter distance might be more appropriate in support of the high availability functionality.

- Global Mirror
  - Asynchronous operation supports long distance replication for disaster recovery.
  - Global Mirror methodology has no impact to applications at the local site.
  - It provides a recoverable, restartable, consistent image at the remote site with a Recovery Point Objective (RPO) typically in the 3–5 second range.

z/OS Global Mirror

z/OS Global Mirror, previously know as Extended Remote Copy (XRC), is a copy function available for the z/OS and OS/390® operating systems. It involves a System Data Mover (SDM) that is found only in OS/390 and z/OS. z/OS Global Mirror maintains a consistent copy of the data asynchronously at a remote location, and can be implemented over unlimited distances. It is a combined hardware and software solution that offers data integrity and data availability. It can be used as part of business continuance solutions for workload movement and for data migration. z/OS Global Mirror function is an optional licensed function.

z/OS Global Mirror is a software-centric remote copy implementation. A DFSMSdfp™ component called System Data Mover (SDM) will copy the writes issued to primary volumes to the secondary devices. Although the main z/OS Global Mirror implementation consists of host resident software, special z/OS Global Mirror support is required in the DS8000 that attaches the z/OS Global Mirror primary volumes.

It is suggested that your secondary storage subsystem be of similar performance to the primary storage subsystem. If not, you might experience production impact due to z/OS Global Mirror not being able to keep up with production data updates.

The SDM is part of the DFSMSdfp software, and must have connectivity to the primary volumes and to the secondary volumes. When primary systems write to the primary volumes, the SDM manages the process of copying those updates to the secondary volumes.

Multiple z/OS Global Mirror sessions can be in effect per z/OS (or OS/390) system (up to 20 with or without coupling them). Multiple instances of the SDM on separate z/OS images are also possible. Each SDM will have one z/OS Global Mirror session that is responsible for a group of volumes. The SDM maintains the update sequence consistency for the volumes participating in the z/OS Global Mirror session, across logical control units (LCUs) in the DS8000 and across DS8000s (as well as with other primary storage subsystems which support z/OS Global Mirror).

A key capability of z/OS Global Mirror is Coupled Extended Remote Copy (CXRC), which allows up to 14 z/OS Global Mirror sessions to be coupled together to guarantee that all volumes are consistent to the same time across all coupled z/OS Global Mirror sessions. An enhancement to CXRC provides the function of being able to cluster up to 13 z/OS Global Mirror coupled sessions within a single logical partition, thus allowing more z/OS Global Mirror sessions to be coupled to a master session.

z/OS Metro/Global Mirror

This mirroring capability implements z/OS Global Mirror to mirror primary site data to a location that is a long distance away, and also uses Metro Mirror to mirror primary site data to a location within the metropolitan area. This enables a z/OS 3-site high availability and disaster recovery solution for even greater protection from unplanned outages of the DS8000.

DS8000 Storage Complexes support z/OS Global Mirror on System z hosts. The z/OS Global Mirror function mirrors data on the DS8000 Storage Unit to a remote location for disaster recovery. It protects data consistency across all volumes that you have defined for mirroring. The volumes can reside on several different Storage Units. The z/OS Global Mirror function can mirror the volumes over several thousand kilometers from the source site to the target recovery site.

In a System z environment, the processor normally accesses the DS8000 storage subsystems on that local site. These disks are mirrored back to the intermediate site with Metro Mirror to another DS8000. At the same time, the local site disk has z/OS Global Mirror pairs established to the remote site to another DS8000, which can be at continental distances from the local Site.

We have covered for the following scenarios:

- Local site disaster

  Production can be switched to the intermediate site. This switch can occur in seconds to minutes with GDPS® type automation. Mirroring to the remote site will be suspended. If operation with the intermediate site disks is expected to be long term, the intermediate site can be mirrored to the remote site with the appropriate communications connectivity.

- Intermediate site disaster

  Local and remote sites continue to operate normally. When the intermediate site returns to normal operation, the intermediate site can be quickly resynchronized to the local site.

- Local site and intermediate site disasters

  The remote site can be brought up to the most recent consistent image, with some loss of inflight updates. FlashCopy consistent image disks to preserve the original recovery point. Production can be started at the remote site.

- Remote site disaster

  Local and intermediate sites continue to operate normally

GDPS/XRC

GDPS/XRC, an industry leading e-business availability solution available through IBM Global Services, is a multi-site solution that is designed to provide the capability to manage the remote copy configuration and storage subsystems, automate Parallel Sysplex operational tasks, and perform failure recovery from a single point of control, thereby helping to improve application availability.

Geographically Dispersed Parallel Sysplex™ (GDPS) is an integrated solution that manages all aspects of switching computer operation from one site to another, planned or unplanned. In a GDPS/XRC configuration, the SDM is placed outside the production sysplex. Normally it is located at the recovery site.

One subset of the GDPS solution is the Remote Copy Management Facility (RCMF) offering RCMF is an automated disk subsystem and remote copy management facility, with a high level user interface. This interface is implemented in the form of ISPF-like displays, and virtually eliminates the tedious and time consuming work with TSO commands. Managing XRC with RCMF could be the first step of a full GDPS implementation.

GDPS/PPRC

GDPS/PPRC is designed to manage and protect IT services by handling planned and unplanned exception conditions, and maintain full data integrity across multiple volumes and storage subsystems. By managing both planned and unplanned exception conditions, GDPS/PPRC can help maximize application availability and provide business continuity. GDPS is capable of providing the following functions:

- Continuous Availability solution
- Near transparent disaster recovery solution
- Recovery Time Objective (RTO) less than one hour for GDPS/PPRC
- Recovery Time Objective (RTO) less than two hours for GDPS/XRC
- Recovery Point Objective (RPO) of zero (optional)
- Protects against metropolitan area disasters

Figure 3-3 shows a simplified illustration of the physical topology of a GDPS/PPRC implementation, which consists of a Parallel Sysplex spread across two sites (site 1 and site 2 in Figure 3-3) separated by up to 100 kilometers of fiber with one or more z/OS systems at each site. The multi-site Parallel Sysplex must be configured with redundant hardware, for example, a Coupling Facility, and Sysplex Timer in each site, as well as alternate cross-site connections.

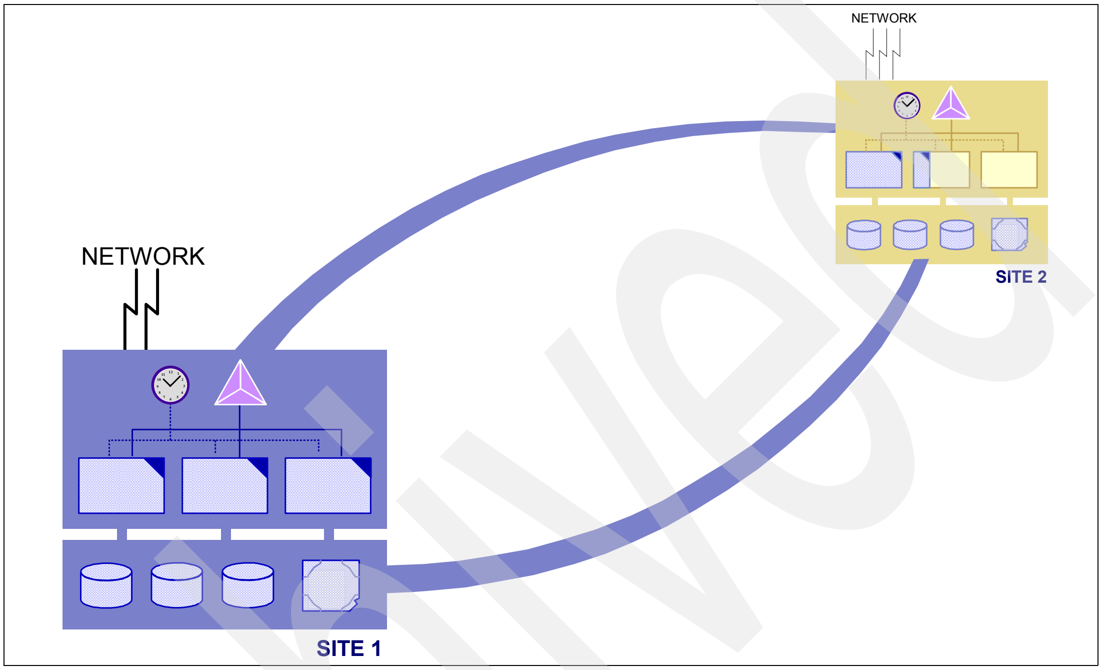


Figure 3-3 GDPS/PPRC topology

All critical data residing on disk storage subsystems in site 1 (the primary copy of data) is mirrored to the disk storage subsystem in site 2 (the secondary copy of data) through PPRC synchronous remote copy.

GDPS/PPRC provides the Parallel Sysplex cluster continuous availability benefits, and it significantly enhances the capability of an enterprise to recover from disasters and other failures, as well as managing planned actions.

GDPS/PPRC HyperSwap

The GDPS/PPRC HyperSwap function provides the ability to switch transparently the applications I/O activity to the secondary PPRC volumes for both planned and unplanned re-configuration. Large configurations can be supported, as HyperSwap has been designed to provide the swap of large number of volumes quickly. The important ability to re-synchronize incremental disk data changes in both directions between primary and secondary PPRC disks is provided as part of this function.

The GDPS/PPRC HyperSwap function is designed to broaden the continuous availability attributes of GDPS/PPRC. This function can help significantly increase the speed of switching sites and switching disks between sites. The HyperSwap function is designed to be controlled by complete automation, allowing all aspects of the site switch to be controlled through GDPS.

Hyperswap can be triggered by planned event (such as an operator command) or by an unplanned event (such as a permanent I/O failure error). Basic HyperSwap proceeds through the following phases:

1. Validation of I/O connectivity to all peers in sysplex with following action based on policy.
2. Freeze and quiesce DASD I/O after verifying access to secondary devices and acknowledge Sysplex members that primary volumes are no longer in use
3. Terminate Peer-to-Peer Remote Copy (PPRC) replication paths from primary copies.
4. Swap UCBs by updating an internal control block called Unit Control Block (UCB).
5. Resume I/O activity by sending commands from master to all members of sysplex.
6. Cleanup is done while application I/O is proceeding.

After the problem that caused the unplanned HyperSwap is rectified, the original configuration may be restored by issuing a HyperSwap command to swap back to the original primary devices.

Basic HyperSwap does not issue a freeze on PPRC suspend events. Therefore, if PPRC were to be suspended between one pair of LSSs, replication on all of the other LSS pairs in the Basic HyperSwap session would not be suspended. This is done because Basic HyperSwap is a continuous availability solution, not a disaster recovery solution, and issuing a freeze does not provide any benefit in this environment. Also, suspending replication on other LSS pairs can only delay the resumption of full duplex mode once the failed component is repaired.

For more information about GDPS, refer to the IBM Redbooks publication *GDPS Family - An Introduction to Concepts and Capabilities*, SG24-6374 and to these Web pages:

- `http://www.ibm.com/servers/eserver/zseries/library/whitepapers/gf225114.html`
- `http://www.ibm.com/services/us/index.wss/so/its/a1000189`

Backup and Restore System utilities of DB2 for z/OS

The Backup and Restore System utilities were introduced in DB2 for z/OS V8 and enriched with DB2 9 for z/OS.

Backup System utility

The Backup System utility fully replaces steps “SET LOG SUSPEND - FlashCopy - SET LOG RESUME” in DB2 V7 with just a one-statement utility, and with much less impact on current activity. It requires z/OS V1.5, DFSMShsm™ with the COPYPOOL construct, and SMS-managed DASD. It can be used to back up the data copy pool (including ICF catalogs defining the DB2 Catalog/Directory and the application data) and the log copy pool (including ICF catalogs for the active logs, BSDSs, and DASD archive logs) for a full backup. It can also back up only the data copy pool, assuming that you will provide all the logs required since the backup to recover the entire subsystem to a given point-in-time.

The Backup System utility performs the following main tasks:

- Allows most updates.
- Disables 32 KB page writes, if the CI size is not 32 KB, to eliminate an integrity exposure.
- Disables system checkpoint, data set creation, deletion, rename, and extension. operations.
- Acquires new PITR lock in X mode to ensure no restore is taking place on other members.
- Records the Recover Based Log Point (RBLP) in the BSDS. This is the starting point for the log apply phase using the Restore System utility.

During the backup, DB2 performs the following operations:

- DB2 invokes the DFSMShsm ARCHSEND service to take full volume DASD-to-DASD copies of the 'DB' COPYPOOL DFSMShsm, schedules multiple tasks, and FlashCopies the volumes in parallel.
- ARCHSEND returns control to DB2 once the logical copies have completed for all volumes in DATABASE copy pool (normally several minutes).
- DB2 updates the BSDS with the system backup information.
- DB2 invokes DFSMShsm ARCHSEND to take full volume copies of the LOG copy pool, in the same manner as occurred for the DATABASE copy pool.
- DB2 then resumes activities.

In DB2 for z/OS V8, the Backup System utility is designed for local subsystem point-in-time recovery and is global in scope. That is, it pertains to an entire DB2 subsystem or data sharing group and not to a limited number of objects. It is not directly usable for a disaster recovery solution, as the copy pools are stored on DASD by generation.

Restore System utility

The Restore System utility can only be used when you want to recover a subsystem or data sharing group to an arbitrary point-in-time or to current. The utility restores only the database copy pool of a data only, and then applies logs until it reaches a point in the log equal to the log truncation point specified in a point-in-time conditional restart control record (SYSPITR CRCR) created with DSNJU003, or to current. The Restore System utility uses the RBLP stored in the BSDS by the Backup System utility as the log scan starting point. The log apply phase uses Fast Log Apply to recover objects in parallel. DB2 handles table space and index space creates, drops and extends, and marks objects that have had LOG NO events as RECP (table spaces and indices with the COPY YES attribute) or RBDP (indices with COPY NO attribute).

To restore a system to a prior point-in-time with the Restore System utility, use the following steps:

1. Stop DB2. If data sharing, stop all members.
2. Use DSNJU003 to create a SYSPITR CRCR specifying the point to which you wish to recover the system. If data sharing, create a SYSPITR CRCR for each member.
3. If data sharing, delete all coupling facility structures.
4. Start DB2. If data sharing, start all members of the data sharing group.
5. DB2 will enter into system recover-pending mode, ACCESS(MAINT), and will bypass recovery except for in doubt URs.
6. Execute the Restore System utility. The utility must be completed on the original submitting member. The Restore System utility performs the following tasks:
   - Restores the most recent database copy pool version that was taken prior to the log truncation point
   - Performs log apply function
7. If a method other than the Backup System utility was used to copy the system, restore the data manually and use RESTORE SYSTEM LOGONLY.
   - This option can be used with z/OS V1R3 and above.
   - Backs up data with another volume dump solution and uses SET LOG SUSPEND/RESUME.
   - Performs log apply function only.

8. Stop DB2 to reset system recovery-pending status. If data sharing, stop all members.
9. Display and terminate any active utilities.
10. Display restricted objects and recover objects in RECP status and rebuild objects in RBDP status.

DB2 9 for z/OS enhancement for Backup System and Restore System utilities

The Backup System and Restore System utilities were added in DB2 for z/OS V8 and use disk volume FlashCopy backups and copy pool z/OS DFSMShsm V1R5 constructs. In DB2 9 for z/OS these utilities are enhanced to use the following new functions available with z/OS V1R8 DFSMShsm.

- Object-level recovery

  The backups taken by the Backup System utility in V8 are used for recovery of the entire DB2 system. DB2 9 for z/OS has an enhancement to allow a DB2 object to be recovered from the system-level backups. Recovery is through the Recover utility, which is now capable of using system-level backups for the restore phase in addition to image copies.

- Dump to tape

  Maintaining multiple copies of all the data on disk can be expensive. DB2 9 for z/OS allows for the backup to be implemented directly to tape.To use this functionality, you must have z/OS DFSMShsm V1R8 or above.

- Incremental FlashCopy

  The physical copying to disk in the background can be improved by the use of incremental FlashCopy. Even if incremental FlashCopy is used, the dump to tape is always a full dump. Incremental FlashCopy has no benefit when dumping to tape except that the dumping to tape might begin earlier, because less data needs to be copied to disk before writing to tape can be started.

For more details on the disaster recovery related to System z refer to the following IBM Redbooks:

- *Disaster Recovery with DB2 UDB for z/OS*, SG24-63700
- *IBM eServer pSeries HACMP V5.x Certification Study Guide Update*, SG24-6375
- *IBM System Storage DS8000 Architecture and Implementation*, SG24-6786

3.2.6 IBM TotalStorage Productivity Center for Replication (TPC-R)

The IBM TotalStorage Productivity Center (TPC) is a suite of software products. TPC-R, as a member of the TPC product family, is a standalone package and does not build on anything related to the TPC Standard Edition.

TPC-R is designed to help administrators manage Copy Services by providing a graphical interface. These data-copy services maintain consistent copies of data on source volumes that are managed by Replication Manager. IBM TPC-R for FlashCopy, Metro Mirror, and Global Mirror support provided automation of administration and configuration of these services, operational control (starting, suspending, resuming), Copy Services tasks, and monitoring and managing of copy sessions.

This applies not only to the Copy Services provided by DS6000™ and DS8000, but also to Copy Services provided by the Enterprise Storage Server (ESS) and SAN Volume Controller (SVC).

TPC-R is designed to be a single product that supports single-site, two-site, and three-site storage replication, as well as a function called Basic HyperSwap*, which is unique to z/OS.

TPC-R provides the following functions:

- Manage Copy Services for the IBM System Storage DS6000 and DS8000.
- Extend support for ESS 800 to manage Copy Services.
- Provide disaster recovery support with failover/failback capability for DS8000, DS6000 and ESS 800.

The optional high availability feature allows us to continue replication management even when one TPC- R server goes down.

The basic functions of TPC-R provide management of the following features:

- FlashCopy
- FlashCopy SE
- Metro Mirror
- Global Mirror
- Metro/Global Mirror

TPC-R supports System z storage (that is, ECKD™ [Extended Count Key Data]). A single instance of TPC-R can manage both open and System z storage on any of the four supported platforms (Linux®, Windows®, AIX®, and z/OS), the only exception being that Basic HyperSwap requires TPC-R be deployed on z/OS.

TPC-R has several new features that are unique to z/OS. This includes the ability to send replication commands through native I/O commands, rather than over TCP/IP (Transmission Control Protocol/Internet Protocol), resulting in greater reliability. This is expected to result in enhanced performance.

The z/OS version of TPC-R supports all of the functions provided by the distributed version of TPC-R, as well as a few functions that are unique to the z/OS version. These include moving reserves and the ability to issue replication commands inband, through the same fabric that is used for I/O requests in addition to sending through a TCP/IP connection. Another unique function is support for Basic HyperSwap, which is designed to help eliminate single-disk failures as a source of application outages by allowing customers to specify a set of storage volumes to be mirrored synchronously.

In the event of a permanent I/O error (if all paths to a device fail), I/O requests are automatically switched to the secondary copy, thereby masking the failure from the application and minimizing the need to restart the application (or system) after the failure. When Basic HyperSwap was developed, designers chose to implement it as a new type of TPC-R replication session. This was done for many of the same reasons that TPC-R was originally ported to z/OS. Implementing Basic HyperSwap in this way allows the new replication session to be managed from the existing product, and also allows TPC-R to check for inconsistencies between replication sessions.

TPC-R comes in several levels:

- TPC-R, (5608-TRA)

  This version includes support for the following features:

  – FlashCopy and FlashCopy SE
  – Planned failover and restart (one direction) for MM and GM

- TPC-R Two Site Business Continuity (BC), (5608-TRB)

  This version includes support for the following features includes support for the following features:

  – FlashCopy and FlashCopy SE
  – Planned and unplanned failover and failback for MM and GM
  – High availability (with two TPC Replication servers)

- TPC-R Three Site Business Continuity (BC), (5608-TRC)

  This version includes support for the following features:

  – FlashCopy and FlashCopy SE
  – Planned and unplanned failover and failback for MM and GM
  – High availability (with two TPC Replication servers)
  – Metro/Global Mirror for DS8000 failover and failback with incremental resynchronization

- TPC-R for System z, (5698-TPC)

  This version includes support for the same features and supported products as 5608-TRA, but TPC-R Server runs on z/OS.

  – Provides SMP/E installer
  – Supports English and Japanese language initially
  – Supports FICON commands in addition to TCP/IP, but TCP/IP attachment is required
  – FICON used only for assigned LSSs, TCP/IP required to manage LSS heartbeat, remote LSSs, and SVC

Note: TPC-R operations are performed using the DS8000 hardware management console. The use of Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) are both supported through the HMC. Ethernet ports on the DS8000 server only support the use of IPv4. For more information, visit the IBM TotalStorage Multiple Device Manager Information Center Web page:

http://publib.boulder.ibm.com/infocenter/tsmdmv11/index.jsp

3.2.7 I/O connectivity

As the amount of data in a data warehouse environment is increasing every day, there is a requirement for providing fast data access to the processor unit. 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 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.

For more information about AMP, see the following Web page:

http://www.almaden.ibm.com/storagesystems/projects/arc/technologies/amp.pdf

For more information about zHPF, see the following Web page:

http://www.ibm.com/support/techdocs/atsmastr.nsf/c6192fb3a432612485256d970082de57/05d19b2a9bd95d4e8625754c0007d365/$FILE/zHPF%20technical%20summary%20for%20customer%20planning%20final.pdf

For more information about FICON, see the following Web page:

http://www.ibm.com/systems/z/hardware/connectivity/news.html

3.2.8 Parallel Access Volumes

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. 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.

With PAV, reads are simultaneous. Writes to different domains (a set of tracks the disk controller is working on) are simultaneous as well. Writes to the same domain, however, are serialized. No double updates are possible to preserve integrity. Large volumes such as 3390 mod 9, 27, and 54 greatly benefit by using PAV. Multiple paths or channels for a volume have been around for many years. Multiple Unit Control Block (UCBs = MVS™ addresses), however, were only introduced with PAVs.

Multiple Allegiance

The DS8000 accepts multiple I/O requests from different hosts to the same device address, increasing parallelism and reducing channel overhead. In older storage disk subsystems, a device had an implicit allegiance (a relationship created in the control unit) between the device and a channel path group when an I/O operation was accepted by the device. The allegiance caused the control unit to guarantee access (no busy status presented) to the device for the remainder of the channel program over the set of paths associated with the allegiance. With Multiple Allegiance, the requests are accepted by the DS8000 and all requests are processed in parallel, unless there is a conflict when writing data to a particular extent of the CDK logical volume.

Figure 3-4 and the following characteristics describe the operations of Multiple Allegiance and PAV:

- With Multiple Allegiance, the I/Os are coming from different system images.
- With PAV, the I/Os are coming from the same system image:
  - Static PAV

    Aliases are always associated with the same base addresses.
  - Dynamic PAV

    Aliases are assigned up front, but can be reassigned to any base address as need dictates by means of the Dynamic Alias Assignment function of the Workload Manager—reactive alias assignment. Figure 3-4 describes the operation of Dynamic PAVs.

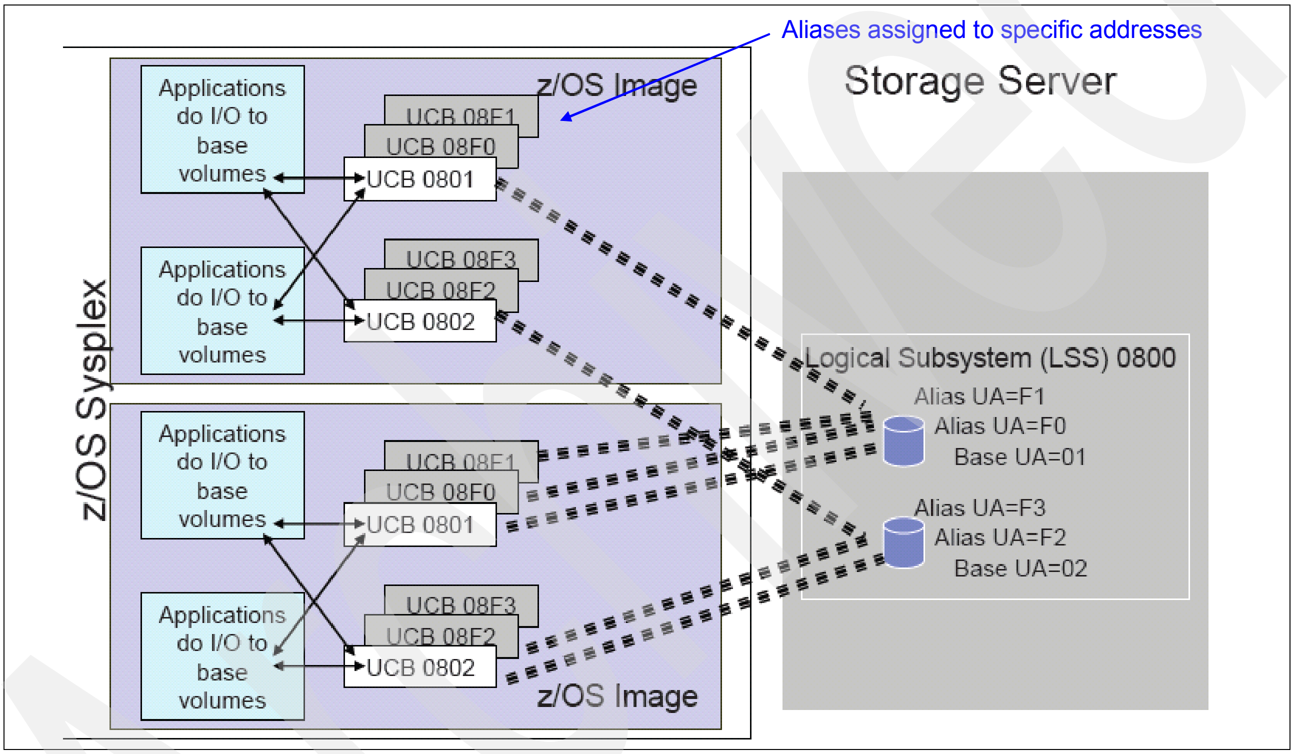


Figure 3-4 Parallel Access Volumes

HyperPAV

With HyperPAV, an on-demand proactive assignment of aliases is possible. HyperPAV allows an alias address to be used to access any base on the same control unit image per I/O base. This capability also allows different HyperPAV hosts to use one alias to access different bases, which reduces the number of alias addresses required to support a set of bases in a System z environment with no latency in targeting an alias to a base. This functionality is also designed to enable applications to achieve equal or better performance than possible with the original PAV feature alone while also using the same or fewer operating system resources. Figure 3-5 on page 42 depicts the HyperPAV operations. For more information about PAV, HyperPAV and DS8000 features, refer to *IBM System Storage DS8000 Architecture and Implementation*, SG24-6786.

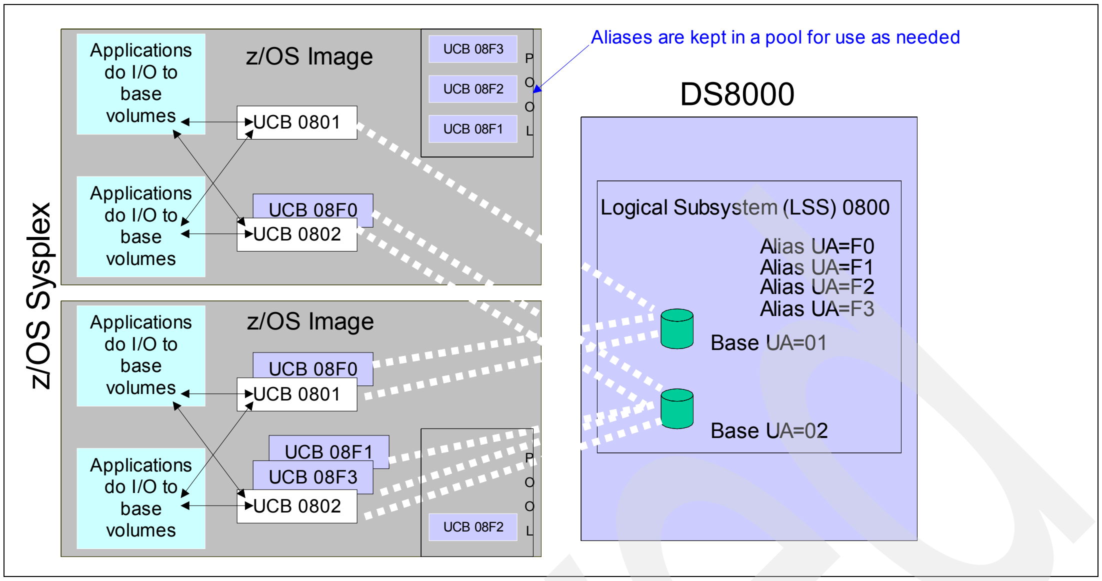


Figure 3-5 HyperPAV

The total cost of ownership of databases and data warehouses has become much more favorable on System z, in some cases costing less than other platforms. This is made true mostly because many different applications can share a single System z, and in some cases share the same DB2 subsystem. This allows the cost of administering System z and DB2 to be spread over multiple application workloads. This may not be possible on other platforms.

IBM continues to introduce innovations that further decrease total cost of ownership (TCO) of a mainframe. The IBM System z Integrated Information Processor (zIIP), designed to maximize resource optimization, is the latest such innovation. It was introduced in 2006 and is also available on z9 and z10 models. It was immediately exploited by DB2 for z/OS V8. See Figure 3-6 on page 43. The zIIP processor is priced lower than general purpose processors and the MIPS it provides do not count toward the software cost of the system. In DB2 V8 and V9 for z/OS, distributed requests over TCP/IP, query requests that use parallelism, and utilities that maintain index structures can exploit zIIPs. BI application costs may directly benefit from DB2 and zIIPs. In DB2 9 for z/OS, remote (DRDA®) native SQL procedures will also make use of zIIPs and more will likely follow. See IBM Redbooks publication *The Business Value of DB2 UDB for z/OS*, SG24-6763.

Also see these papers by Julian Stuhler on DB2:

- V8

  ftp://ftp.software.ibm.com/software/data/pubs/papers/DB2_for_zOS_V9_Business_Value_White_Paper.pdf

- DB2 9

  http://www.ibm.com/software/os/systemz/newsletter/mainstreamed11_uk.html

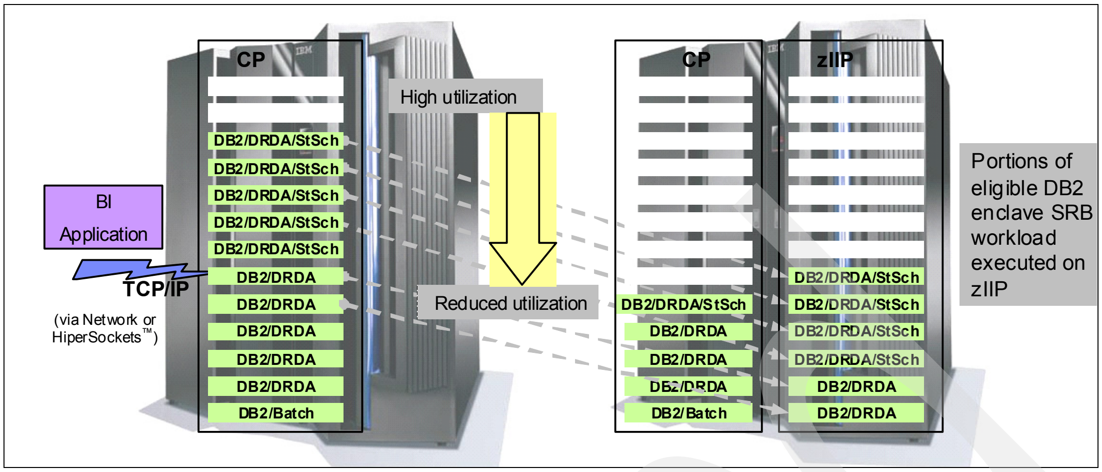


Figure 3-6 The specialty engines

Specialty engines

Additional TCO improvements can be achieved by making use of the other specialty processors available on System z. The Integrated Facility for Linux (IFL) is used for Linux applications. You can consolidate Linux servers on a single z10 partition (LPAR) or you can move distributed Linux applications to System z to exploit HiperSockets™ for better performance. The System z Application Assist Processor (zAAP) is used to run Java™ applications.

zNALC and DB2 Value Unit Edition

To drive new workload growth on new and existing DB2 for z/OS, IBM is offering customers DB2 for z/OS a one time charge (OTC) pricing for qualified net new workloads on z/OS. The net new workloads are limited to those that qualify for zNALC (System z New Application Charge) (for example: commercial packaged applications, data warehouses, and InfoSphere™ applications running in a partition, machine, or sysplex). The VUE offering excludes existing workloads.

Refer to the following links for more information about zNALC and DB2 Value Unit Edition

- http://www.ibm.com/software/data/db2/zos/edition-vue.html
- http://www.ibm.com/servers/eserver/zseries/swprice/znalc.html

3.2.9 IBM System z10

The system z10 is designed from the ground up to help dramatically increase data center efficiency by significantly improving performance and reducing power, cooling costs, and floor space requirements. It offers unmatched levels of security and automates the management and tracking of IT resources to respond to ever-changing business conditions. The following list details the key features of System z10:

- A single z10 is equal to nearly 1,500 x86 servers
- Up to 85% less energy costs
- Up to 85% smaller footprint
- Consolidates x86 software licenses at up to a 30-to-1 ratio.
- Mainframe goes Quad-Core
- z10 brings discipline to data center chaos:
  - Just-in-time capacity to meet ever-changing business conditions
  - Automated management of system performance to favor high-value transactions

Figure 3-7 shows the evolution to System z10 in comparison with previous generation machines.

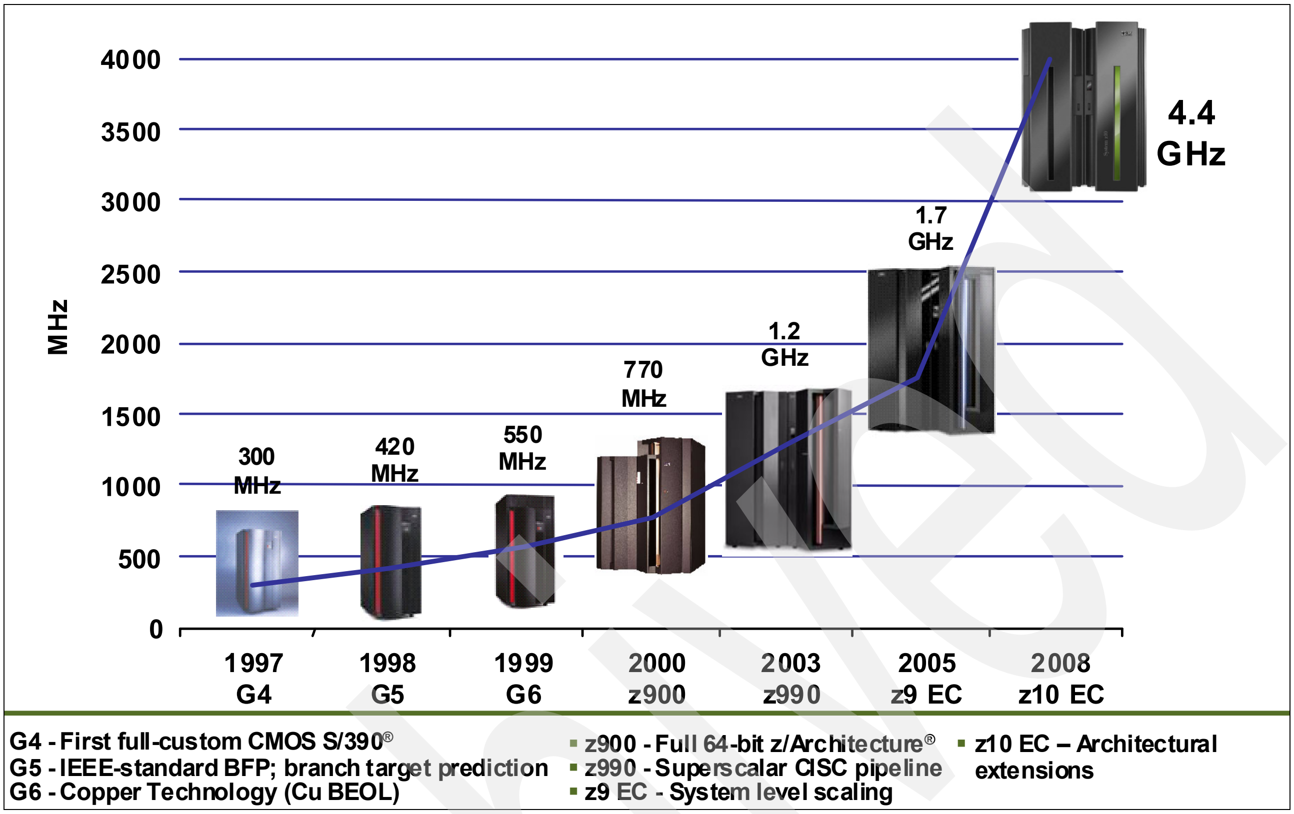


Figure 3-7 Evolution of System z

The z10 also supports a broad range of workloads. In addition to Linux, XML, Java, WebSphere, InfoSphere, and increased workloads from SOA implementations, IBM is working with Sun™ Microsystems and Sine Nomine Associates to pilot the Open Solaris™ operating system on System z, demonstrating the openness and flexibility of the mainframe.

You can find more on features of System z (z9 BC, z9 EC, z10 BC, and z10 EC™) on the following Web pages:

http://www.ibm.com/systems/z/hardware/z9bc/features.html
http://www.ibm.com/systems/z/hardware/z9ec/features.html
http://www.ibm.com/systems/z/hardware/z10bc/features.html
http://www.ibm.com/systems/z/hardware/z10ec/features.html

3.2.10 Existing System z customer base

The mainframe has been a solid platform delivering value for years. It is a proven performer in both OLTP (online transaction processing) and data warehouse scenarios. Where there is a mainframe installed today with a preponderance of critical IT delivery and support to users, one should take a look at the following points that support implementing a data warehouse Solution on System z:

- Existing infrastructure

  Most of the data feeding the warehouse resides on z/OS and is being continuously updated .

- Platform preferences and skills in-house

  Regardless of available platforms this is a System z-centric enterprise and wants to use its existing skills.

- Budget/Cost

  The customer has capacity on System z and does not intend to assume of cost of a new platform, skills, and associated software.

- Market trends in warehousing solutions and vendors

  Gartner[1] substantiates that System z is viable as a data warehouse solution. Customer requirements do not dictate an alternative. For more information, see ftp://ftp.software.ibm.com/software/data/db2bi/data_warehousing_whitepaper.pdf.

- Complexity

  The sheer volume of data involved and its associated issues if it must be replicated and moved as well as other factors.

- Competitive influences

  System z solution is adequate to address competitive situations.

- Mergers and acquisitions or consolidations

  System z in the new joint venture has dominance or the customer wishes to reduce the number of platforms and servers they currently maintain.

- Data or application sources

  The majority of the data is on System z and is constantly being added to, updated, and changed. It is too large and volatile to re-host.

- Business processes required and business drivers

  Customer infrastructure and business process requirements (for example, intranet or Internet infrastructure is setup for System z with high affinity to warehouse data).

3.2.11 DB2 for z/OS with additional data warehousing capabilities

Several DB2 enhancements, like enhanced index options, new partitioning options, advanced SQL, star join enhancements, and MQT support are especially suited for data warehousing.

Tablespace partitioning

The large volume of data stored in DW environments can introduce challenges to database management and query performance. The tablespace partitioning feature of DB2 9 for z/OS has the following characteristics:

- Maximize availability or minimizing run time for specific queries
- Can have 4096 partitions, a partition is a separate physical data set
- Allow loading and refreshing activities, including the extraction, cleansing and transformation of data, in a fixed operational window
- Increase parallelism for queries and utilities
- Accommodate the growth of data, an universal table space can grow automatically up to 128 TB and adds the functionality of segmented table spaces.

[1] Gartner, Magic Quadrant for data warehouse DBMS Servers, 2006, Publication Date: 25 August 2006, ID Number: G00141428

Very large database

DB2 for z/OS can contain a large amount of data. As many as 64,000 databases can exist within a single DB2 subsystem or data sharing group. While the number of tables in a table space could number in the thousands, a reasonable limit for operational ease is 100 tables per table space or one partitioned table per table space. The number of tables in a subsystem can be millions, much more than used by any current customer. The maximum table size for partitioned tables is 128 TB, unless LOBs are used, where the limit is orders of magnitude larger. For partitioned tables, only one table per table space is allowed, but the limits are in the exabyte range, again larger than any customer can afford. These partitioned table spaces can have as many as 4096 files for the data and the same number for each partitioned index, thus the number of files could be 500 million. Thus DB2 can easily cater to the growing need of a DW environment.

Star schema enhancements

A common data model used in DW environments is the star schema, in which a large central fact table is surrounded by numerous dimension tables. Queries generally provide filtering on the independent dimensions that must be consolidated for efficient access to the fact table.

DB2 for z/OS Version 8 contains several enhancements to improve the performance of star schema queries:

- In-memory workfiles for efficient access to materialized dimensions/snowflakes
- Improved cost formula for join sequence determination
- Predicate localization when OR predicates cross tables

DB2 9 for z/OS further enhances star schema query performance with a new access method, Dynamic Index ANDing, for simpler index design, more consistent performance, disaster avoidance, and improved parallelism.

Query parallelism

Reduce the response time for data or processor-intensive queries by taking advantage of the ability of DB2 to initiate multiple parallel operations when it accesses data in a DW environment. In addition, in DB2 9 more of the DW work can be done in parallel (with increased throughput) and is eligible for zIIP processing (lowering the cost).

DB2 for z/OS supports three types of parallelism:

- I/O parallelism
- CPU parallelism
- Sysplex parallelism

Materialized query tables

Materialized query tables can simplify query processing, greatly improve the performance of dynamic SQL queries, and are particularly effective in data warehousing applications, where you can use them to avoid costly aggregations and joins against large fact tables.

The DB2 optimizer uses partial or entire MQTs to accelerate queries. Its operation and its access path are also kept, which allows an easy refresh of the MQT content without having to specify the source query again. The DB2 optimizer is aware of these MQTs. If you were to write queries against the base tables on which this MQT is based, the optimizer sees if it can satisfy the request from the MQT. If it can, it automatically rewrites the query to go to the MQT. You save CPU time and quicken query response time for users by using MQTs.

OLAP functions

New SQL enhancements are made in DB2 9 for z/OS for improving OLAP functions in a data warehouse. The following OLAP expressions were introduced in DB2 9 for z/OS:

- RANK and DENSE_RANK
- ROW_NUMBER

Tablespace and index compression

DB2 for z/OS uses the hardware-assisted compression instruction of System z for compressing table spaces. DB2 9 for z/OS can also compress index spaces using software techniques. A data warehouse has a large amount of data, with many indexes for query performance. Thus compressing table and index spaces in a data warehouse saves large amounts of disk space (and often CPU cycles as well).

INDEX on expression

DB2 9 for z/OS supports the creation of indexes on an expression. The DB2 optimizer uses such an index to support index matching on an expression. In certain scenarios it can enhance the query performance. In contrast to simple indexes, where index keys are composed by concatenating one or more table columns specified, the index key values are not exactly the same as values in the table columns. The values have been transformed by the expressions specified.

CLONE tables

To overcome the availability problem when running certain utilities like LOAD REPLACE in a DB2 for z/OS environment. For information about DB2 9 utilities, see IBM Redbooks publication *DB2 9 for z/OS Performance Topics*, SG24-7473. Cloning feature was introduced in DB2 9 for z/OS. A clone of a table can be created using an ALTER TABLE SQL statement with an ADD CLONE clause. The clone can be used by applications, SQL, or utilities, thereby providing high availability.

When you create clone tables, the purpose is to switch from the original base table to your newly created and maybe reloaded clone table. Use the EXCHANGE SQL statement to initiate the swap.

XML support

DB2 9 for z/OS provides pureXML™, a native XML storage technology providing hybrid relational and XML storage capabilities. This provides a huge performance improvement for XML applications while eliminating the need to shred XML into traditional relational tables or to store XML as Binary Large Objects (BLOBs), methods other vendors use. DB2 9 pureXML uses z/OS XML System Services for high-performance parsing with improved price performance using zAAPs and zIIPs.

PureXML includes the following capabilities:

- XML data type and storage techniques for efficient management of hierarchical structures inherent in XML documents.
- pureXML indexes to speed search subsets of XML documents.
- New query language support (SQL/XML and XPath) based on industry standards and new query optimization techniques.
- Industry-leading support for managing, validating, and evolving XML schemas.
- Comprehensive administrative capabilities, including DB2 utilities and tools.
- Integration with popular application programming interfaces (APIs) and development environments.

- XML shredding, publishing, and relational view facilities for working with existing relational models.
- Proven enterprise-level reliability, availability, scalability, performance, security, and maturity that users expect from DB2.

3.2.12 ETL on the same platform

ETL (extract, transform, and load) is the acronym given to the processes that extract information from the OLTP system, transform it according to the needs of the data warehousing environment, move it to the platform that houses the data warehouse, and load this data into the data warehouse.

The process of extracting information from the OLTP system runs by necessity on the same platform as the OLTP system. The process of transforming it to conform to the needs of the data warehouse is also usually performed on the platform where the OLTP system resides. This is because many of the techniques used in the transformation process reduce the amount of data moved to and loaded into the data warehouse.

Regardless of where the transformation process is performed, it is necessary to move the data to the platform housing the data warehouse. There are many methods that can be employed in this movement of data. Whatever method is employed, if the data warehouse is physically distinct from the platform housing the OLTP system, it is necessary to transmit this data over a communications path of some kind. Often, this is the single most expensive component of the entire ETL process.

Obviously, if the data warehouse resides on the same platform as the OLTP system, sending data over a communication path is unnecessary. With the parallel sysplex capabilities of the System z, it is a simple matter to have the data warehouse on the same platform as the OLTP system and, at the same time, be on separate processors from the OLTP system.

System z can handle the different characteristics of OLTP and BI workloads within one logical database system. This can be achieved by DB2 subsystems, optimized for different workloads within a data sharing group. Data sharing is a DB2 feature of exploiting System z Parallel Sysplex technology, and therefore sharing the workload between multiple DB2 subsystems accessing the same data

See Figure 3-8 on page 49. This figure shows three subsystems at the top that are optimized to handle transactional workload. At the bottom are two subsystems that are optimized for BI. All subsystems are in the same data sharing group.

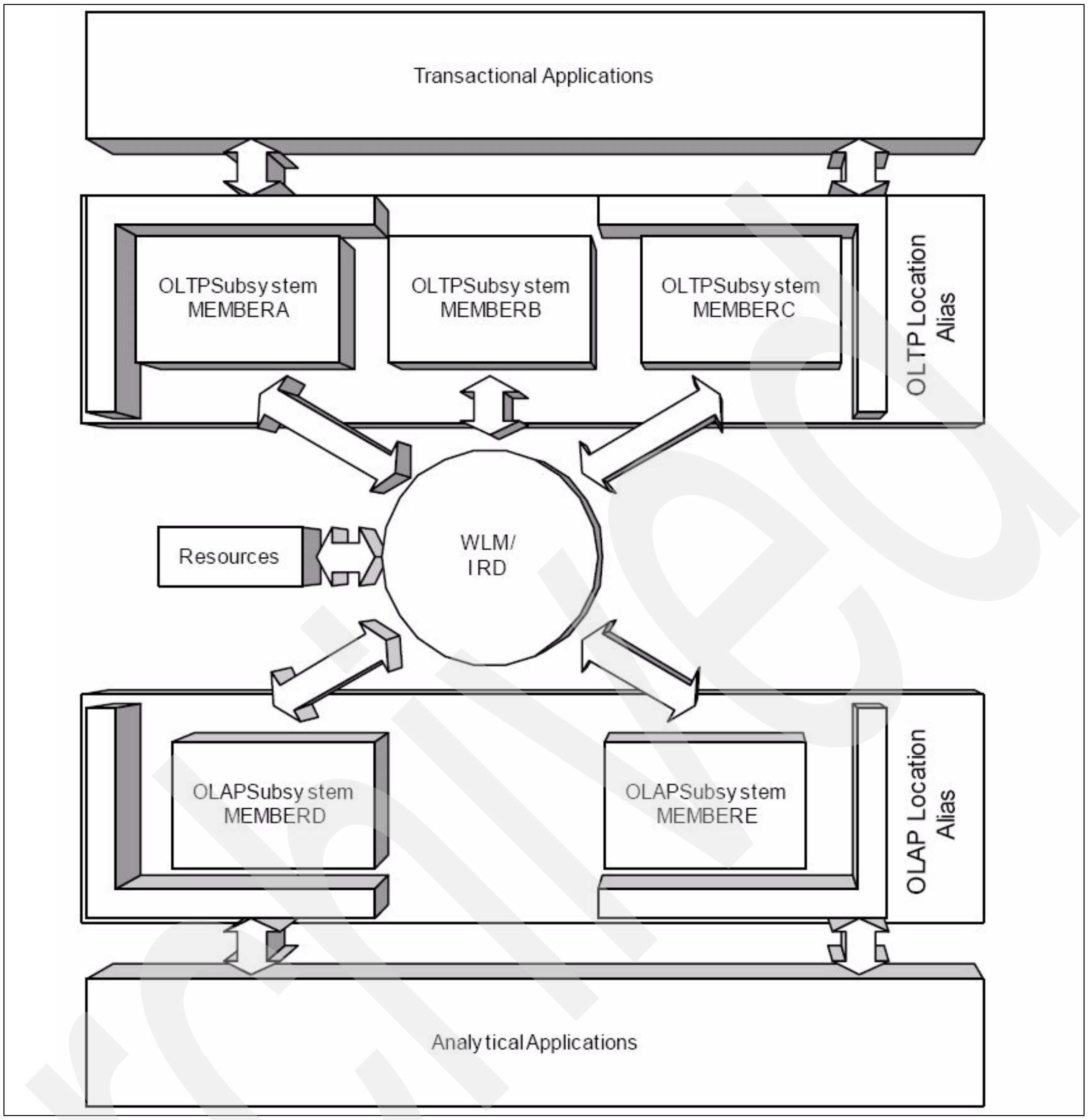


Figure 3-8 Data sharing and mixed workload

If the subsystems are in the same LPAR, the WLM can be configured to assign the correct amount of resources to the subsystems. It can be configured in a way that OLTP subsystems always have a guaranteed amount of resources, no matter how much workload is assigned to the data warehouse members. Further, WLM can be configured to prioritize short-running queries in the BI subsystems.

If the subsystems are not in the same LPAR, the Intelligent Resource Director (IRD) can manage resources among LPARs. WLM can manage those resources among subsystems within the same LPAR.

The figure also shows that the transactional systems are routed against the data sharing members that are optimized for OLTP workload, and the query and reporting and analytical applications are routed against the data sharing members that are optimized for BI queries. This is achieved by location alias definitions in DB2. Each application sees detailed transactional data as well as data warehouse aggregates. But the DB2 subsystems of member D and E are configured with a focus on parallel query execution, while members A, B, and C are used for normal, short-running transactions.

This System z/DB2 configuration is optimal for the different types of workload that occur if OLTP and BI data is mixed in the same system. It is highly scalable. Detailed data can be accessed in the same way as aggregates, which also allows in-database transformations.

If data movement to another system or platform is desired for some reason, IBM InfoSphere Replication provides the ability to stream high volumes of data with low latency to create near real time Warehouses or IBM InfoSphere DataStage® can be used to provide complex transformations of the data before loading them into the warehouse.

In addition to DB2 for z/OS as an essential core component for hosting data in a data warehouse, other components are required to build a comprehensive solution, mainly to address the following issues:

- Efficient access to multiple data sources
- ETL processing
- Data integration and cleansing
- Managing incremental updates
- Analytics and reporting

InfoSphere Classic Federation

A data warehouse may collect, consolidate and aggregate data from various sources and various source types, not all being naturally relational data sources, such as DB2 for z/OS.

Some data might reside on System z in the following locations:

- IMS
- CICS® controlled VSAM data sets
- ADABAS
- IDMS
- Flat files

This data (or at least aggregates of it) needs to be integrated into the data warehouse so that all information becomes available in a single place.

InfoSphere Classic Federation can access this data in a way that makes it unnecessary to set up a specialized extraction process for each of the different data sources (such as hierarchical or relational databases).

The InfoSphere Classic Federation Data Server is installed where the data sources reside.

For more information, see the following Web page:

http://www.ibm.com/software/data/infosphere/classic-federation-server-z/

A connector is configured and started for each data source that needs to be accessed. These connectors are then used by the data server to access the source data from IMS, VSAM, ADABAS, or other supported sources. The data server maps the various source data structures to a relational structure. So all data accessed through the InfoSphere Classic Federation Data Server looks like one relational database, even if the source is structured hierarchically.

Mapping between source and target definitions is done with the Classic Data Architect, which is an Eclipse-based workstation tool. It allows importing IMS DBD or COBOL Copybooks to obtain the structure definitions of source data. The user can select which information to include in a target table which is simulated as a relational structure by the data server. No further configuration or coding is necessary.

An ETL server may access this data through JDBC™ or ODBC to move it to the data warehouse in the form of staging tables or aggregates. The ETL Server and the WS Classic Federation DataServer are primarily used for either an initial load or a full re-load of the DW. Data in legacy data sources can be updated as well, but this is usually not required within a data warehouse environment. See Figure 3-9 for an overview of components that come with and interact with InfoSphere Classic Federation Server.

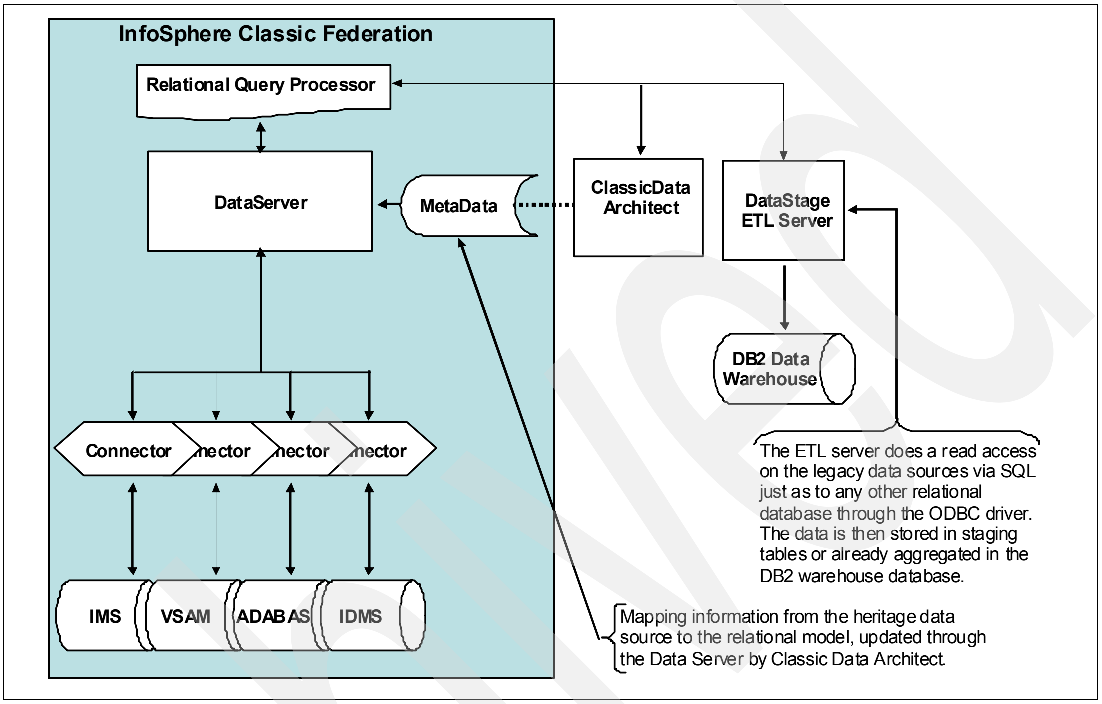


Figure 3-9 InfoSphere Classic Federation

IBM InfoSphere Data Event Publisher

Once the data is loaded into the data warehouse, it usually must be updated incrementally. It is not necessary to replace the entire content. InfoSphere Data Event Publisher and InfoSphere Classic Data Event Publisher, respectively, are used to detect the changes in DB2 sources and heritage data sources and to provide the information about the changes to the ETL server. For more information, see the following Web page:

http://www.ibm.com/software/data/infosphere/support/classic-data-event-publisher-z/

The Classic Data Event Publisher and the Classic Federation have much code in common. If you installed and configured Classic Federation before, you can extend this configuration to allow event publishing. Again, you have to map legacy data structures to relational tables by using Classic Data Architect. You can even configure the server so that this information is used for federation and event publishing at the same time, with just one running data server.

Instead of using a connector to directly access the traditional data sources, a change capture agent is used to detect all manipulations to the data source. In some cases, this is a logger exit (like, for example, with IMS), in other cases it is a log reader. The Change Capture Agent sends the change information to the correlation service, which maps the current data structure to a relational structure, which is designed with Classic Data Architect. Distribution Service and Publisher are then used to send the information about the changes through WebSphere MQ to the DataStage data integration server. The DataStage server is able to read the MQ messages directly from the queue, just as they are transmitted. These events can then be stacked up in staging tables and applied to the data warehouse in a composite update, which is run in a batch window. See Figure 3-10 on page 52.

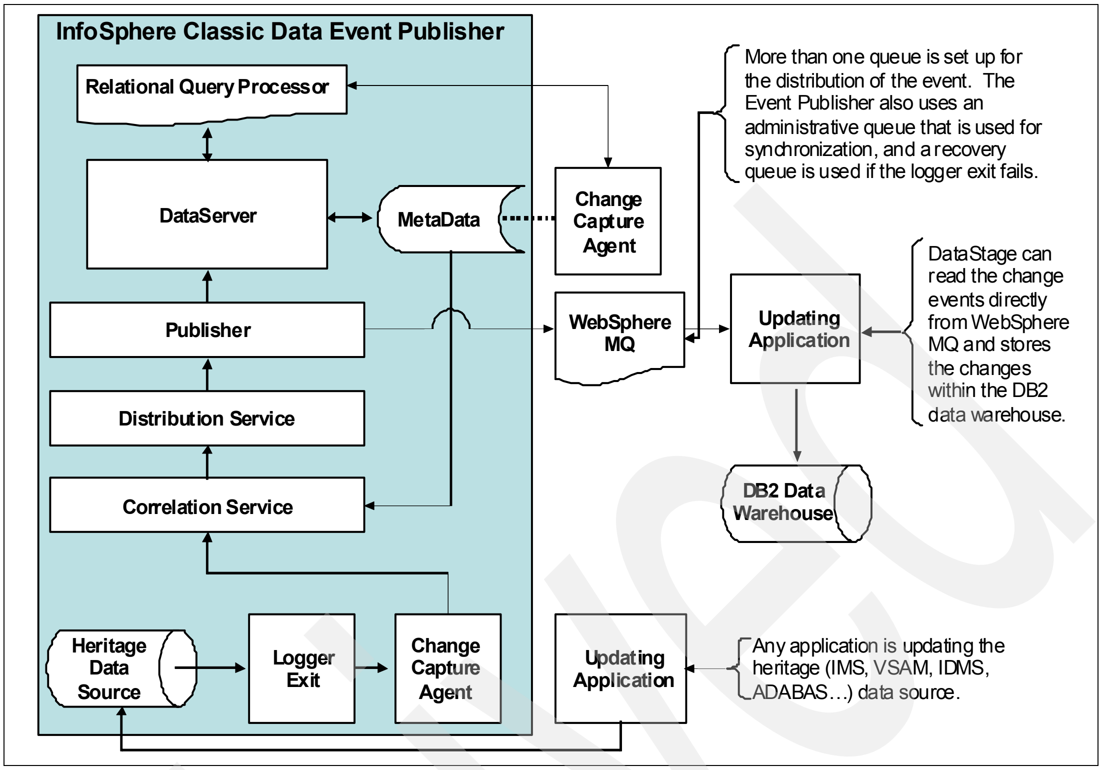


Figure 3-10 InfoSphere Classic Data Event Publisher

InfoSphere Data Event Publisher for z/OS and InfoSphere Replication Server for z/OS

InfoSphere Data Event Publisher for z/OS and IBM InfoSphere Classic Data Event Publisher for z/OS offer efficient changed-data solutions for DB2 for z/OS databases.

A Q Capture program reads the DB2 log files for transactions containing changes to the sources in your publications. It places messages containing those transactions or messages containing individual changes on queues. You provide the destination for the messages. You can publish XML messages to a Web application, use them to trigger events, and more. You can either start the Q Capture program with the default settings or modify those settings for your environment.

Figure 3-11 on page 53 shows a simple configuration for Q capturing and event publishing: a queue capture control program reads change information from DB2 for z/OS logs and (based on definition in control tables) make these changes available in a queue.

With InfoSphere Data Event Publisher for z/OS, you can propagate data changes to DataStage and other integration solutions for populating a data warehouse, data mart, or operational data store with the changes.

For more information, see the following Web page:

http://www.ibm.com/software/data/infosphere/classic-replication-server-z/

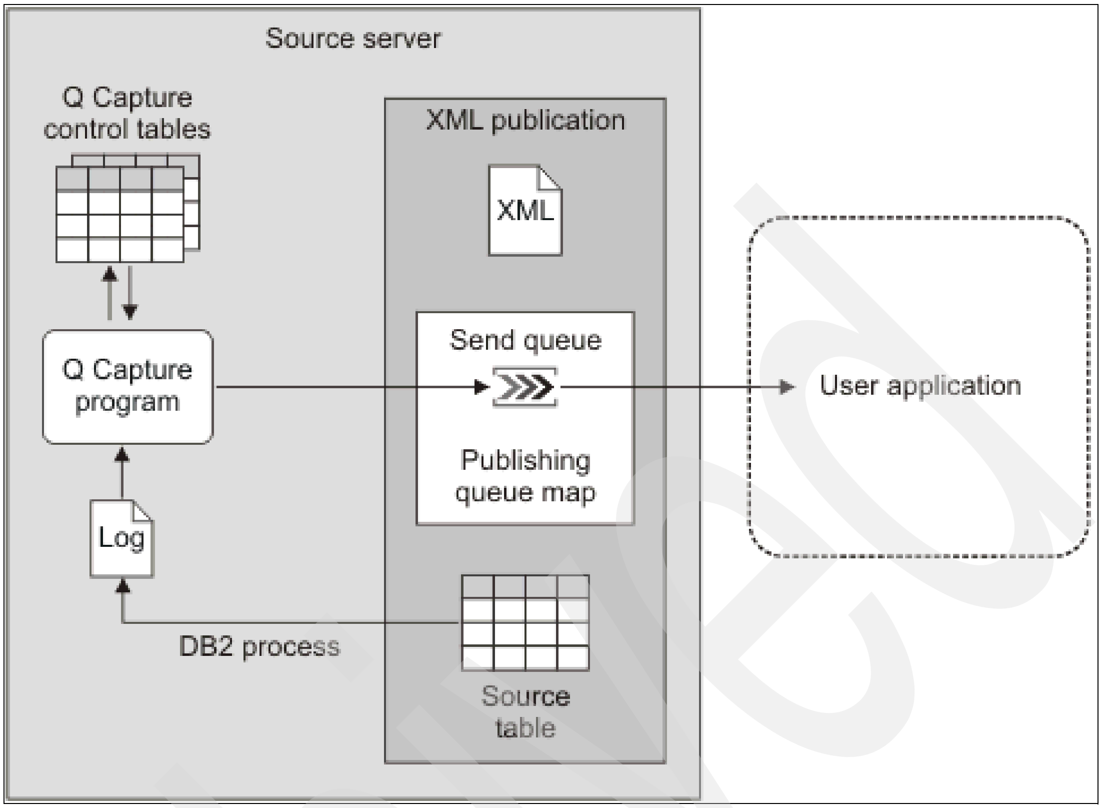


Figure 3-11 Event publishing and Q capturing

InfoSphere Information Server for System z

The IBM InfoSphere Information Server is a framework for services around data transformation, data quality assurance, metadata management, SOA services ,and other operations.

For more information, see the following Web page:

http://www.ibm.com/software/data/infosphere/info-server/overview/

Figure 3-12 on page 54 depicts the four key integration functions in Information Server. The following list explains how these components can be used in a data warehouse environment:

- Understand data

  This function is about analyzing data, and determining the meaning and relationships of information. Web-based tools can be used to define and annotate fields of business data. Monitoring functions can be used to create reports over time. A metamodel foundation (and the IBM Metadata Server) helps in managing changes in the OLTP data model and their implication in the warehouse model and derived reports.

- Cleanse data

  In a data warehouse environment, multiple data sources maintained by different applications are to be integrated. Consequently, the data values in the transactional system may be stored in different formats and independent systems are likely to duplicate data. The QualityStage component in Information Server supports consolidation, validation, and standardizing of data from these multiple data sources, and thereby helps to build a consistent, accurate, and comprehensive warehouse model.

- Transform data

  Transforming data in the context of an ETL process is a major requirement for a comprehensive data warehouse solution. The DataStage component in Information Server provides a variety of connectors and transformation functions for this purpose. High speed join and sort operations as well as the parallel engine help in implementing high-volume and complex data transformation and movement. A comprehensive set of tools are offered to monitor, manage, and schedule ETL jobs.

- Deliver data

  In addition to the components already mentioned, Information Server comes with many ready-to-use native connectors to various data sources, both located on distributed or mainframe systems.

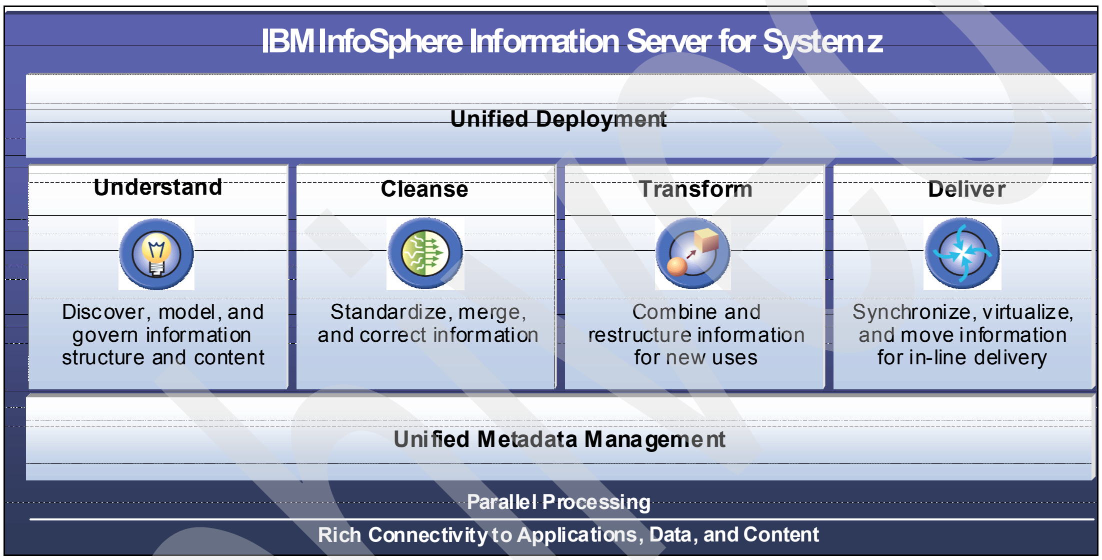


Figure 3-12 Information Server for System z

Information Server 8.0.1, also called IBM InfoSphere FastTrack, comes with a server and a runtime component.

For more information, see the following Web page:

http://www.ibm.com/software/data/infosphere/fasttrack/

For a System z-based implementation, the server component may be installed on Linux on System z. The client components may be installed on a Windows system and are used to design and model data and ETL jobs for DataStage and QualityStage.

Note: There is an Information Server version available for z/OS (running with z/OS UNIX® System Services) and an MVS version of the product. Both are currently based on product levels older than the one we used on Linux for System z. When running multiple jobs concurrently, the version on Linux for System z may have advantages.

IBM Cognos, Alphablox, DataQuant and QMF

Reporting, BI analytics, and queries are an integral part of a data warehouse solution. At the time of writing this book, the following products are available on System z:

- Cognos 8 BI Server (now available for Linux on System z)
- Alphablox (available for Linux on System z)
- DataQuant (available on z/OS) and QMF™ (available on z/OS)
- SAS (runs on System z or other platforms)

Nearly all applications on other platforms can connect to DB2 for z/OS. DB2 Connect™ is an important part of the connectivity. Other products are as follows:

- Informatica
- Hyperion
- Business Objects
- Micro Strategies

4

Configuration

Putting together the right components could be critical for a successful project. In this chapter we list hardware and software components used for this benchmark, and describe the environment and configuration parameters used.

4.1 Introduction

Organizations continue to build increasingly robust data warehouses and BI environments. They often need to expand these environments as their businesses grow. Data is one of the major factors that needs to be considered. However, processing power demands are also growing. The limits of data warehouse environments are pushed every year. These large environments need to be tested and tuned. For this purpose, IBM has built a robust system environment that is described in this chapter.

Data Warehouse and OLTP systems have different requirements. They differ in workload, data model, query types, and history of data. It is important to configure the system for the best performance with the right parameters. You can find these configuration parameters in this chapter. After setting up recommended hardware components and configuration parameters, customers can prepare their environment and be ready to deploy.

On distributed platforms, the whole environment is running on a separate box, while on the System z platform there are multiple workloads and systems. For this reason, many System z features support workload balancing. Parallel Sysplex provides resource sharing, workload balancing, and continuous availability benefits while GDPS technology provides system-level automation, which enables 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. FICON distance and bandwidth capabilities also make it an essential and cost effective component of data high availability and disaster recovery solutions.

Workload Manager (WLM) makes it possible to combine different types of workloads with different completion and resource requirements. It helps the system make the best use of its resources and maintain the highest possible throughput. With workload management, you define performance goals and assign a business importance to each goal. You define the work goals in business terms, and the system decides how much resource, such as CPU and storage, should be given to it to meet the goal. Workload Manager will constantly monitor the system and adapt processing to meet the goals.

The identification of work requests is supported by middleware and the operating system. WLM provides constructs to separate the work into distinct classes. The work classification and WLM observation of how the work uses the resources provide the base to manage the system. This management is done based on the goals that the installation defines for the work. After classifying the work into distinct classes, the installation associates a goal with each class. The goal determines how much service the work in the class is able to receive. For more information about Workload Manager, see Chapter 11, “Workload management” on page 211.

4.2 Environment

This section lists the hardware, operating systems, subsystems, and storage that we used for our 50 TB benchmarking project.

4.2.1 Hardware configuration

We used an IBM System z10-E56 processor and three IBM DS8300 storage units.

Table 4-1 Partition HW configuration

Partition	P17	V64
Memory	512G Central	12G Central 4G Expanded
CPs	56 (Will convert processors from GP to zIIPs at will)	2 to 8
IP address	129.40.178.17	129.40.178.64

The total number of CPs for this benchmark was 56. The Linux LPAR took 2 to 8 CPs, if activated. The number of CPs varied for each test. At times the general processors (GPs) were converted to zIIPs for different measurements.

Table 4-2 Storage DASD configuration

Sub	Type	Cache	# mod 3	# mod 9	# mod 54	TB	Device #
J10	2124-932	128GB	24	72	1692	90	5000-5F94
J20	2124-932	128GB	24	72	1692	90	6000-6F94
J30	2124-932	128GB	24	72	1692	90	7000-7F94

For details, see 4.2.4, “Storage configuration” on page 64.

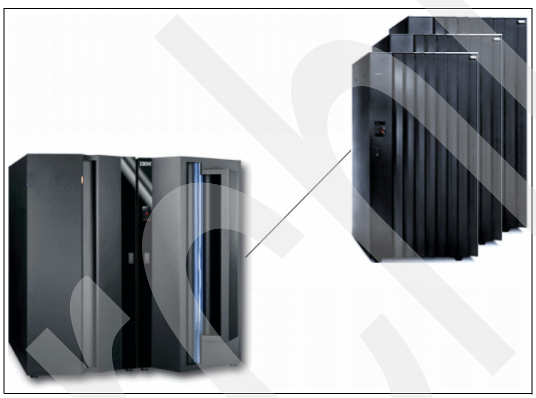


Figure 4-1 IBM System z and DS8000 storage units

4.2.2 Operating systems

z/OS software information

- Operating System
  - z/OS V1R9
  - UNIX Systems Services
  - C/C++ compiler and Language Environment® available
  - RACF®
  - DFSMS™
  - DFSORT™
  - RMF™
  - WLM
  - TCP/IP
  - Tivoli® OMEGAMON®
- Subsystems
  - DB2 9 for z/OS

z/VM software information

- Operating System
  - z/VM® 5.3
  - RACF
  - TCP/IP

Linux on System z

- Operating System
  - SuSE Linux Enterprise Server 10 SP2
  - C/C++ compiler available
- Middleware
  - WebSphere Application Server 31 bit V6.1 fixpak 17
  - IBM HTTP Server 31 bit V6.1 fixpak 17
  - DB2 Enterprise Server 9.5. fixpak 1
- z/VM
- SUSE® Linux Enterprise Server 10
- 2 CPs during installation phase
- 6-8 CPs during test phase
- 16 GB memory (12 GB +4 GB ext)
- Hipersocket connection to z/OS LPAR
- ext3 file system

z/OS operating system configuration

- User catalog
  - UCAT.VLDB
- Data set high-level qualifier
  - VLDB
- HFS sizes are in 3390 cylinders. DASD is SMS managed. All system setup data sets are named 'PEL.VLDB.*'. We use concatenated PARMLIBs, PROCLIBs, and so forth, rather than 'SYS1.*' data sets to externalize the project from the base system data sets.
  - PEL.VLDB.INSTALL' - setup JCL.
  - PEL.VLDB.PARMLIB'- system parameters
  - PEL.VLDB.PROCLIB'- system procedures
  - PEL.VLDB.VTAMLIB'
  - 'PEL.VLDB.VTAMLST'

Linux on System z Operating System Configuration

On our Linux system, the default user IDs were used.

- DB2 instance user
  - db2inst1
- Fenced user
  - db2fenc1
- DB2 administration server user
  - dasusr1

File systems are named <devicename>1. For example: dasda1, dasdb1......dasdl1.

All file systems are defined as ext3. All DASD are mod 9s with 6.8 GB of space. A 26 GB linear logical volume was created from 4 of the disk devices. This is mounted on /vldbtemp and can be used for temporary space.

Table 4-3 File systems

Device	File System Name	Mount point	Comments
6003	/dev/dasda1	/	
6103	/dev/dasdb1	/opt	
6203	/dev/dasdc1	/home	
6303	/dev/dasdd1	swap	
6403	/dev/dasde1		
6503	/dev/dasdf1		
6603	/dev/dasdg1		
6703	/dev/dasdh1	/vldbtemp	/dev/vldbvg/vldblv LVM for temp space, 26GB
6803	/dev/dasdi1	/software_downloads	
6903	/dev/dasdj1		available
6A03	/dev/dasdk1		available
6B03	/dev/dasdl1		available

Example 4-1 Display of the filesystem table

```
vldb1:/ # cat /etc/fstab
/dev/dasda1          /                     ext3       acl,user_xattr        1 1
/dev/dasdc1          /home                 ext3       acl,user_xattr        1 2
/dev/dasdb1          /opt                  ext3       acl,user_xattr        1 2
/dev/dasdi1          /software_downloads   ext3       acl,user_xattr        1 2
/dev/vldbvg/vldblv   /vldbtemp             ext3       acl,user_xattr        1 2
/dev/dasdd1          swap                  swap       defaults              0 0
proc                 /proc                 proc       defaults              0 0
sysfs                /sys                  sysfs      noauto                0 0
debugfs              /sys/kernel/debug     debugfs    noauto                0 0
devpts               /dev/pts              devpts     mode=0620,gid=5       0 0
```

Table 4-4 Minimum storage requirements for software components

Software Component	Disk Space	Comment
DB2 LUW 9.5.0.1	800 MB	
HTTP Server 6.1	200 MB	(part of the WAS 6.1 package)
Cognos BI Server 8.3	2000 MB	BI Server and Transformer
WebSphere Application Server 6.1.0.17	2000 MB	31/32 Bit version
Tivoli Directory Server 6.1	700 MB	w/o Web Administration Toolkit
Temp	10000 MB	
optional: Power Cubes	3000 MB	depends on availability of dimensional models
optional: ITM (monitoring)		maybe use rstatd instead

4.2.3 DB2 subsystem configuration

DASD volumes for our DB2 subsystem (VLD9)

- VLDBS1-VLDBS4
  - DB2 libraries, DB2 system catalogs/directories, BSDS
- VLDBL1-VLDBL8
  - DB2 Active logs.
- VLDW001-VDW540
  - DB2 workfile data sets (DSNDB07)
- VL0001-VL1908
  - Application DB2 table spaces/indexes (SMS managed - DB2DATA storage group. Use HLQ of VLDB29V)
- VLP001-VLP108
- VLDB data volumes (SMS managed - PROJECT storage group. Use HLQ of VLDB)

Table 4-5 Naming Convention for VLD9 (non-data sharing)

Entity	Value
Group name	n/a
Catalog alias	VLDB
Group attach name	n/a
Location name	VLD9LOC
Generic LU name	n/a
IRLM group name	n/a
Subsystem name	VLD9
Command prefix	-VLD9
BSDS	VLDB29.BSDS01 VLDB29.BSDS02
Active log prefix	VLDB29.LOGCOPY1 VLDB29.LOGCOPY2
Archive log prefix	VLDB29.ARCLG1 Note: archiving has been set to off; OFFLOAD=NO in ZPARM VLDB29.ARCLG2
Work file database	DSNDB07
Procedure names	VLD9MSTR VLD9DBM1 VLD9DIST VLD9SPAS
Subsystem ZPARM	DSNZVLD9
LU name	VLD9LU
TCPIP Ports	PORT# 446 RESYNC PORT# 5021
IRLM subsystem name	IRLM
IRLM procedure name	VLD9IRLM
IRLM ID	1

- High Level Qualifiers
  - VLDB29 - Used for DB2 system data sets.
  - VLDB29V - Used for DB2 application tables/indexes (SMS managed)
- Customized libraries:
  - PEL.VLDB.VLD9.SDSNSAMP (all DB2 installation jobs are here)
  - VLDB29.DBRMLIB.DATA
  - VLDB29.RUNLIB.LOAD
  - VLDB29.SRCLIB.DATA
- To Start
  - VLD9 STA DB2
- To Stop
  - VLD9 STO DB2
- To access SPUFI
  - From ISPF primary option menu, select DB2 and make sure HLQ is VLDB29
- # Active logs:
  - 3 pairs (Note: archiving has been set to off; OFFLOAD=NO in ZPARM)

DB2 Performance Expert is installed and available from the main DB2 panel under PE.

DB2 Administration Tool is installed and is available from the main DB2 panel under A or by executing the command: TSO ADB.

ZPARMs configuration

Based on *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637, it is recommended to change some default DSNZPARMs. Table 4-6 compares the recommended ZPARMs with the ones used for benchmark:

Table 4-6 Recommended parameters compared to the benchmark

Recommended Parameter	Benchmark parameter	Description
CDSSRDEF=ANY	1 (can be overwritten during bind)	Allow parallelism for DW; ANY: parallelism, 1: no parallelism
CONTSTOR=NO	NO	For best performance, specify NO for this parameter. To resolve storage constraints in DBM1 address space, specify YES. See also: MINSTOR
DSVCI=YES	YES	The DB2-managed data set has a VSAM control interval that corresponds to the buffer pool that is used for the table space. YES is the default and the preferred setting.
MGEXTSZ=YES	YES	Secondary extent allocations for DB2-managed data sets are to be sized according to a sliding scale
MINSTOR=NO	YES	Recommendation: For best performance, specify NO for this parameter. To resolve storage constraints in DBM1 address space, specify YES. See also: CONTSTOR
OPTIXIO=ON	ON	“OPTIXIO=ON: Provides stable I/O costing with significantly less sensitivity to buffer pool sizes. (This is the new default and recommended setting)”

Recommended Parameter	Benchmark parameter	Description
PARAMDEG=X	0 - let db2 figure out	#Processors <= X <= 2*#Processors. If concurrency level is low, the ratio can be higher.
SRTPOOL=8000	51,200,000	(The Sort Pool is defined as 8 MB)
STARJOIN=	DISABLE	“The STARJOIN and the SJTABLES parameter define the join processing of a query as described in 7.6, “Star schema processing” on page 126. If you use this parameter check the performance improvements carefully.”
MXQBCE	1023	“It is the max number of combinations considered by the optimizer at prepare time when joining N tables. If needed, set to this value to (2**N – 1)., where N is the TABLES_JOINED_THRESHOLD. The MXQBCE default is 1023 in DB2 9, with N=10.”

4.2.4 Storage configuration

IBM has a wide range of storage products offerings. The IBM System Storage DS8000 series, with its capacity, scalability, performance, virtualization, and other new features, is well-suited for a data warehouse environment.

DS8000 models: The DS8100 (Models 921 and 931) features a dual two-way processor complex and support for one expansion model.

The DS8300 (Models 922, 932, 9A2, and 9B2) features a dual four-way processor complex and support for up to four expansion models. Models 9A2 and 9B2 support two IBM System Storage System logical partitions (LPARs) in one storage unit.

The DS8000 enhanced caching algorithm, Adaptive Multistream Prefetching (AMP), can dramatically improve sequential workload performance, such as those used in data warehouse and BI environments. AMP optimizes cache efficiency by incorporating an autonomic, workload-responsive, self-optimizing prefetching technology. AMP adapts both the amount of prefetch and the timing of prefetch to maximize the performance of the system. It also solves two problems that plague most other prefetching algorithms:

- Prefetch wastage

  Prefetch wastage occurs when prefetched data is evicted from the cache before it can be used.

- Cache pollution

  Cache pollution occurs when less useful data is prefetched instead of more useful data.

AMP demonstrates optimal sequential read performance, maximizing the aggregate sequential read throughput of the system. The amount of data prefetched for each stream is dynamically adapted according to the application's needs and the space available in the SEQ list. The timing of the prefetches is continuously adapted for each stream to avoid misses and avoid any cache pollution.

Even though this project benchmarked a 50 TB data warehouse, the 50 TB were used only for table data. Total capacity of storage used was 270 TB to support MQT tables, indexes, logs, temporary space, and FlashCopy images. This configuration allowed us to spread the table data across multiple disks, so that better seek times could be achieved.

For this purpose three dual frame DS8300s (rather than two 3-frames) were chosen to increase the IOPS capacity and bandwidth and to decrease the floor weight loading. Each DS8300 has a full set of 300 GB 15 KRPM disk drive modules (DDMs) that give us 90 TB with 128 GB cache, 32 4-GB FICON channels on 16 host channel adapter cards. The calculated bandwidth is 3808 MBps by configuring 119 KB transfer size and 32 k I/O per second per controller.

Positioning FATA with Fibre Channel disks

It is essential to understand the differences between FATA and Fibre Channel (FC) drive characteristics and mechanisms and the advantages and disadvantages of each to implement FATA target applications properly.

FC drives were designed for the highest levels of IOPS and MBps performance, integrating advanced technologies to maximize rotational velocities and data transfer rates, while lowering seek times and latency. In addition, the FC interface provides robust functionality to process multiple I/O operations concurrently, of varying sizes, in both directions at once. The FATA slower drive mechanisms result in both lower IOPS and MBps performance compared to FC. The FATA drive is not designed for fast access to data or handling large amounts of random I/O. However, FATA drives are a good fit for data mining, Data Warehouse, and many bandwidth applications, because they provide comparable throughput for short periods of time.

Without a doubt, the technical characteristics and performance of FC disks remain superior to those of FATA disks. However, not all storage applications require these superior features and FATA disks offer a tremendous cost advantage over FC. First, FATA drives are cheaper to manufacture, and because of their larger individual capacity, they are cheaper per gigaByte (GB) than FC disks. In large capacity systems, the drives themselves account for the vast majority of the cost of the system. Using FATA disks can substantially reduce the total cost of ownership (TCO) of the storage system. We suggest using the DS8000 FATA drives for data that has a lower usage (20% or lower read/write duty cycles) than the enterprise drives (read/write duty cycles up to 80%). Also, from a performance perspective, FATA drives have a lower random seek performance due to the lower RPM, and therefore, longer seek times than the enterprise drives.

An important factor to keep in mind is that the FATA drives used in the DS8000 protect themselves by reducing I/O if needed, based on the temperature registered by the internal sensors. When reduced, the performance of the drives can drop by up to 50%, resulting in much higher disk access times until the disk returns to its normal temperature. For best savings and performance, the combination of a storage DDM is recommended. Together with a tape library, an archiving solution could be implemented. For more information, see 5.3, “Information retention and archiving” on page 85.

Each disk drive module (DDM) is hot plugable and has two indicators. The green indicator shows disk activity, while the amber indicator is used with light path diagnostics to allow for easy identification and replacement of a failed DDM.

The DS8000 allows the following choice:

- Three different Fibre Channel (FC) DDM types:
  - 73 GB, 15k RPM drive
  - 146 GB, 15k RPM drive
  - 300 GB, 15k RPM drive
- One Fibre Channel Advanced Technology Attachment (FATA) DDM drive:

  500 GB, 7,200 RPM drive

For best performance, the 15 K RPM drives have been chosen for 50 TB benchmark with the 300 GB capacity.

RAID (Redundant Array of Independent Disks) is set up of multiple disk drives that provide fault tolerance and high performance. RAID-5 and RAID-10 are the most common types used and supported on the DS8300. While RAID-10 could provide some performance boost (RAID-5 has a penalty of calculating and writing the parity), this is not the case for Data Warehouse, because this is a write-once read-many environment. The capacity needed for RAID-10 is almost twice the RAID-5 number because of mirrored data. For the benchmark, RAID-5 was used.

Fully-populated DS8300 2-frames with 300 GB DDMs give us 24 6+P+S and 24 7+P arrays/ranks. Combination of two 6+P+S and two 7+P ranks has made each extent pool. The 12 pools were used to create 12 LCUs per subsystem. Each LCU has 6 3390-9 volumes, 141 3390-54 volumes, 2 3390-3 volumes, and 107 PAV addresses, for an average of 13.375 PAVs per channel. The purpose of the 3390-3 volumes is to use all of the space. Our system had 32 channels per subsystem but the maximum number of channels in a channel group is 8. Therefore, the 32 channels were divided into 4 channel groups of 8 channels each. Each channel group was connected to 3 LCUs on the subsystem. So channel group A (chpids 0–7) serviced LCUs 0-3, channel group B (chpids 8-F) serviced LCUs 4–6, and so forth.

Readers who are interested in the technical details can refer to the tables and figure below. Table 4-7 compares the storage capacities of various emulated DASD models. Table 4-8 on page 67 shows how we configured our storage groups, and Figure 4-2 on page 67 compares the amounts of space used by each storage group.

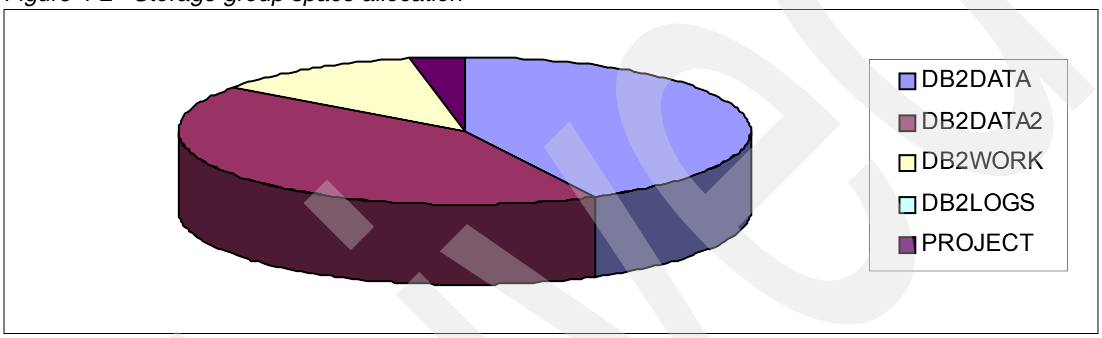
Table 4-7 DASD Emulation Models comparison

Type	# Cylinders	# Tracks	Space in GB	DB2 (GB)	DB2 pages
3390 mod1	1,113	16695	0.946	0.764	200340
3390 mod2	2,226	33390	1.892	1.528	400680
3390 mod3	3,339	50085	2.838	2.293	601020
3390 mod9	10,017	150255	8.514	6.878	1803060
3390 mod54	60,017	982800	51.689	44.989	11793600

Table 4-8 SMS Storage group configuration

SMS Storage group	Range from	Range to	Total volumes available	DASD model type	Available space in GB, per volume	Available space in TB	Remarks
DB2DATA	VL0001	VL1908	1908	mod54	44.99	83.83	DB2 Application data
DB2DATA2	VF0001	VF1908	1908	mod54	44.99	83.83	Load input data
DB2WORK	VDW001	VDW540	540	mod54	44.99	23.73	DSNDB07 table spaces
DB2LOGS	VLDBL1	VLDBL8	8	mod9	6.88	0.05	DB2 log data sets
PROJECT	VLP001	VLP108	108	mod54	55.69	5.87	Performance data

Figure 4-2 Storage group space allocation



5

VLDB build tips and pointers

This chapter summarizes our experiences in building a Very Large Database (VLDB) Data Warehouse (DW) environment at the IBM Benchmark Center, Poughkeepsie, New York. This chapter reviews the key issues that many VLDB designers and administrators encounter, and discusses how we addressed some of those issues for the benchmark. We also cover the design methods, strategies, and tools used to build a large DW on System z.

This chapter contains the following topics:

- Storage management strategy
- Partition strategy
- Compression strategy
- Loading the tables
- Building non-partitioned secondary indexes
- Building materialized query tables (MQT)
- Bufferpool strategy

5.1 Introduction to VLDB

There is no widely accepted definition for VLDB. As recently as the mid-1990s, a decision support database of 300 to 500 gigabytes (GB) was considered a VLDB. By the late 1990s, databases of over a terabyte (TB) were not uncommon. With the growth of eBusiness and Web-driven commerce, data warehouses are growing even more rapidly. For this book, we define a VLDB as a database that exceeds 10 terabytes in total size.

5.2 Building a VLDB data warehouse

Building a database with 50 TB of raw data is a fairly big challenge in a test situation. It is an even bigger challenge in a production environment, where the batch window may last only a short time. According to WinterCorp, “the size of the world’s largest databases has tripled every two years since 2001.” For a discussion of VLDBs, see the Winter TopTen Program:

http://wintercorp.com/VLDB/2005_TopTen_Survey/TopTenProgram.html

In this chapter we discuss the tips and techniques we used in building our 50 TB VLDB database.

5.2.1 Storage management strategy

For our test, we chose to use the IBM Storage Management Subsystem (SMS) to manage the allocation of our DB2 data sets. We had three management options: user, DB2, or DFSMS.

- User

  A user can manually assign the VSAM clusters to a specific volume serial number, by running an IDCAMs job for each DB2 object.

```
DEFINE CLUSTER  ( NAME(VLDB29.DSNDBC.DSNDB07.DSN4K001.I0001.A001) -
        LINEAR   REUSE   CYL(5824 5824) CONTROLINTERVALSIZE(4096)
SHAREOPTIONS(3 3) ) -
  DATA  ( NAME(VLDB29.DSNDBD.DSNDB07.DSN4K001.I0001.A001))
  CATALOG(VLDB29)
```

- DB2

  We can allow DB2 to select the volume serial numbers for the VSAM clusters by creating a DB2 STOGRP with specific volumes in the create statement.

```
CREATE STOGROUP  .......  VOLUMES ( volser1, volser2, ....')
```

- DFSMS

  We can let SMS select the volume serial numbers for database allocation without assistance from DB2 by creating a STOGROUP and specifying ‘*’.

```
CREATE STOGROUP  .......  VOLUMES ( '*')
```

The user option gives you the most control over the physical storage of tables and indexes. But in a VLDB DW, with a large of number of objects to manage, this is not practical. In our test environment, table spaces alone needed over 14,000 data sets.

The benefits of using a DB2-managed storage group, are as follows:

- DB2 will allocate and delete data sets as needed when a table space and indexes are created and dropped.
- When needed, DB2 can extend individual data sets. For segmented and simple table spaces, DB2 might automatically create a new data set when it reaches its maximum size of 2 GB.
- When you reorganize a table space, DB2 deletes and then redefines associated data sets, reclaiming fragmented space.

A DB2-managed storage group also makes managing auxiliary storage much easier. However, the DBA still has to manage the DB2 storage groups and decide which volumes should be used for each DB2 storage group.

The DB2 storage group approach assigns storage until the first volume is full and then moves over to the next volume, so data are not spread across multiple volumes at a time.

The third option, an SMS-managed storage group, is both DB2 and SMS managed. This has all the advantages of a DB2-managed storage group, and it exploits SMS for data set management. DB2 allows SMS to manage the allocation, placement, and management of the data sets. The data set allocation is based on the SMS storage class (STORCLAS), data class (DATACLAS), and management class (MGMTCLAS) specifications. SMS automatic class selection (ACS) routines will determine which DATACLAS, MGMTCLAS, or STORCLAS to use for a particular data set, based on the high level qualifier (HLQ) of the name of the data set. By following standard naming conventions for table spaces and indexes, you can easily manage placement for DB2 data sets.

For our testing, we defined the SMS STORCLAS as in Table 5-1. We created an ACS routine that selected the appropriate STORCLAS for each data set. We decided to store user data, DB2 work files, DFSORT work files, and other data sets in separate SMS storage pools. This separation helps to plan for storage space and monitor and manage data growth easily

Each SMS storage pool was spread across a large number of disk volumes. This allowed the spreading of data across multiple disk volumes, which improves disk seek time.

Table 5-1 SMS storage classes

SMS STORCLAS	Purpose	Number of volumes	Size of volume (GB)	Total size (TB)
DB2EXAD	User table spaces and indexes	1908	42	80
DB2WORK	Work file database DSNDB07	540	51	28
DB2SORT	Sort work and temp data sets	540	51	28
DB2ARC	Flash copy backup	1908	42	80
PROJECT	Other data sets	108	51	6

There is another reason for using SMS-managed storage. Data sets must be SMS-managed if they are larger than 4 GB. Our largest two tables required partitions larger than 4 GB. So we defined the SMS data class (DATACLAS) with extended addressability (EA) and extended format (EF) specifications. See 4.2.4, “Storage configuration” on page 64 for information about how a DS8000 can be organized to achieve a high I/O throughput of over 3.7 GB/s.

Enhancement in DB2 9: In an environment with many DB2 subsystems and other subsystems (CICS, MQ, and so forth), ACS routines can get complex. DB2 9 has a new ease-of-use enhancement. In DB2 9, CREATE STOGROUP, has optional DATACLAS, MGMTCLAS, and STORCLAS parameters. Using these, DB2 can exploit SMS without an SMS ACS routine. We did not use this feature because our test environment was quite simple.

For our benchmark, DB2 storage groups (STOGROUPs) were created using DDL, similar to the one in Example 5-1.

Example 5-1 DB2 storage group DDL

```
CREATE STOGROUP TPCHSG VOLUMES ('*') VCAT VLDB29V;
```

If DFSMS is being used, then the parameter VOLUMES ('*') indicates that DB2 should use SMS. VLDB29V is the first level qualifier of data sets that DB2 will allocate. The SMS ACS routine will use the HLQ of the data set name to determine the STORCLAS. However, if the dataclas or storclas parameter is used, there is no need to specify the HLQ. When we created the stogroup for our database, we did not use DATACLAS or STORCLAS. We used VOLUME(*) and specified the HLQ.

Sliding secondary allocation size

In DB2 Version 8, IBM added a ZPARM called MGEXTSZ (with a global scope), which stands for optimize extent sizing. It enables the use of a "sliding secondary quantity for DB2 managed page sets. If MGEXTSZ is set to YES, DB2 will adjust the size of the secondary extent when extending the data set, in such a way that the maximum allowable data set size is reached before the maximum number of extents for that data set (255) is reached. The MGEXTSZ parameter only impacts the allocation of secondary extents. When calculating secondary quantities, DB2 uses a sliding scale. With the sliding scale, the initial secondary allocations are small and later secondary allocations are larger. This feature reduces the possibility of an outage caused by lack of space.

In our test environment, we set MGEXTSZ to YES.

5.2.2 Partition strategy

DB2 Version 9 supports partitioning of both the tables (table spaces) and indexes. The ability to split a table space and index among multiple partitions presents operational advantages.

Data partitioning

For our benchmark, we decided to use index-controlled partitioning. Through V7, DB2 supported only this type of partitioning of a table space. In this type, partitioning is enforced using a partition index (PI). In DB2 9 this is still supported. Even though table-controlled partitioning is newer and now recommended, we used the index partitioning because we built the database only once and did not update or add records. Table 5-2 on page 73 shows a comparison of both methods supported (in V8 and 9).

Table 5-2   Table versus index controlled partitioning

Table-controlled partitioning	**Index-controlled partitioning**
A partitioning index is not required Clustering index is not required.	A partitioning index is required Clustering index is required.
Multiple partitioned indexes can be created on a table space.	Only one partitioned index can be created on a table space.
A table space partition is identified by both a physical partition number and a logical partition number.	A table space partition is identified by a physical partition number.
The high-limit key is always enforced.	The high-limit key is not enforced if the table space is non-large.

We decided to use a partitioned table space to make use of the following advantages.

- DB2 will use a higher degree of parallelism for read-only queries, batch jobs, and utilities when batch jobs are run in parallel and each job goes after different partitions. Parallel processing is most efficient when you spread the partitions over different disk volumes and allow each I/O stream to operate on a separate channel.
- Utilities like LOAD and REORG can be run against selected partitions without affecting the availability of the rest of the partitions. This is easier to manage and it increases database availability. We had a large number of partitions because our table spaces were large.
- DB2 obtains locks at partition level for certain operations. This reduces lock contention and increases concurrency.
- Spreading data across multiple devices. Each partition of a table space can be assigned to a different storage group. We did not exploit this in our benchmark, because the SMS storage pools were already spread across a large number of disk volumes.
- Planning for growth. New partitions can be added as needed and the new partition becomes available immediately.

To achieve maximum parallelism in our benchmark, all large tables except LINEITEM were created with 2000 partitions. The 30 TB LINEITEM table was created with 4000 partitions.

Details: We used 2000 partitions for most tables because we needed a large number of partitions to hold all the data. We made it simple by using the same number of partitions for nearly all tables. The exception was LINEITEM, which needed more partitions due to the size of each partitioned index object.

Even the small tables like NATION were partitioned. For example, NATION had 25 rows and it was partitioned into 25 parts. We did this to increase the possibility of DB2 using read query parallelism, particularly if NATION happened to be the first table DB2 processes.

Table-controlled partitioning versus index-controlled partitioning

It is suggested to use table-controlled partitioning instead of index-controlled partitioning. Table-controlled partitioning will eventually replace index-controlled partitioning. We used index-controlled partitioning for our tests because we only built the database once and we did not update or add records into these tables.

Time dimensional partitioning

Choosing an appropriate partition key is important. The choice will impact the loading of the table, query performance, and the maintenance of the table. Every DW will have a time dimension. For some of the tables, partitioning the table on the time dimension (for example, one partition per week, or one partition per month) could be a good strategy.

Some points to consider:

- Most queries will have a time dimension. DB2 uses partition scanning instead of table scanning, resulting in less resource consumption and faster response.
- Consistent query response times over time. Adding history data does not lengthen query response time, because new data goes to new partitions.
- When data is added, data updates only happen to the latest partition. All other partitions are not affected. This increases data availability.
- After the first full backup, subsequent backups need to be done only for the last partition. This is a big advantage in a VLDB environment.
- When archiving less-used data, you can archive by partition, oldest first. You do not need to scan through every partition looking for data to archive.
- Not everything is good about this approach, especially for parallelism. Typically, in a DW, most of the queries will be requests for recent data. This causes the queries to be sent to last the last few partitions, reducing degree of parallelism and thus affecting performance. This may also affect concurrency.

  We designed this study for high performance. In contrast, a production system might use a partition by time approach for operational efficiencies, with slightly less performance. This performance expense can be mitigated by finer granular partitions.

For our benchmark, we did not partition our tables on a time dimension.

Page size

Unlike an OLTP environment, DW queries are dominated by prefetch accesses. For sequential prefetch, a large page size is an advantage, because DB2 can read a large volume of data in a single I/O. An OLTP environment generally is dominated by random reads. A smaller page size is better for OLTP workloads.

DB2 allows page sizes of 4 K, 8 K, 16 K, and 32 K. A bigger page is better for our tests. But DB2 has a limitation. One page can only hold a maximum of 255 rows. Therefore, if we use a large page size for a table with a small record length, we will be wasting a lot of space in each page. For each table space, we used the largest possible page size that did not waste too much space. For our study, we used an 8 K page size for tables that were less than 60 bytes and a 16 K page size for the rest. We used a 4 K page size for all indexes.

Example: The average row length of the Customer table after compression is 91 bytes. To determine the pagesize for this table, we used the following computation:

```
Pagesize = rowlength * 255
Pagesize =  91 bytes * 255 = 23,205 bytes
```

If we had chosen a pagesize of 32 K, we would have a lot of wasted disk space, because the maximum number of rows per page is only 255. We would have wasted almost 10000 bytes per page, as shown here:

```
Bytes wasted = 32,768 - 23,205 = 9,563 bytes
```

On the other hand, with a pagesize of 16 K, DB2 would be able to fit 180 rows on each page:

```
Bytes per page (16k) =  16, 384 bytes  /  91 bytes  =  180.04 rows
```

The size of the data page is determined by the buffer pool in which you define the table space.

Greater than 4 KB CI size

DB2 table spaces and index spaces are defined as VSAM linear data sets. Through DB2 V7 every page set has been allocated in control intervals of 4 KB, even though VSAM allows CI sizes as a multiple of 4 up to 32 KB for linear data sets, and DB2 has chained the CIs up to the required page size.

DB2 V8 introduces support for CI sizes of 8, 16, and 32 KB. The vary DS control interval (DSVCI) ZPARM specifies whether you want DB2-managed data sets to have a variable VSAM CI. If DSVCI is set to the default value of YES, all new table space page sets will be allocated by DB2 with a CI corresponding to the page size. The new CI sizes reduce integrity exposures, relieve some restrictions on concurrent copy and the use of striping, and provide the potential for reducing elapsed time for table space scans. For more information, see the abstract for IBM Redpaper 4187, *Disk Storage Access with DB2 for z/OS*, available at the following Web page:

http://www.redbooks.ibm.com/abstracts/redp4187.html

> **Note:** If DSVCI=YES (the default), DB2 creates a DB2-managed data set for a table space. This table space has a VSAM CI that corresponds to the bufferpool used for the table space. If DSVCI=NO, DB2 always uses a fixed CI size of 4 KB, regardless of bufferpools used by the table space.

In our test environment, we set DSVCI=YES.

Greater than 4 KB page size for indexes

Prior to DB2 9, the size of an index page was limited to 4 KB. The size of an index page limits the number of index keys that the index page can accommodate and can cause contention in indexes that split frequently. DB2 9 lifts these restrictions by offering expanded index page sizes of 8 KB, 16 KB, and 32 KB. An index page size that is greater than 4 KB accommodates more index keys per page and can reduce the frequency of index page splits. You can use the INDEXBP option on both CREATE DATABASE and ALTER DATABASE statements to specify 4 KB, 8 KB, 16 KB, or 32 KB index buffer pools. You can also use the BUFFERPOOL keyword on the CREATE INDEX statement to specify 8 KB, 16 KB, and 32 KB buffer pools. The larger page size also helps with the compression of indexes. We used the standard 4 K page size for our indexes. We used a 4 K page size for all indexes.

DSSIZE

DSSIZE specifies the maximum size of each partition. The maximum number of partitions for a table space depends on the page size and on the DSSIZE. For our tables, the number of partitions and page size and were determined as discussed earlier. We needed to use DSSIZE for this database because some partitions were larger than 4 GB. We chose a DSSIZE large enough to accommodate all rows, as shown in Table 5-3.

Table 5-3 Maximum number of partitions

DSSIZE	Page size			
	4 KB	8 KB	16 KB	32 KB
1 GB - 4 GB	4096	4096	4096	4096
8 GB	2048	4096	4096	4096
16 GB	1024	2048	4096	4096
32 GB	512	1024	2048	4096
64 GB	256	512	1024	2048

DSSIZE versus LARGE options: If DSSIZE or LARGE options are not specified in a create table space statement, the maximum size of table space is 64 GB. LARGE option identifies that each partition of a partitioned table space has a maximum partition size of 4 GB, which enables the table space to contain more than 64 GB of data. DSSIZE Specifies the maximum size for each partition (1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, and 64 GB). With DSSIZE, the NUMPARTS or MAXPARTITIONS clause must be specified. NUMPART specifies the number of partitions. MAXPARTITIONS is the maximum number of partitions in partition-by-growth table. The preferred method to specify a maximum partition size of 4 GB and larger is the DSSIZE clause. The LARGE clause is for compatibility of older releases of DB2.

Table 5-4 lists the tables we created, the page size, DSSIZE, and number of partitions.

Table 5-4 Tables used for the benchmark

Table Name	No. of rows	Max / Avg row size	Page size	DSSIZE	No. of parts
CUSTOMER	7.5 billion	252 / 93	16 KB	2 GB	2000
ITEM	300 billion	172 / 73	16 KB	16 GB	4000
VENDOR	500 million	219 / 90	16 KB	2 GB	2000
PURCHASE ORDER	75 billion	159 / 56	8 KB	8 GB	2000
ITEMPART	10 billion	189 / 58	8 KB	2 GB	2000
ITEMSUPP	40 billion	235 / 58	8 KB	4 GB	2000
TERRITORY	5	193 / 106	4 KB		5
COUNTRY	25	198 / 102	4 KB		25
LOCATION	163	44 / 31	4 KB		163

We used data compression to take advantage of space saving and query performance. See 5.2.3, “Compression strategy” on page 78. For a creating table space, we used a DDL like Example 5-2.

Example 5-2 table space DDL

```
CREATE TABLESPACE TSCUST IN TPCH50TB
  USING STOGROUP TPCHSG
  PRIQTY 363535
  DSSIZE 2G
  PCTFREE 0 FREEPAGE 0
  NUMPARTS 2000 COMPRESS YES
  BUFFERPOOL BP16K1;
```

Tables were created using DDLs like Example 5-3.

Example 5-3 Table DDL

```
CREATE TABLE CUSTOMER
  (
  C_CUSTKEY BIGINT NOT NULL,
  C_NAME VARCHAR(25),
  C_ADDRESS VARCHAR(40),
  C_GEOKEY INTEGER,
  C_NATIONKEY INTEGER,
  C_PHONE CHAR(15),
  C_ACCTBAL DECIMAL(12,2),
  C_MKTSEGMENT CHAR(10),
  C_COMMENT VARCHAR(117)
  )
  IN TPCH50TB.TSCUST;
```

Partition indexes were created using DDLs like Example 5-4.

Example 5-4 Partition index DDL

```
CREATE UNIQUE INDEX C_CK ON CUSTOMER
  (C_CUSTKEY, C_NATIONKEY, C_MKTSEGMENT)
  BUFFERPOOL BP2
  USING STOGROUP TPCHSG
  PRIQTY 83000
  FREEPAGE 0 PCTFREE 0
  CLUSTER
  (
     PART  1   VALUES (3750000     ),
     PART  2   VALUES (7500000     ),
     PART  3   VALUES (11250000    ),
     PART  4   VALUES (15000000    ),
     PART  5   VALUES (18750000    ),
     PART  6   VALUES (22500000    ),
     ...............................,
     ...............................,
     ...............................,
     PART 1995   VALUES (7481250000),
     PART 1996   VALUES (7485000000),
     PART 1997   VALUES (7488750000),
     PART 1998   VALUES (7492500000),
     PART 1999   VALUES (7496250000),
     PART 2000   VALUES (9999999999));
```

Universal tablespace

Universal tablespace (UTS) is a new tablespace type introduced in DB2 9. UTS is a combination of partitioned and segmented tablespace schemes. Unlike a segmented table space, a UTS can only hold one table. Some benefits of universal tablespaces:

- Better space management in conjunction with varying-length rows. This is because a segmented space map page has more information about free space than a partitioned space map page.
- Improved mass delete performance. Mass delete in a segmented tablespace organization tends to be faster than in other types of tablespace organizations like partitioned or simple table spaces. Although in DB2 9 you can no longer create simple table spaces, DB2 continues to support existing ones.
- DB2 reuses all or most of the segments of a table after you commit the dropping or mass delete of it.

DB2 supports two types of UTS:

- Range-partitioned UTS

  Range-partitioned tablespaces are based on partitioning ranges, A range-partitioned universal tablespace uses the segmented space map page organization. Partition ranges are implemented using table-controlled partitioning syntax. The maximum size of a range-partitioned universal tablespace is 128 TB.

- Partition-by-growth UTS

  Partition-by-growth table spaces divide the available space into separate partitions, but are managed by DB2. The table space begins as a single-partition table space and DB2 automatically adds a new partition when it needs more space to satisfy an insert. The MAXPARTITIONS parameter in the CREATE TABLESPACE, specifies the maximum number of partitions that a partition-by-growth table space can grow to. A partition-by-growth table space can grow up to 128 TB.

Index partitioning

Up through DB2 V7, an index could not be physically partitioned. DB2 V8 introduced partitioning for a secondary index. When an index is partitioned, it is partitioned in the same way as the underlying data in the table.

Data-partitioned secondary indexes (DPSI)

A data-partitioned secondary index is a non partitioning index that is physically partitioned according to the partitioning scheme of the table. Characteristics include these:

- A DPSI has as many partitions as the number of partitions in the table space.
- Each DPSI partition contains keys for the rows of the corresponding tablespace partition only.

The use of partitioned indexes promotes partition independence. Therefore, it can reduce lock contention and improve index availability, especially for utility processing, partition-level operations (such as dropping or rotating partitions), and recovery of indexes.

However, the use of data-partitioned secondary indexes does not always improve the performance of queries. For example, for a query with a predicate that references only the columns of a data-partitioned secondary index, DB2 must probe each partition of the index for values that satisfy the predicate if index access is chosen as the access path. Take into account data access patterns and maintenance practices when deciding to use a data-partitioned secondary index. Replace a non-partitioned index with a partitioned index only if there are perceivable benefits such as improved data or index availability, easier data or index maintenance, or improved performance.

Non-partitioned secondary indexes (NPSI)

A non-partitioned secondary index is any index that is not defined as a partitioning index or a partitioned index. A NPSI has one index space that contains keys for the rows of all partitions of the table space.

For performance reasons, we decided use NPSI instead of DPSI.

5.2.3 Compression strategy

As the name implies, VLDBs will have large volumes of data. Typically, data warehouses tend to have a large number of indexes. Therefore, both data and index compression are critical in a VLDB DW.

Data

Data compression in DB2 has been around since V3. It uses the hardware compression feature of System z. We suggest use of data compression on a table space if the table space has the following characteristics:

- Is large enough to provide a significant space savings (for instance larger than 10 MB) or pages saved is greater than 5 MB.
- Compresses by at least 30%.
- Can fit at least one more row on a page. That is, if the row length is less than compression ratio times the page size value.

With data compression, the buffer pool memory use generally improves, as buffers contain compressed data, and as getpages are reduced for sequential access. Data warehousing queries are dominated by sequential prefetch accesses, which generally benefit from DB2 compression. With data compression, data on disk, DB2 log, buffer pool, and image copies are compressed.

We decided to use data compression (tablespace compression) to take advantage of storage savings and improved query performance. Refer to Chapter 7, “Data and index compression” on page 105 for a detailed study of the space saving effects of data compression.

Index

DB2 9 introduced index compression. DB2 uses a hybrid compression algorithm that uses prefix compression, as well as other methods, to compress index pages. DB2 only compresses leaf pages, which represent the vast majority of the index space.

Unlike data compression, index compression is not assisted by hardware. Only data on the disk is compressed. Index compression is used only for storage savings. We did not initially use index compression for our benchmark study, because we wanted to form a baseline.

> **Note:** Unlike table compression, the index compression mechanisms do not require a compression dictionary. Because DB2 performs dictionary-less compression, newly created indexes may begin compressing their contents immediately.

5.2.4 Loading the tables

For our tests we needed to load 50 TB of raw data. This was one of our most critical activities. The data for the DB2 tables was generated using an in-house data generation application. It created 50 TB of raw data while maintaining application relational integrity between the tables. For loading data, we decided on the following approach.

- Sort the input data set in the clustering sequence (in our case, partition key). Load does not sort data. This can avoid a reorg after the loading.
- Build secondary indexes after loading the base table and building the partitioned index. Building secondary indexes during load will slow down the load process. The indexes were created with DEFER=YES prior to the load activity.
- Load one partition of one table space per load job. Each job handled a much smaller volume of data, and had a much shorter elapsed time than loading the entire table in one job. Fixing errors and restarting jobs, in case of failure, went much faster.

  This approach created a large number of jobs to manage. We used in-house developed scripts for this. The scripts automated the generation of LOAD JCLs. We wanted to keep the CPUs at 100% busy, because this is the fastest way to load data.

- We did not use the SORTKEYS option. We only had one index (PI) created at load time and the input data was already sorted in ascending order of the index. Hence SORTKEY did not provide any advantages. Besides, our tables were so big that they used up all the virtual storage during sorting.
- Sort work data sets were not specified in the JCL. We avoided the manual calculation of sort work data sets by letting DB2 determine them, by setting UTSORTAL=YES in the DB2 system parameters (ZPARM). See the discussion below.
- We collected inline statistics while loading a table. To save running a RUNSTATS utility later on the table, we used the STATISTICS option of the LOAD utility. This collected and saved the statistics in DB2 catalog tables.
- To improve performance of our initial load, we turned off DB2 logging using the LOG NO option.
- The LOG NO option sets the COPY-pending restriction on the partition. This restriction is normally reset by taking an image copy of the partition. This is DB2's mechanism to ensure data recoverability. For our test environment, we were using FlashCopy backups instead of image copy backups. Hence we used NOCOPYPEND option on LOAD, which specifies that LOAD is not to set the table space in the COPY-pending status, even though LOG NO was specified.

The REXX™ scripts generated DB2 LOAD utility JCLs similar to Example 5-5.

Example 5-5 DB2 LOAD utility JCL

```
//LDP0020 JOB CLASS=A,TIME=NOLIMIT,REGION=0M,MSGCLASS=H,
//       MSGLEVEL=(1,1),USER=LSJCRUZ
/*JOBPARM SYSAFF=*
//************************************************************
//*** LOAD FOR PART BY PARTITION ****
//************************************************************
//JOBLIB DD DSN=VLDB29.SDSNLOAD,DISP=SHR
//LDP&0     EXEC  PGM=DSNUTILB,PARM='VLD9,LDP0020'
//UTPRINT  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
//SYSUT1   DD DISP=(NEW,DELETE,DELETE),
//            SPACE=(CYL,(1000,1000)),UNIT=3390
//SORTOUT  DD SPACE=(CYL,(2000,2000))
//SYSRECRA DD DISP=SHR,DSN=VLDBDATA.TPCP20.TBL.PART.N0020
//SYSIN    DD *
  LOAD DATA  LOG NO NOCOPYPEND
      STATISTICS TABLE (ALL) SAMPLE 20 INDEX (ALL)
      INDDN SYSRECRA
      SORTDEVT SYSDA
      EBCDIC  CCSID(00037,00000,00000)
      FORMAT DELIMITED COLDEL  X'4F'  DECPT   X'4B'
  INTO TABLE "TPCH50TB"."PART" PART 0020 REPLACE
     ( "P_PARTKEY"
      POSITION(*)            INTEGER
     , "P_NAME"
      POSITION(*)            VARCHAR
     , "P_MFGR"
      POSITION(*)            CHAR(00025)
     , "P_BRAND"
      POSITION(*)            CHAR(00010)
     , "P_TYPE"
      POSITION(*)            VARCHAR
```

```
, "P_SIZE"
 POSITION(*)           INTEGER
, "P_CONTAINER"
 POSITION(*)           CHAR(00010)
, "P_RETAILPRICE"
 POSITION(*)           DECIMAL
, "P_COMMENT"
 POSITION(*)           VARCHAR
)
```

Load using SORTKEYS

Using a SORTKEYS clause in the load statement causes keys extracted during the reload phase to be passed in memory. When sorting completes, the sorted keys needed for building indexes are passed in memory to the build phase. This allows for shorter elapsed time for load utility processing. The disadvantage is that the LOAD utility is not restartable following a failure during reload, sort, or build. Based on the restart requirements and load times available during the batch window, you need to decide whether to use a SORTKEYS clause. We did not use it, as explained on the previous page.

Sort work data sets in utility jobs

Computing the correct size of sort work data sets for utility jobs is challenging. If not computed correctly, it could result in jobs abending or wasting too much temp space.

Two new DB2 system parameters (ZPARMs), ‘utility sort data set allocation (UTSORTAL)’ and ‘ignore SORTNUM stat (IGNSORTN)’ have been introduced through APARs PK41899 (for DB2 9) and PK45916 (for DB2 V8) to solve this challenge. DB2 utilities that invoke a sort (CHECK, LOAD, REBUILD, REORG, and RUNSTATS) exploit these new parameters. If UTSORTAL is set to YES, DB2 will use real-time statistics to determine the sort work data set sizes and allocate them dynamically. If real-time statistics data are not available, DB2 will use a space prediction algorithm to compute the size. IGNSORTN will take effect only when UTSORTAL is set to YES. If IGNSORTN is set to YES, DB2 will ignore the SORTNUM specification in utility jobs.

5.2.5 Building non-partitioned secondary indexes

We used non-partitioned secondary indexes (NPSI). See “Index partitioning” on page 78. The indexes were created with the DEFER YES option. DEFER YES causes the description of the index to be added to the DB2 catalog. DB2 will not build the index but puts the NPSI in Rebuild Pending status. We used the REBUILD INDEX utilty to build the indexes. CREATE INDEX (with DEFER NO) uses the RDS Sort component to sort the keys, and REBUILD INDEX uses an external sort (DFSORT). The RDS Sort component is great for smaller sorts that require only a few sort runs, but when the table is of significant size, the external sort far outperforms it.

REBUILD INDEX has improved performance with parallel partition key extract and parallel index build. Beginning with DB2 V8, non-unique indexes offer yet another advantage for CREATE INDEX DEFER in that the data is still available. A new feature of REBUILD INDEX SHRLEVEL CHANGE in DB2 9 enables these non-unique indexes to be built with a minimum of data outages. These advantages are not available with SQL CREATE INDEX. NPSIs were created using DDLs like Example 5-6 on page 82.

Example 5-6 Non-partitioned secondary index DDL

```
CREATE INDEX C_NK ON CUSTOMER
   (C_NATIONKEY ASC)
   USING STOGROUP TPCHSG
   PRIQTY  1048576 SECQTY 104858
   FREEPAGE 0 PCTFREE 0
   PIECESIZE 1G
   BUFFERPOOL BP2
   DEFER YES;
```

Secondary indexes were built using the DB2 REBUILD INDEX utility. We used JCLs similar to Example 5-7 for building indexes.

Because we were building large indexes, we needed large sort work data sets. The default maximum size of a sequential data set is 65,535 tracks. To create larger data sets, we specified DSNTYPE=LARGE in the DD card for each sortwork file. This will create large format sequential data sets.

HIPRMAX=0. HIPERMAX specifies (for DFSORT) the maximum amount of Hiperspace™ to be used for Hipersorting. Hiperspace resides in expanded storage, or in central storage for 64-bit real mode, and is backed by auxiliary storage (if necessary). In a normal sorting, I/O processing is reduced for Hipersorting, which improves elapsed time at the cost of some CPU time.

To avoid a runstats run after the building index, we used inline statistics collection while rebuilding the index (Example 5-7) , using the STATISTICS option.

Example 5-7 REBUILD INDEX Job JCL

```
//NPICNK JOB CLASS=A,TIME=NOLIMIT,
//         REGION=5M,MSGCLASS=H,MSGLEVEL=(1,1)
/*JOBPARM SYSAFF=*
//JOBLIB DD DSN=VLDB29.SDSNLOAD,DISP=SHR
//***************************************************
//      EXEC  PGM=DSNUTILB,
//    PARM='VLD9,NPICNK,,STATSLVL(SUBPROCESS)'
//UTPRINT  DD SYSOUT=*
//STPRIN01 DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=A
//*  DATASET FOR SORT INPUT & OUTPUT FOR INLINE STATS
//ST01WK01 DD UNIT=SYSALLDA,SPACE=(CYL,(1000,1000)),
//     DISP=(NEW,DELETE,DELETE)
//SW01WK01 DD UNIT=SYSDA,SPACE=(CYL,(60000)),
//     DISP=(NEW,DELETE,DELETE),DSNTYPE=LARGE
//SW01WK02 DD UNIT=SYSDA,SPACE=(CYL,(60000)),
//     DISP=(NEW,DELETE,DELETE),DSNTYPE=LARGE
//SW02WK01 DD UNIT=SYSDA,SPACE=(CYL,(60000)),
//     DISP=(NEW,DELETE,DELETE),DSNTYPE=LARGE
//SW02WK02 DD UNIT=SYSDA,SPACE=(CYL,(60000)),
//     DISP=(NEW,DELETE,DELETE),DSNTYPE=LARGE
.............................................
.............................................
.............................................
//SW18WK02 DD UNIT=SYSDA,SPACE=(CYL,(60000)),
//     DISP=(NEW,DELETE,DELETE),DSNTYPE=LARGE
//*
//SORTDIAG DD DUMMY
//DFSPARM DD *
```

```
 HIPRMAX=0
/*
//SYSIN    DD *
 REBUILD INDEX (TPCH50TB.C_NK)
 SORTDEVT SYSDA
 STATISTICS KEYCARD
```

5.2.6 Building materialized query tables (MQTs)

MQTs can save a lot of processing time for queries that are frequently executed, because results are pre-tabulated and queries do not have to plow through millions of rows each time. An MQT is a real (materialized) table built from the result set of a query. Like any other table an MQT can have indexes. Use RUNSTATS to collect and store table statistics. MQTs primarily exist for performance. Properly selected, MQTs perform a set of calculations that can then be used over and over by subsequent queries, but without going through the same operations again and again. The qualifying rows do not need to be fetched again, saving input time. Also, the calculations do not need to be redone, which saves CPU time.

MQTs can be either user-maintained or DB2-maintained. For DB2-maintained MQTs, DB2 will update MQTs as new data is added to the the underlying table(s).

Refer to Chapter 10, "Using materialized query tables" on page 183 for a detailed discussion about the MQTs we built, how we loaded them, and the query performance they provided.

5.2.7 Bufferpool strategy

The way that data is assigned to buffer pools can have a significant impact on performance. Here are some of the guidelines we followed.

DB2 handles large buffer pools efficiently. Searching in large buffer pools does not use any more processor resources than searching in smaller pools. To avoid paging, it is suggested that you set the total buffer pool size to less than the real storage that is available to DB2. In general, larger buffer pool sizes provide the following advantages:

- Result in a higher bufferpool hit ratio, which can reduce the number of I/O operations.
- Achieve higher transaction rates with the same response time.
- Prevent I/O contention for the most frequently used disks, particularly the catalog tables and frequently referenced user tables and indexes.
- Be beneficial when a DB2 sort is used during a query, because I/O contention on the disks containing the work file table spaces is reduced.

We suggest using multiple bufferpools for each page size, especially where they are used for different components. You can isolate data in separate buffer pools to favor certain applications, data, and indexes. However, if you allocate too many, memory is used less efficiently and tuning is harder. In a production environment, avoid small bufferpools and allocate 4–7 per page size. The benefits are as follows:

- You can favor certain data and indexes by assigning more buffers. For example, you might improve the performance of large buffer pools by putting indexes into separate pools from data.
- You can customize buffer pool tuning parameters to match the characteristics of the data. For example, you might want to put tables and indexes that are updated frequently into a buffer pool with different characteristics from those that are frequently accessed but infrequently updated.

- You can put work files into separate buffer pools. Doing this provides better performance for sort-intensive queries. Applications that use created temporary tables use work files for those tables. Keeping work files separate allows you to monitor temporary table activity more easily. It is also common to pin small tables and indexes into their own bufferpools. This is quite helpful for BI workloads because look up and dimension tables are common in BI processing.
- This process of segregating different activities and data into separate buffer pools has the advantage of providing good and relatively inexpensive performance diagnosis data from statistics and accounting traces.

We computed the minimum number of pages (VPSIZE) for each bufferpool using the following formula:

```
VPSIZE = 64 * # of parallel tasks * # of concurrent tables (assigned to the same
BP) being accessed within a query * # of concurrent queries
```

We allocated a proportionately larger number of bufferpools for larger tables. Table 5-5 shows the bufferpool allocation we used for our benchmark. A production environment, with more dynamic applications, would probably use fewer.

VPSEQT and VPPSEQT

DW queries are dominated by sequential prefetch accesses. Setting sequential steal threshold (VPSEQT) to a high value will benefit sequential prefetch. We set the (VPSEQT) to 90% and virtual buffer pool parallel sequential threshold (VPPSEQT) to 100%. See Table 5-5.

Table 5-5 Bufferpool allocation

Bufferpool	Object	VPSIZE	VPSEQT	VPPSEQT
BP0	DB2 Catalog	20000	80	50
BP1	Small table space	20000	90	100
BP2	Small Indexspace	20000	90	100
BP4	Partitioning Index CUSTOMER	300000	90	100
BP5	Partitioning Index LINEITEM	300000	90	100
BP7	SORT Work File	1000000	90	100
BP8	Partitioning Index ORDERS	300000	90	100
BP10	Partitioning Index PART	300000	90	100
BP11	Partitioning Index PARTSUPP	300000	90	100
BP12	Partitioning Index SUPPLIER	300000	90	100
BP20	Secondary Index CUSTOMER	800000	90	100
BP21	Secondary Indexs LINEITEM	2000000	90	100
BP22	Secondary Indexs ORDERS	1500000	90	100
BP23	Secondary Indexs PART	800000	90	100
BP24	Secondary Indexs PARTSUPP	1000000	90	100
BP25	Secondary Indexs SUPPLIER	800000	90	100
BP8K0	DB2 catalog	108000	80	50
BP8K1	tablespace ORDERS	3000000	90	100

Bufferpool	Object	VPSIZE	VPSEQT	VPPSEQT
BP8K2	tablespace PART	1000000	90	100
BP8K3	tablespace PARTSUPP	2000000	90	100
BP16K0	DB2 catalog	10000	80	50
BP16K1	tablespace CUSTOMER	1000000	90	100
BP16K2	tablespace LINEITEM	4000000	90	100
BP16K3	tablespace SUPPLIER	1000000	90	100
BP32K	DB2 catalog	10000	80	50

Sequential steal threshold (VPSEQT)

VPSEQT is a percentage of the buffer pool that might be occupied by sequentially accessed pages. This threshold is checked before stealing a buffer for a sequentially accessed page instead of accessing the page in the buffer pool. If the threshold has been exceeded, DB2 tries to steal a buffer holding a sequentially accessed page rather than one holding a randomly accessed page. Setting the threshold to 0% would prevent any sequential pages from taking up space in the buffer pool. In this case, prefetch is disabled, and any sequentially accessed pages are discarded as soon as they are released. Setting the threshold to 100% allows sequential pages to monopolize the entire buffer pool; however, this is not recommended for real-world data.

Virtual buffer pool parallel sequential threshold (VPPSEQT)

VPPSEQT is a portion of the buffer pool that might be used to support parallel operations. It is measured as a percentage of the sequential steal threshold (VPSEQT). Setting VPPSEQT to zero disables parallel operation.

5.3 Information retention and archiving

Data warehouse databases are growing tremendously. Because of the need for historical data or legal requirements, terabytes of data are kept online. But not all of this data is accessed frequently, nor does it need to be kept on fast media. Archiving is the process of moving selected inactive data to another storage location that is accessed only when necessary. This way it is possible to maintain data as inexpensively and effectively as possible. Archiving and information retention not only reduce hardware costs, but also enhance systems performance while enabling organizations to better manage risk and streamline regulatory compliance.

5.3.1 Background

Archiving helps boost the performance by reclaiming disk space, increase manageability (faster recovery times, less objects to maintain), and shorten time for realization of projects. An accumulation of historical transactions, the advent of data warehousing, and other factors contribute to the accumulation of inactive data, which is less likely to be accessed. However, businesses cannot necessarily get rid of inactive data altogether for several reasons. Some might need the data to comply with government requirements. Other businesses might want the data because they anticipate building trend analysis. And many keep inactive data to maintain a complete history of their customers. As the size of a database grows, the percentage of inactive data often grows.

Archive is different from backup, in that archiving moves data to other media, while backup copies data. Archive retrievals are selective, whereas backups are not. Archives are application-oriented, while backups are data store-oriented. However, archives can be a component of a backup and recovery scheme. Another difference is that online backups are done with connections to a database while offline backups are done without any connections. Online archives can be accessed by an application but have slower response time, while offline archives use offline storage that needs to be mounted.

Terms

Types of archives include table and file archives. Table archives, commonly referred to as history tables, are stored in DB2 tables and accessed using SQL. File archives are stored in flat files in a different data store (for example, a different DBMS) from the active data. Choices when storing file archives include the following options:

- Disk (high cost)
- Tape (lower cost)
- Optical (lower cost)

Table archives are used when fast or SQL access to the archived data is necessary. File archives, which can be burned on CDs or stored on tape, are used when SQL access is not needed, and access is less time sensitive.

Referential constraints are important in archiving. Some referential constraints are enforced by DB2, while others are enforced by the application. When data is archived, the entire referential set of data is involved. Keeping track of application-enforced referential constraints becomes crucial. Archive metadata (information about the archived data) needs to be captured at the time of archiving to aid in retrieving the archived data later.

DB2 can help with your data warehouse's growth by taking advantage of UTSs partition-by-growth feature in DB2 9. DB2 for z/OS can also save you additional disk space by using DB2 hardware data compression for table archives. Compression commonly yields a 50% reduction in data size. For instance, if you compress the 30 GB of archived data, you reduce your disk requirements by 15 GB. You have now reduced your disk space requirements from 100 GB to only 65 GB, which results in a 35% savings.

Using IBM Optim Data Growth Solution to compress your archived DB2 data

You can save even more disk space by compressing your active data. When using DB2 hardware data compression, be aware of trade-offs that exist. For example, indexes are not compressed, tablespace scans can incur significant increases in CPU usage, and compression dictionaries consume DB2 address space memory. For more about compression, refer to Chapter 7, "Data and index compression" on page 105.

If your DB2 databases are growing large and contain data that you consider to be inactive, you can move, or archive, this data to DB2 archive tables or to archive flat files. By doing so, you move this data from highly used and possibly highly expensive data storage to a different, possibly less expensive, data store. At the same time, IBM Optim™ Data Growth Solution allows you to maintain access to the archived data, either for online applications that are run on a regular basis or for accessing it less frequently on another storage medium.

After you archive data, you might want to selectively retrieve part or all of the archived data. This process differs from backups, because you generally recover all data from backups.

A table archive exists as a set of DB2 tables to which data can be archived, yet quickly and frequently accessed. Table archives are maintained in the same data store as the active data and provide quick access to retrieval of the archived data. Table archives in relational databases are often referred to as history tables.

A file archive is a traditional flat-file data store to which data is archived and referenced or used infrequently. These archives bring an expected delay in data retrieval. However, the wide variety of choices for storage provides a method for greatly reducing costs. Tape and optical storage, for example, provide relatively lower storage costs than disk space.

Retrieving archived data

Being able to retrieve the data that you archive is an important part of any archiving scheme. Most businesses want the capability to access archives on demand; However, you can control access programmatically as well. Archived data is not usually retrieved to the original active data tables. You can use SQL to access data in table archives. When you create file archives, the data must be easily accessible as needed. You will not usually have to retrieve an entire archive unit. To implement retrieving capabilities for file archives, you can use a load utility or a 4GL program to browse the file archives. Consider the requirements for selectively retrieving from a single archive and from multiple archives. When retrieving from multiple archives, expect the schemas to vary.

The destinations for the retrieved data can be flexible, depending on how you implement your strategy. Destinations can include new or existing retrieve tables, or even active data tables.

Much of the inactive data is not unusable. However, it is less likely to be accessed. Most of the time the reasons to maintain inactive data could be the government requirements or to meet Sarbanes-Oxley compliance. Also, in a data warehouse environment much of the historical and low granularity data may not be accessed frequently. To achieve savings of storage, the cheaper media could be used.

Online storage

Online storage is best suited for applications that require constant instantaneous access to data, such as databases and frequently accessed user data. Primary storage stores most current data and data that is most often accessed. This data requires continuous availability and typically has high performance requirements. Business-critical data should be stored on a Fibre Channel (FC) disk implemented in enterprise-class storage solutions.

Online archiving

Online archiving is used for applications that require quicker access compared with offline storage (such as tape), but do not require the continuous, instantaneous access provided by online storage. Secondary storage stores business-important information, but can often tolerate lower performance and potentially slightly less than 24/7 availability. It can also be used to cache online storage for quicker backups to tape. Secondary storage represents a large percentage of a company's data and is an ideal fit for FATA technology. See "Positioning FATA with Fibre Channel disks" on page 65. Data storage implementations best suited to use FATA technology reside at the "near-line" or secondary location within the networked storage hierarchy and offer a cost-effective alternative to FC disks at that location. Positioned between online storage and offline storage, near-line storage or secondary storage is an optimal cost/performance solution for hosting cached backups and fixed data storage.

Offline archiving

Offline archiving is used for applications where infrequent serial access is required, such as archives for long-term storage. For this type of storage, tape remains the most economical solution. It has lowest MBps performance and require mount time.

5.3.2 Archiving strategy

An archiving strategy is a plan that ensures that you are archiving the correct data in the correct way. Archiving without an archiving strategy might result in poor overall system performance.

The most successful archiving implementations are those that are well-planned and take several important aspects into account. The following sections provide recommendations for creating a well-planned archiving strategy, as well as several pitfalls to avoid that can create problematic archiving schemes. Begin by assessing your needs for archiving. Answer the following questions:

- What data do we need to archive?
- When do we need to archive this data?
- How long do we need to maintain the archived data?
- Do we need SQL access to the archived data?
- Should the archived data be stored in DB2 table archives or in flat-file archives?
- Under which circumstances and how often will we need to retrieve archived data from either table archives or file archives?
- Do we require archived data that is stored in table archives to be moved to flat-file archives (in other words, do we require a multitiered archiving strategy)?

After you have answered these questions, you can begin to design your archiving strategy.

Recommendations for designing a successful archiving strategy

The success of your archiving strategy depends on a number of factors. The suggestions in this section can help you address these factors. To design a successful archiving strategy, make the following considerations:

- Assign an applications team to maintain control over the archiving operations for your installation.
- Understand the schemas, the data, and how they are accessed.
- Understand your archive retrieval requirements.
- Assess the retention requirements for archived data. That is, determine how long you need to keep archived data. This factor strongly influences the design of your strategy. For example, if you require some archived data to be accessed periodically, and other archived data to be accessed infrequently, you might determine that a multitiered strategy is required in which old archives can be deleted after a certain period of time.
- Identify truly active data (versus inactive data) early in the design process. If archived data later requires updating, it really becomes active data again. The assumption is that archived data will not be updated because once it is modified, it no longer represents an accurate snapshot of that data.
- Establish a consistent scheduling process for your archiving. The best strategies archive data on a regular basis to maintain storage goals and to help in retrieval consistency. You can still run single archive operations on an as-needed basis.
- Consider all users and their requirements for authorization to access the archived data.
- Understand that most archives involve multiple tables, and in many cases can include thousands of tables.

Archiving data can lower total cost of ownership (TCO) and save space. This is especially likely in a data warehouse environment with a large amount of historical data. Archiving allows you to maintain data as inexpensively as possible yet still have effective access to it.

5.3.3 Archiving tool: IBM Optim solution—Enterprise Data Management

IBM Optim Data Growth Solution provides a single solution suite for managing enterprise application data throughout every stage of the information life cycle. You can apply business rules to archive, subset, access, store, retain, and protect your enterprise data. Optim capabilities are based on a consistent and proven data management methodology that aligns with your business objectives and scales across applications, databases, operating systems, and hardware platforms.

IBM Optim supports all major enterprise databases and operating systems, including DB2, Oracle®, Sybase, SQL Server®, Informix®, IMS, VSAM, Windows, UNIX, Linux, and z/OS. It supports the key ERP and CRM applications in use today: Oracle E Business Suite, PeopleSoft® Enterprise, JD Edwards® EnterpriseOne, Siebel®, and Amdocs CRM, as well as your custom and packaged applications.

Optim's comprehensive database archiving capabilities (archive, browse, restore, and delete) support managing data in production environments and address critical business issues for data growth management, information governance, business continuity, data retention compliance, e-discovery, as well as application upgrades, migrations, and retirements. Optim's Test Data Management capabilities (extract/move, edit, access, compare, convert/mask) support managing data in non-production (development, testing, and training) environments and address business issues for improving application reliability and protecting data privacy.

Protecting privacy data in production and non-production environments

Depending on the industry, operations, and types of applications, many production and non-production databases process sensitive information. The challenge is to provide the appropriate protection, while meeting business needs and ensuring that data is managed on a need-to-know basis.

Sensitive application data must be protected, whether it resides in a production system or non-production (development, testing, and training) environment. Most production environments have established standard security and access restrictions at the network level, the application level, and the database level to protect against data breaches. Physical entry access controls can be extended by implementing multi-factor authentication schemes. However, these protective measures cannot be simply replicated across every environment because the methods that protect data in production may not necessarily meet the unique requirements for protecting non-production environments.

What makes non-production environments so vulnerable? The answer lies in the nature of how non-production databases are created and used. For example, applications must be tested outside the production environment so that when testing reveals application errors, the live production system is not affected. Realistic data is essential for testing application functionality and to ensure accuracy and reliability.

In addition, adequate application development, testing, or training coverage can require multiple environments. Most often, one or more non-production environments can be created by simply cloning copies of the production database. By definition, this means that sensitive information is propagated from a secure production environment to one or more vulnerable non-production environments.

Industry analysts recognize that data privacy in the application development, testing, and training, as well as other non-production environments, is essential. They also concur that as a best practice, masking or de-identifying the data is a viable approach. De-identifying data in non-production environments is simply the process of systematically removing, masking, or transforming data elements that could be used to identify an individual. Data de-identification enables developers, testers, and trainers to use realistic data and produce valid results, while still complying with privacy protection rules. Data that has been scrubbed or cleansed in such a manner is generally considered acceptable to use in non-production environments and ensures that even if the data is stolen, exposed, or lost, it will be of no use to anyone.

IBM Optim Test Data Management and Data Privacy Solutions provide comprehensive capabilities for improving testing efficiencies and de-identifying application data that can be used effectively across non-production environments. Optim's data masking technology preserves the integrity of the data and produces consistent and accurate results that reflect the application logic. Masked data can be propagated accurately across multiple non-production environments to generate valid test results.

For more information about Optim, see *DB2 Security and Compliance Solutions for Linux, UNIX, and Windows,* SG24-7555.

6

Balancing a data warehouse

The principle of balanced data warehouse systems is to balance carefully a defined combination of resources related to the data warehouse. These resources are as follows:

- Processors
- Memory
- I/O bandwidth
- Storage

The resources are balanced to make a system perform well with a lower cost. A balanced system avoids bottlenecks that limit overall performance. Balancing also reduces the risk of oversizing single components.

6.1 Balanced data warehouse systems

The goal of a balanced data warehouse system is to provide a prescriptive and quality approach based on data collected from this study. By using the methodology when implementing a data warehouse, you can reduce your total time to deploy and lower the total cost of ownership of the warehouse. The prescriptive approach used by this methodology minimizes the complexity of warehouse design and implementation through tested designs and practices that increase the quality and manageability of the warehouse.

The balanced data warehouse system provides many benefits:

- The methodology takes a complex concept and breaks it down into more easily understood steps.
- Scaling the warehouse is simplified. As business requirements change, the balanced warehouse system methodology simplifies the process to determine the resources required to grow the warehouse to meet new workload demands.
- Over time, workloads deployed on consistent balanced data warehouse systems will lead to improved sizing and capacity planning processes for the data warehouse.
- The balanced data warehouse system methodology provides a prescriptive approach to implementing a data warehouse solution in z/OS. The prescriptive approach is intended to reduce the risk of sizing, deployment, integration, and planning for growth.
- A consistent approach and configuration allows more focused quality testing, therefore reducing the number of variables and reducing risk.

For more information, see the IBM Infocenter article *What is the Balanced Configuration Unit (BCU)?*, at the following Web page:

http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.bcu.doc/c0012182.htm

6.2 The balanced data warehouse system methodology

This methodology should be used to implement all new data warehouses, as well as to redesign and grow existing warehouses. When you are defining growth and upgrading technologies, apply these concepts to balance the warehouses. This methodology should be considered for upgrading existing systems to maintain processor, memory, storage, and I/O configuration within the solution.

This methodology is derived from testing using all IBM equipment. But if your organization has storage components that are non-IBM, the methodology can still be applied.

6.2.1 What does balancing a data warehouse system mean?

As with all computing systems, a data warehouse is a collaboration of processors, memory, disk, operating system, database engine, applications, data model, and business requirements. A balanced collaboration is the key to success, where all the components are chosen to fit with each other for a mixture of compatibility, performance, and reliability reasons.

A data warehouse consists of one or more LPARs implemented on a shared-data architecture. With the exception of the storage component, this architecture provides a simple approach of designing, sizing, and configuring a system by looking at the resource and performance requirements for an LPAR. By providing guidelines for the amount of processing that can be satisfied by an LPAR with a given set of resources, you can effectively size the overall solution to meet your requirements. This building block approach makes it simpler to ensure that systems are configured in a balanced manner.

Avoid bottlenecks in computer systems

To some degree, bottlenecks exist in all computer systems. For example, if a server can drive a peak I/O capability of 50,000 I/O operations per second, yet it is attached to a storage subsystem that is only capable of 25,000 I/O operations per second, the server may be under-used for some extreme I/O intensive workloads. However, if the applications using the server are more CPU intensive, it may be possible to fully use all the processing power of the server. With these considerations in mind, system architects and designers endeavor to design a configuration to meet the performance requirements and service levels required of the system. This is exactly the approach advocated by the balanced data warehouse system methodology.

Based on Business Intelligence workloads

Intellectual capital has been gathered over the years that IBM has been architecting and implementing data warehouses. Experience has been gained about the ratios of system resources that are required to deliver a balanced data warehouse system. By using these ratios the systems will generally have the right balance of CPU performance, memory performance, and storage performance.

With a design based on these balanced system guidelines, there will always be situations that may not perform as optimally as though the solution had been specifically designed to match it. The ratios presented in this chapter are general guidelines. Some configuration customization may be appropriate to come up with the final resource requirements.

6.3 Memory

Buffer pools are the largest objects in DB2 virtual storage. It is important to support the buffer pools with sufficient central memory. If this is not achieved, paging takes place, which degrades performance significantly. It is a good practice to allocate sufficient central memory to eliminate paging activities. For systems with a small amount of central memory, it is better to use smaller buffer pools. Hit ratios are lower with smaller buffer pools and DB2 accesses data from external storage. However, database I/O operations are more efficient than paging. Sequential I/O operations are more prominent in data warehouse workloads with each sequential prefetch process accessing 256 K of data (DB2 9). In contrast, paging I/O takes place one page at a time because it is impossible to predict where the next page fault will occur. In essence, it is trading one sequential I/O operation with 64 random I/O operations, and it is clear sequential I/O operations come out to be the winner by a large margin.

Although virtual storage tuning is an ongoing exercise even with DB2 9, storage used for buffer pools has been moved above the 2 GB bar in V8. This enables implementation of large buffer pools, significantly larger than the 2 GB limit previously imposed. As mentioned previously, each page of a buffer pool should be backed by a page in central memory. As such, there is a practical limit with the current architecture of System z. For example, z10 supports a maximum of 1.5 TB of central memory. In theory, a system can be purchased with this amount of memory but the cost could be prohibitive for most customers. As a result, one of the objectives of this study was investigating the benefits of using various buffer pool sizes, including some that are larger than most installations.

6.3.1 Methodology

The objective was determining a relationship between CPU capacity and central memory (main storage) for various workloads. Tests performed using different workloads, and the number of processors was changed between measurements. Our memory configuration was also modified between runs. When memory increased, buffer pool sizes increased proportionally.

A key performance indicator is the buffer pool hit ratio. It is logical that hit ratios get better as buffer pool sizes increase, but one should not expect a linear relationship between the two variables. Beyond a threshold, the system reaches diminishing returns where increasing buffer pool sizes does not yield noticeable performance benefits. Because the buffer pools are backed by central memory, changes to buffer pool sizes reflect the effect of memory configuration changes. Hence, the buffer pool hit ratio is a good indication of memory effectiveness.

In our project, no attempts were made to optimize individual buffer pool sizes. When more memory becomes available, the optimal distribution method is increasing the size of each buffer pool by an amount proportional to its need. Some buffer pools with lower hit ratios may benefit from a large increase, but buffer pools experiencing higher hit ratios are probably not good candidates for an increase. This kind of tuning is time consuming. It is also heavily dependent on the workload being measured. It was felt that individual buffer pool tuning would not produce a general relationship that the study was looking for. However, in a production environment, tuning for a specific workload can lead to substantial performance improvements.

6.3.2 Measurement results

Table 6-1 shows the buffer pool hit ratios of batch report queries using an 8-CP configuration. This is a workload similar to the one used in the processor scalability measurements.

Table 6-1 Report Queries—8 CPs

Buffer Pool	16 GB	1 GB
BP2	100%	100%
BP4	25%	15%
BP5	5%	5%
BP7	100%	100%
BP8	92%	92%
BP10	91%	91%
BP11	20%	20%
BP12	67%	67%
BP22	46%	45%
BP8K1	-18%	-22%
BP8K2	0%	-1%
BP16K1	2%	1%
BP16K2	56%	56%
BP16K3	0%	0%

Although one scenario has significantly more memory, it does not lead to any noticeable performance differences. Other than BP4, which observes a 10% jump in hit ratio, the rest of the buffer pools exhibit near identical results.

The hit ratios of BP8K1 and BP8K2 are shown as negative numbers. The data comes from DB2PE accounting reports. There is a large amount of dynamic prefetch I/O operations in this buffer pool. Because a dynamic prefetch assumes pages read in are likely to be used by the requestor, there is no guarantee all pages brought in will be used. In this particular measurement, a good percentage of the pages are not required by the queries. As a result, the number of pages read is higher than the number of GETPAGES issued by the queries. When this happens, DB2PE will report the hit ratio as negative.

For this particular workload, 1 GB of memory is sufficient. By allocating 16× more memory, as shown in the middle column of Table 6-1 on page 94, buffer pool hit ratios are virtually identical. It can be concluded that an 8 CP LPAR requires no more than 1 GB of memory per buffer pool for this workload.

It is possible to reduce memory further until significant performance differences are observed. Configurations of 512 MB, 256 MB, and others, could be tried. In practice, 1 GB of memory is becoming available in many installations, so the study did not test smaller memory configurations.

Buffer pool hit ratios for 16 CP measurements are shown in Table 6-2. The numbers between the two columns are quite similar, suggesting that moving up from 1 GB to 16 GB of memory does not lead to any noticeable performance differences. Based on this set of data, one can conclude that 1 GB of memory is sufficient for 16 CPs. This is a step up in CPs from the previous table.

Table 6-2 Report Queries—16 CPs

Buffer Pool	**16 GB**	**1 GB**
BP2	100%	100%
BP4	25%	15%
BP5	5%	5%
BP7	100%	100%
BP8	92%	92%
BP10	91%	91%
BP11	21%	20%
BP12	66%	66%
BP22	47%	44%
BP8K1	-15%	-21%
BP8K2	0%	-1%
BP16K1	2%	1%
BP16K2	56%	56%
BP16K3	0%	0%

It can be argued that the gigantic size of the database affects the sensitivity of additional memory. Because the database occupies 50 TB of storage, 16 GB of memory constitutes only a tiny percentage of the database. In fact, increasing 1 GB to 16 GB reflects a bump from 0.002% to 0.032% in terms of memory to disk ratio. Although memory increased by 16-fold, these tiny percentages make it unlikely to see a big improvement in buffer pool hit ratios.

The largest amount of memory used in balanced warehouse system measurements was 16 GB. An even larger memory configuration with 64 GB was used for the processor scalability measurements. These two categories of measurements shared workloads that were virtually identical. Data from a processor scalability run was added to Table 6-3. As the table shows, only one additional buffer pool, BP16K1, showed incremental benefits. Compare the hit ratios between the 1 GB and 64 GB columns. The overall impression is that it is not cost-effective to increase memory by 64× to see minor improvements in hit ratios.

Table 6-3 Report Queries—16 CPs, multiple measurement categories

Buffer Pool	64 GB	16 GB	1 GB
BP2	100%	100%	100%
BP4	34%	25%	15%
BP5	5%	5%	5%
BP7	100%	100%	100%
BP8	93%	92%	92%
BP10	91%	91%	91%
BP11	21%	21%	20%
BP12	68%	66%	66%
BP22	47%	47%	44%
BP8K1	2%	-15%	-21%
BP8K2	1%	0%	-1%
BP16K1	15%	2%	1%
BP16K2	57%	56%	56%
BP16K3	7%	0%	0%

Table 6-4 on page 97 shows the addition of the Cognos query workload to the batch report queries. The Cognos workload consists of lighter weight queries with smaller CPU consumption. As a result, processing is dominated by the batch queries. As expected, results from these tests are similar to the data in Table 6-1 on page 94.

Table 6-4 Report queries and Cognos Queries—8 CPs

Buffer Pool	16 GB	1 GB
BP2	100%	100%
BP4	26%	16%
BP5	5%	5%
BP7	100%	100%
BP8	92%	92%
BP10	91%	91%
BP11	20%	20%
BP12	67%	67%
BP22	49%	47%
BP8K1	5%	3%
BP8K2	0%	-1%
BP16K1	4%	2%
BP16K2	56%	56%
BP16K3	0%	0%

Two analytic queries (Q02 and Q12) were tested to determine the impact of larger memory configurations. In contrast to the report queries used in previous tests, these queries reflect analysts' work flow. They are more ad hoc in nature and generally take more time to complete. They represent a workload with vastly different characteristics than the report queries. See Table 6-5.

Table 6-5 Analytics Queries—8 CPs

	Q02		Q12	
Buffer pool	**16 GB**	**1 GB**	**16 GB**	**1 GB**
BP2	100%	100%		
BP7			57%	44%
BP8			35%	35%
BP11	-85%	-87%		
BP12	83%	83%		
BP23	0%	0%		
BP8K2	0%	0%		
BP16K2			0%	0%
BP16K3	1%	1%		
BP16K3	0%	0%		

The results from these queries show a similar picture as the report queries. Most of the buffer pools exhibit little or no difference between 1 GB and 16 GB. Only one buffer pool, BP7, indicates a 13% improvement in hit ratio for Q12.

6.3.3 Recommendations

Results from all the test cases indicate 1 GB is sufficient for 16 CPs. However, there could be some elements of very large databases in play given the sizes of the buffer pools are only a tiny fraction of the database. In practice, larger buffer pool configurations are probably more appropriate. Also, different workloads have different data access characteristics. It is advisable to start with a smaller configuration, perform buffer pool monitoring and tuning, and add additional memory if necessary. For an 8 CP configuration, consider starting with 8 GB of memory and adjust up and down the sizes of the buffer pools based on monitoring results.

6.4 I/O capacity

I/O configuration plays a vital part in the performance characteristics of a data warehouse application. Not only is storage required to house the database, work files, and other objects, but also sufficient bandwidth is necessary to ensure that applications deliver good response times to users. Traditionally, capacity planning for a new application focused on CPU requirements only, while I/O requirements came as an afterthought, because processors are the most expensive resources. However, without careful consideration for I/O requirements, an installation may not achieve the full potential of a data warehouse system.

Many production organizations have departments that specialize in storage management. They assign storage space for database objects, but they might not take into account the I/O bandwidth. For convenience, they might use volumes with consecutive addresses or addresses close to one another and belong to the same logical control unit. This approach optimizes space use but it does not deliver the performance characteristics most applications require. When a DBA sizes a new application, in addition to total space requirements it is also important to convey I/O bandwidth requirements to the storage management department. Ideally, they work together to determine how the volumes would be spread across multiple logical control units, and for larger scale data warehouse deployment across multiple disk subsystems.

System z supports mixed workloads well. As opposed to using a dedicated server approach as in distributed platforms, it is common to add resources to an existing System z server for new applications or for growth of current applications. Ease of scalability is an advantage on the mainframe. To deploy a new data warehouse application, processors are simply added to an existing server for the additional capacity. Likewise, the purchase of a new disk subsystem or expansion of an existing disk subsystem may be performed for a new application. The addition of the new disk subsystem is not tied to the addition of new processors. In some cases sufficient I/O capacity is available in the existing disk subsystems so that no purchase of additional capacity is necessary. Clearly there is a relationship between processor and I/O capacity, but the advantage of a System z solution is that resources required for new data warehouse applications can be managed independently.

6.4.1 I/O operations

One measurement of I/O requirements is the number of I/O operations per second (IOPS). In System z, it refers to the number of start I/O operations to an I/O device. This is different from internal I/O operations measured by disk devices, which is measured at a more atomic level. This number is shown as the DASD Rate in an RMF Monitor I report.

IOPS is more prevalent for OLTP applications. These workloads access several records in a transaction and the amount of data retrieved is relatively small. On the read side, each I/O operation usually accesses a 4 K page. The pages accessed in a transaction are generally random. On the write side, DB2 typically batches the updated pages and writes them out with sequential I/O operations. But the read to write ratio of most applications is quite high, suggesting that random single page I/O operations dominate the access pattern.

A single DS8000 subsystem is capable of supporting tens of thousands of IOPS while delivering good response times. OLTP applications place a greater demand on this metric, and customer benchmarks have shown consistently IOPS is the appropriate measurement to gauge I/O performance. When IOPS increases, I/O response time goes up slightly until a threshold is reached and response time deteriorates quickly.

A data warehouse exhibits a different I/O access profile. It has been found that IOPS does not accurately measure the I/O response times of data warehouse applications. Instead sequential bandwidth turns out to be a better metric.

6.4.2 Sequential bandwidth

Sequential bandwidth measures the amount of data transfer in a unit of time for data that resides in adjacent locations. It is generally expressed in MBps or GBps. For sequential I/O operations, each operation transfers a large amount of data. With this approach, protocol overhead is reduced because each operation transfers significantly more data than a random I/O operation. In DB2 9, each sequential I/O operation fetches 256 KB of data, much larger than a typical 4 K page in a random I/O. For utilities, it goes even higher to 512 KB for each I/O operation.

Tablespace scans are common in data warehouse data accesses. A large amount of data is digested by queries to determine business trends. Insight about business can best be made if weeks or months of sales data are analyzed. All of these call for frequent sequential I/O operations in data warehouse queries.

The read-to-write ratio is even higher in data warehouse applications. Traditionally, other than daily or weekly refreshes, there are minimum write I/O operations to the warehouse. But this is changing with the advent of near real time data feeds. Operational data is copied to warehouse databases multiple times a day. To maintain high availability of services to users, load utilities are generally not used. Instead, ETL applications use SQL statements to insert data to a database. This decreases the read to write ratio, but for capacity planning read I/O sequential bandwidth is still the driving factor.

Disk subsystems normally do not provide equal read and write bandwidths for sequential I/O operations. For a DS8000 subsystem, there is a non-volatile cache to speed up write I/O operations. But this cache is significantly smaller than the general purpose cache. Also, there is staging logic to speed up sequential read operations.

It should be noted that these discussions center on total sequential bandwidth across disk subsystems. A large table space is carved up into many partitions and different queries access different partitions concurrently. Occasionally, there are hot spot partitions and tuning may be required. For data warehousing, total sequential bandwidth refers to the aggregation of I/O operations across all database objects. In contrast, the DB2 logging constraint in OLTP workloads refers to the sequential bandwidth of a single volume. Because I/O to the active log data set is sequential and there is only one active log data set for each DB2 image, maximum throughput depends on maximum single volume sequential write bandwidth.

6.4.3 Measurement results

Table 6-6 shows the results of multiple measurements to quantify the I/O requirements for various functions. For reference purposes, IOPS data is listed in the table under column "DASD Rate". A derived metric, I/O per MIPS, is calculated by dividing the number of I/O operations by the CPU capacity in a measurement. The maximum value is 2.2 across all measurements. This suggests a 5,000 MIPS LPAR requires 11,000 IOPS. A DS8000 disk subsystem is capable of much higher traffic. As mentioned previously, IOPS or any metric derived from it is not a good indicator of demand to the I/O subsystem for data warehouse workloads.

Table 6-6 I/O Bandwidth

	DASD Rate	I/O/MIPS	MBps	MBps/MIPS
Load	16,398	0.64	2,052	0.08
Scan	23,385	0.92	6,113	0.24
MQT system refresh	18,228	2.20	1,277	0.15
MQT user refresh	18,807	0.84	4,117	0.18
Q01	2,083	0.16	364	0.03
Q02	18,611	1.87	1,579	0.16
Q04	21,436	1.50	2,812	0.20
Q06	12,994	0.92	3,395	0.24
Q12	15,056	1.12	3,337	0.25
Report Queries	14,818	1.57	265	0.03

There are multiple workloads listed in the table. Loading refers to data loading of the line item table. Scanning relates to an I/O intensive query performing a tablespace scan of the line item table with minimum business logic. There are two approaches to populating an MQT, either controlled by DB2 internally or explicitly managed by users using the load with cursor technique. Both approaches were tested in the study. In addition, a collection of analytic queries were run to study the I/O effect. These queries accessed a sizable portion of each qualified table. For the final set of data, results from the memory tests described in this chapter were used to capture the I/O behavior. They are listed as Report Queries in the queries.

6.4.4 Load

Contrary to intuition, data loading is a CPU-intensive process. In this study, non-partitioned indexes were not built during the loading process. The build was deferred until all data was loaded to a table. Only the partitioned index was built during data loading. Because there was one partitioned index object for each partition, contention was minimized when multiple load jobs ran concurrently with each job acting on a separate partition.

Loading measurement data shows a requirement of 2,000 MBps split evenly between read and write I/O operations. Among all the categories of measurements, this is the only one with a heavy demand of write I/O operations. The others are mainly reads. Normalized by CPU capacity, it comes out to 0.08 MBps per MIPS. For a 5,000 MIPS LPAR, it comes out to 400 MBps, only a fraction of the full capacity of a DS8000 disk subsystem.

6.4.5 Scan

This is a purely theoretical exercise to learn how fast the study system could scan a table. It does not relate closely to any business queries as minimum business logic is involved. The scanning query consumed massive amount of CPU cycles due to a high degree of query parallelism. From an I/O perspective, it read in 6,000 MB of data per second, easily labeled as the most I/O-intensive test case. Because it consumed significant CPU cycles, the MBps per MIPS metric comes out to 0.24. For a 5,000 MIPS LPAR, it required 1,200 MBps bandwidth.

6.4.6 MQT refresh

Whether the system refresh or user refresh approach was used to populate a materialized query table (MQT), sequential bandwidth requirements were similar. This is probably due to similar logic exercised within DB2. There is more control during user refresh with respect to degree of parallelism. As a result, the user refresh process drove the processors to higher use and consumed more MIPS. In either case the sequential bandwidth requirements came in between 0.15 and 0.18 MBps per MIPS. For a 5,000 MIPS LPAR, it came to between 750 and 900 MBps. (MQT refresh using one of the replication tools would be different.) See Chapter 10, “Using materialized query tables” on page 183 for additional information.

6.4.7 Analytic queries

The demand for system resources by analytic queries is unpredictable. Different queries could access different tables with varying amounts of data. Even multiple instances of the same query can have different execution times. An instance asking for data from New York will fetch significantly more rows than an instance requesting data from Rhode Island.

The five test queries exhibited a wide range of sequential bandwidth requirements, ranging from 364 MBps for Q01 to 3,395 MBps for Q06. From this information, it is clear that Q01 is CPU bound with substantially more business logic. Normalized by CPU capacity, the range comes out to between 0.03 and 0.25 MBps per MIPS. For a 5,000 MIPS LPAR, the requirement is between 150 and 1,250 MBps.

6.4.8 Report queries

The report queries are similar to the ones used in processor scalability measurements. They were used in the memory measurements to determine CPU and memory relationship. They were lighter in weight and performed fewer scans. As such, their demand on sequential bandwidth was also less. They consumed 265 MBps sequential bandwidth during the test. Bandwidth per MIPS comes out to 0.03. For a 5,000 MIPS LPAR, the I/O requirement is 150 MBps.

6.4.9 Recommendations

A wide spectrum of sequential bandwidth was observed for the various workloads, ranging from 0.03 to 0.25 MBps per MIPS. One can plan for the peak load by allocating 0.25 MBps per MIPS for a data warehouse application. However, this may be excessive, given that peak requirements may only last for a short duration. A slightly lower value, such as 0.20 MBps per MIPS, could be a reasonable compromise. It would be a less expensive solution; however certain processes may run slightly below optimal speed.

Another factor influencing these numbers is the buffer pool hit ratios. In this study the database is of an enormous size, dwarfing the buffer pools. In a production environment, buffer pool hit ratios could be higher and a larger percentage of data could be found in the buffer pools. This cuts down the number of I/O operations and reduce sequential bandwidth requirements. One should take this into account during a sizing exercise.

6.5 Total storage space

The standard approach of calculating the size of a data warehouse uses the raw data size as input. In our study the raw data takes up 50 TB of data, hence it is called a 50 TB data warehouse study. To support a data warehouse in operation, whether in this study or in production, significantly more storage (disk space) is required. Not only is storage required to house the tables, but also additional storage is allocated for indexes, work files, and MQTs. It will be valuable if a ratio of total storage to raw data is available for planning purposes. This ratio varies across workloads, but general guidelines are useful. To this end, we provide suggestions here based on the results of this study as well as from customer experiences.

6.5.1 High availability

One technique of providing high availability of data is using data mirroring for DASD using RAID 1 architecture. This is expensive because a double amount of storage is required. So System z has been exploiting RAID 5 storage technique since the RAMAC® days. The latest technology DS8000 storage subsystems continue with the RAID 5 architecture. In this scheme, only a small fraction of additional storage is necessary to support parity information. Single failure of disks is compensated by the availability of parity data, and no outage is incurred. The amount of space available to applications depends on the configuration of a DS8000 disk subsystem. For sizing purposes, one should use the available storage rather than the total installed hard drive storage in the DS8000.

6.5.2 Data compression

Compression of table data is highly recommended for data warehouses given the prominence of table scanning. It also improves query performance because there are fewer pages to read from a table space. Compression savings is 50% or better for most tables. As a result, generally storage requirement for table data is only 50% of raw data or less.

In this study, the compression savings for the tables was between 43% and 63%.

6.5.3 Index compression

Index compression is a new function in DB2 9. It uses a different algorithm than data compression, and the savings varies widely. It also requires some tuning to determine the appropriate buffer pool size for a compressed index. With this tuning effort optimal storage savings can be achieved.

In this study, the compression savings for the indexes was between 12 and 61%.

6.5.4 Work files

Work files are mainly used by DB2 to sort data. It is difficult to come up with an equation to determine the amount of storage for work files. Complexity of queries, amount of data accessed, and concurrency level all affect work file storage requirements. Usage of MQT also adds to work file demand. Consider allocating 10% of the raw data size as work files. Monitor the usage of work files by queries and look for running out of storage condition, which shows up as unavailable resources with name 4 K if the work files are defined with 4 K page size. Add more work files as needed.

6.5.5 MQT

Storage usage for MQT is driven by the aggregation level of data. An MQT storing sales data aggregated at monthly level will have fewer rows. On the other hand, data aggregation at the daily level will have significantly more rows. The number of dimensions is also a strong factor in determining MQT size. For example, summarizing data by products and date only yields two dimensions. But requesting data by products, date, stores, and customers will create many more rows.

Another consideration is the number of MQTs in a data warehouse. Obviously presence of more MQTs will benefit a larger number of queries, but there is a price to it. Maintenance cycles take longer and the storage requirement goes up.

Little information is available to determine MQT storage requirement based on database size.

6.5.6 Recommendations

Our total storage requirement is probably about two to two-and-a-half times the raw data size with data compression. Compressed table data takes about 0.5× of raw data, indexes about 1×, work files and MQTs another 0.5× to 1× depending on query sort requirements, query concurrency, and usage of MQTs. With index compression, further savings could be realized. However, because this is a new feature in DB2 9, there is little customer data available to determine the actual savings and performance effects.

7

Data and index compression

Compression has been in use with databases for a long time and DB2 is no exception. DB2 for z/OS has used some form of compression since version 2 (software) and version 3 (hardware). With the introduction of DB2 for z/OS version 8, there has been a renewed interest in data compression. Today, with more customers looking to DB2 and System z as the database and platform of choice for hosting their data warehouse, compression can offer significant savings in terms of disk storage for data and indexes. The benefits of compression can be fully realized once you understand how it can impact your data warehouse.

This chapter describes data and index compression, the benefits compression offers, and how a data warehouse can take advantage of these benefits. In addition to describing data and index compression in detail, the tests conducted on smaller databases during the 50 TB data warehouse benchmark are examined.

7.1 Overview of compression

The use of hardware-assisted compression became available for table spaces on z/OS in DB2 Version 3 in 1993. Compression results in a reduction in the amount of storage required for your data warehouse. As a result of compression, you achieve an additional benefit of disk I/O reduction. Fewer I/O operations are required to access the same amount of data in the compressed form. However, the initial reason for choosing compression should be disk savings, with performance as an added benefit.

In data warehouse applications it is common to apply extensive data compression to table spaces, which reduces disk space requirements. On the other hand, to improve performance, it is also common to have several indexes defined on the table with the purpose of solving queries with index only access. Star schema implementations require specific indexes for supporting star schema processing such as star join and pair-wise access methods. It is not uncommon to end up with more disk space requirements for indexes than for table spaces.

DB2 9 for z/OS introduced the ability to create (and alter) indexes to also use compression. DB2 9 also introduced larger page sizes for indexes. DB2's index compression uses the new DB2 9 index page sizes that are larger than 4 K. The larger index pages sizes are necessary because DB2 9 index compression compresses only the index on disk. Index compression provides a solution to these requirements, helping you further manage the cost of ownership.

7.1.1 Compression techniques

Various techniques for compression have been developed over the last 30 years. Without going into all of the details, it is important to understand the two major forms of compression.

Lossless compression

Compression reduces the number of characters or bits stored to disk. What this means is that when data is compressed, the characters or bits are not all stored on disk. Instead, based on the technique used, specific characters or bits are stored.

Lossless compression implies you get back exactly what you started with after expanding the compressed data. Lossless compression thus does not guarantee compression of all input data, and the percentage of compression varies based on the input data.

Compression in data warehousing must be lossless. DB2 guarantees compression to be lossless, so anything compressed, data or indexes, uses lossless techniques.

Lossy compression

Contrary to lossless compression, lossy compression results in the loss of some information each time compression is applied. The decompressed data is close to the original, but not exactly the same.

Lossy compression can result in a much smaller file, at the cost of the original form. Thus, lossy compression is used by a number of applications. Examples of applications that use lossy compression are multimedia applications that compress sound, images, or videos so as to have a much smaller file. JPEG is an example of lossy compression.

7.1.2 Hardware and software compression

Compression can be implemented using either hardware or software. DB2 9 on z/OS uses hardware compression for data, but uses software compression for indexes. The difference between the two is described in the following sections.

Hardware compression

Hardware compression has the compression algorithms built into the hardware. Thus, minimal CPU overhead is required to compress and decompress data. A key point is that hardware compression keeps getting faster as chip speeds increase, although software compression speed increases at the same time.

Other advantages of hardware compression on System z are as follows:

- Reduces CPU overhead, saving valuable CPU bandwidth
- Higher data throughput
- Faster than software compression
- Less costly than software compression
- Runs as a black box, performing compression and decompression

CMPSC instruction

The CMPSC instruction is the connection between the application (in this case DB2) and the hardware-assisted compression. The instruction operands include the address and length of the character string, the location of the dictionary, and an indication of the operation (compression or expansion). For compression, the character string is examined and a symbol string is returned. For expansion, the symbol string is read and the expanded output is returned in the character string.

The CMPSC instruction compresses or expands data as specified by a bit in general register 0. CMPSC can be used to compress any randomly or sequentially used data, provided that there is some degree of repetition of character strings (which may be actual characters or bytes of numeric or graphic data) in the data, so that the data can be kept on disk in compressed form, thus saving disk space. CMPSC also has a symbol translation option that allows the instruction to be used to compress VTAM® network data.

Software compression

Software compression, as the name implies, uses software algorithms to compress the information. When compared with hardware compression, software compression has a higher CPU cost on System z. However, software compression is the most widely used form of compression, even on other IBM platforms such as System p, and it works well.

7.2 Data compression

Data compression, or table space compression, uses the Ziv-Lempel compression algorithm, which requires a dictionary. DB2 9 on z/OS uses the hardware compression feature described earlier to compress table spaces or the partitions of a partitioned table space.

Data compression offers significant benefits:

- Reduction in the storage space used by the data warehouse
- Reduced elapsed time for most data warehouse type queries
- Reduced I/O time
- More effective use of buffer pool space
- Higher buffer pool hit ratio under certain conditions

DB2 for z/OS compresses rows within a page, so that each data page consists of compressed rows. It uses the hardware instruction along with a data dictionary to give the most efficient compression available. The compressed data can also be encrypted, thereby saving space and implementing security requirements at the same time. The encryption tool was recently changed to be able to compress and encrypt efficiently. It will compress the data, then encrypt it.

With a 50% compression rate, a compressed page contains twice the rows that an uncompressed page would contain. This means that each I/O retrieves twice as much compressed data as it would if the data was uncompressed. The data remains compressed in the buffer pool, which means that DB2 for z/OS can cache twice as much compressed data in its buffer pool as it would if the data was uncompressed. Finally, when data is modified in a row that is compressed, the information logged about that data change is also compressed, thus reducing log volume for both the active logs and archive logs.

Not all data on a compressed page is decompressed; just the rows needed by the application. Combined with the use of the hardware instruction to perform the decompression, this serves to limit the amount of additional CPU needed to access compressed data.

The larger amount of data retrieved in each I/O gets compounded with the DB2 9 for z/OS increased prefetch quantities. This provides significant elapsed time reductions for all types of sequential processes, including the typical BI queries that make use of table scans and index range scans. This also includes sequential processes for utility access, providing benefits in terms of faster reorganizations, faster unloads, and faster recovery.

7.2.1 The dictionary

To perform data compression, DB2 uses a compression dictionary. DB2 also creates a decompression dictionary used for decompressing the data. A dictionary is stored as additional pages in a table space or table space partition and are cached to memory for quick access when the table space (or table space partition) is opened.

The dictionary usually occupies 64 K of storage, depending on the page size. That is, 16 pages at 4 K, 8 pages at 8 K, 4 pages at 16 K, or 2 pages at 32 K. In all currently supported versions of DB2, the storage used for the compression dictionary is taken from above the 2 GB bar.

Compression can be specified at the partition level for each table space. Thus, every partition could be specified to be compressed, resulting in as many dictionaries. If you have 4096 partitions, and all partitions are compressed, you would create 4096 dictionaries, taking a lot of valuable storage. Thus, when determining implementing compression, it is important to take into consideration all factors that could impact compression.

> **Note:** Index compression does not use a dictionary. Index compression, and how it works, is described in 7.3, “Index compression” on page 112.

Building the compression dictionary is one of the critical components of data compression. The better the dictionary reflects the data, the higher the compression rate achieved. The dictionary is built through use of either the LOAD or the REORG utilities, which are described in 7.2.2, “Activating data compression” on page 109. These are the only two utilities, and the only two ways, that can be used to create a dictionary.

Creating or rebuilding the dictionary can be a CPU-intensive task. If the existing dictionary results in an acceptable compression rate, it would not be recommended to rebuild the dictionary. It is important to always remember this when running a LOAD or a REORG, as the dictionary will be rebuilt on invocation of the utility. If the existing dictionary is good, then the KEEPDICTIONARY keyword can be used with the LOAD/REORG utilities to keep the existing dictionary and not create a new one.

Details on whether compression is active for an index or table space and metrics describing how effective compression is can be found in the DB2 catalog.

Compression Catalog Information

Catalog table	Column	Description
SYSINDEXES	COMPRESS	Compression is in use (Y) or not in use (N)
SYSTABLEPART	COMPRESS	“Y” compression in use “blank” compression not used Can be at partition level
SYSTABLEPART	*PAGESAVE	% pages saved 0 no savings + is an increase Includes overhead bytes
SYSTABLEPART	AVGROWLEN	Average row length with or without compression
SYSTABLES	PCTROWCOMP	% rows compressed within total rows active

*There are other columns in history tables

Figure 7-1 Compression details available in DB2 Catalog

7.2.2 Activating data compression

Compression is implemented in DB2 at the table space level. If a table space has multiple partitions, compression can be defined for a specific partition, for multiple partitions, or for all partitions.

A new table space is defined with the COMPRESS YES clause during the table space definition. If the table space exists, the table space must be altered with the ALTER command using the COMPRESS YES clause.

Once the table space has been defined ALTERed with the COMPRESS YES clause, a LOAD or a REORG has to be run so as to build the dictionary and implement compression in your database. Even with compression turned on, nothing is compressed in the table space until the dictionary has been created.

To turn off compression, the opposite is to be done. That is, the table space is altered using the COMPRESS NO clause and a LOAD or a REORG needs to be executed to decompress the data.

The following is a brief summary of the commands used for compression.

Create tablespace

The command used to define the table space as compressed during the initial table space definition is as displayed in Example 7-1 on page 110

Example 7-1 Create tablespace command

```
CREATE TABLESPACE tablespace IN database
...
...
COMPRESS YES
...
...;
```

Alter tablespace

The command used to change the table space to compressed if the table space exists is displayed in Example 7-2

Example 7-2 Alter tablespace command

```
ALTER TABLESPACE database.tablespace COMPRESS YES;
```

The LOAD utility

The LOAD utility can be used to create the compression dictionary and compress the rows in the table space. The compression dictionary and the compression of the rows are done during the reload phase of the LOAD utilities execution. The LOAD utility is designed to use the first “x” number of rows to create the compression dictionary. However, there are no rows compressed while the LOAD utility is building the dictionary. These rows are inserted into the table space as is, in its uncompressed format. Once the dictionary is created using the “x” number of rows, subsequent rows loaded are considered for compression. DB2 does not compress a row if the compressed row is longer than the original uncompressed row.

The REORG utility

The REORG utility can also be used to create the compression dictionary and compress the rows in the table space. The REORG utility creates a more accurate, and therefore more efficient, dictionary than the LOAD utility. This is because the REORG utility sees all the rows when building the dictionary during the UNLOAD phase. Unlike the LOAD utility, the REORG utility compresses all the rows in the table space during the RELOAD phase, as it has the compression dictionary available during the RELOAD phase.

Any row inserted after the dictionary is built will be compressed, assuming the compressed row is shorter than the original row. The REORG utility should thus be your first choice for building a better dictionary, resulting in better compression.

7.2.3 DSN1COMP

DB2 has a mechanism in place to help estimate the disk savings that could be achieved if compression is activated for a table space. DSN1COMP is a stand-alone utility provided by DB2. This utility estimates space savings that could possibly be achieved by DB2 data compression in both table spaces and indexes.

DSN1COMP can be run against the following types of data sets that contain uncompressed data:

- DB2 full image copy data sets
- VSAM data sets that contain DB2 table spaces
- Sequential data sets that contain DB2 table spaces (for example, DSN1COPY output)

DSN1COMP cannot be run against the following type of data sets

- Compressed data
- Data sets that contain LOB table spaces
- Catalog (DSNDB06)
- Directory (DSNDB01)
- Workfiles (DSNDB07)

DSN1COMP can be executed as a z/OS job and does not require authorization. However, if any of the data sets are RACF-protected, the authorization ID of the job must have the RACF authority for the data set. To ensure that DB2 has not allocated the DB2 data sets, it is suggested that the DB2 STOP DATABASE command be issued to stop the database. Using DSN1COMP with image copies or DSN1COPY outputs can make gathering compression information less intrusive.

When choosing a VSAM LDS option to run against, be careful if you are using an online REORG (REORG SHRLEVEL CHANGE). Online REORG flips between the I0001 and J0001 for the fifth qualifier of the VSAM data sets. You can query the IPREFIX column in SYSTABLEPART or SYSINDEXPART catalog tables to discover which qualifier is in use.

Time should be taken to use the correct execution time parameters to insure the results obtained are actually results that can be used. Keywords like PAGESIZE, DSSIZE, FREEPAGE, and PCTFREE should be set exactly as the object you are running DSN1COMP against to insure that estimates made by DSN1COMP are accurate. If dictionaries will be built using the REORG utility as suggested, REORG should be specified at DSN1COMP runtime. If not, DSN1COMP will assume the dictionary will be built by the LOAD utility. If a full image copy is being used as input to DSN1COMP, make sure the FULLCOPY keyword is specified to obtain the correct results.

The output from a DSN1COMP table space execution includes the following information:

- Statistics with and without compression and the percentage you should save in kilobytes
- Number of rows scanned to build the dictionary
- Number of rows processed to deliver the statistics in the report
- The average row length before and after compression
- The size of the dictionary in pages
- The size of the table space in pages, both before and after compression
- Percentage of pages that would have been saved

Example 7-3 shows the DSN1COMP report run against one of the table space partitions used during the 50 TB data warehousing benchmark study.

Example 7-3 DSN1COMP Report for table space

```
DSN1999I START OF DSN1COMP FOR JOB DSN1COMP STEP1
 DSN1998I INPUT DSNAME = dddddd.DSNDBD.tbsp1234.ts123456.I0001.A001    , VSAM
 DSN1944I DSN1COMP INPUT PARAMETERS
           4,096  DICTIONARY SIZE USED
               0  FREEPAGE VALUE USED
               0  PCTFREE VALUE USED
       1,000,000  ROWLIMIT REQUESTED
                  ESTIMATE BASED ON DB2 LOAD METHOD
             255  MAXROWS VALUE USED

 DSN1940I DSN1COMP COMPRESSION REPORT
         148,677  KB WITHOUT COMPRESSION
          69,260  KB WITH COMPRESSION
              53  PERCENT OF THE BYTES WOULD BE SAVED
```

```
          682  ROWS SCANNED TO BUILD DICTIONARY
    1,000,000  ROWS SCANNED TO PROVIDE COMPRESSION ESTIMATE
        4,096  DICTIONARY ENTRIES

          155  BYTES FOR AVERAGE UNCOMPRESSED ROW LENGTH
           73  BYTES FOR AVERAGE COMPRESSED ROW LENGTH

            4  DICTIONARY PAGES REQUIRED
        9,524  PAGES REQUIRED WITHOUT COMPRESSION
        4,469  PAGES REQUIRED WITH COMPRESSION
           53  PERCENT OF THE DB2 DATA PAGES WOULD BE SAVED

DSN1994I DSN1COMP COMPLETED SUCCESSFULLY,         9,474  PAGES PROCESSED
```

7.3 Index compression

Implementing index compression requires a good understanding of its principles. Because data compression has been around DB2 since version 3, it is easy to get confused when applying data compression analogies to index compression. In general, a data warehouse environment is a good candidate for index compression. As a guideline, you may consider index compression where a reduction in index storage consumption is more important than a possible increase on CPU consumption.

In this section we explain how index compression works and show its differences from data compression. We also examine the tests conducted on smaller databases during our 50 TB data warehouse benchmark. For more on index compression, refer to *Index Compression with DB2 9 for z/OS*, REDP-4345.

7.3.1 How index compression works—page compression

DB2 9 for z/OS index compression compresses at the page level whereas DB2's data compression for table spaces is at the row level. Index compression uses a hybrid compression algorithm that uses prefix compression, as well as other methods, to compress index pages. Index compression does not use a compression dictionary (unlike table compression).

Prefix compression is made possible by the fact that the keys on each index page are stored in sorted order from the left byte to right, either in ascending or descending order. The left-most bytes that are common from one key to the next constitute the prefix. The prefix length will vary throughout the index, but generally, the longer the prefix is, the better the compression will be. Also, some index page elements, such as the key map are no longer written to disk because this information can be reconstructed when the compressed page is read from disk again.

DB2 only compresses leaf pages, which represent the vast majority of the index space. Figure 7-2 on page 113 shows a schematic representation of the b-tree index structure. Only the leaf pages, in shadow, are compressed. Leaf pages are normally the large majority of the index pages. Note that in small indexes (as in small date), compression is not useful.

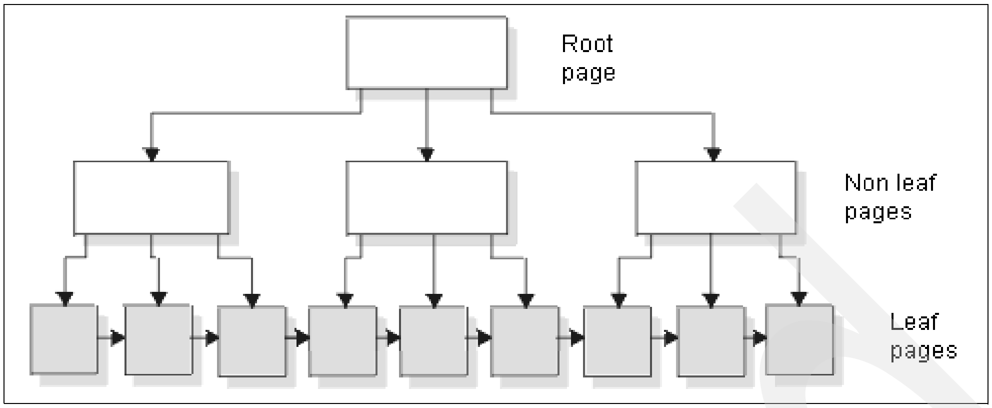


Figure 7-2 B-tree schematic representation

Index compression is made possible by the ability of creating indexes with page sizes bigger than 4 K. The index compression implementation is a hybrid approach:

- On disk, pages are always 4 K.
- You cannot define a compressed index in a 4 K buffer pool. Pages are stored on page sizes of 8 K, 16 K, or 32 K. The following pages detail the impact of buffer pool selection, as the maximum disk saving is influenced by this choice.

There are a few restrictions for index compression, as DB2 9 for z/OS does not support compression of the following indexes:

- Indexes in 4 K buffer pools
- Versioned indexes

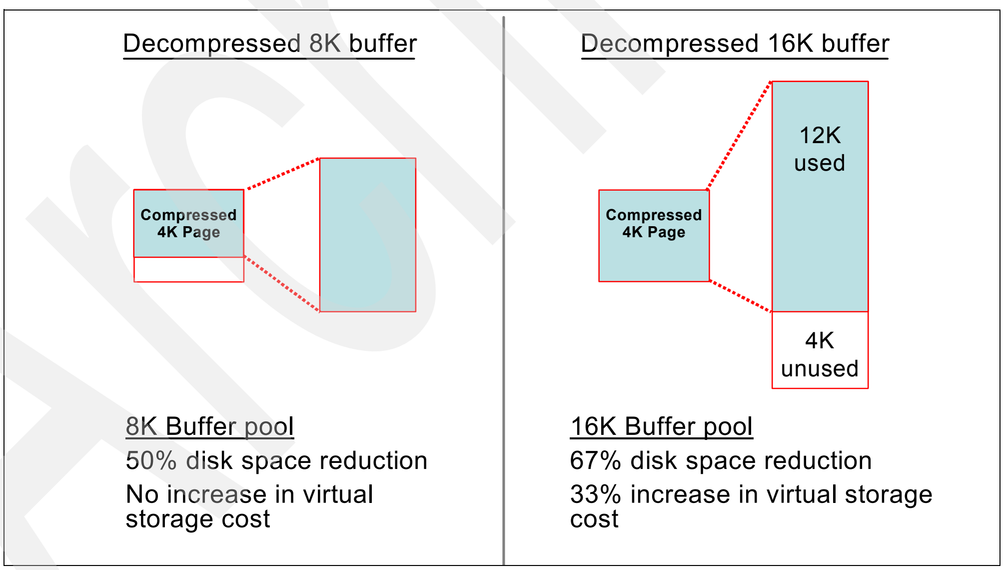


Figure 7-3 Decompress into 8 K or 16 K buffer

Unlike data compression, log records for index keys and image copies for indexes are not compressed. Index compression does not use the Ziv-Lempel algorithm, does not use a compression dictionary, and is not assisted by hardware. Buffer pools for indexes contain expanded pages. That is, pages are decompressed when read from disk. However, similar to data compression, an I/O bound scan will run faster.

Table 7-1 shows the major implementation differences between index and data compression.

Table 7-1 Comparing index and data compression

	Index compression	Data compression
Technique	Prefix compression	Ziv-Lempel hardware assisted
Dictionary	Not needed, compression starts immediately after definition of the index. Altering an index to COMPRESS YES requires rebuilding the index.	Must be built before getting data compressed
Level	Page	Row
Data on disk	Compressed	Compressed
Data on DB2 log	Not compressed	Compressed
Data in buffer pool	Not compressed	Compressed
Image copies	Not compressed	Compressed

For more information, see *Index Compression with DB2 9 for z/OS*, REDP-4345 at the following Web page:

http://www.redbooks.ibm.com/abstracts/redp4345.html

7.3.2 Activating index compression

Index compression is implemented in DB2 9 using the CREATE or ALTER index statements. It is similar to specifying compression for a table space, that is, using the COMPRESS YES/NO options in the DDL.

To turn off compression, the opposite is to be done. The index is to be altered using the COMPRESS NO clause.

When an index is altered to specify compression, the index is put in a REBUILD PENDING state. To use it, a REORG INDEX, REBUILD INDEX, or a REORG TABLESPACE has to be issued.

The following is a brief summary of the commands used for index compression.

Create index

The command to define the index as compressed during creation is shown in Example 7-4.

Example 7-4 Create index command

```
CREATE ... INDEX index-name
...
COMPRESS YES/NO
...;
```

Alter index

To change the index to compressed if it already exists, see Example 7-5.

Example 7-5 Alter index command

```
ALTER INDEX index-name COMPRESS YES;
OR
ALTER INDEX index-name COMPRESS NO;
```

Note: Remember to rebuild the index or REORG the index or table space once an ALTER is performed.

7.3.3 DSN1COMP

We discussed DSN1COMP in detail in 7.2.3, “DSN1COMP” on page 110. With DB2 9 for z/OS, DSN1COMP has been enhanced to support index compression. In this section we look at how DSN1COMP works on an index.

The output from a DSN1COMP index execution includes the following information:

- Number of leaf pages scanned
- Number of keys and RIDs processed
- How many kilobytes of key data were processed
- Number of kilobytes of compressed keys produced
- Reports broken down for possible percent reduction and buffer pool space usage for both 8 K and 16 K index leaf page sizes

One of the options to look at is the LEAFLIM option. The LEAFLIM option of DSN1COMP limits how many leaf pages are scanned by DSN1COMP. If this option is not specified, all leaf pages in the index are scanned. Obviously, not using the option could cause DSN1COMP to run longer.

7.4 Analysis and performance of the benchmark

Tests were executed against smaller databases during the 50 TB benchmarks to assess the effect of compression on table spaces and indexes. A limited set of performance measurements was taken during the execution of these tests.

Performance due to compression depends on a number of factors, which can result in a various compression results in different environments. The data contained in a table space or index can highly contribute towards the compression ratio and the order of columns chosen for indexes can affect the compression ratio. Thus, it is important to note that user experiences will vary for data and index compression, and the following results should be used to understand compression for very large databases.

We will see in the following pages the results achieved during the compression study. We start by describing the test philosophy and follow up with our results. We analyze space savings achieved due to data compression and due to index compression. We then look at the CPU and elapsed time results and analyze why we see interesting results for a few of the various queries that were executed.

We show here how compression can help you even if you use a large database. This holds not only for storage but also for saving CPU processing time.

7.4.1 Test philosophy

Queries for the tests were run against the databases using the following three methods:

- Against uncompressed table spaces and indexes
- Against compressed table spaces and uncompressed indexes
- Against compressed table spaces and indexes

To execute queries with the above philosophy, the environment was set up with three databases as follows:

- The first database contained uncompressed table spaces and uncompressed indexes. See Example 7-6 for the SQL on one of the tables.

Example 7-6 Uncompressed table spaces and indexes

```
CREATE TABLESPACE TSCUST IN TPCH100G
USING STOGROUP TPCHSG
PRIQTY 134766 SECQTY 13476
PCTFREE 0 FREEPAGE 0
COMPRESS NO
NUMPARTS 20
BUFFERPOOL BP16K1;

CREATE UNIQUE INDEX C_CK ON CUSTOMER
(C_CUSTKEY, C_NATIONKEY, C_MKTSEGMENT)
BUFFERPOOL BP4
USING STOGROUP TPCHSG
PRIQTY 1660 SECQTY 200
FREEPAGE 0 PCTFREE 0
CLUSTER PARTITIONED
COMPRESS NO;
```

- The second database contained compressed table spaces and uncompressed indexes. See Example 7-7 for the SQL on one of the tables.

Example 7-7 Compressed table spaces and uncompressed indexes

```
CREATE TABLESPACE TSCUST IN TPCH100H
USING STOGROUP TPCHSG
PRIQTY 134766 SECQTY 13476
PCTFREE 0 FREEPAGE 0
COMPRESS YES
NUMPARTS 20
BUFFERPOOL BP16K1;

CREATE UNIQUE INDEX C_CK ON CUSTOMER
(C_CUSTKEY, C_NATIONKEY, C_MKTSEGMENT)
BUFFERPOOL BP4
USING STOGROUP TPCHSG
PRIQTY 1660 SECQTY 200
FREEPAGE 0 PCTFREE 0
CLUSTER PARTITIONED
COMPRESS NO;
```

- The third database contained compressed table spaces and compressed indexes. See Example 7-8 on page 117 for the SQL on one of the tables.

Example 7-8 Compressed table spaces and indexes

```
CREATE TABLESPACE TSCUST IN TPCH100C
USING STOGROUP TPCHSG
PRIQTY 134766 SECQTY 13476
PCTFREE 0 FREEPAGE 0
COMPRESS YES
NUMPARTS 20
BUFFERPOOL BP16K1;

CREATE INDEX C_CK ON CUSTOMER
(C_CUSTKEY, C_NATIONKEY, C_MKTSEGMENT)
BUFFERPOOL BP16K4
USING STOGROUP TPCHSG
PRIQTY 1660 SECQTY 200
FREEPAGE 0 PCTFREE 0
CLUSTER PARTITIONED
COMPRESS YES;
```

Data was loaded onto the tables using the LOAD utility. The same sequential files were used as input for the load of the uncompressed and compressed tables. Example 7-9 shows the LOAD statement used to load the data onto the uncompressed table.

Example 7-9 LOAD statement used with an uncompressed table

```
LOAD DATA REPLACE LOG NO NOCOPYPEND
    STATISTICS TABLE (ALL) SAMPLE 20
    INDDN SYSRECRA SORTKEYS 25000000
    SORTDEVT SYSDA SORTNUM 12
    EBCDIC  CCSID(00037,00000,00000)
    FORMAT DELIMITED COLDEL  X'4F'  DECPT   X'4B'
INTO TABLE TPCH100G.CUSTOMER
```

Example 7-10 shows the LOAD statement used to load the data onto the compressed table.

Example 7-10 LOAD statement used with a compressed table

```
LOAD DATA REPLACE LOG NO NOCOPYPEND
    STATISTICS TABLE (ALL) SAMPLE 20
    INDDN SYSRECRA SORTKEYS 25000000
    SORTDEVT SYSDA SORTNUM 12
    EBCDIC  CCSID(00037,00000,00000)
    FORMAT DELIMITED COLDEL  X'4F'  DECPT   X'4B'
INTO TABLE TPCH100C.CUSTOMER
```

As you can see, both LOAD statements are exactly the same except for the database and table space they are being executed against.

7.4.2 Impact of compression on storage

Compression is used for table spaces and indexes to reduce disk space requirements. This is the primary concept behind using compression for our data warehouses. Compression offers other benefits, described earlier, but this section looks at the savings in terms of percentages we saw for tests conducted during this study.

Storage savings for table spaces

Figure 7-4 shows the percentage savings achieved on the various table spaces used for the compression tests. The percentage displayed is an average from the compression achieved across all the partitions of each table.

As we can see, the lowest storage savings achieved due to table space compression was 43%, whereas the highest storage savings achieved due to table space compression was 62%. We can thus see a range of savings. This range depends on a number of factors that could affect compression, as detailed in 7.2, “Data compression” on page 107.

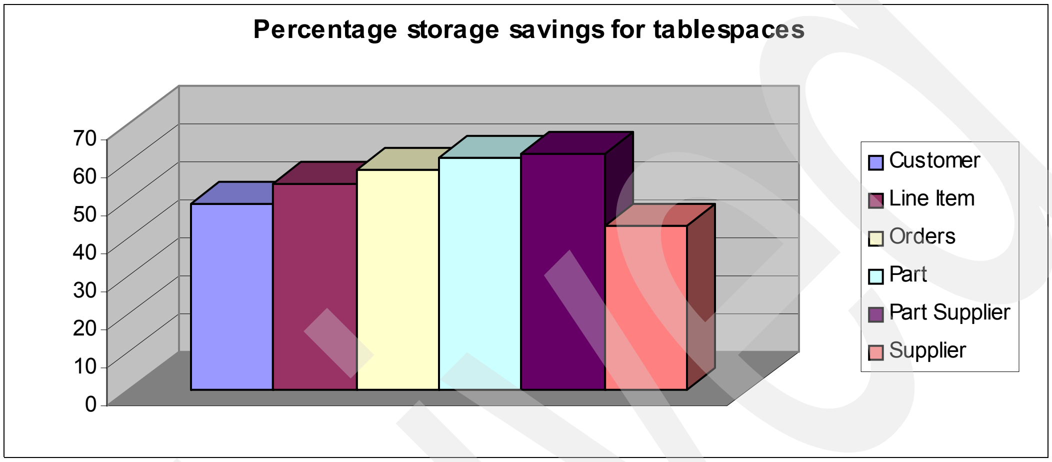


Figure 7-4 Percentage storage savings for table spaces

Storage savings for indexes

In our study, we compressed the indexes and saw 30.76% compression. We calculated this by building 100 GB databases to determine the savings. Figure 7-5 shows the percentage savings achieved on the various indexes on each table used for the compression tests. The number of indexes created on each table varied, and Figure 7-5 displays the percentage if an index was created for a table. Thus, the customer table had two indexes created in it, with the first index resulting in a storage savings of 22% and the second index resulting in a storage savings of 61%. On the other hand, the Line Item table had five indexes. The storage savings achieved for these indexes were 25%, 31%, 51%, 31%, and 30%.

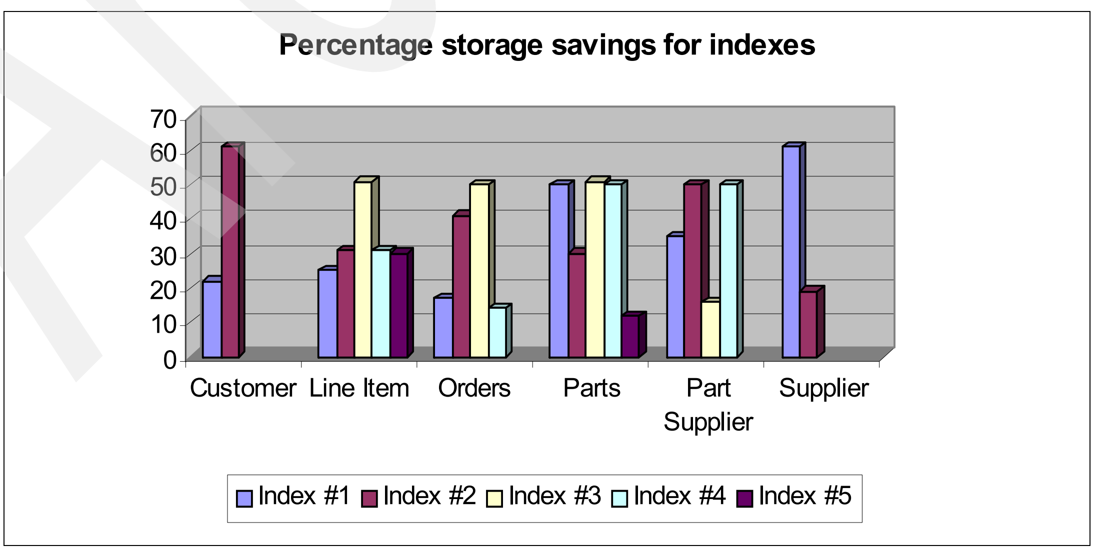


Figure 7-5 Percentage storage savings for indexes on the various tables

As we can see, the lowest storage savings achieved due to index compression was 14%, whereas the highest storage savings was 61%. We also see that for each table we have indexes with a storage savings due to index compression of close to 50%. Once again, this depends on a number of factors that could affect compression, as described in 7.3, "Index compression" on page 112.

7.4.3 Impact of compression on processing time

When considering index compression, take into consideration the impact on CPU time depending on the type of workload running. Unlike data compression, index compression has trade-off's in terms of CPU time based on workload. We look at the CPU time and elapsed time in this section while running a standard set of queries against the three databases we created and described in 7.4.1, "Test philosophy" on page 116.

Elapsed time observations

A set of runs were made to compare query performance with and without compression. Figure 7-6 shows the elapsed times and CPU times of the queries. There were three configurations in the testing. The left bars in each group indicate query performance in a configuration where there was no compression against the tables and indexes. The middle bars correspond to performance of queries against a database with compressed tables and uncompressed indexes. The third configuration utilized a database with compressed tables and indexes, and the performance of queries running in that configuration are depicted by the right bars.

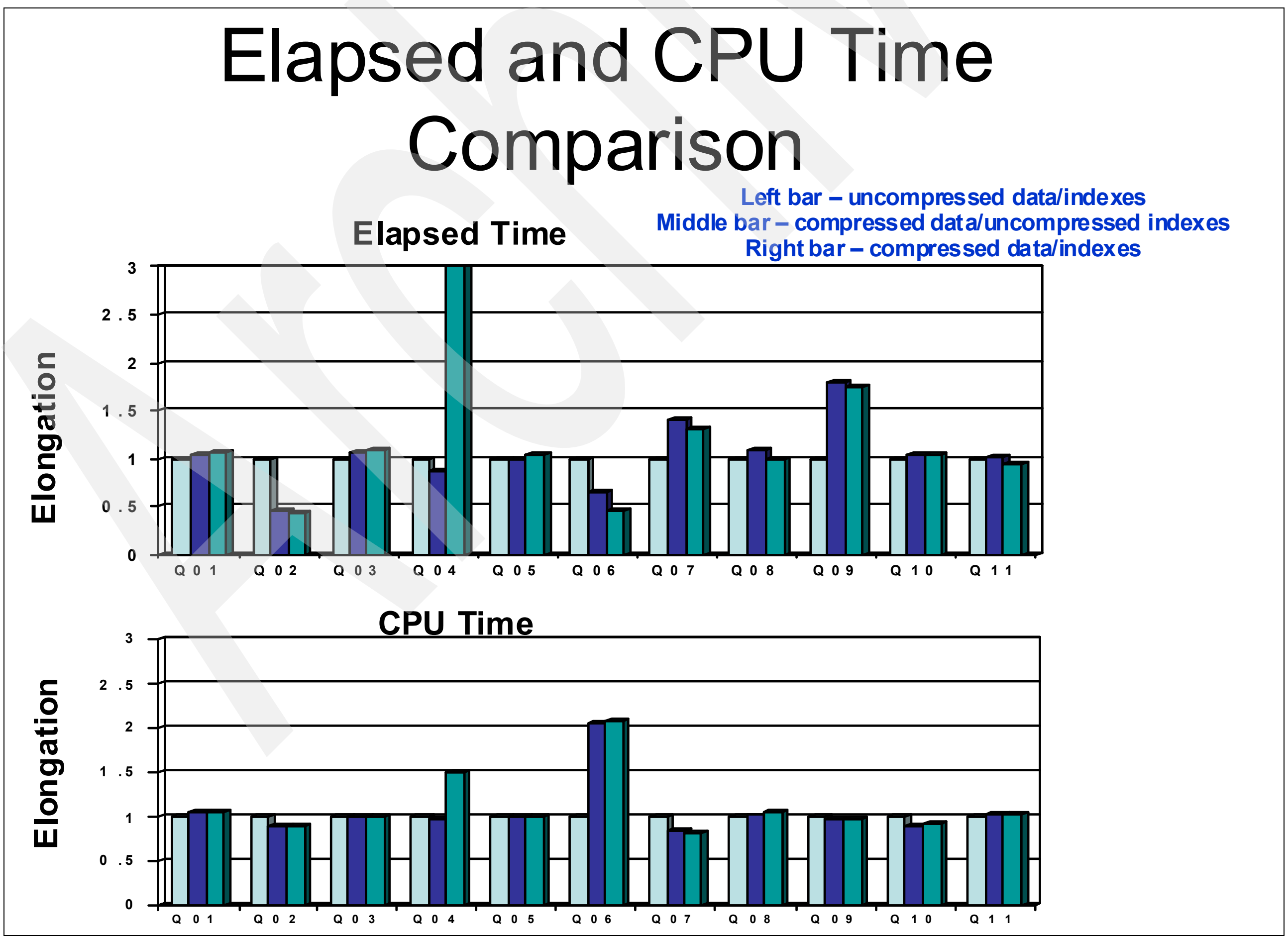


Figure 7-6 Elapsed times and CPU times for the queries

Elapsed times and CPU times are similar across these configurations for a number of queries. However, some queries exhibit different performance characteristics. Q02 is an I/O intensive query. With compressed data, this query read a smaller number of pages, reducing elapsed time significantly. For queries that spend a large portion of time scanning table data, compression tends to reduce their elapsed times. The degree of reduction depends on the time to access the pages versus the time spent on the rest of query processing.

There is a larger percentage of queries scanning tables and indexes in data warehouse workloads. With compressed data, this could lead to faster query processing time at the expense of higher CPU consumption. Data from the DB2PE accounting reports generated in this study indicates fewer GETPAGE requests by queries for the compressed tables.

Another benefit of data compression is the potential increase in buffer pool hit ratios. With compressed data, a page contains more data rows. For a table that achieves 50% disk savings, the same buffer pool holds twice as many rows. This increases the frequency of finding a page in a buffer pool and reduces the number of I/O operations to disks. However, in this study, the size of a table, taking multi-terabytes or more, is significantly larger than a buffer pool, which is defined with only gigabytes of virtual storage. Holding twice as many rows in a buffer pool does not have any noticeable effect to the hit ratios. Please read chapter 6, Balancing a Data Warehouse, to learn more about the effect of using various amount of memory for buffer pools.

There is not much difference in CPU time among the different versions of Q02. Although data expansion takes up some CPU cycles and there could be a noticeable increase in CPU time for some queries, the actual increase depends on the ratio of time spent in data expansion versus query processing. In this particular query, only a small portion of CPU time was spent in data expansion.

When table data is compressed, consecutive rows in the clustering sequence are closer together because there are fewer pages in the page set. As a result, data clustering ratio improves. In some cases this leads to the optimizer to choose an index different from the one that would have been used if the data is not compressed. In general this leads to a positive outcome. But occasionally the new index may yield a less efficient access path.

Along the same line, compressed indexes take fewer pages. This changes the statistics used by the optimizer for access path selection. As in the data compression case, most of the times the new access path is more efficient, but at times it may not lead to better query processing time. One such case is depicted by Q04.

Q06 is a query that scans the largest table in the database. Although there was business logic applied to each qualified row, it was lightweight and a significant portion of time was spent in expanding the data rows. This leads to a large jump in CPU time. On the other hand, elapsed time drops dramatically. This is common for queries that do scanning primarily with only a small amount of business logic.

Similar to the change of cluster ratios, more dynamic prefetch operations were observed in many queries. As data is compressed, the distance between adjacent accessed rows, as measured by the number of pages between them, becomes smaller. This increases the likelihood of triggering the sequential detection algorithm and switching to dynamic prefetch operations.

Lower hit ratios were observed for some buffer pools containing compressed index pages. Although larger pages sizes, 8K and 16K, were used for compressed index buffer pools, the total amount of virtual storage allocated to these buffer pools did not change. As a 4K index page on a disk is expanded into an 8K or 16K page in a buffer pool, depending on the index compression ratio, it is possible the buffer pool page is not completely filled with index data.

With the same amount of virtual storage allocated for a buffer pool, fewer index entries are held in the buffer pool. This leads to a lower hit ratio.

7.5 Other considerations

7.5.1 When should data compression be considered?

There are situations where compression may not be the right choice. Choosing the right objects that benefit from compression must be considered. Some table spaces do not realize any benefit from compression. If rows are small already and a page contains close to the 255 maximum rows per page, making the rows smaller will not gain any additional rows on the page. You will be paying the cost of compressing and ending up with a bunch of free space in the page of which you will not be able to take advantage.

Then there is the other extreme: a table with large rows. If compression is not sufficient enough to increase the number of rows per page, there is no advantage, or disk savings, using compression. A 4,000 byte row with 45% compression is still 2,200 bytes. With only a maximum of 4054 bytes of available in a data page, the compressed length is still too large to fit a second row into a single page. If you are using large rows, do not give up on compression. Remember that DB2 has page sizes other than 4 K. Getting good compression and saving on space may require you to move to a larger 8 K or 16 K page size. You may only be able to fit one 3,000 byte row into a 4 k page but you can fit 5 of them into a 16 K page and you just saved the equivalent of a 4 k page by simply increasing the page size.

7.5.2 What happens when an insert or update is performed?

A compressed data page remains in its compressed format even after it is brought into the DB2's buffer pool. All SQL DML operations must be performed against the compressed data in the pool. An insert is the most straightforward SQL operations. A data row is compressed as it is inserted. However, if a row needs to be updated, the row has to be expanded, updated, and re-compressed to complete the operation making the UPDATE potentially the most expensive SQL operation when compression is turned on. All changed rows, including inserted rows, are logged in their compressed format so you might save a few bytes on your logs. The larger page sizes available in DB2 may result in better compression. The resulting row after compression is variable in length, so you might be able to fit more rows with less wasted space in a larger page size.

7.6 Conclusion

The importance of compression to a data warehouse should be clear. For table space compression, there is disk savings with the potential for performance improvements. For indexes, where they are a critical resource to the successful performance of a data warehouse, compression is an important factor in the reduction of disk storage cost. Whether you are interested in compressing your data, indexes, or both, the reality is compression can provide a wealth of benefits. There are few reasons in DB2 9 not to take advantage of compression. However, it requires that you take the time to ensure it will actually accomplish the goal you have in mind. Don’t mandate it or simply reject it without giving it its proper analysis, using tools such as like DSN1COMP.

7.7 Resources

The following references were used in this chapter:

IBM Redbooks publications

- *DB2 for OS/390 and Data Compression*, SG24-5261
- *Index Compression with DB2 9 for z/OS*, REDP-4345
- *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637

Other publications

DB2 for z/OS Data and Index Compression, Presentation by Willie Favero

Online resources

- DB2 for z/OS

  http://www.ibm.com/software/data/db2/zos/

- *Getting the most out of DB2 for z/OS and System z* - Willie Favero's blog

  http://it.toolbox.com/blogs/db2zos

8

Reporting and analysis with Cognos

Cognos 8 Business Intelligence (BI) for Linux on System z enables customers to make more informed, faster, and more aligned decisions on the strategic platform of choice for mission-critical applications. The broad range of BI capabilities ensures all users receive the most relevant information how, when, and where it is needed. The open, enterprise-class Cognos 8 platform provides IT with cost-effective scaling to meet growing user demands. Decades of expertise in both mainframe and BI mean world-class technology and services to drive customer success.

In this study, we focused on the reporting and analysis features incorporated within the Cognos 8 BI product. Using these technologies, Cognos reports were developed against the 50 TB data warehouse that resided in DB2 for z/OS 9. These reports were run using a varying number of virtual users to simulate reporting activity from a typical call center environment. The performance measurements taken under mixed load scenarios, and the tuning suggestions derived from the test results, are outlined in this chapter.

8.1 Business Intelligence with Cognos

Cognos 8 BI provides organizations with the ability to use 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 BI and performance management. Cognos 8 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, Cognos 8 BI represents a complete range of BI capabilities in an open, enterprise-class platform, built on a service-oriented architecture (SOA). 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 Cognos 8 BI platform.

The Cognos 8 BI architecture can be separated into three distinct tiers: the presentation tier, the application tier, and the data tier.

Presentation tier

The presentation tier of Cognos 8 BI conforms to these operating criteria:

- 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 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 hundreds of thousands of users, and does not require lots of IT resources.
- Designed to fit into any Web environment. It works with Microsoft® IIS, IBM WebSphere, and Apache, to name a few. Many customers have existing Web infrastructure in place they want to keep using—such as firewalls—to ensure security or load balancing routers that spread requests across server farms. The Cognos platform fits within and uses an existing infrastructure without requiring any modifications
- Provides openness to integrate BI within the business users existing environment and toolsets. Providing customers access to BI information right at the point of decision, and delivering BI within their environment rather than a siloed application truly drives adoption. To accomplish this integration with portals, Microsoft Office business process software and enterprise search are provided, with the ability to integrate with other applications through open interfaces.

Application tier

The Cognos 8 BI platform is based on an SOA architecture. Services in the Cognos 8 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 Cognos 8 configuration, is capable of servicing an incoming request. The benefit is 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. That capacity is fully configurable.

The dispatcher receives requests from the gateways discussed in the presentation tier, and forwards them to the appropriate service to handle the request. When you install Cognos 8 BI on a server within a configuration, the dispatcher simply registers itself within that configuration 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, and allows the system to expand without having to revisit or touch the existing servers in the configuration.

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 usage.

We 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 query service that is the data access layer). This independence enables the presentation tier to render whatever data is provided by the query service, so there is no hard-link between a data source and a BI capability. Any SQL or any MDX source can be provided to any BI capability. This is powerful, as it gives you a open and scalable architecture. Cognos can add in new capabilities in one area that can use all the power of another area.

In addition, the architecture has a single open API. All communications between services takes place on a high performance 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 uses the same API that is exposed in the software development kit. You can use the development kit to use these APIs with the knowledge that they are used within the architecture.

Data tier

To be successful, all data must be accessible to the capabilities regardless of where the data resides (data tier).

The Cognos 8 platform accesses the data through the single query service which understands and uses the data source strength by using a combination of open standards such as SQL99, native SQL, and native MDX to optimize data retrieval for all these different data providers

The Cognos 8 platform provides flexibility in terms of how to source 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.

Cognos 8 BI supports four main BI capabilities:

- Reporting
- Analysis
- Dashboards
- Scorecards

Reporting is often the best understood and the most broadly adopted capability. Cognos 8 BI supports all the types of reporting an enterprise requires including transactional reports, managed reports, high fidelity reports (like those you might provide to customer), business/ad hoc reporting and analytical reporting, and so on. Features such as scheduling, briefing books, flexible object embedding, interactive charting abilities and so on provide users the ability to customize BI output in virtually any desired style or quality. Cognos 8 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 to which they want to pay attention.

The Cognos 8 BI analysis functionality enables users to explore information and perform multidimensional analysis against Cognos 8 BI data. This feature provides the ability to analyze and report against any OLAP data source and dimensionally modeled relational data. Cognos 8 BI analysis studio offers an engaging and highly flexible interface, and 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 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. Cognos 8 lets you develop a complete view of the business for truly actionable BI.

8.1.1 Cognos 8 BI for Linux on System z

Cognos 8 BI for Linux on System z offers the comprehensive BI 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 identify and respond to critical business trends.

At the time of writing this book, Cognos 8.3 was available for Linux for System z, which 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. Also note that at the time of writing this book, Cognos scorecarding was not available on System z.

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

8.1.2 Cognos software components

Cognos 8 BI comes with several product components and interacts with existing components on both z/OS and Linux for System z.

Figure 8-1 on page 127 shows an overview of the Cognos 8 BI architecture on Linux for System z. Each of the components is described in the following sections.

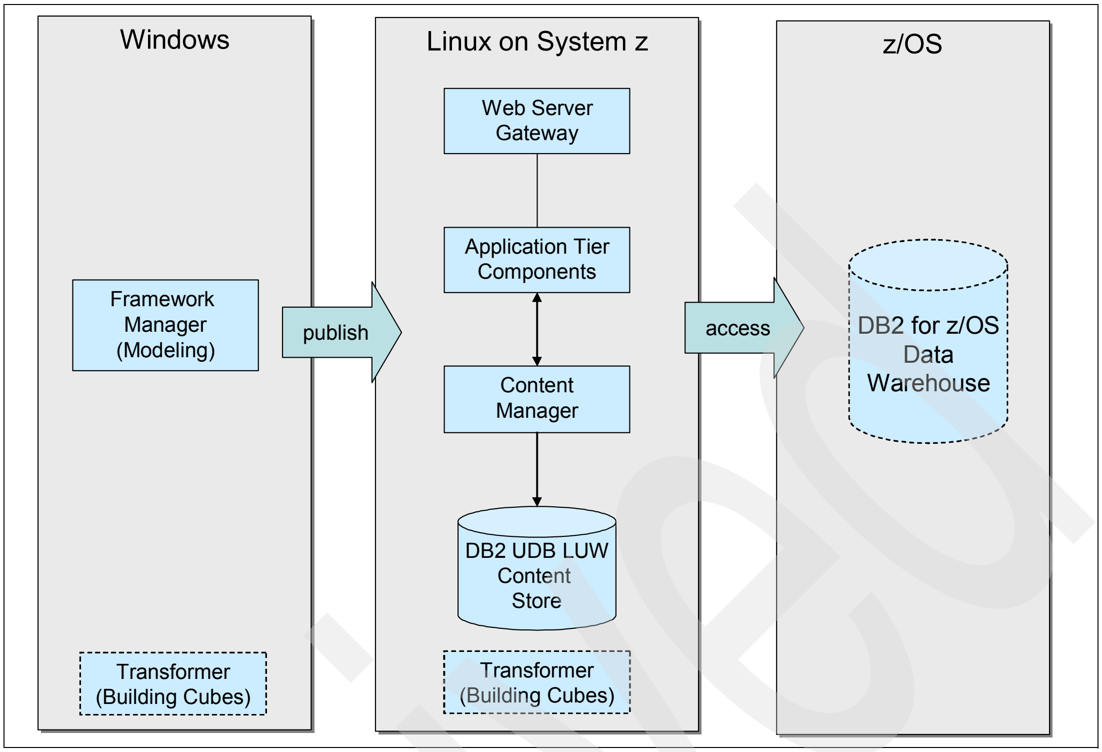


Figure 8-1 Cognos software components

Modeling

Cognos 8 Framework Manager is a natural starting point in developing Cognos user information. Cognos 8 Framework Manager is the window-based tool used to create models of source data (in the form of metadata) that is exposed to Cognos studio users. In developing the framework model, users define relationships between data entities, create new query subjects, create calculated query items and filters, and so forth. 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.

Web server (gateway)

Cognos Connection, the Web portal for Cognos BI Server, is accessed by users through Web browsers. Cognos 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 Cognos application tier components. For scalability, multiple Web servers may be installed.

Application tier components

The Cognos application tier contains one or more Cognos servers. A Cognos 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:

- Dispatcher

  The dispatcher starts all Cognos 8 services configured and enabled on a computer, then routes requests. The dispatcher is a multithreaded application that uses one or more threads per request.

- Content manager service

  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.

- Report service

  The report service manages interactive requests to run reports and provides output for a user in Cognos Connection or a studio.

Again, many Cognos server instances may be installed for scalability.

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 data that Cognos 8 needs to operate. This includes report specifications, published models (metadata), and the packages that contain them. It also contains connection information for data sources, information about the external namespace, and the Cognos namespace itself. The Cognos Content store database can be installed with other Cognos 8 components or installed on a separate computer. For Cognos 8.3 for Linux on System z, DB2 on Linux for System z is the recommended database for use as a content store.

Cognos 8 BI Transformer and PowerCubes

Cognos 8 BI Transformer provides the ability to easily construct, build and publish Cognos PowerCubes (OLAP data cubes) as caches of data for exploration and analysis by broad communities of business users. These cubes convert high volumes of data into hierarchical-based compact cubes that deliver high performance and predictable response times. They are accessible by any of the user capabilities including analysis, reporting, mobile, and so forth.

The transformer component comes with a user interface that runs on Windows. However, extracting the data from the referenced data source and building the cube can also run on Linux for System z.

Data source

Data sources, also known as query databases, are relational databases, dimensional cubes, files, or other physical data stores that can be accessed through Cognos 8. Application tier components use data source connections to access data sources.

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

8.2 Delivering BI throughout the organization

Information-based decision making has played a key role in helping businesses drive improvements in revenue and profitability for the last two decades. Data Warehousing and BI software have been key enablers in this movement. Following that trend, information-based decision making is moving further down the organization to customer-facing employees, customer service, and sales, as well as out to customers themselves. This new trend is helping to drive more efficient and effective employees and better customer loyalty, or *stickiness* (to borrow a term from the World Wide Web). The concept of improving efficiency and effectiveness by empowering customer-facing employees with better information is known as Operational BI.

8.2.1 Operational BI

There are two major characteristics of Operational BI: driving key performance indicators (KPIs) out to customer service and sales, and empowering these front line employees with better information.

The first characteristic of Operational BI, as mentioned above, is to drive KPIs down the organization to align customer-facing employees (sales and customer service representatives) with organizational goals. This allows organizations to influence behavior throughout the work week instead of retroactively evaluating employees at a point in time when they can no longer achieve those organizational goals. This keeps employee moral high and helps to achieve closer adherence to organizational policies.

The second characteristic, and perhaps the greater opportunity for organizations, is to deliver BI content to those customer service and sales people. What does that mean? It means being able to deliver customer information, not just data, to customer-facing employees to optimize each and every customer interaction. Simply put, it is applying the principals of good data warehousing practices and BI application design to information that is accessed by lower levels of the organization. This does not mean ad-hoc access for customer service. This means well-designed screens supporting the customer service and sales process that effectively provide enough information about the customer so that the customer service agent can understand the value and behaviors of the particular customer. Do they pay on time? How much revenue has this customer generated? Do they complain all the time? What types of products do they buy and what is the product mix?

Many customer service systems are designed to provide billing and transaction support, that is, data, with little thought about usable information. The billing and transaction orientation of the customer service systems makes customer service agents less effective at handling more sophisticated questions that involve account information over time. Delivering Operational BI applications makes customer-facing employees more efficient and more effective, and helps drive corporate differentiation. Simple changes in presentation, graphics, and data orientation can have a significant impact.

8.2.2 Driving information out to the customer

Another theme of Operational BI is to drive better decision making to the customer. This can eliminate customer support calls making the organization more efficient. It empowers the customer and can differentiate the organization. One of the greatest assets that an organization has is the information about its customers. Enabling customer access to their own information in a secure and usable manner can create organizational loyalty that would be harder to achieve with the infrequent customer service contact model.

8.2.3 Cognos 8 BI for Linux on System z, Enterprise BI and Operational BI

Cognos for Linux on System z is an Enterprise BI platform. It provides a full set of capabilities for delivering BI. What role can it play in an Operational BI context? That is exactly what the IBM team wanted to understand. Can an enterprise BI platform be used within the context of Operational BI and, especially, will it scale?

Implementing an Enterprise BI platform means that you have a fully integrated set of components and there is significantly less development work to be done. That means a quicker time to market and, potentially, a higher return on investment. Additionally, consistency between Operational, Tactical, and Strategic BI output could be achieved by using the same core metadata used to build the Operational BI system.

8.2.4 Scalability aspects for very large databases

As organizations consider expanding the BI access to customer service and sales and even to customers, you can understand the implications on infrastructure and the need for a dynamic and scalable architecture. The test scenarios used in this book were designed to support a high level of Operational BI concurrency and also validate that production Operational Workloads could be maintained within Service Level Agreements (SLAs) at the same time that BI workloads were being run.

It can be observed that operational BI queries are more like online transactional processing (OLTP) queries in that they return a limited amount of data, do not require huge aggregations, and typically responds in seconds.

These queries are typically performed in the context of a specific subject (such as a customer or an order) where the number of records returned (and the amount of data accessed) could be independent from the size of the database. Consider a large data warehouse with information about customers and orders. While not generally true, it is likely that the amount of data stored per customer is independent from the number of customers. For each customer order, it is likely that the number of ordered parts (line items) is fairly constant and independent from the number of orders or the number of customers.

Figure 8-2 shows these characteristics and illustrates the data accessed in tables of a data warehouse database for traditional BI queries (left figure) compared to the data accessed for operational BI queries (right figure). While traditional BI queries typically require scanning and sorting of a large amount of data, operational BI queries can benefit from index access.

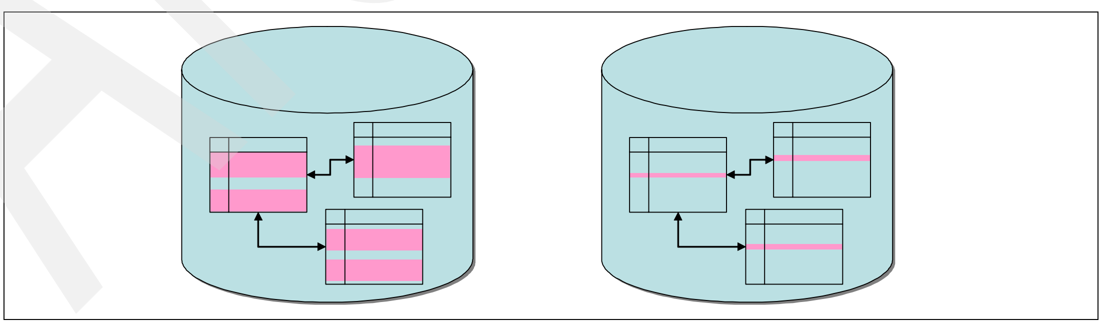


Figure 8-2 Amount of data accessed for traditional (left) and operational (right) BI queries

With adequate database optimization in place and efficient index access to this set of data, System z and DB2 can play their strengths in processing this type of transaction efficiently. As a result, even for the 50 TB data warehouse database we used for the book, we were able to achieve good response times that were not significantly longer than tests with a smaller database size.

8.3 System environment and configuration options

This section describes 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 below 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 Cognos BI Server documentation for recommended configuration decisions, as it provides a far more comprehensive discussion of the various options.

8.3.1 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 8-1 gives an overview of the product components that were installed and used for this portion of the study.

Table 8-1 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) 16 GB physical memory 4 or 8 CPs (IFLs) assigned to Linux
HTTP Server (gateway)	IBM_HTTP_Server/6.1.0.17 Apache/2.0.47
Application Server	WebSphere Application Server 6.1.0.17, 31 Bit Installed in /opt/IBM/WebSphere/AppServer. Two application server profiles defined: AppSrv01 and AppSrv02. JVM™ heap size is 768M for both servers. Thread pool for Web sessions is set to 500 with the option to exceed the limit, if required.
Java Runtime Environment	We use the JDK™ that comes with the Application Server. java -version reports: Java(TM) 2 Runtime Environment, Standard Edition (build pxz31dev-20080315 (SR7)) IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux s390-31 j9vmxz3123-20080315 (JIT enabled) J9VM - 20080314_17962_bHdSMr JIT - 20080130_0718ifx2_r8 GC - 200802_08) JCL - 20080314

Component	Product and Version
Database for Cognos 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 "SQL09051" with level identifier "03020107". Informational tokens are "DB2 v9.5.0.1", "s080328", "MI00227", and Fix Pack "1".
Data source for relational data	DB2 9 for z/OS (where our 50 TB data warehouse database resides)
LDAP Server	Tivoli Directory Server 6.1 We also investigated authentication in Cognos using Tivoli Directory Server on the same Linux system. Tivoli Directory Server comes with an additional instance of DB2 for Linux for System z. For our final measurements, we didn’t use LDAP authentication, though. Therefore, this book doesn’t describe LDAP setup and interaction between Cognos BI and an LDAP user directory.
Cognos BI Server	Cognos BI Server version 8.3 The install image and driver version is: c8bisrvr_zlinux31_8.3.81.523_ml.tar We installed the server components twice on the same machine in the following directories (for details on the two server setup see 8.3.2, “Topology and setup” on page 132) /opt/IBM/cognos /opt/IBM/cognos2

8.3.2 Topology and setup

The 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 8-3 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.

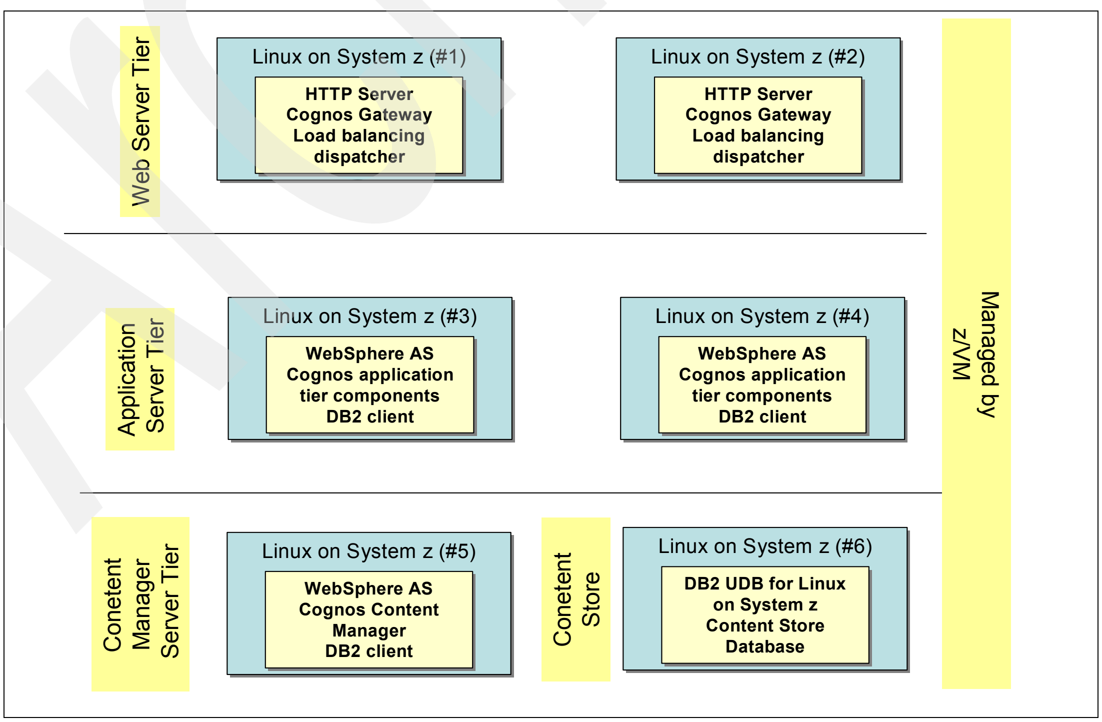


Figure 8-3   Options to distribute Cognos components over multiple Linux systems

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 Cognos content store database), if this is desired. It also allows users to manage and tune the server components individually without having to consider the potential impacts to other components on the same Linux system.

Because Cognos 8.3 BI Server for Linux on System z runs in a 31 bit Application Server environment, the virtual heap size that can be assigned to an instance of the Java virtual machine is limited by 2 GB (due to other limitations, it is actually slightly lower). The Cognos 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.

Note: There can only be one active instance of the Content Manager at any time. You can, however, have multiple Content Manager instances installed and configured to implement fail-over support.

For this study, the decision was made to install all components on a single Linux for System z instance. See Figure 8-4.

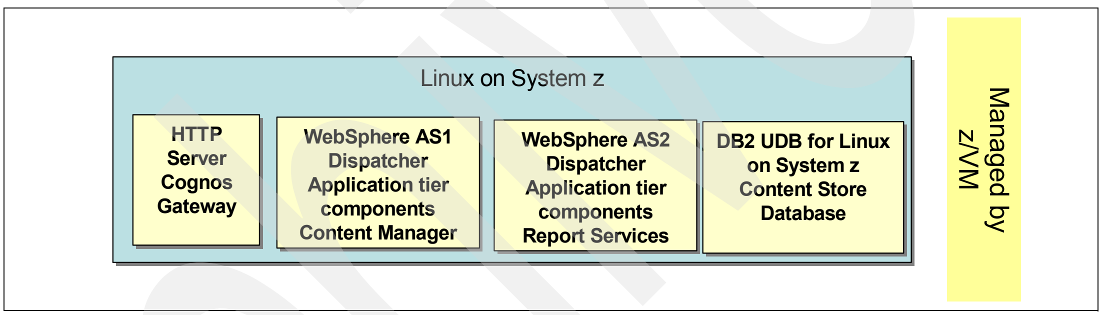


Figure 8-4 Cognos topology used for this book

This decision was made based on the assumption that Linux is 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).

To set up a system like the one outlined in Figure 8-4, two WebSphere Application Server profiles are needed, and the 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 require their 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 Cognos (**`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. Both 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 Cognos components, two instances of the code were installed (that is, the installation procedure was run twice). For one installation (/opt/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/cognos2), only the application tier components were selected. The resulting EAR file was deployed in the AppSrv02 profile.

Figure 8-5 shows the configuration of the first instance. 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.

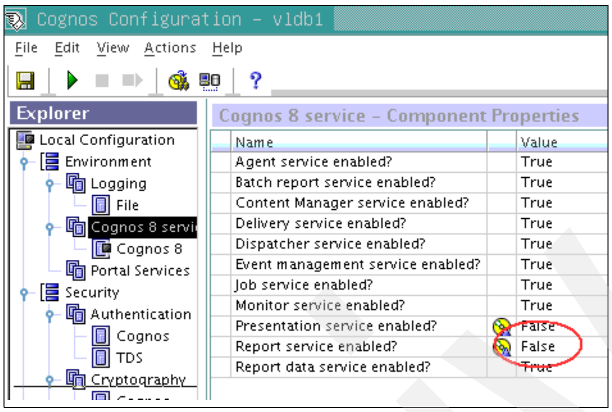


Figure 8-5 Cognos configuration for first instance

Figure 8-6 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.

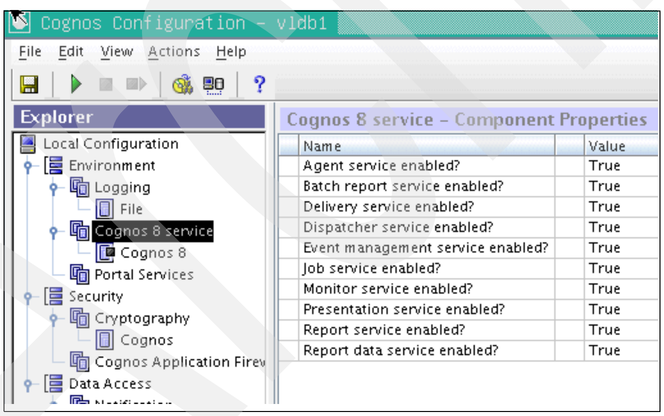


Figure 8-6 Cognos configuration for second instance

Once the configuration of the two servers is complete, the two dispatchers appear in Cognos Administration, as in Figure 8-7. Note that one server uses application port 9080 and the other uses 9606. Both can now be configured individually.

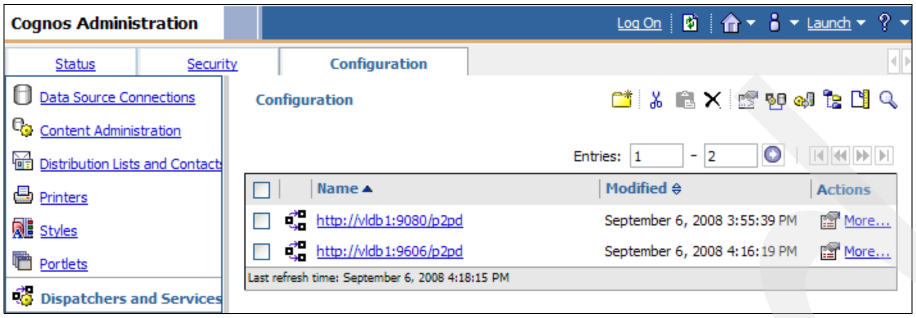


Figure 8-7 Cognos Administration showing our two Cognos dispatchers

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

8.3.3 Tuning Cognos gateway

Cognos offers several options for the gateway implementation in the HTTP server, including a CGI gateway, a servlet gateway, and also an Apache connector.

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

Example 8-1 shows the configuration settings for the IBM HTTP Server. This configuration loads the corresponding shared library with the implementation (mod2_cognos.so) and defines the location as /cognos8/cgi-bin/cognos_module.

Example 8-1 Configuration for mod2 gateway in file httpd.conf

```
LoadModule cognos_module /opt/IBM/cognos/c8/cgi-bin/mod2_cognos.so
...
<Location /cognos8/cgi-bin/cognos_module>
SetHandler cognos-handler
</Location>
```

In Cognos configuration, the corresponding gateway URI is defined to point to the previously defined location setting. See Figure 8-8.

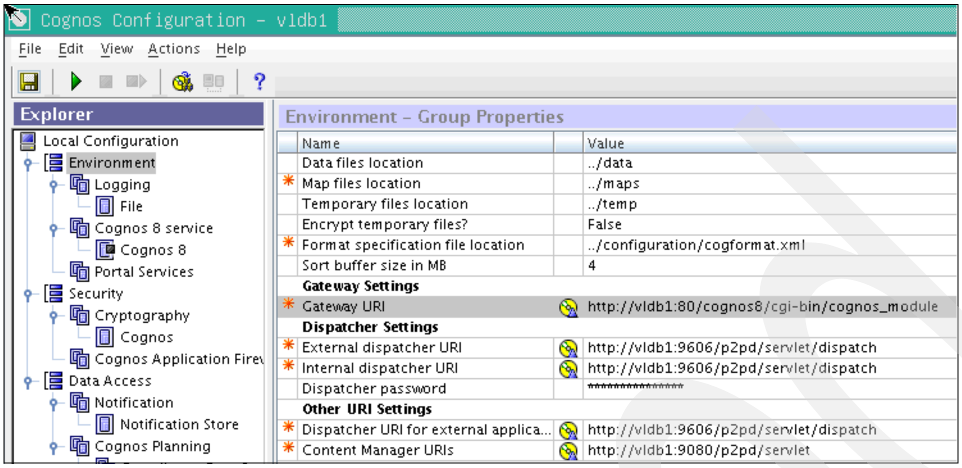


Figure 8-8 Configuration of the mod2 gateway in Cognos Configuration

To cope with the expected number of concurrent Web sessions, the configuration of the IBM HTTP Server was modified to increase the number of maximum clients to 600. See Example 8-2.

Example 8-2 HTTP server settings in httpd.conf

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

8.3.4 Tuning Cognos report service

Whenever a user requests a new report to be created using Cognos Connection, this request ultimately is processed by a (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. The number of report service processes multiplied by the number of threads also controls the number of connections to the data sources.

For this testing, the number of processes used for the report service was varied between five and 16. 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).

To configure the number of report service processes (both for peak and off-peak hours), use Cognos Administration. Select the server and the service (report service). See Figure 8-9 on page 137 for an example where fourteen report service processes were configured.

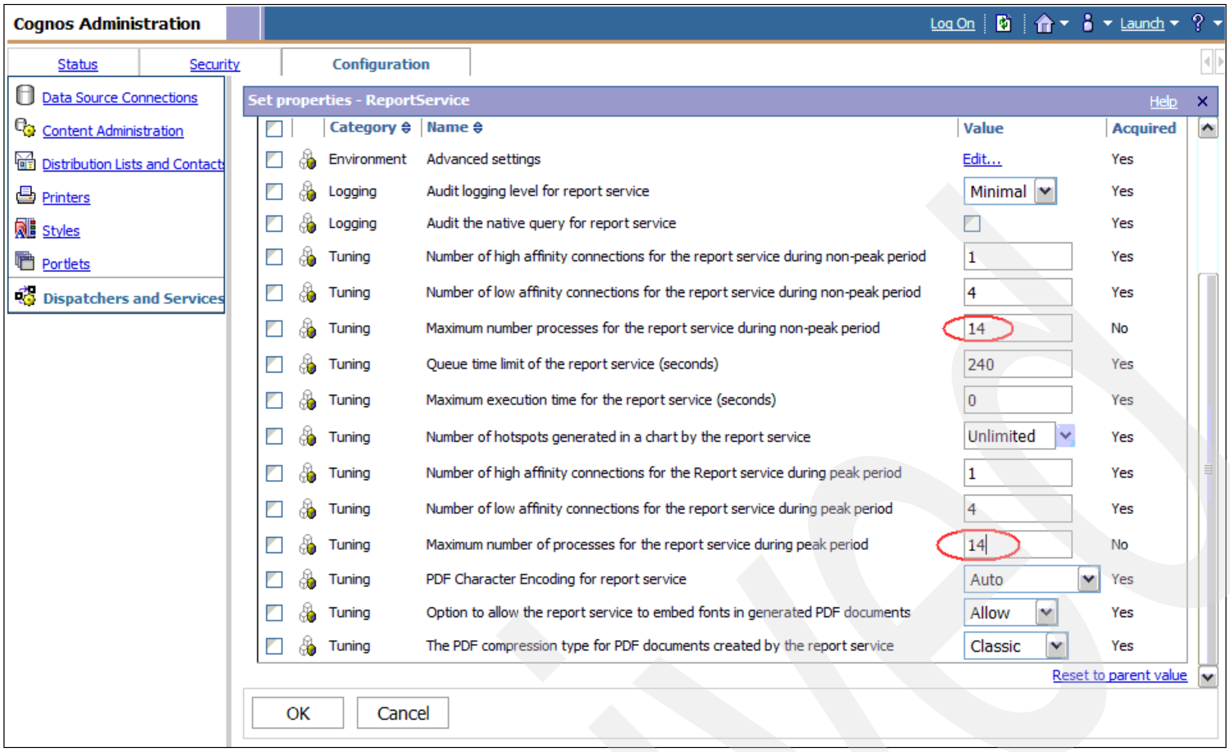


Figure 8-9 Configuration of report service processes in Cognos Administration

8.3.5 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 Cognos BI Server can run with this default setting, frequent Java garbage collection can be observed during periods of high load. This impacts performance.

The virtual memory assigned to the Java process includes both the native and the JVM heap. This memory is limited to less than 2 GB for the 31 Bit environment. Altering the JVM heap size is, therefore, a trade-off between assigning memory to Java classes and to native libraries (for example, for Cognos or DB2).

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 8-10.

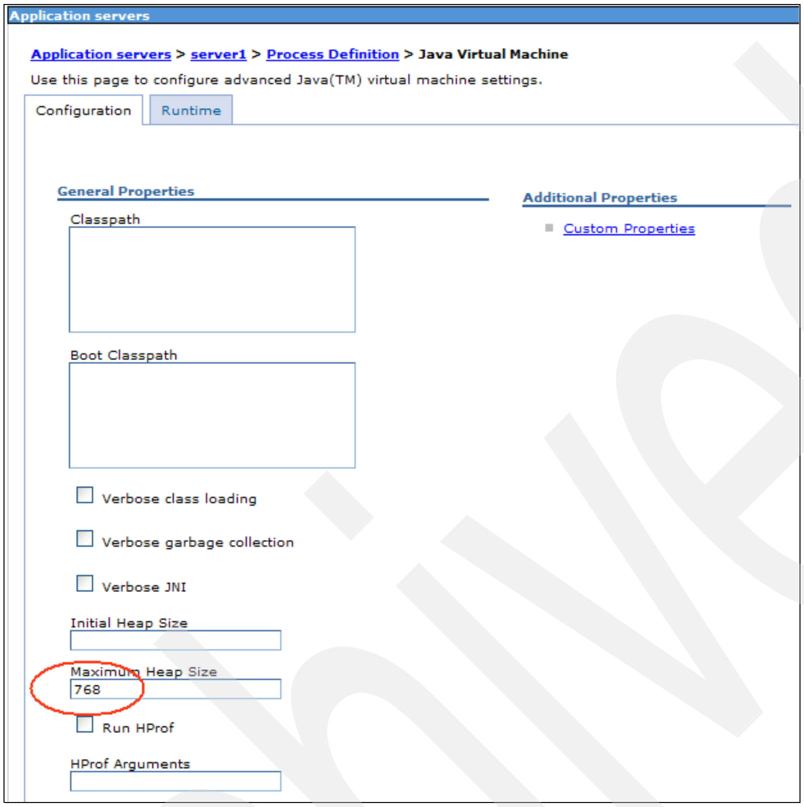


Figure 8-10 Setting JVM heap size for the Application Server

The Web container in 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 run time. Creating new threads is 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**. See Figure 8-11.

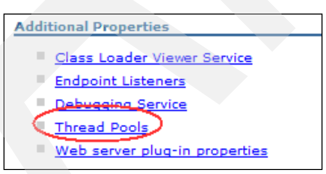


Figure 8-11 Thread pool configuration in WebSphere Application Server

The maximum size was changed to 512. See Figure 8-12.

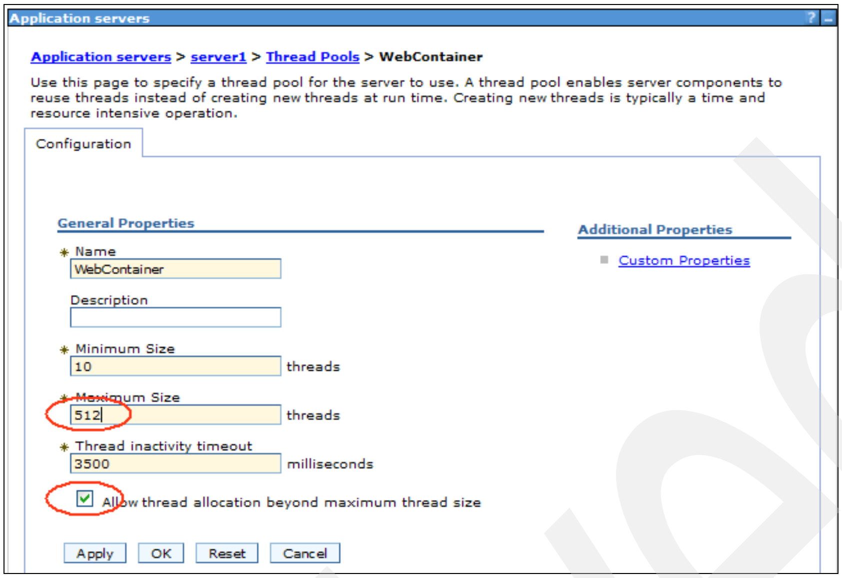


Figure 8-12 Setting for the maximum size of the thread pool in WebSphere Application Server

8.3.6 Tuning DB2 for Linux on System z

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

As shown below, 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 Cognos C8SE installation directory.

Example 8-3 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
```

8.4 The model and reports used for this book

This section describes the Cognos meta model and the reports we used for this book. They build the foundation for the tests described in 8.5, "Testing the operational BI application" on page 148.

8.4.1 The relational model

Cognos 8 Framework Manager is a window-based tool used to create models of source data in the form of metadata that is exposed to Cognos Studio users. Using this centralized metadata model, users can apply consistent rules and criteria to data irrespective of the source. Modelers start by creating projects that contain all of the objects within the model. Folders and namespaces can be used to improve the organization and the naming of objects within the model. Relationships between data objects (query subjects) are used to define which columns the objects join together on, the join cardinality, and so forth. Of course, connections to the source data must be established through Framework Manager data sources. Other objects, such as calculations and filters, can be created and exposed to users as standalone objects in the model or embedded within a query subject. Users can then incorporate these reusable functions and filters (represented as objects) into their reports. To learn more about creating new projects within Cognos Framework manager, refer to the Cognos installation documentation and the book *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637.

Table 8-2 Relationships between Framework Manager VLDB model objects

Relationship between query subjects	Relationship defined between query items
LINEITEM->ORDERS	L_ORDERKEY->O_ORDERKEY
LINEITEM->ALLTIME	L_COMMITDATE->A_CALENDARQUARTER
LINEITEM->PART	L_PARTKEY->P_PARTKEY
LINEITEM->SUPPLIER	L_SUPPKEY->S_SUPPKEY
LINEITEM->CUSTOMER	L_CUSTKEY->C_CUSTKEY
ORDERS->ALLTIME	O_ORDERDATE->A_CALENDARDT
ORDERS->CUSTOMER	O_CUSTKEY->C_CUSTKEY
PART->MFGBRAND	P_BRAND->M_BRAND
CUSTOMER->GEOGRAPHY	C_GEOGRAPHYKEY->G_GEOGRAPHYKEY
GEOGRAPHY->NATION	G_NATIONKEY->N_NATIONKEY
NATION->REGION	N_REGIONKEY->R_REGIONKEY

The example in Table 8-2 shows the relationships created between query subjects in Cognos 8 Framework Manager. The project created for these tests is shown in the mapping below (Figure 8-13 on page 141). Each query subject is created and connected to its respective object within DB2 for z/OS. A sample relationship definition between the CUSTOMER table and LINEITEM table is provided in Figure 8-13 on page 141. It shows the join between C_CUSTKEY and L_CUSTKEY is a 1..1 to 1..N cardinality relationship. This means that each L_CUSTKEY in LINEITEM will match one C_CUSTKEY in CUSTOMER. Secondly, each C_CUSTKEY in CUSTOMER may match multiple L_CUSTKEY's in LINEITEM. These relationships tell Cognos BI how the objects you wish to incorporate into a report relate to each other, and they provide the information necessary for the drag and drop and query generation technologies to work properly.

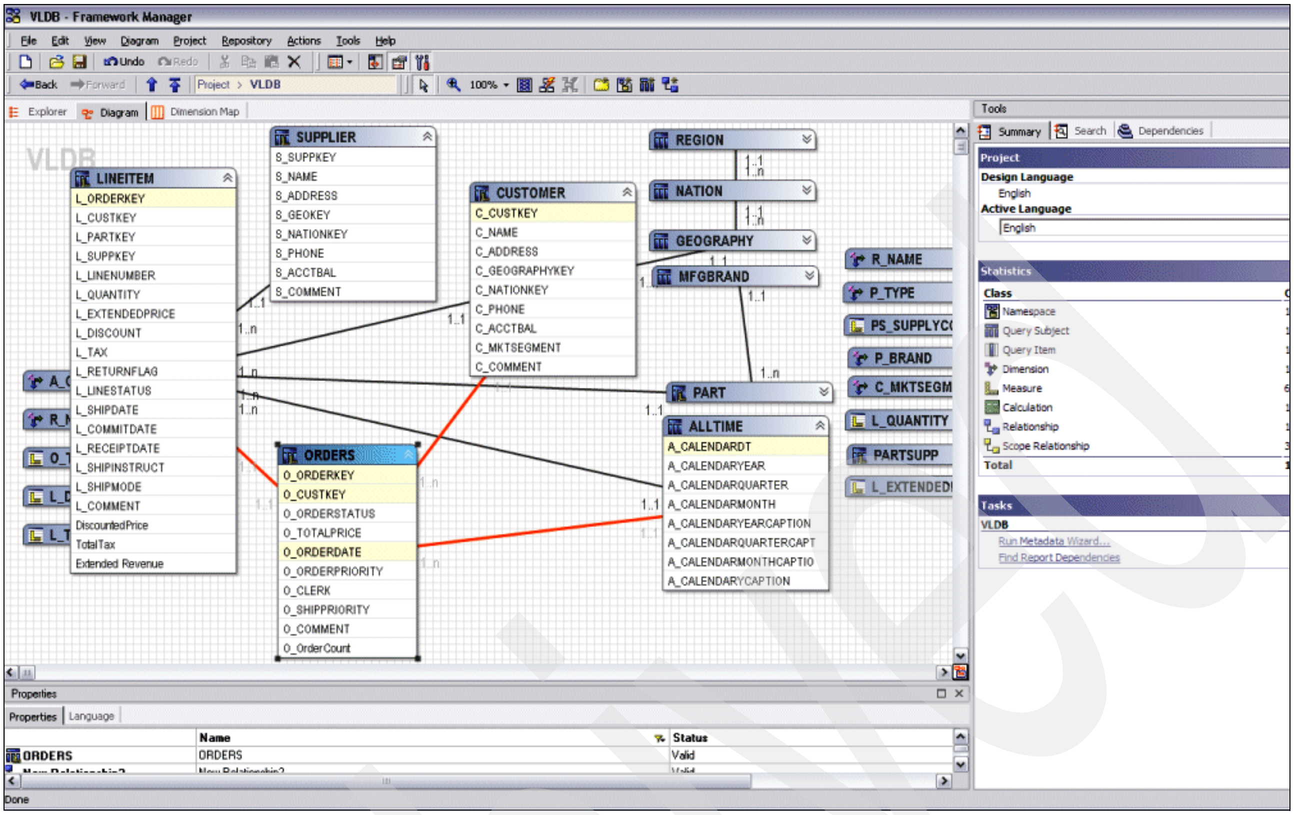


Figure 8-13 VLDB model in Cognos 8 Framework Manager

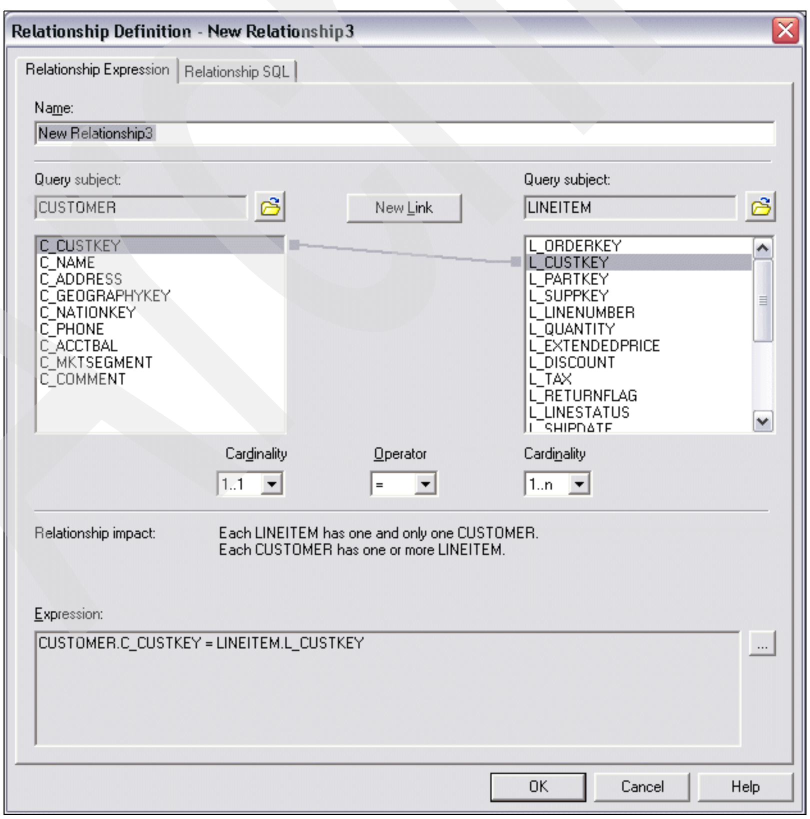


Figure 8-14 Cognos 8 Framework Manager relationship definition example

To adjust the granularity of date reporting within Cognos, the table ALLTIME was created. See Figure 8-14 on page 141. This table gave report creators the ability to transform data information from other tables into their respective days of the week, quarters, years, and so on. In the following example, the date column in ISO date notation from the ORDERS table can be transformed into multiple formats using information in the ALLTIME table. Note that this table provided seven years of dates in the range from '1992-01-01' to '1998-12-31'

Example 8-4 Definition of the ALLTIME table for time based modeling

```
CREATE TABLE TPCH50TB.ALLTIME (
"A_CALENDARDT" DATE NOT NULL,
"A_CALENDARYEAR" INTEGER WITH DEFAULT NULL,
"A_CALENDARQUARTER" INTEGER WITH DEFAULT NULL,
"A_CALENDARMONTH" INTEGER WITH DEFAULT NULL,
"A_CALENDARYEARCAPTION" CHAR(6) WITH DEFAULT NULL,
"A_CALENDARQUARTERCAPTION" CHAR(7) WITH DEFAULT NULL,
"A_CALENDARMONTHCAPTION" CHAR(8) WITH DEFAULT NULL,
"A_CALENDARYCAPTION" CHAR(10) WITH DEFAULT NULL
)
```

The date '1992-01-01' from the ORDERS table could be joined to the ALLTIME table on the A_CALENDARDT field, and a report could drill down to different levels using the information in Example 8-5.

Example 8-5 ALLTIME entry for a specific date

```
A_CALENDARDT: 1992-01-01
A_CALENDARYEAR: 92
A_CALENDARQUARTER: 1
A_CALENDARMONTH: 1
A_CALENDARYEARCAPTION: 1992
A_CALENDARQUARTERCAPTION: 1992Q1
A_CALENDARMONTHCAPTION: JAN
A_CALENDARYCAPTION: WED
```

8.4.2 Operational BI reports used in this book

To simulate an operational BI application, two reports were created that could support tactical decision making during customer interaction in (for example) a call center.

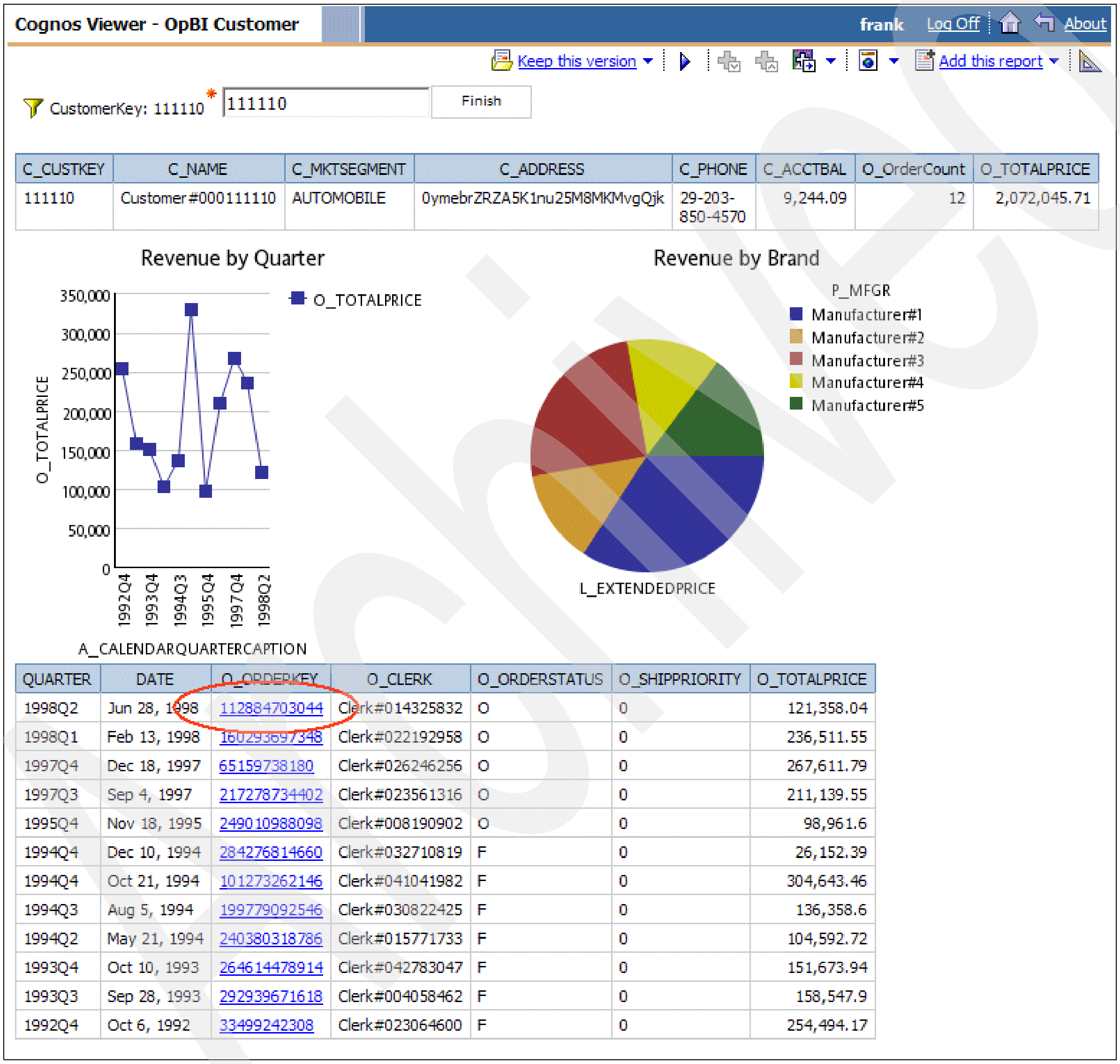
Assume that a call center agent of a retailer receives a call from a customer who complains about a certain shipment. In addition to accessing the transactional system (which was not simulated in this setup), the call center agent automatically receives additional operational BI information for this customer in a page such as the one in Figure 8-15 on page 143.

The page includes four sections

- General information about the selected customer. This information could already include aggregated and computed information based on previous interaction with the customer. An example may be the total revenue for the customer, or a computed "score" based on previous sales
- A chart (on the left side) illustrating revenue by quarter for this particular customer. This information may help the call center agent understand how revenue with this customer has been generated over time.

- Another chart (on the right) illustrates which brands the customer has purchased. This information might give additional insight in the customer's purchase behavior and could support the call center agent when proposing new or additional products to buy.
- Finally, the table at the bottom gives a historical overview of orders the customer completed in the past. Either through the 'native' system or through a link to another system, the call center agent may directly access detailed information for each of these transactions.

To implement this type of application, a Cognos report was created (Figure 8-15) based on the model described in 8.4.1, "The relational model" on page 140.



C_CUSTKEY	C_NAME	C_MKTSEGMENT	C_ADDRESS	C_PHONE	C_ACCTBAL	O_OrderCount	O_TOTALPRICE
111110	Customer#000111110	AUTOMOBILE	0ymebrZRZA5K1nu25M8MKMvgQjk	29-203-850-4570	9,244.09	12	2,072,045.71

QUARTER	DATE	O_ORDERKEY	O_CLERK	O_ORDERSTATUS	O_SHIPPRIORITY	O_TOTALPRICE
1998Q2	Jun 28, 1998	112884703044	Clerk#014325832	O	0	121,358.04
1998Q1	Feb 13, 1998	160293697348	Clerk#022192958	O	0	236,511.55
1997Q4	Dec 18, 1997	65159738180	Clerk#026246256	O	0	267,611.79
1997Q3	Sep 4, 1997	217278734402	Clerk#023561316	O	0	211,139.55
1995Q4	Nov 18, 1995	249010988098	Clerk#008190902	O	0	98,961.6
1994Q4	Dec 10, 1994	284276814660	Clerk#032710819	F	0	26,152.39
1994Q4	Oct 21, 1994	101273262146	Clerk#041041982	F	0	304,643.46
1994Q3	Aug 5, 1994	199779092546	Clerk#030822425	F	0	136,358.6
1994Q2	May 21, 1994	240380318786	Clerk#015771733	F	0	104,592.72
1993Q4	Oct 10, 1993	264614478914	Clerk#042783047	F	0	151,673.94
1993Q3	Sep 28, 1993	292939671618	Clerk#004058462	F	0	158,547.9
1992Q4	Oct 6, 1992	33499242308	Clerk#023064600	F	0	254,494.17

Figure 8-15 Cognos report used for operational BI application (customer analysis)

To get a better understanding of the underlying database operations required to create a page such as the one shown in Figure 8-15, Cognos Report Studio can be used (**Tools → Show Generated SQL/MDX**) to view the SQL statement that is executed when the report is run.

These SQL queries are generated automatically by the Cognos BI server based on the information in the model, the published package, the database system, as well as additional options set for the query in the report specification.

Example 8-6, shows the SQL query that is run on DB2 for z/OS to get data for the first section of the report shown in Figure 8-15 on page 143.

Example 8-6 Query 1 for the operational BI report

```
select "T1"."C0" "C0" , "T1"."C1" "C1" , "T1"."C2" "C2" , "T1"."C3" "C3" , "T1"."C4" "C4" , "T1"."C5" "C5"
, "T0"."C5" "C6" , "T0"."C6" "C7"
 from (
select "CUSTOMER"."C_CUSTKEY" "C0" , "CUSTOMER"."C_NAME" "C1" , "CUSTOMER"."C_MKTSEGMENT" "C2" ,
"CUSTOMER"."C_ADDRESS" "C3" , "CUSTOMER"."C_PHONE" "C4" , sum("ORDERS3"."O_OrderCount") "C5" ,
sum("ORDERS3"."O_TOTALPRICE") "C6"
 from "TPCH50TB"."CUSTOMER" "CUSTOMER", (
select "ORDERS"."O_CUSTKEY" "O_CUSTKEY" , "ORDERS"."O_TOTALPRICE" "O_TOTALPRICE" , 1 "O_OrderCount"
 from "TPCH50TB"."ORDERS" "ORDERS") "ORDERS3"
 where "CUSTOMER"."C_CUSTKEY" = ? and "CUSTOMER"."C_CUSTKEY" = "ORDERS3"."O_CUSTKEY"
 group by "CUSTOMER"."C_CUSTKEY", "CUSTOMER"."C_NAME", "CUSTOMER"."C_MKTSEGMENT", "CUSTOMER"."C_ADDRESS",
"CUSTOMER"."C_PHONE") "T0", (
select "CUSTOMER"."C_CUSTKEY" "C0" , "CUSTOMER"."C_NAME" "C1" , "CUSTOMER"."C_MKTSEGMENT" "C2" ,
"CUSTOMER"."C_ADDRESS" "C3" , "CUSTOMER"."C_PHONE" "C4" , "CUSTOMER"."C_ACCTBAL" "C5"
 from "TPCH50TB"."CUSTOMER" "CUSTOMER", (
select "ORDERS"."O_CUSTKEY" "O_CUSTKEY" , "ORDERS"."O_TOTALPRICE" "O_TOTALPRICE" , 1 "O_OrderCount"
 from "TPCH50TB"."ORDERS" "ORDERS") "ORDERS3"
 where "CUSTOMER"."C_CUSTKEY" = ? and "CUSTOMER"."C_CUSTKEY" = "ORDERS3"."O_CUSTKEY") "T1"
 where "T1"."C0" = "T0"."C0" and ("T1"."C1" = "T0"."C1" or "T1"."C1" is null and "T0"."C1" is null) and
("T1"."C2" = "T0"."C2" or "T1"."C2" is null and "T0"."C2" is null) and ("T1"."C3" = "T0"."C3" or "T1"."C3"
is null and "T0"."C3" is null) and ("T1"."C4" = "T0"."C4" or "T1"."C4" is null and "T0"."C4" is null) FOR
FETCH ONLY
```

The query in Example 8-7 shows the statement that is used to build the chart for revenue per quarter shown in Figure 8-15 on page 143. There are additional operations applied to the result set to compute the data required to build the chart.

Example 8-7 Query 2 for the operational BI report

```
select "ALLTIME"."A_CALENDARQUARTERCAPTION" "A_CALENDARQUARTERCAPTIONkey" ,
sum("ORDERS3"."O_TOTALPRICE") "O_TOTALPRICE"
 from "TPCH50TB"."ALLTIME" "ALLTIME", "TPCH50TB"."ORDERS" "ORDERS3"
 where "ORDERS3"."O_CUSTKEY" = ? and "ALLTIME"."A_CALENDARDT" =
"ORDERS3"."O_ORDERDATE"
 group by "ALLTIME"."A_CALENDARQUARTERCAPTION" FOR FETCH ONLY
```

The query in Example 8-8 shows the statement that is used to build the chart Revenue by Brand shown in Figure 8-15 on page 143. Additional computing in Cognos takes place before the pie chart can be rendered.

Example 8-8 Query 3 for the operational BI report

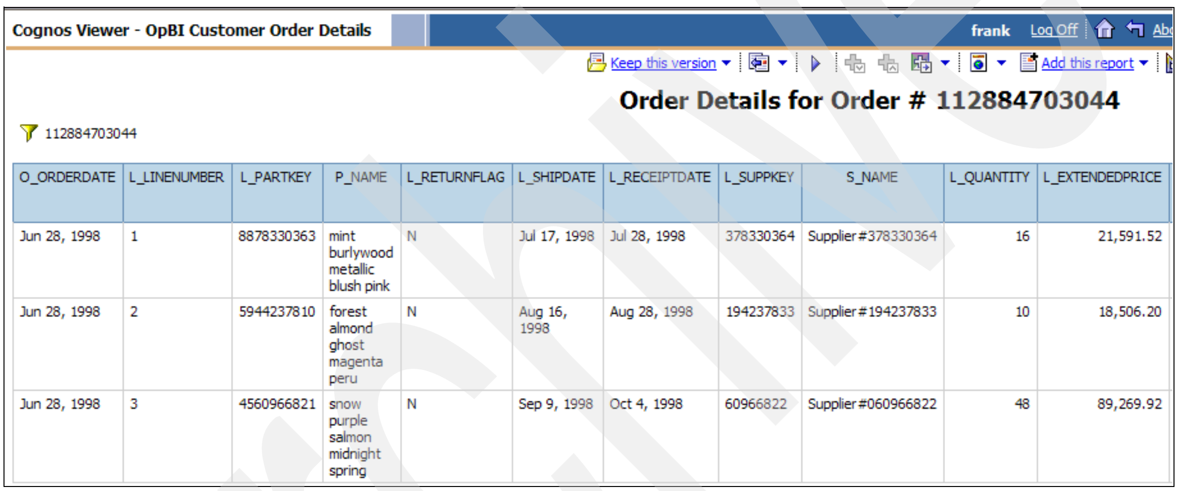
```
select "MFGBRAND"."P_MFGR" "P_MFGRkey" , sum("LINEITEM"."L_EXTENDEDPRICE")
"L_EXTENDEDPRICE"
 from "TPCH50TB"."MFGBRAND" "MFGBRAND", "TPCH50TB"."LINEITEM" "LINEITEM",
"TPCH50TB"."PART" "PART"
 where "LINEITEM"."L_CUSTKEY" = ? and "MFGBRAND"."P_BRAND" = "PART"."P_BRAND" and
"PART"."P_PARTKEY" = "LINEITEM"."L_PARTKEY"
 group by "MFGBRAND"."P_MFGR" FOR FETCH ONLY
```

The SQL statement in Example 8-9 is used to get the data for the table at the bottom of the report shown in Figure 8-15 on page 143.

Example 8-9 Query 4 for the operational BI report

```
select "ORDERS3"."O_ORDERKEY" "O_ORDERKEY" , "ORDERS3"."O_CLERK" "O_CLERK" ,
"ORDERS3"."O_ORDERSTATUS" "O_ORDERSTATUS" , "ORDERS3"."O_SHIPPRIORITY"
"O_SHIPPRIORITY" , "ORDERS3"."O_TOTALPRICE" "O_TOTALPRICE" ,
"ALLTIME"."A_CALENDARQUARTERCAPTION" "A_CALENDARQUARTERCAPTION" ,
"ALLTIME"."A_CALENDARDT" "A_CALENDARDT"
 from "TPCH50TB"."ORDERS" "ORDERS3", "TPCH50TB"."ALLTIME" "ALLTIME"
 where "ORDERS3"."O_CUSTKEY" = ? and "ALLTIME"."A_CALENDARDT" =
"ORDERS3"."O_ORDERDATE"
 order by "A_CALENDARQUARTERCAPTION" desc, "A_CALENDARDT" desc FOR FETCH ONLY
```

We use an additional report (Order Details) to show how a user may navigate from the initial overview page in Figure 8-15 on page 143 to a more detailed view of a specific (historical) order. Figure 8-16 shows the report that was built for this purpose. It basically contains a list with rows derived from multiple underlying database tables.



O_ORDERDATE	L_LINENUMBER	L_PARTKEY	P_NAME	L_RETURNFLAG	L_SHIPDATE	L_RECEIPTDATE	L_SUPPKEY	S_NAME	L_QUANTITY	L_EXTENDEDPRICE
Jun 28, 1998	1	8878330363	mint burlywood metallic blush pink	N	Jul 17, 1998	Jul 28, 1998	378330364	Supplier#378330364	16	21,591.52
Jun 28, 1998	2	5944237810	forest almond ghost magenta peru	N	Aug 16, 1998	Aug 28, 1998	194237833	Supplier#194237833	10	18,506.20
Jun 28, 1998	3	4560966821	snow purple salmon midnight spring	N	Sep 9, 1998	Oct 4, 1998	60966822	Supplier#060966822	48	89,269.92

Figure 8-16 Cognos report used for operational BI application (order details)

The SQL statement in Example 8-10 shows how data is retrieved to display the report page in Figure 8-16.

Example 8-10 Query for the order details report

```
with "LINEITEM2" as (
select "LINEITEM2"."L_ORDERKEY" "L_ORDERKEY" , "LINEITEM2"."L_PARTKEY" "L_PARTKEY" ,
"LINEITEM2"."L_SUPPKEY" "L_SUPPKEY" , "LINEITEM2"."L_LINENUMBER" "L_LINENUMBER" ,
"LINEITEM2"."L_QUANTITY" "L_QUANTITY" , "LINEITEM2"."L_EXTENDEDPRICE" "L_EXTENDEDPRICE" ,
"LINEITEM2"."L_DISCOUNT" "L_DISCOUNT" , "LINEITEM2"."L_TAX" "L_TAX" , "LINEITEM2"."L_RETURNFLAG"
"L_RETURNFLAG" , "LINEITEM2"."L_SHIPDATE" "L_SHIPDATE" , "LINEITEM2"."L_RECEIPTDATE"
"L_RECEIPTDATE" , "LINEITEM2"."L_EXTENDEDPRICE" * (1 - "LINEITEM2"."L_DISCOUNT")
"DiscountedPrice" , ("LINEITEM2"."L_TAX" * (1 - "LINEITEM2"."L_DISCOUNT")) *
"LINEITEM2"."L_EXTENDEDPRICE" "TotalTax" , "LINEITEM2"."L_EXTENDEDPRICE" * (1 -
"LINEITEM2"."L_DISCOUNT") + ("LINEITEM2"."L_TAX" * (1 - "LINEITEM2"."L_DISCOUNT")) *
"LINEITEM2"."L_EXTENDEDPRICE" "Extended_Revenue" , "LINEITEM2"."L_QUANTITY" *
"PARTSUPP"."PS_SUPPLYCOST" "Extended_Supply_Cost" , "LINEITEM2"."L_EXTENDEDPRICE" * (1 -
"LINEITEM2"."L_DISCOUNT") - "LINEITEM2"."L_QUANTITY" * "PARTSUPP"."PS_SUPPLYCOST" "Gross"
```

```
 from "TPCH50TB"."LINEITEM" "LINEITEM2", "TPCH50TB"."PARTSUPP" "PARTSUPP"
 where "PARTSUPP"."PS_PARTKEY" = "LINEITEM2"."L_PARTKEY" and "PARTSUPP"."PS_SUPPKEY" =
"LINEITEM2"."L_SUPPKEY")
select distinct "ORDERS4"."O_ORDERDATE" "O_ORDERDATE" , "LINEITEM2"."L_LINENUMBER"
"L_LINENUMBER" , "LINEITEM2"."L_PARTKEY" "L_PARTKEY" , "PART"."P_NAME" "P_NAME" ,
"LINEITEM2"."L_RETURNFLAG" "L_RETURNFLAG" , "LINEITEM2"."L_SHIPDATE" "L_SHIPDATE" ,
"LINEITEM2"."L_RECEIPTDATE" "L_RECEIPTDATE" , "LINEITEM2"."L_SUPPKEY" "L_SUPPKEY" ,
"SUPPLIER"."S_NAME" "S_NAME" , "LINEITEM2"."L_QUANTITY" "L_QUANTITY" ,
"LINEITEM2"."L_EXTENDEDPRICE" "L_EXTENDEDPRICE" , "LINEITEM2"."L_DISCOUNT" "L_DISCOUNT" ,
"LINEITEM2"."DiscountedPrice" "DiscountedPrice" , "LINEITEM2"."L_TAX" "L_TAX" ,
"LINEITEM2"."TotalTax" "TotalTax" , "LINEITEM2"."Extended_Revenue" "Extended_Revenue" ,
"LINEITEM2"."Extended_Supply_Cost" "Extended_Supply_Cost" , "LINEITEM2"."Gross" "Gross"
 from "TPCH50TB"."ORDERS" "ORDERS4", "LINEITEM2", "TPCH50TB"."PART" "PART",
"TPCH50TB"."SUPPLIER" "SUPPLIER"
 where "ORDERS4"."O_ORDERKEY" = "LINEITEM2"."L_ORDERKEY" and "LINEITEM2"."L_SUPPKEY" =
"SUPPLIER"."S_SUPPKEY" and "PART"."P_PARTKEY" = "LINEITEM2"."L_PARTKEY"
 order by "L_LINENUMBER" asc  FOR FETCH ONLY
```

The Cognos query for this report is configured to use a common table expression. While this may make the query look a bit complex at first glance, it can actually be processed quite efficiently in DB2 for z/OS. Even with the 50 TB database as a source, this type of query responds within a few seconds.

8.4.3 Tuning with MQTs

As described in Chapter 10, “Using materialized query tables” on page 183, the creation of effective, efficient MQTs can improve query performance for long running queries (especially in the DW/BI space where data set sizes are generally large, and some amount of data latency may be tolerated). Report authors will have access to the Cognos queries generated by the reports, but they may not have access to the information that describes the sources of the underlying data. Data modelers understand which environments the data sources reside in, but they generally do not have an idea at modeling time of what queries will be generated by the reports that will access the data. Therefore, the proactive creation of MQTs to aid in BI reporting response time will require some coordination of effort between the folks that create the reports and work with the generated queries, and the folks that model the data sources or have access to the data sources.

The following set of images show a sample, long running report that was created within Cognos. This report accesses data that resides solely within the 50 TB database in DB2 for z/OS.

Figure 8-17 on page 147 shows the report page used to enter the selection predicate criteria used in this query. The associated MQT definitions are outlined in Chapter 10, “Using materialized query tables” on page 183.

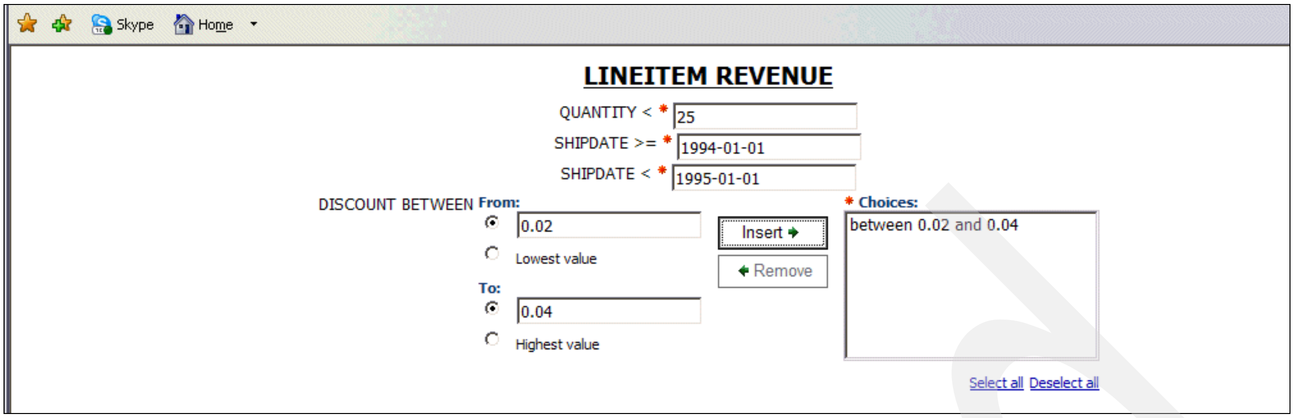


Figure 8-17 Cognos LINEITEM report entry page

After submitting this report with the targeted MQT disabled for query optimization, this long running query took just over two hours to return results. See Figure 8-18. This behavior was expected due to the massive amount of data that was being analyzed and the lack of a perfect index to address this specific query. After enabling an MQT for query optimization, this report returned in under five seconds.

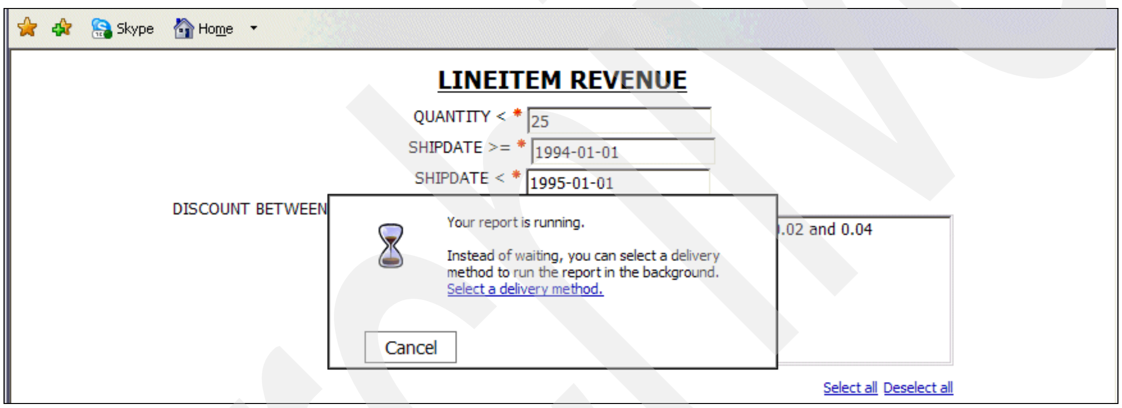


Figure 8-18 Long running report without MQT in Cognos BI

In creating effective MQT definitions to handle complex Cognos queries, users must understand the query syntax that is generated by Cognos under the covers, as well as the source information for the underlying data. In this example, all data came from the same DB2 for z/OS instance. The generated SQL was viewed through Query Explored in Cognos by choosing the Generated SQL/MDX option under the Properties pane. In Figure 8-19 on page 148, the SQL was shown after eliminating the common table expression information.

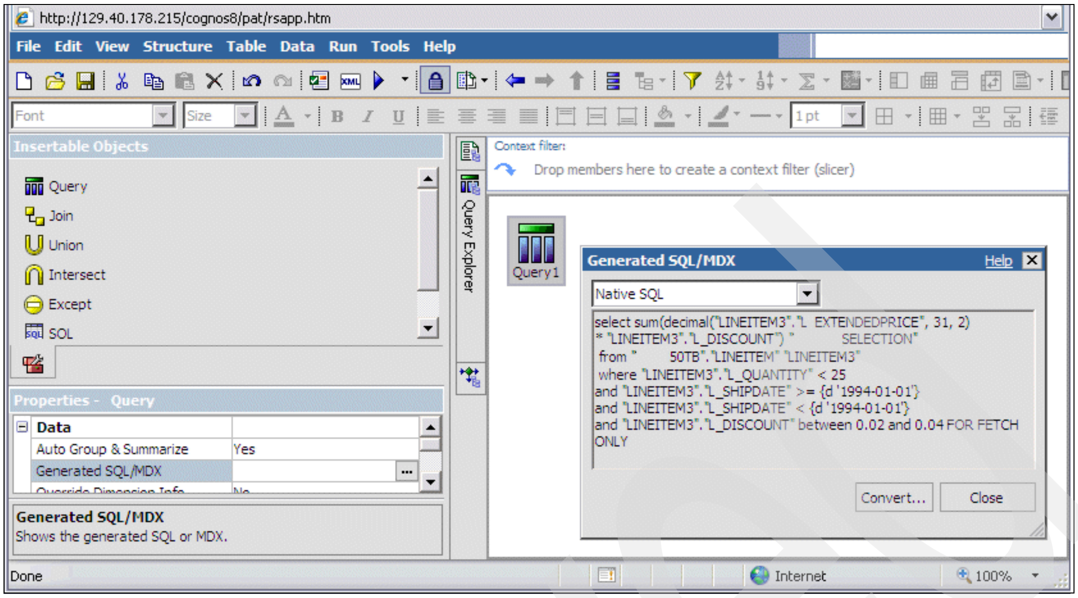


Figure 8-19 SQL statement in Cognos report studio

8.5 Testing the operational BI application

For this book, the model and reports described in 8.4, “The model and reports used for this book” on page 140 were defined and tested with varying numbers of virtual users running in parallel. The following sections describe the setup assumptions that were made and testing outcomes.

8.5.1 Test scenario

For this environment, a scenario was defined where users run a new version of a report OpBI Customer in package VLDB-Model (query). The sequence followed by each user to run a report is illustrated in Figure 8-20 on page 149.

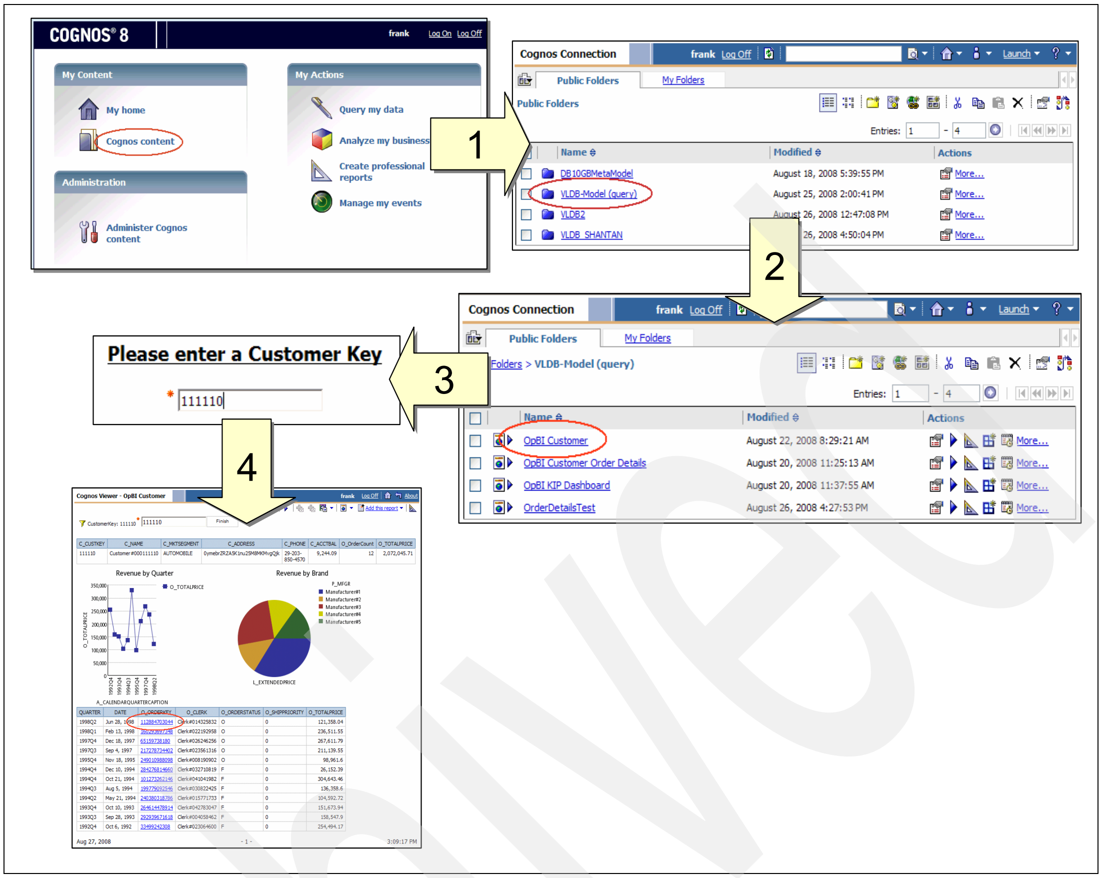


Figure 8-20 Test scenario for Cognos report testing

The steps highlighted in Figure 8-20 illustrate the following activities:

1. When no authentication is required (which is the case in this setup), users typically arrive at the Cognos 8 welcome page and click **Cognos content**
2. The next page shows a list of available (published) packages in the public folder. The user selects **VLDB-Model (query)** to continue.
3. The folder VLDB-Model (query) contains a set of reports. The user selects **OpBI Customer**.
4. Because this report is a prompted report, it asks for a customer key number. In a real application, it is likely that this customer number would be automatically submitted from the transactional system or retrieved otherwise. In this scenario, the user has to key in a valid customer number (111110 in Figure 8-20). After the value is submitted, the operational BI Customer report is shown.

8.5.2 Simulating call center agent interaction

The sequence described in Figure 8-21 on page 150, is run for each of the virtual users, simulating a call center agent. It is assumed that in real world, a user may take one second to read a page and make the next click.

Figure 8-21 on page 150 depicts two kinds of activities, The steps shown in the blue area include navigation in Cognos connection. This is where the user finds the right report to run, enters customer key numbers, and so on. These interactions are required to run the report, but are not accessing the DB2 for z/OS database.

This study is more focused on the interaction that starts after the customer key is provided and the report request is submitted (green area at the bottom). The time it takes to run the underlying database query and render the complete HTML report is what is being reported as transaction time.

One may argue that the navigation steps (in the top, blue area) should be included in a transaction as well. Alternatively, it can be argued that the navigation steps should be ignored because a real user would just repeat step 4 over and over again or create a shortcut in the browser, leading to report creation.

For these tests, the definition of a transaction and the established test sequence was a trade-off that took into account some expected real-life interaction as well as report execution and rendering on System z.

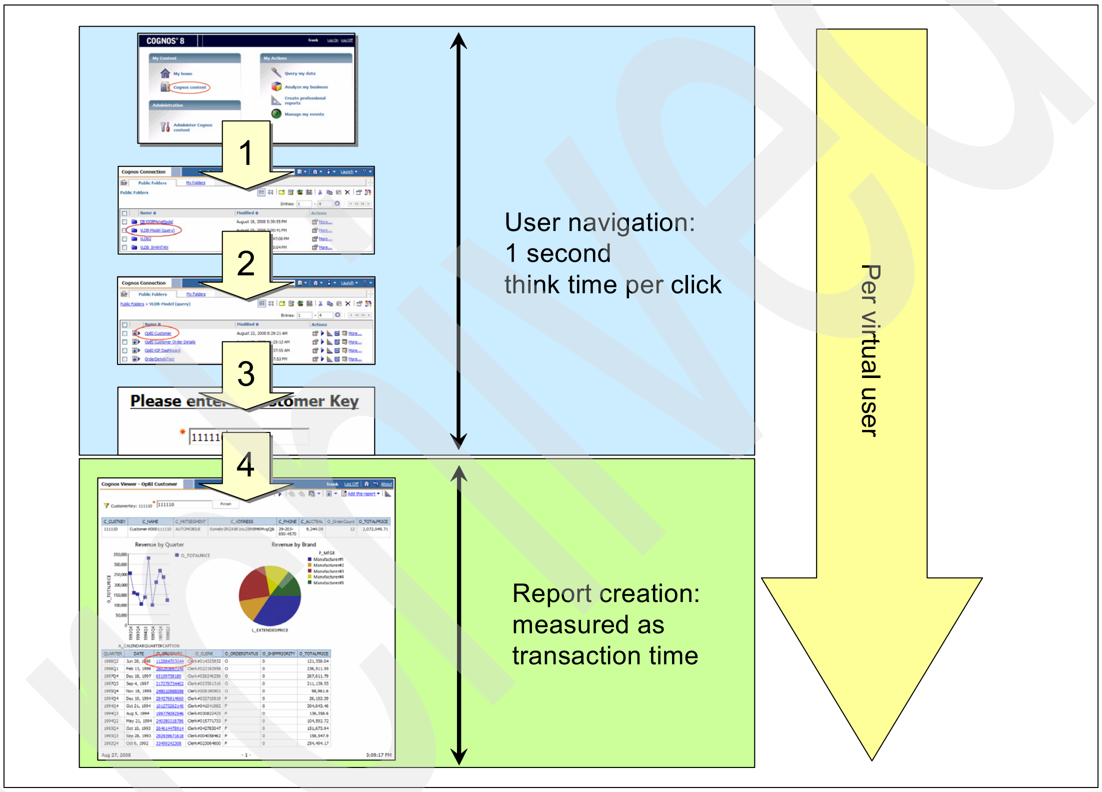


Figure 8-21 Activities per virtual users

Figure 8-21illustrates a single interaction of one user. The following information is a explanation of how multiple users would presumably act over time. Figure 8-22 on page 151 shows the call center agent receiving a customer call. The user is assumed to access operational BI information using the reports similar to the sequence previously described. After some amount of dialogue, the call center agent may make a decision (based on the customer request) to trigger some action against a transactional system.

While customer call durations vary heavily, an assumption was made that call length averages one to two minutes each. For test case setup purposes, this would imply that one virtual user would submit a request for a particular user every one to two minutes.

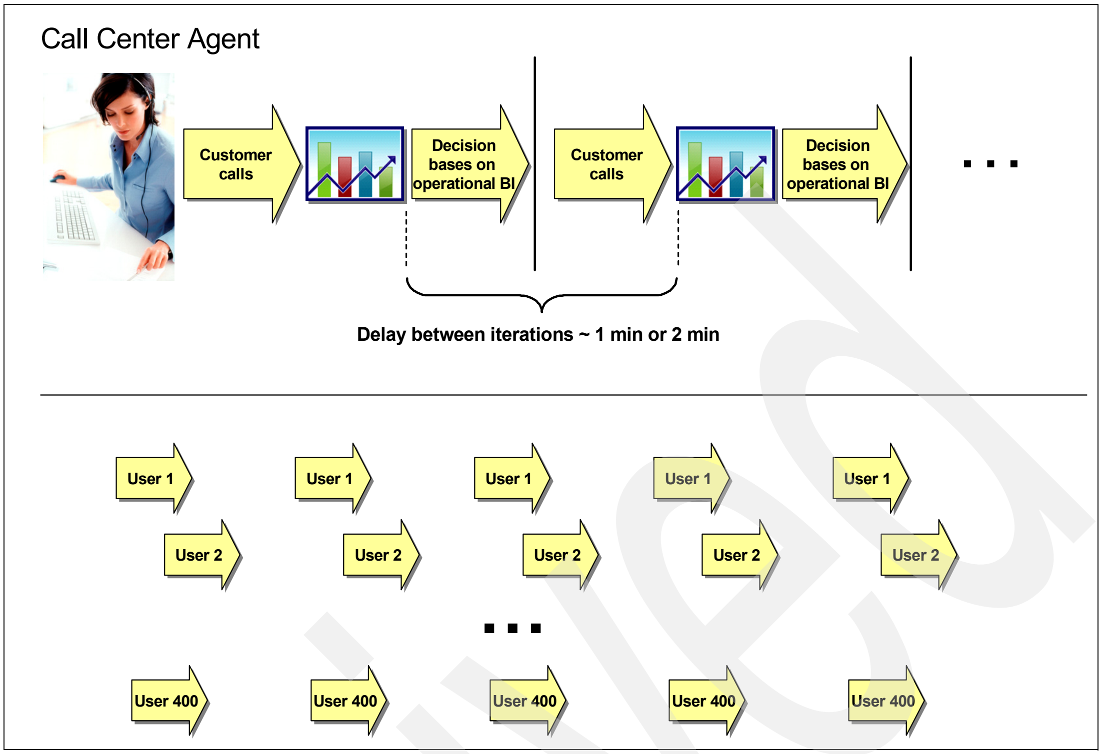


Figure 8-22 Iteration per virtual user

This iteration pattern leads to a set of (randomly) distributed requests from all involved users. See the illustration at the bottom of Figure 8-22. Obviously, the number of users that can be supported within a given environment is typically higher. There is a higher delay between iterations because waiting users do not significantly contribute to resource consumption (except for keeping Web sessions open). Tests were also run with no delay between iterations to determine an artificial limitation on the number of users that could run with delays.

8.5.3 Data for testing and caching considerations

Both DB2 for z/OS and Cognos BI Server use caching techniques to improve response times for requests asking for previously processed data.

The decision was made here to use a prompted Cognos report (one that asks for a parameter, such as a customer or order key number), and to vary the input value for the tests. To do so, a query such as the one shown in Example 8-11 was used to select a set of 4,500 valid customer/order key combinations from the 50 TB database.

Example 8-11 SQL query to select customer/order key combinations

```
SELECT
   C.C_CUSTKEY,
   (SELECT O.O_ORDERKEY FROM TPCH50TB.ORDERS O WHERE O.O_CUSTKEY = C.C_CUSTKEY
    FETCH FIRST ROW ONLY)
FROM
   TPCH50TB.CUSTOMER C
WHERE
   MOD(C.C_CUSTKEY, 1111111) = 0 AND EXISTS
    (SELECT O.O_ORDERKEY FROM TPCH50TB.ORDERS O WHERE O.O_CUSTKEY = C.C_CUSTKEY)
```

Because the customer table itself is partitioned using the customer key as a partition key, it is assumed that the selection gives a fair distribution over the data in the table.

Depending on the size of the associated buffer pool, the buffer pool hit ratio in DB2 for z/OS might still be high because all of the data that is retrieved for this set of customer key values may fit into it. If so, future iterations of the test may benefit from previous iterations. However, this would tend to happen in a real life scenario as well. It is likely that a smaller subset of customers will more frequently and repeatedly be accessed than others.

Additional tests used non-prompted reports (reports that can be run without additional parameter values). For these tests, the query definition was modified in Cognos 8 BI Report Studio to set Use Local Cache to **No**. See Figure 8-23.

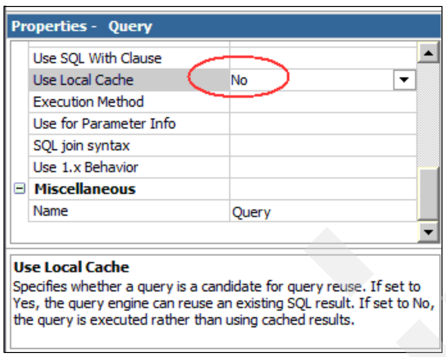


Figure 8-23 Disabling local cache for non-prompted report

With this setting, Cognos 8 BI would avoid reusing a previously computed result, and instead always send the request to the database.

8.5.4 Understanding asynchronous interaction with Cognos BI

The following section provides a more detailed understanding about the technical interaction between components when a new report is requested in Cognos BI Server.

Interaction with Cognos BI Server in these tests occur through a Web browser from the time users select a report of interest until the answer set is rendered as a Web page. This is just one option. Cognos 8 BI offers a rich set of interactive and batch report creation options, including creation of a PDF file, running scheduled reports, and sending them per mail to a distribution list.

For the interactive case, it is important to understand an aspect called *asynchronous interaction* that takes place between the Web browser and Cognos 8 BI when report creation takes longer than some number of seconds. This asynchronous interaction must be considered here because the test cases must be flexible enough to interact efficiently with Cognos BI Server.

> **Note:** The details of asynchronous interaction described here may be inaccurate or incomplete. This potentially simplified view worked for this test scenario. Other scenarios might trigger different, more complex interaction patterns.

Figure 8-24 shows an interaction diagram (simplified) with the involved components:

- The Web browser (including Java Script code)
- The HTTP Server with the Cognos gateway that routes requests to a dispatcher
- The Cognos BI Server with the report server component and the asynchronous processes for interactive report creation

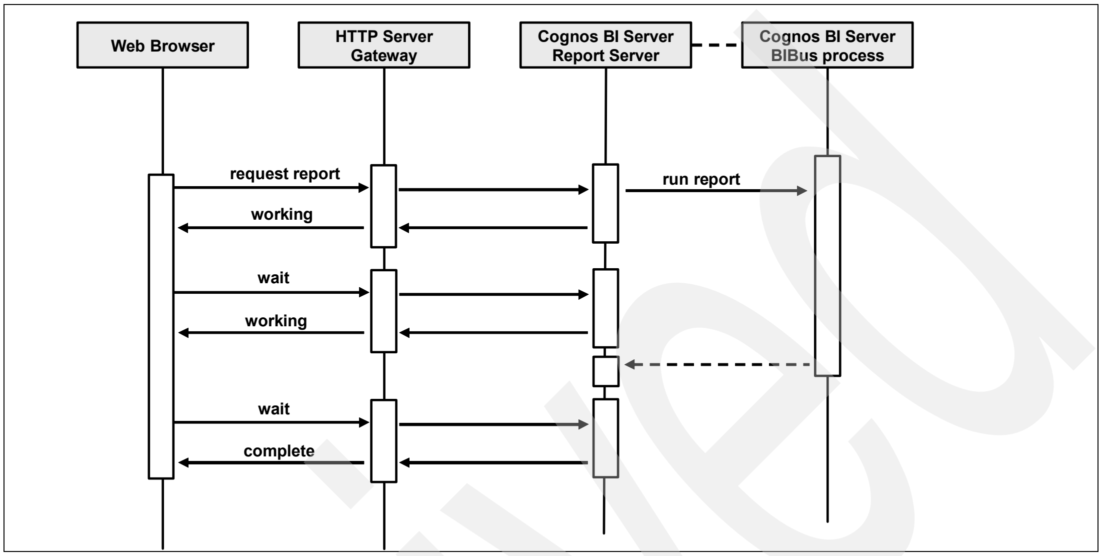


Figure 8-24 Asynchronous interaction between Web browser and Cognos BI server

The Web browser sends a request to create a new report. This essentially occurs when a user clicks on a Web page link in a Cognos Connection Web page. The HTTP server forwards this request to the dispatcher in the Cognos 8 BI application tier, and the report is executed using the report service. If all processes available for the report services are busy, the request may be queued for later processing. In any case, the Cognos BI Server responds to the Web browser with a message, indicating either that the report is complete or that it is still working. In the latter case, the Web browser displays an hourglass symbol and continues to send wait requests to the Cognos BI Server. After the report finally becomes available, Cognos BI Server responds with a complete message to a wait request from the Web browser.

This communication between the Web client and server may take place multiple times. Thus, any test case or script which attempts to simulate this type of user interaction through the Web browser must adhere to this communication protocol.

For our tests, it turned out to be beneficial to monitor the Cognos server log file (cogserver.log) for any error messages regarding asynchronous interaction during load test runs. Example 8-12 shows an error message that could point to an issue with an unsuccessful communication with the report server.

Example 8-12 Errors in cogserver log

```
816     1       Audit.RTUsage.RSVP      Response        ReportService
Failure RSV-BBP-0036 The request 'asynchWait_Request' failed because a previous
request that returned a fault cancelled the conversation.
```

8.6 Testing with Rational Performance Tester

IBM Rational® Performance Tester is a multi-user load testing and performance testing tool for validating Web application scalability. It creates, executes, and analyzes tests to validate the reliability of complex, Web-based applications.

It has the following features:

- Delivers both high-level and detailed views of tests with a rich, tree-based text editor
- Performs capacity planning tests to ensure optimal investment in hardware and IT infrastructure.
- Delivers automatic identification and support for dynamic server responses
- Enables large, multi-user tests with minimal hardware resources.
- Diagnoses the root cause of performance bottlenecks by quickly identifying slow performing lines of code and integrates with Tivoli composite application management solutions to identify the source of production performance problems.
- Supports HTTP, Siebel, SAP®, SIP, Entrust, Citrix and SOA/Web Services Supports Windows, Linux and z/OS as distributed controller agents
- Supports Linux and Windows

Rational Performance Tester was used because it provides the features and capabilities needed to test the complex interactions with Cognos BI Server. Furthermore, it is able to render numerous types of reports which provide insight into performance and scalability behavior over the entire test cycle.

More and more customers are using load and scalability test tools in their environment and in the context of a system verification tests long before they go into production with a complex combination of components. This is done to ensure that service level agreements (SLAs) and stability criteria are met. Therefore, using a tool such as Rational Performance Tester is advised when it comes to product level testing.

For a good introduction of the concepts and details relating to Rational Performance Tester, refer to *Using Rational Performance Tester Version 7*, SG24-7391.

8.6.1 Test setup

Three Windows workstations with Intel® Xenon dual core processors were used for Rational Performance Tester. One workstation was installed with RPT workbench and accessed through the Remote Desktop. The other two workstations acted as agents for the test. See Figure 8-25 on page 155.

The agents were connected to the Linux on System z instance where all of the Cognos 8 BI components were installed, as described in 8.3.1, “Product components” on page 131. Additionally, a connection was established from the Linux on System z LPAR to the z/OS LPAR where the 50 TB DB2 for z/OS database was running.

A dedicated, high-speed performance network was employed, along with an additional administration network, to access the systems within the environment.

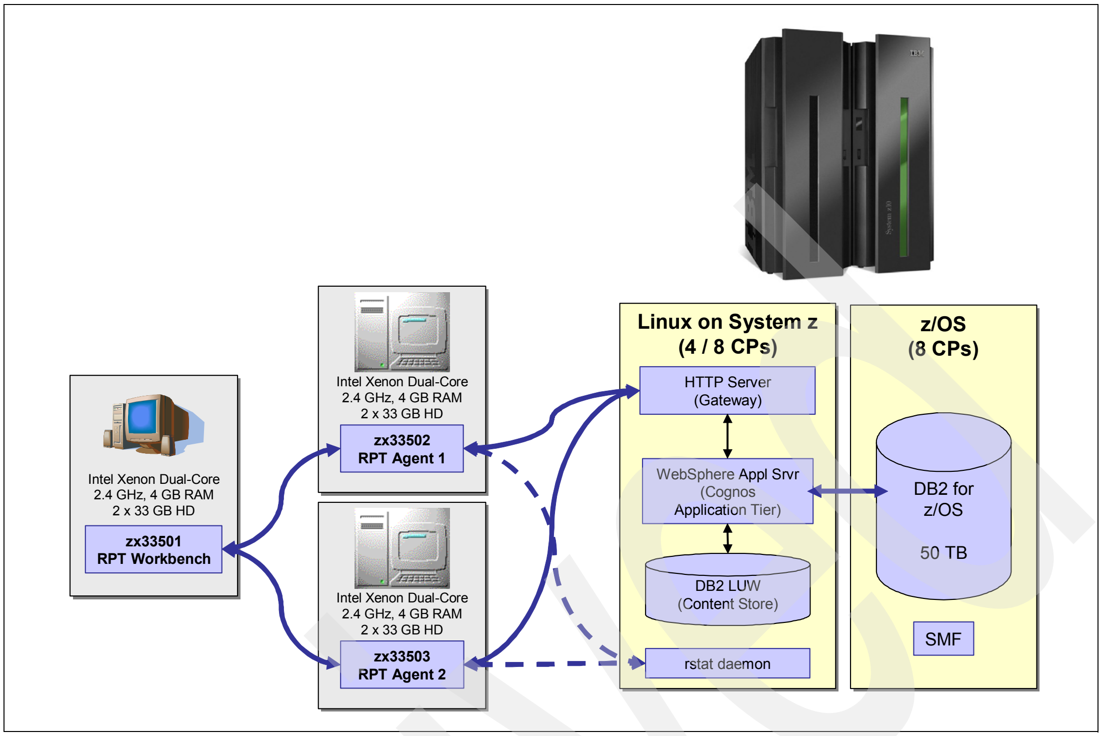


Figure 8-25 Test environment used for the book

During our tests, it turned out that a single RPT agent was sufficient to drive the tests we performed. We used the second RPT agent just as a backup system.

8.6.2 Test sequence

The test sequence we ran mainly used the operational BI report and the assumptions outlined in 8.5, "Testing the operational BI application" on page 148.

To prepare the system environment on Linux for System z, the steps illustrated in Figure 8-26 on page 156 were applied before each test run to create reproducible results.

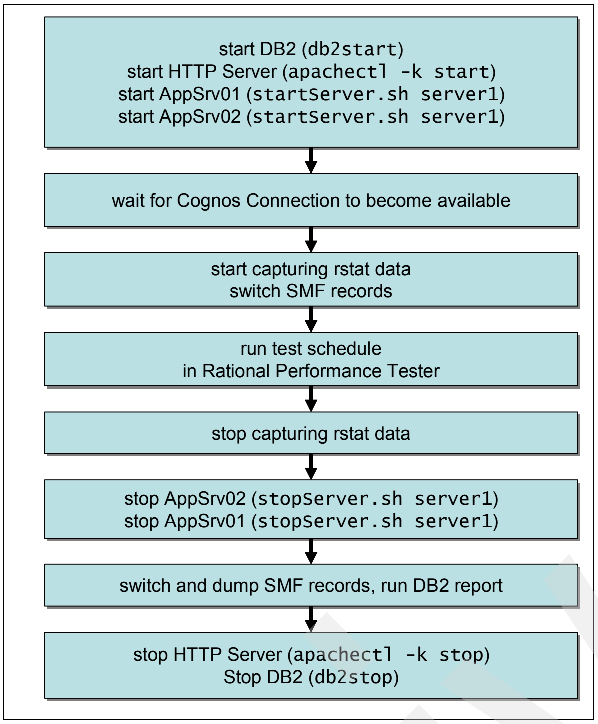


Figure 8-26 Steps performed for each test run

Because the Cognos content manager component runs on AppSrv01, it is indicated to start the corresponding WebSphere Application Server first and then proceed with AppSrv02.

For capturing SMF data appropriately, we stopped both application servers first to ensure that database connection still kept alive in connection pools are closed before we switch SMF records.

8.6.3 Performance tests

With Rational Performance Tester, we first recorded the test sequence with the built-in HTTP recording function. See Figure 8-27. The result is a performance test with all the HTTP request and responses.

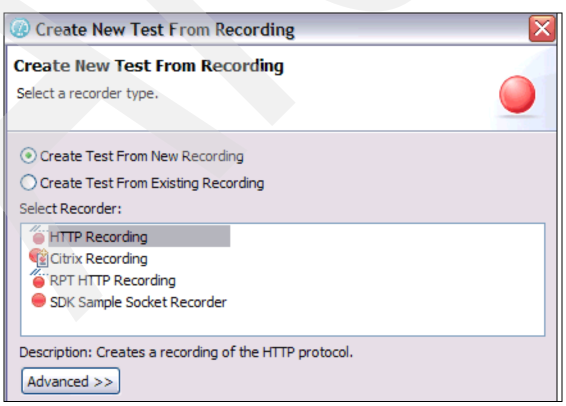
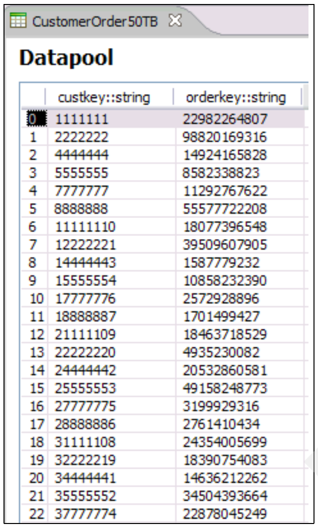


Figure 8-27 HTTP recording of a performance test in Rational Performance Tester

Because we run and test a prompted report (a Cognos report which requests an input value, in our case the customer key number), we created a data pool with a set of valid values for the test. These values have been created as described in 8.5.3, “Data for testing and caching considerations” on page 151. In Rational Performance Tester, we defined a data pool with these values (Figure 8-28) that are used for the performance tests.


CustomerOrder50TB

Datapool

	custkey::string	orderkey::string
0	1111111	22982264807
1	2222222	98820169316
2	4444444	14924165828
3	5555555	8582338823
4	7777777	11292767622
5	8888888	55577722208
6	11111110	18077396548
7	12222221	39509607905
8	14444443	1587779232
9	15555554	10858232390
10	17777776	2572928896
11	18888887	1701499427
12	21111109	18463718529
13	22222220	4935230082
14	24444442	20532860581
15	25555553	49158248773
16	27777775	3199929316
17	28888886	2761410434
18	31111108	24354005699
19	32222219	18390754083
20	34444441	14636212262
21	35555552	34504393664
22	37777774	22878045249

Figure 8-28 Data pool definition for customer and order numbers

The recorded performance test was adapted and enhanced with the interaction required to handle the asynchronous interaction described in 8.5.4, “Understanding asynchronous interaction with Cognos BI” on page 152 as well as loop and transaction handling.

Figure 8-29 gives an overview of the elements in the performance test we used (refer to the numbers in circles in the figure).

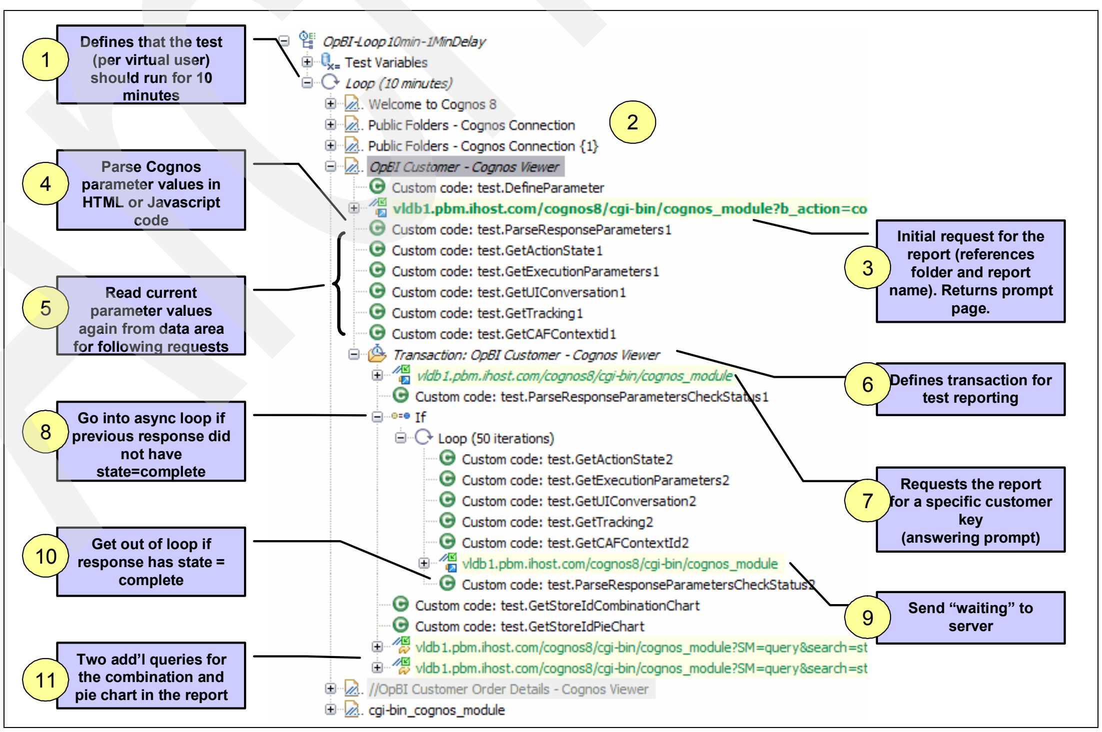


Figure 8-29 Performance test in Rational Performance Tester

1. The outermost loop defines how long or often a single virtual user should perform the enclosed test sequence. In our case, we typically ran for 10 or 15 minutes. This is also the place where delays between iterations were defined (0, 1 or 2 minutes).
2. As defined in 8.5.1, “Test scenario” on page 148, there a some navigational steps to get from the Cognos welcome page to the report. This sequence of requests were not modified and used mostly unchanged from the previous recording.
3. The initial request for the report contains references to the folder name and the report name and returns the prompt page to enter the customer key number.
4. A piece of custom code is used to parse the response of the previous request and extract various parameter values. These values are stored in data areas for subsequent requests.
5. A set of custom code snippets are used to read again the parameter values (such as action state and tracking) from the corresponding data area elements. This is required because following HTTP communication with the Cognos BI Server needs these values to be set and match with the values retrieved in the previous response.
6. A transaction in Rational Performance Tester defines the set of enclosed elements that are considered for transaction reporting. In our case, we defined a transaction as the sequence between submitting the prompt page and until all parts of the reports are received. You can have multiple transactions defined for a test and can also control them from inside custom code.
7. The element in step 7 is essentially the major HTTP request for the report as it submits the customer key value from the data pool and requests report creation.

   Figure 8-30 shows an example for the test element details of this request. The section in green are the customer key which are substituted from an associated data pool. The key values in orange are substituted by the return values of previous custom code elements.

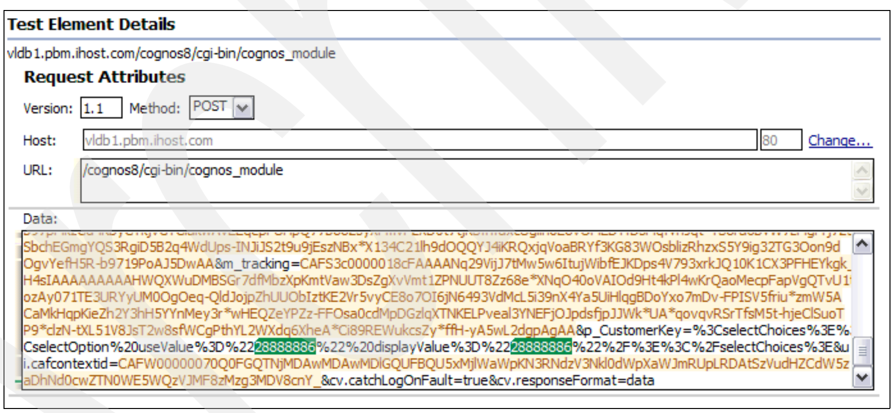


Figure 8-30 Test element details in Rational Performance Tester

8. Due to the potential asynchronous interaction between client and server, these steps use a condition (if), based on the status value from the previous response and a loop to mimic the client/server interaction for longer running reports.
9. Inside the loop, step 9 continues to send wait messages to the server until the report becomes available.
10. The custom code in step 10 (see Example 8-13 on page 159) shows how the response from the previous request is parsed and in case Cognos BI server reports a status complete how the breakLoop() function is used to exit the loop. Note that the code snippet refers to code in other custom code element (DefineParameter).

Example 8-13 Custom code snippet to check status and exit the loop, if required

```
IDataArea dataArea = tes.findDataArea(IDataArea.VIRTUALUSER);
ITestLogManager testLogManager = tes.getTestLogManager();
DefineParameter.readParametersHtml(args[0], dataArea, testLogManager);
String status = (String) dataArea.get(DefineParameter.ASYNCSTATUS_KEY);
tes.getTestLogManager().reportMessage("loop - status: " + status );

if( status.equals("complete"))
{
   tes.getLoopControl().breakLoop();
}
```

11. Two additional requests are required to retrieve the images for the combination and the pie chart in the report. Values for storeID parameters are again captured in responses from previous requests.

8.6.4 Performance test schedules

The performance schedule defines how a performance test should be run. See Figure 8-31 on page 160. It includes definitions of the following terms:

- User load (the number of virtual users with ramp-up time definition)

  We ran between 30 and 900 virtual users and used a ramp-up definition so that a new user starts every 600 milliseconds or every second

- Think time

  We varied the think time between 75% and 125% of the time specified in the performance test.

- Resource monitoring

  We monitored CPU use on Linux for System z with accessing the rstat daemon on Linux. Unfortunately, due to setup issues, we were not able to fully capture CPU on the system for the test runs and instead correlated this data later manually.

- Statistics and problem determination

  We used default settings for statistics and problem determination. To track custom code and interaction with Cognos BI server during the tests, we used the ITestLogManager interface in custom code (see Example 8-13) to write messages to the test log events.

- Which agents to use

  We ran all tests on a dedicated Rational Performance Tester agent machine.

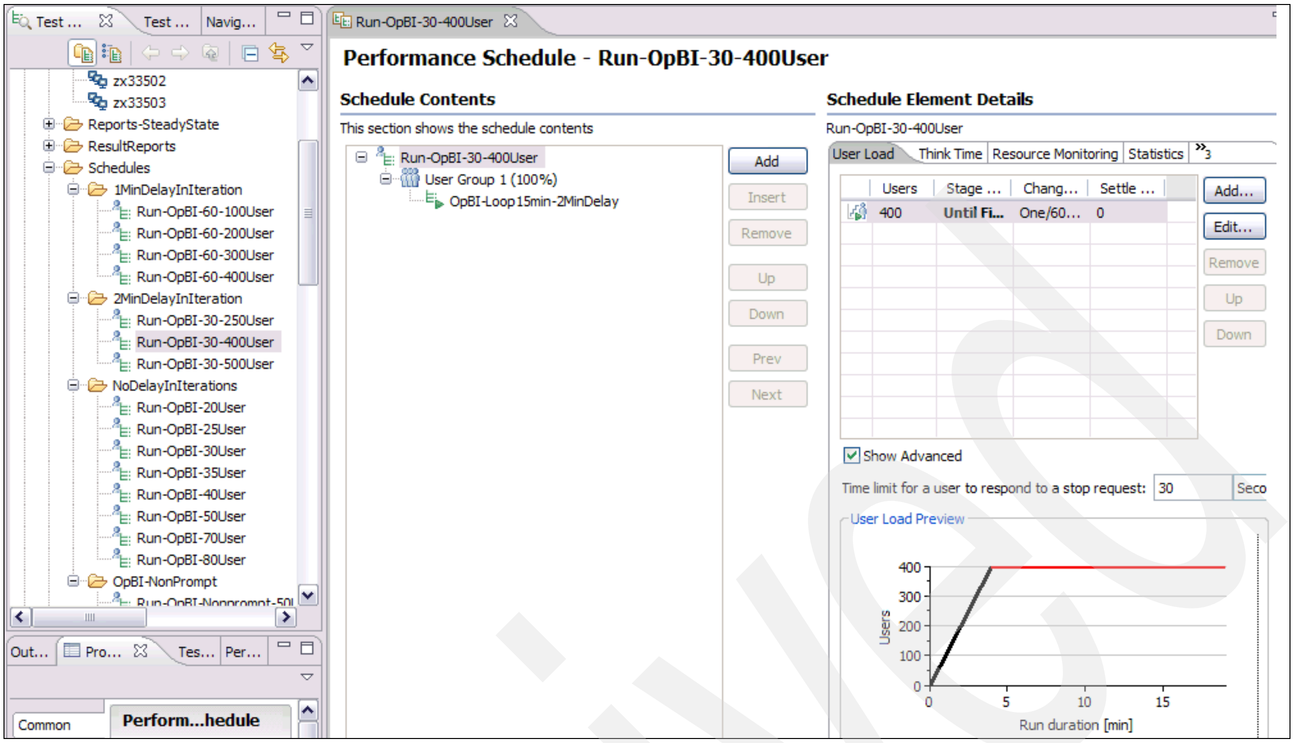


Figure 8-31   Performance schedule for the 400 users test

8.6.5 Session handling with Rational Performance Tester and Cognos BI Server

For test design with Rational Performance Tester, it is important to understand the implication of defining a loop (iterations) in the performance schedule compared to defining a loop in the performance test itself.

Figure 8-32 on page 161 illustrates these two options. On the left hand site, the performance schedule defines a loop for the performance test which in turn run a set of HTTP requests. On the right hand side, the performance schedule invokes the performance test only once and a loop in the performance test causes repeating execution of the HTTP requests.

From an execution point of view, both approaches result in the same set of HTTP requests. However, the major difference is in session handling. By default, Rational Performance Tester closes all connections (and the session) with a server when a performance test completed and continues to use existing connections when running in a loop within a performance test. Particularly when testing Cognos reports, you have to make sure that session handling is configured in a way that your tests can cope with an increasing number of new session requests.

While both options are valid and possible, for our tests, it turned out to be beneficial to loop in the performance test rather than in the performance schedule.

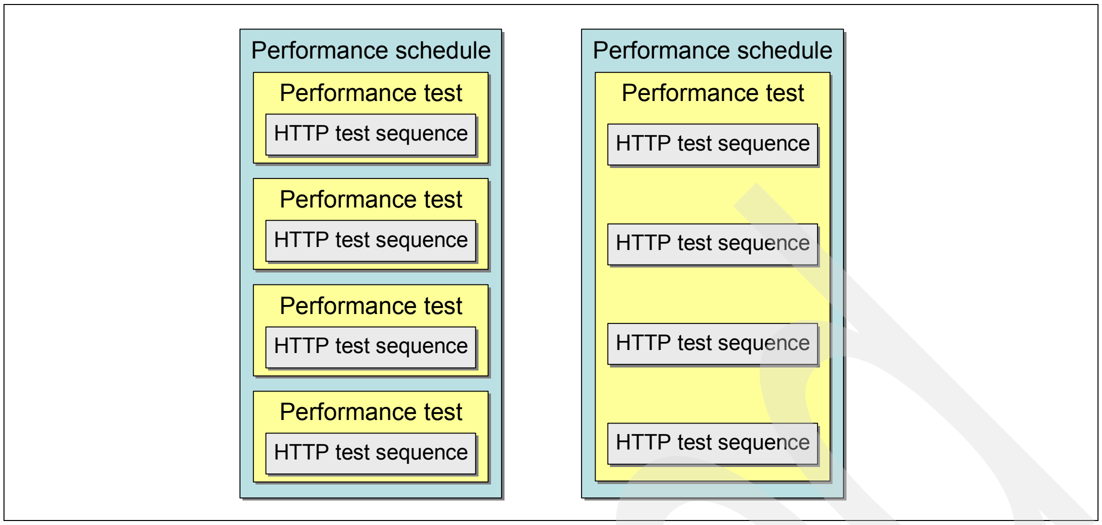


Figure 8-32 Looping in performance schedule versus looping in performance test

8.7 Test results

The objective of the tests for this book was not to create official performance measurements for operational BI reporting with Cognos on Linux for System z. To do so, we would have to set up a more controlled environment, apply tuning to the involved components, and use a variety of operational BI reports (we used two for our test). Results are also heavily dependent on the model and the report definition. Adding more complex figures, creating PDF output, or defining more complex aggregation for the reports could easily change the results and lead to great differences. Therefore, we have not published the detailed figures that we recorded during our experiments for this book. Instead we have summarized the highlights and emphasized our test methodology.

8.7.1 Test result summary

The objective for this book was to investigate whether Cognos 8 BI on System z and DB2 9 for z/OS could cope with the demands of operational BI in the context of a large 50 TB database.

- Tests were successfully run with 400 concurrent users simulating call center agents accessing an operational BI report with two-minute delays between iterations. See scenario description in 8.5, “Testing the operational BI application” on page 148. For these tests, a 1.75 seconds average response time was achieved for query and report creation per user over a 15 min. run (steady state), at 56% CPU use on Linux for System z.
- DB2 for z/OS provides efficient access to operational BI data, and CPU use on z/OS remained at (5-10%) during these tests.

The configuration and the operational BI reports were able to support

- 400 users with two minute delays between iterations (1.75 s average transaction time)
- 200 users with one minute delays between iterations (1.63 s average txn time)
- 50 users with no delay between iterations (4.02 s average txn time)

When varying the number of CPs assigned to Linux for System z

- Using 8 CPs instead of 4 CPs increased throughput by ~66%

A simpler report based on Order Details (see 8.5, “Testing the operational BI application” on page 148) was able to run with

- 900 users with two minute delays between iterations (0.6 s average transaction time)

8.7.2 Test details

Figure 8-33 shows how transaction duration time (for a definition, see 8.5, “Testing the operational BI application” on page 148) changes with the number of simultaneous users accessing the system.

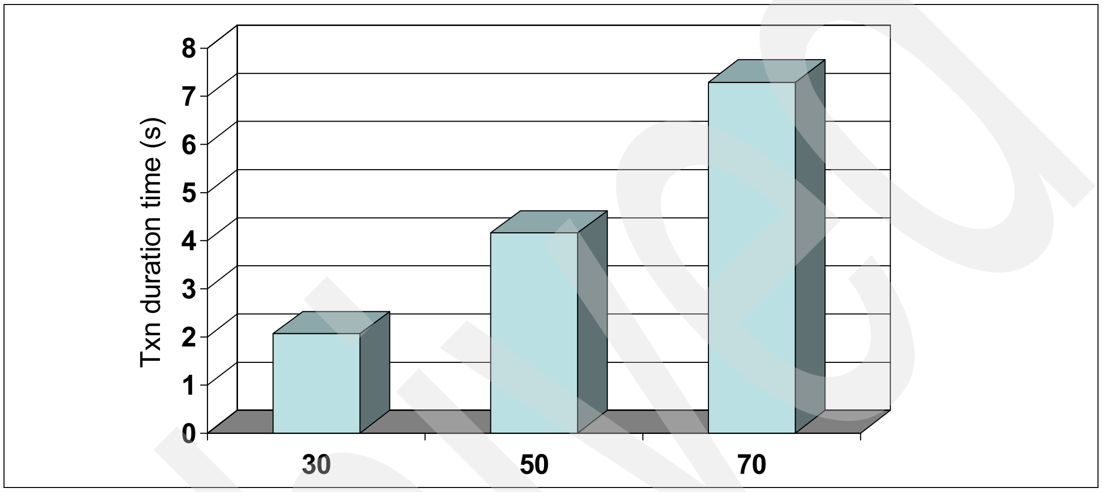


Figure 8-33 Scalability with number of concurrent users

Figure 8-34 shows average CPU use on Linux for System z during tests where all users are accessing the system to request reports.

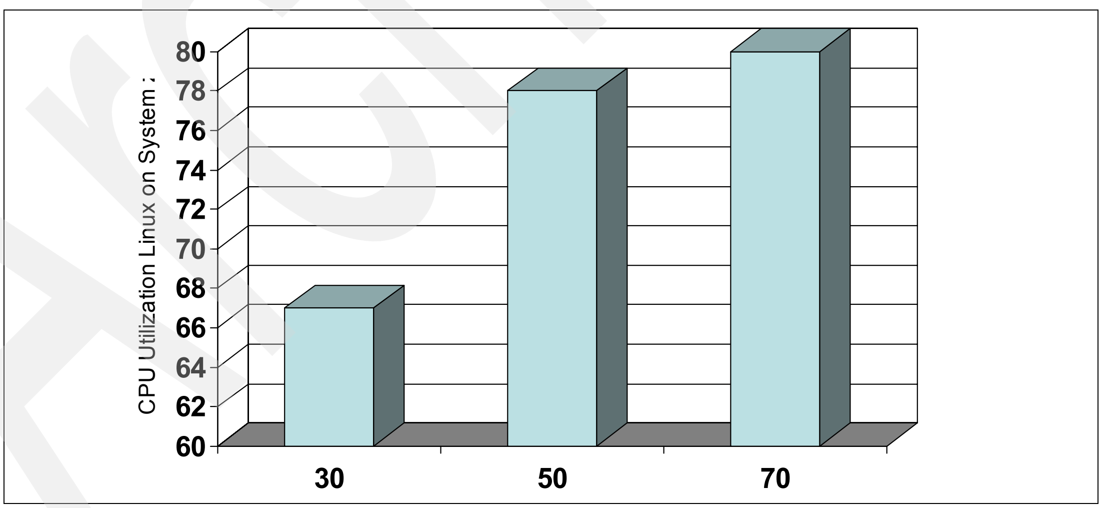


Figure 8-34 CPU use on Linux for System z for simultaneous users

Figure 8-35 shows the scalability of the environment after moving from 4 CPs on Linux for System z to 8 CPs. The number of CPs for z/OS were unchanged. Because CP use did not exactly match for test cases with 4 CPs and 8 CPs, the transaction rate was normalized to 100% (for example, a rate of 100 transactions per minute at 60% would equal a rate of 166.6 at 100%).

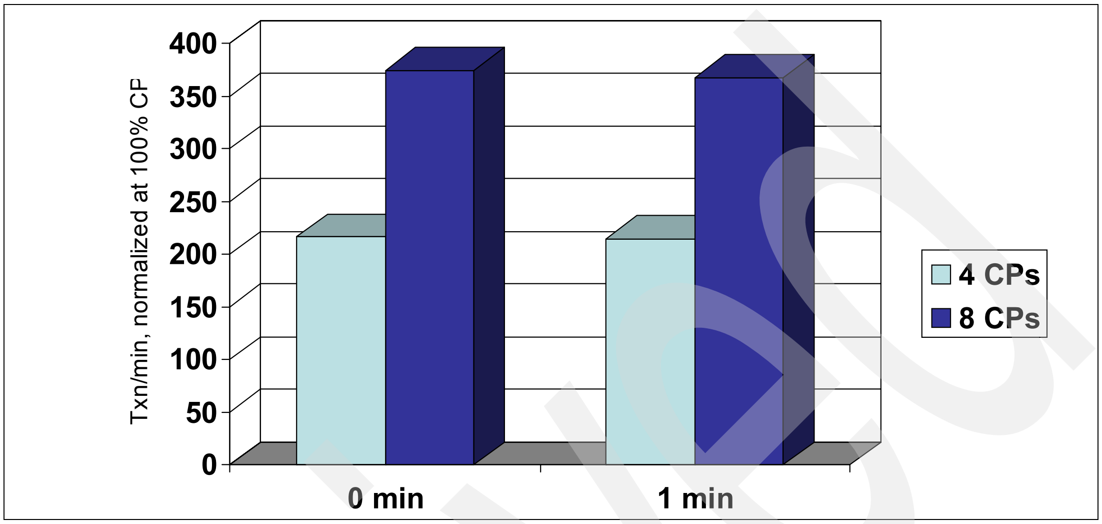
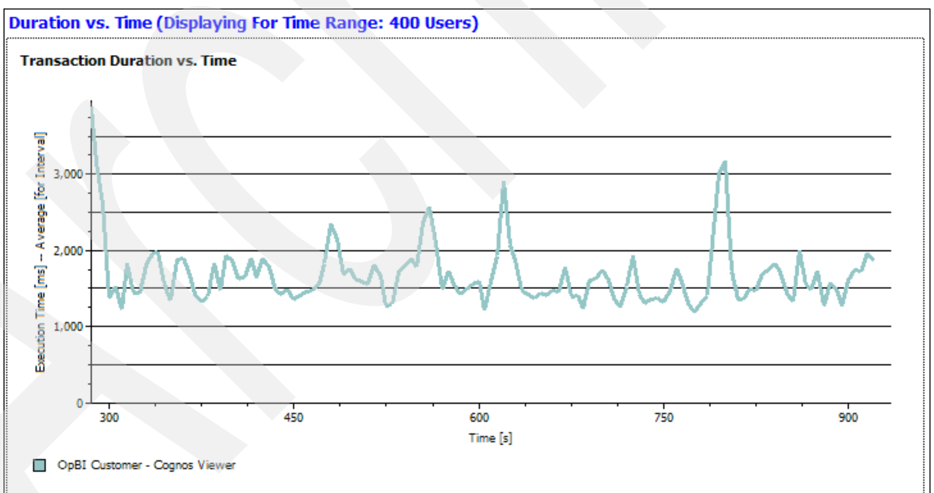


Figure 8-35 Impact of doubling CP capacity on Linux for System z on transaction rate

Another metric of interest was the transaction behavior over time. Figure 8-36 shows transaction duration in milliseconds during steady state time for a test with 400 users. While there are some peaks in response times, transaction duration stays quite stable over the entire test run.



	Execution Time [ms] -- Minimum [for Run]	Execution Time [ms] -- Average [for Run]	Execution Time [ms] -- Maximum [for Run]	Execution Time [ms] -- Standard Deviation [for Run]	Completed -- Rate [per second] [for Run]	Attempts -- Count [for Run]
OpBI Customer - Cognos Viewer	891	1,751.1	9,750	722.6	3.21	2,049

Figure 8-36 Rational Performance Tester transaction report for 400 user test

Figure 8-37 shows data of the simpler order details test (only one SQL query and report without graphics) run with 900 users.

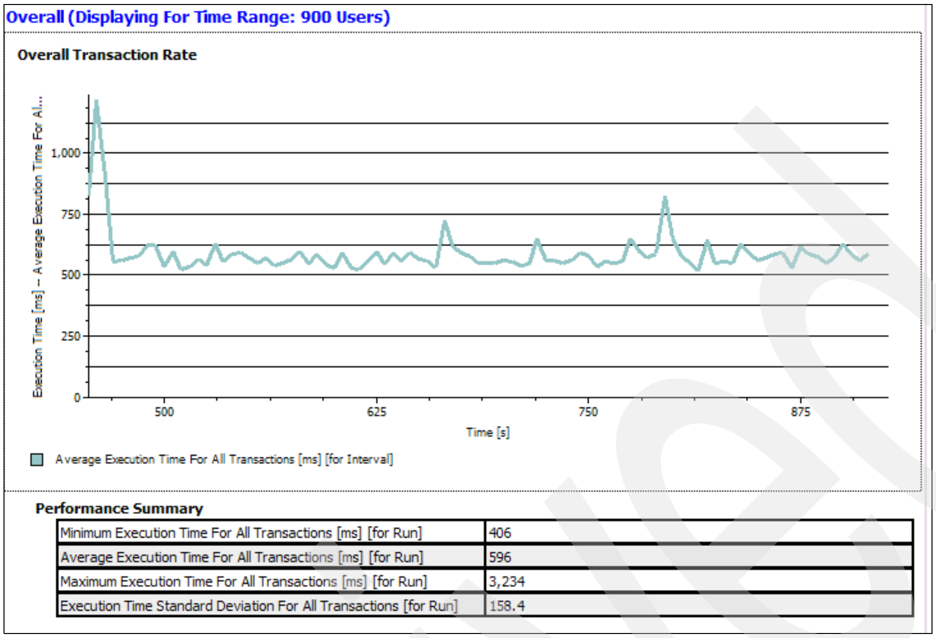


Figure 8-37 Rational Performance Tester transaction report for 900 user test

A final point of interest was the amount of memory consumption during the test, particularly for the two Java Virtual Machines (JVMs) and the report service processes. Figure 8-38 on page 165 shows the characteristics during a test with 300 users and a one minute delay per iteration. The chart at the top right was taken with JPerfmeter, and the green curve at the top shows CPU use on Linux for System z.

The screen shot at the bottom is based on output from top, a command line tool on Linux for System z. Fourteen report service processes were configured for this run (14 processes named BIBusTKServerManager in the list). There are also two Java processes for the two Application Server profiles, AppSrv01 and AppSrv02. They both consume about 1.5 to 1.7 GB virtual memory. DB2 for Linux for System z contributed 16% (related to one of 8 CPs) CPU use. Overall memory use was about 13 GB, leaving 3.2 GB free with no swap space being used. Another test using sixteen report servers consumed closer to the 16 GB memory limit.

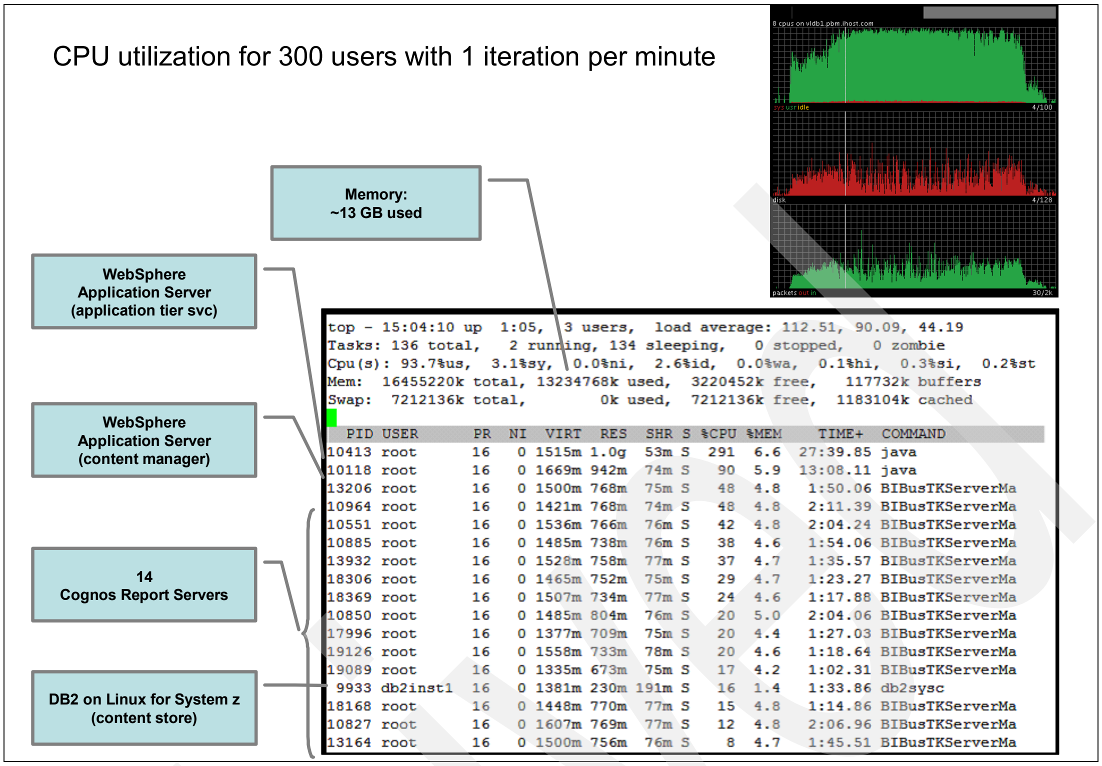


Figure 8-38 CPU and memory use snapshot taken during a 300 users test

9

Scalability

As businesses grow, their user and data processing needs also grow. Business ventures, such as mergers, acquisitions, new services, or new government regulations, can accelerate how quickly the data processing needs of the business grow. As rapid growth occurs, companies need a way to scale their business successfully.

Scalability is a important capability for enterprises across the world today. Businesses need to be able to scale as needed without impacting existing processes or requiring major changes to the way business is executed. The ease with which a system is scalable should be one of the major factors that determines the database that hosts an enterprise data warehouse.

This chapter examines processor and data scalability for tests conducted on the 50 TB data warehouse benchmark.

9.1 Overview

Scalability is the ability of a system to maintain its performance levels with a proportional increase in hardware, software and workload.

The database supporting an organizations data warehouse must be scalable to support the growth due to increases in data and increase in the number of transactions. The ability to support this rapid growth will enable a business to run successfully.

Scalability can be thought of as scale up and scale out. This evaluation demonstrates the scale up capabilities of DB2 on System z, both from a processor and data perspective. These terms are described in the 9.1.1, "Scale up" on page 168 and 9.1.2, "Scale out" on page 168.

There were two different sets of tests executed to measure processor scalability. The first set of tests was to execute a concurrent user workload, which simulates a customer workload to measure processor scalability. This is discussed in detail in 9.2, "Concurrent user processor scalability" on page 168. The second set of tests was to execute a single query against various processor configurations. This is discussed in detail in 9.3, "Single query processor scalability" on page 175.

Data scalability tests were executed in a manner similar to the single query processor scalability tests. Single queries were executed against the same configuration to go after larger quantities of data. This is discussed in detail in 9.4, "Data scalability" on page 179.

> **Attention:** Performance data in this chapter was obtained in a controlled environment with specific performance benchmarks and tools. This information is presented along with general recommendations to assist the reader to have a better understanding of IBM products. Results obtained in other environments may vary. The data presented here does not predict performance in a specific installation environment.

9.1.1 Scale up

Scaling up, also referred to as scaling vertically, means adding additional resources to the current system. Examples would be processors and memory.

9.1.2 Scale out

Scaling out, also referred to as scaling horizontally, means adding new systems to the existing system. Scaling out is a way to improve scalability when a single system cannot be scaled up any longer, but additional resources are required for the business.

During the 50 TB benchmark, scalability has been demonstrated through scale up (processor scalability and data scalability). Scale out was not in scope for this project.

9.2 Concurrent user processor scalability

Processor scalability is the adding of additional processors to the system so as to increase available processing capacity. The workload throughput for a concurrent user workload should improve relative to processors. In an ideal scenario, the improvement is expected to be linear.

A concurrent user workload was used to simulate a customer workload and tests executed against the 50 TB database to demonstrate processor scalability. The test methodology, workload characterization and performance results and observations are described in the sections that follow.

9.2.1 Test methodology

To measure the performance of the system with various configurations, it is necessary to have a uniform methodology and a repeatable workload. With a uniform methodology and a repeatable workload, the measurements can be compared for accuracy. During our tests, each of the workloads are run according to the same set of rules, to assure that results can be compared between each measurement. Neither changes to setup or operation, nor unique tuning activities are done to favor any configuration.

To measure performance for concurrent user processor scalability, an LSPR-like SI (Single Image) start to finish methodology was used. Large System Performance Reference (LSPR) is the IBM assessment of relative processor capacity in an unconstrained environment. A start to finish methodology measures time from the start of processing of the workload till the end of processing of the workload. This methodology is used in a batch environment.

> **Note:** For more information about LSPR, visit the following Web page:
>
> http://www.ibm.com/systems/z/advantages/management/lspr/

To ensure consistency, reliability, and repeatability of the results across measurements, each test scenario was executed multiple times. We were looking to be repeatable within 2% of the internal throughput rate and consistently achieved it within 1%. Internal throughput rate is defined in “Performance metrics” on page 170. The steps taken to ensure consistency, reliability, and repeatability from measurement to measurement are as follows:

1. Recycle DB2.
2. Execute the priming jobs.

   The jobs executed for our measurements prepared the system for the execution.
3. Flush the buffer pools to set them to zero.
4. Reset the buffer pools to the required values.
5. Start the measurements when the workload is started.

   The steps taken as part of starting the measurements are as follows:

   a. Switch SMF.
   b. Start RMF.
   c. Start DB2 statistics for this measurement. Example 9-1 shows the command used to start the DB2 statistics for the measurement. The same command effectively stopped the DB2 statistics and cut a record, as it is a modify command.

      Example 9-1 Command used to start/stop (modify) the DB2 statistics

      ```
      VLD9 MOD TRA(STAT) CLASS(1,3,4) TNO(1)
      ```

   d. Start measurement.
6. End the measurement.

   The steps taken as part of ending the measurements are as follows:

   a. Stop DB2 statistics for this measurement.
   b. Stop RMF.

Processor scaling parameters

Concurrent user processor scalability was measured with the workload being executed against multiple configurations varying the number of processors, that is, a 4-way, 8-way, 16-way, and 32-way configuration. The workload used to measure processor scalability was a set of queries in three categories (small, medium, and large) based on their CPU consumption. The number of queries in each category and number of initiators for each query category was proportionately scaled based on the number of processors for each configuration.

Table 9-1 displays the scaling parameters for the workloads created to measure processor scalability. Measurements were done using the general purpose processors. As can be seen, the number of initiators and number of queries varies with the n-way in a proportional manner. Thus, for a 4-way configuration, the number of initiators is 1:2:2 and the number of queries is 350:234:48 for small, medium, and large respectively. For an 8-way configuration, the number of initiators is 2:4:4 and the number of queries is 700:468:96 for small, medium, and large respectively. The ratio for small, medium, and large queries was kept at about 55:35:10.

Table 9-1 Concurrent user processor scalability parameters

Number of General Purpose Processors	Number of Initiators			Number of Queries		
	Small	Medium	Large	Small	Medium	Large
4	1	2	2	350	234	48
8	2	4	4	700	468	96
16	4	8	8	1400	936	192
32	8	16	16	2800	1872	384

Performance metrics

For our tests, the CP time and response (elapsed) time was measured from the start of the execution till the end of execution of all the jobs. The workload was not memory or I/O constrained. The concentration for the tests was only on the processor itself. The workload is described in 9.2.2, “Workload characterization” on page 171.

To measure performance itself, the two metrics usually used in the industry are the external throughput rate (ETR) and internal throughput rate (ITR). *External throughput rate* is an elapsed time measure and focuses on system capacity. *Internal throughput rate* is a processor time measure and focuses on processor capacity.

- ETR: Number of transactions per unit of time.

  For our tests:

  ETR: Number of queries per minute

- ITR: Number of transactions per CPU busy time.

  The formula is as follows:

  ITR = (ETR / CPU%) * 100

We use then internal throughput rate from the 4-way configuration measurement as the baseline to calculate the ideal internal throughput rate of the various configurations. We then compare with the ITR measured during the tests. Section 9.2.3, “Performance results” on page 172 describes the results in detail.

9.2.2 Workload characterization

Understanding the workload is crucial to any performance measurement analysis. Moreover, for an accurate measurement, there should be consistency between the workloads used for tests against various configurations. In this section we discuss the workload characteristics.

The maximum degree of parallelism for the concurrent user processor scalability measurements was set to five from measurement to measurement, to have consistency between measurements.

Query categories

The workload was a combination of queries in three categories. The categorization was done based on the average CPU processing time consumed.

- Small queries were defined as having a CPU processing time of 0.3 to 5 seconds.
- Medium queries were defined as having a CPU processing time of 17 to 30 seconds.
- Large queries were defined as having a CPU processing time of 20 to 55 seconds.

Medium sized queries used more CPU cycles and generally accessed more data than small sized queries. Large queries used more CPU cycles and generally accessed more data than medium sized queries. All the queries were designed to be highly CPU intensive and did not have any memory or I/O constraints.

Query profile

The queries for the measurements in each of the three categories described were a mix of various types. Our aim for the measurements was to get a good mix of queries so as to ensure a good customer type workload. The queries for our workload were read-only dynamic SQL and generally processor intensive queries, so as to use maximum CPU.

The queries in the small category ranged from two table joins to eight table joins, with all having a group by and order by clause. The queries were a mix of correlated and non-correlated.

The queries in the medium category ranged from a two-table to a five-table join of various types, having a mix of group by and order by clauses. Once again, the queries were a mix of correlated and non-correlated.

The queries in the large category ranged from a single table to a two-table join have a mix of group by and order by clauses.

Query DB2 statistics

The DB2 Performance Monitor (DB2PM) statistics report provides detailed information regarding the work performed by the DB2 subsystem. We used this report to extract important information for each of the bufferpool sizes within each of the query categories. Table 9-2 on page 172 displays the query DB2 statistics.

Table 9-2 Query DB2 statistics

Characteristic	Small			Medium			Large		
	4 K	8 K	16 K	4 K	8 K	16 K	4 K	8 K	16 K
Bufferpool hit ratio	17.72	86.00	58.41	40.50	41.21	43.23	4.20	0	0
No. of getpages	19722	345	13669	289404	3512	65861	1064985	12533	0
No. of synchronous reads	648	56	96	4437	73	929	42	12173	0
No. of seq prefetch requests	0	0	0	0	0	0	0	0	0
No. of list prefetch requests	0	0	0	0	0	0	0	0	0
No. of dynamic prefetch requests	695	1	1556	6148	203	3141	31367	84	0
No. of pages read asynchronously	14352	14	10170	75679	3157	24593	1002610	631	0

- The buffer pool hit ratio is a measure of how often a page access (a getpage) is satisfied without requiring an I/O operation.
- Number of getpages shows the number of times DB2 requested a page from the buffer manager.
- The number of synchronous reads is the synchronous read I/O operations performed by DB2.
- Sequential prefetch reads a sequential set of pages.
- List prefetch reads a set of data pages determined by a list of RIDs taken from an index.
- With dynamic prefetch, DB2 can automatically adjust between multi-page prefetch and single page synchronous read, as needed for optimal I/O performance.
- The number of asynchronous reads is the asynchronous read I/O operations performed by DB2.

9.2.3 Performance results

In this section we discuss the results for the concurrent user processor scalability tests executed during the 50 TB benchmark. We executed the tests as described in 9.2.1, “Test methodology” on page 169 and 9.2.2, “Workload characterization” on page 171. User experiences may vary based on the actual environment in which the tests are executed.

Table 9-3 on page 173 displays the external throughput rate, CPU use percentage, internal throughput rate, and the ideal internal throughput rate for the measurements taken against the various n-way configurations listed. We use the internal throughput rate from the 4-way configuration measurement as the baseline to calculate the ideal internal throughput rate of the various configurations. Thus, the ideal ITR for a 8-way configuration is twice the ideal ITR of a 4-way configuration and the ideal ITR for a 32-way configuration is 8 times the ideal ITR of a 4-way configuration. The values for the external throughput rate, internal throughput rate and the ideal internal throughput rate are per minute, that is, queries per CPU busy minute.

Table 9-3 Processor scalability workload throughput measurements

Number of processors	External Throughput Rate (per min)	CPU %	Internal Throughput Rate (per min)	Ideal ITR (per min)
4	21.90	96.67	22.62	22.62
8	43.68	96.72	45.18	45.24
16	85.98	98.03	87.72	90.48
32	159.6	96.40	165.7	180.96

Table 9-3 shows that the ideal internal throughput rate we calculated with the 4-way as the baseline is close to the internal throughput rate from the measurements for each of the n-way configurations. Figure 9-1 compares the measured internal throughput rate against the ideal internal throughput rate for each of the n-way configurations.

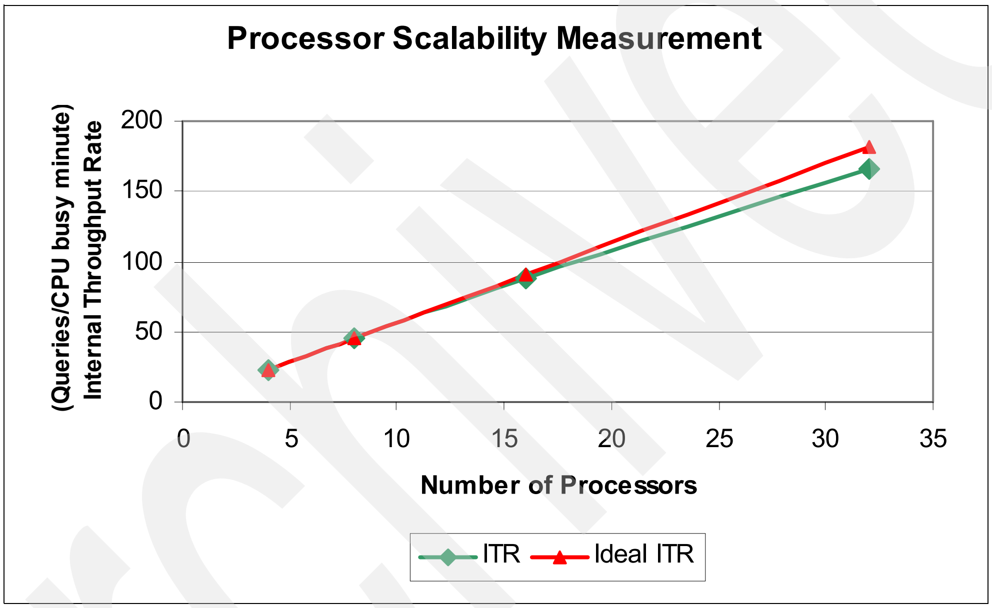


Figure 9-1 Number of CPs versus internal throughput rate

Processor use for each of the n-way measurements were consistently high between 96%–98%. The internal throughput rate, which is a measure of the external throughput rate with respect to processor use, is close to the external throughput rate. On comparison with the calculated ideal internal throughput rate, the results show we have achieved near linear scalability for these measurements.

These impressive throughput results can help customers scale their concurrent user requirements considerably with the existing performance levels.

Table 9-4 on page 174 displays the average CP time and the average response (elapsed) time in seconds per query for each of the three categories of small, medium, and large queries. These values are from the measurements described in Table 9-1 on page 170. We see that for each of the query categories, the average CP time and the average response time within a category with a proportionately scaled system is consistent.

Table 9-4 Processor scalability time measurements

Number of General Purpose Processors	Average CP Time per query (seconds)			Average Response Time per query (seconds)		
	Small	**Medium**	**Large**	**Small**	**Medium**	**Large**
4	1.30	18.86	23.00	4.49	14.57	53.82
8	1.34	19.02	22.63	4.71	14.35	68.31
16	1.31	19.53	23.38	4.36	14.61	60.01
32	1.37	20.25	24.61	4.82	15.72	74.02

9.2.4 Observations

The throughput results from the concurrent user processor scalability tests are impressive and can help customers. Customers looking to scale their systems can expect DB2 for z/OS to help maintain their existing performance levels based on the type of workload they are executing.

In an ideal scenario, the workload throughput should improve with additional processors. This ideal improvement is expected to be linear. However, in the real world, the performance improvement is not linear but has diminishing returns. This is because when all the processors in an n-way are working, there are times when the processors need to wait for exclusive access to a shared resource. There is also a need for processing capacity to control and orchestrate the interaction between various resources. Thus, as we move from a 4-way to a 8-way, a 8-way to a 16-way and so on, we do not see a linear improvement in performance throughput. This effect is known as the SMP[1] effect (or the n-way effect).

In comparing the LSPR ratios for various n-ways, the SMP effect can be seen. See 9.2.1, “Test methodology” on page 169 for a brief description and a link for LSPR.

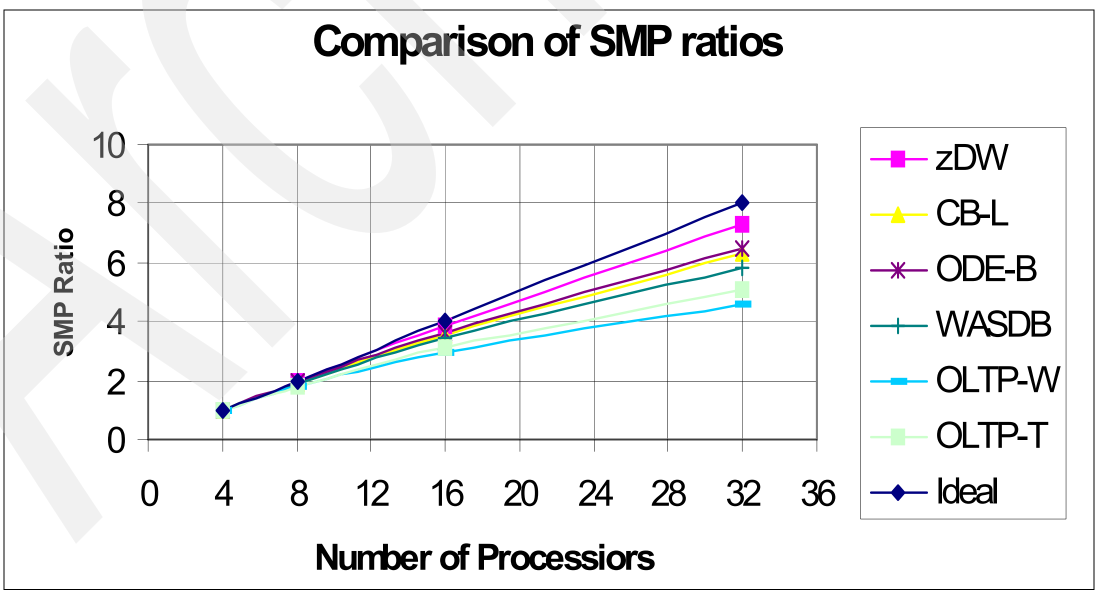


Figure 9-2 Comparison of SMP ratios

[1] Symmetric multiprocessing effect, in contrast with the massively parallel processing (MPP) effect.

Figure 9-2 on page 174 displays an interesting comparison. If we took the processor scalability measurement results of our workload (represented by zDW in Figure 9-2 on page 174), and plotted them along side the traditional single image LSPR workloads, we see impressive scaling characteristics. The ratios for each of the n-way configurations were calculated using the internal throughput rates of the 4-way configuration as the base. The same method was used to determine the ratio for each of the SI LSPR workload's with the 4-way configuration as the base against which the ratio was determined. The ideal ratio is a multiplication factor of the number of CPs. With the above method, we see the characteristics demonstrated by the zDW workload, which is better than all the other LSPR workloads and close to the ideal scenario.

Table 9-5 shows a sample of how the ratios were determined. For example, the CB-L workload LSPR ratio for a 4-way 2097-704 processor is 6.84, which was considered to have a base value of 1. The 8-way ratio is thus a factor of the 4-way, which is 1.91. The ratios for the other workloads were similarly calculated.

Table 9-5 Table for ratio determination

Number of CPs	Ideal	zDW ratio against base of 1	CB-L	CB-L ratio against base of 1
4	1.00	1.00	6.84	1.00
8	2.00	2.00	11.90	1.91
16	4.00	3.88	22.00	3.53
32	8.00	7.33	39.58	6.34

Note: The LSPR ratio's were based on a single z/OS V1 R8 image on System z10 EC while our executions were on z/OS V1 R9. We used the LSPR ratio for the 2097-704, 2097-708, 2097-716 and 2097-732 for each of the 4-way, 8-way, 16-way and 32-way configurations. Visit the following Web page for more information about LSPR:

http://www-03.ibm.com/systems/z/advantages/management/lspr/

With the above observations, we see a reduced SMP effect as we achieve near linear scalability for CPU intensive workloads. This is an impressive and scalable system.

9.3 Single query processor scalability

Section 9.2, "Concurrent user processor scalability" on page 168 used a workload that simulated a customer workload. We measured processor scalability using single query processor scalability tests, which are discussed in this section. We discuss the single query processor scalability measurements to demonstrate processor scalability. We measure the response time and CP time due to an increased number of processors being available to a single query. The test methodology, workload characterization, performance results, and observations are described in the sections that follow.

9.3.1 Test methodology

The methodology followed for our tests are consistent across measurements as described in 9.2.1, "Test methodology" on page 169. This is so that the measurements can be compared for accuracy. Neither changes to setup or operation, nor unique tuning activities are done to favor any configuration.

Measurements were taken based on the start to finish methodology. Repeatability tests were executed for each of the measurements for consistency and reliability of results across measurements. We were looking to be repeatable within 2% between measurements and consistently achieved it within 1%. Steps for each measurement is described in 9.2.1, “Test methodology” on page 169.

Performance metrics

The CP time and the response time for each execution of the query in each of the n-way configurations was measured from the start of query execution to the end of query execution. Section 9.3.3, “Performance results” on page 176 describes the results in detail.

9.3.2 Workload characterization

A single query was executed against configurations of a 8-way, 16-way, 32-way, and a 56-way to measure processor scalability. The processing capacity was increased and it was expected that the processing time reduced proportionally. This test was executed with two different sets of queries specifically chosen so that the value of the additional processors being available for an improved performance can be used. The queries required intensive CPU processing time.

The degree of parallelism was set to ANY for the single query processor scalability measurements allowing the optimizer to determine and choose the best degree of parallelism for the respective measurement. Using all the available CPU cycles to the maximum extent possible was important to the performance measurement. The queries ran with the highest degree of parallelism available and being CPU intensive queries, consumed all the CPU cycles, resulting in an accurate measurement.

9.3.3 Performance results

In this section we discuss the results for the single query processor scalability tests executed during the benchmark. We executed the tests as described in 9.3.1, “Test methodology” on page 175 and 9.3.2, “Workload characterization” on page 176. User experiences may vary based on the actual environment in which the tests are executed.

Figure 9-3 on page 177 displays the results for the response time measurements (in minutes) for query #1 against the n-way configurations. When executed against a 16-way configuration, the response time decreased proportionally in comparison with the response time against a 8-way configuration. The percentages displayed in each of the bars for the 16-way, 32-way, and 56-way configurations represents the performance ratio when compared against the 8-way configuration as a base. Figure 9-4 on page 177 displays the results for the CPU time measurements (in minutes) for query #1 against the various n-way configurations. We see excellent CPU time results with consistent CPU usage in the various configurations for the same query.

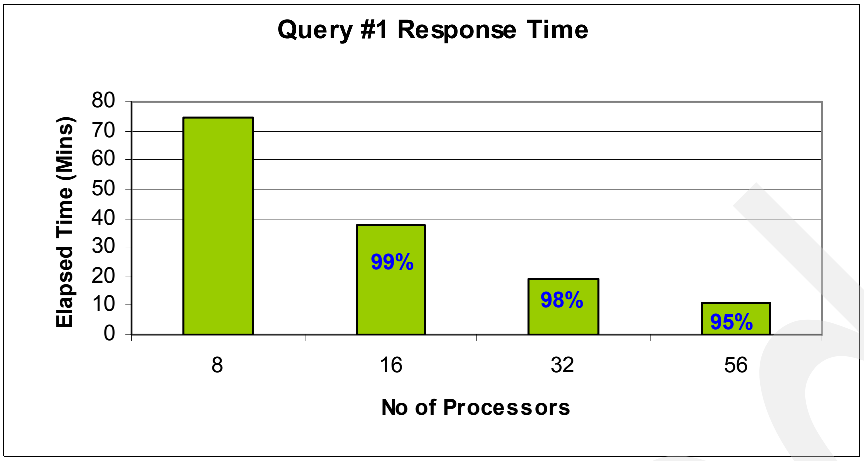


Figure 9-3 Query #1 response time against multiple processors

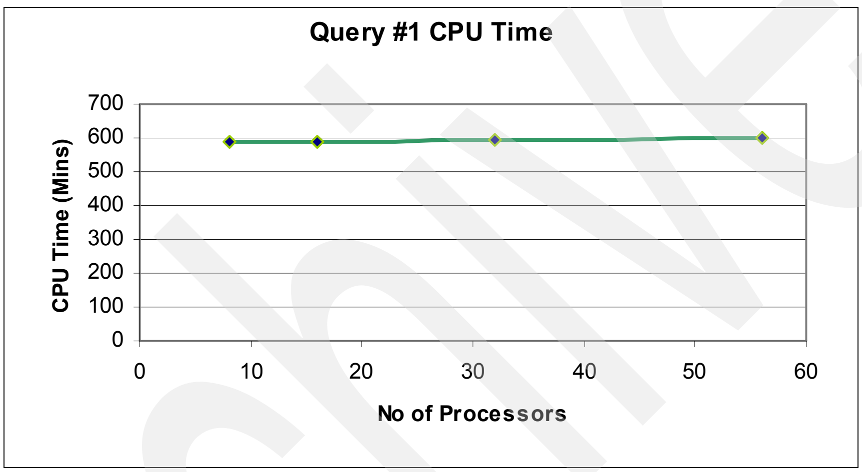


Figure 9-4 Query #1 CPU time against multiple processors

Figure 9-5 on page 178 displays the results for the response time measurements (in minutes) for query #2 against the n-way configurations. When executed against a 16-way configuration, the response time decreased proportionally in comparison with the response time against a 8-way configuration. The percentages displayed in each of the bars for the 16-way, 32-way, and 56-way configurations represent the performance ratio when compared against the 8-way configuration as a base. Figure 9-6 on page 178 displays the results for the CPU time measurements (in minutes) for query #2 against the different configurations. We see excellent CPU time results with consistent CPU usage in the various configurations for the same query.

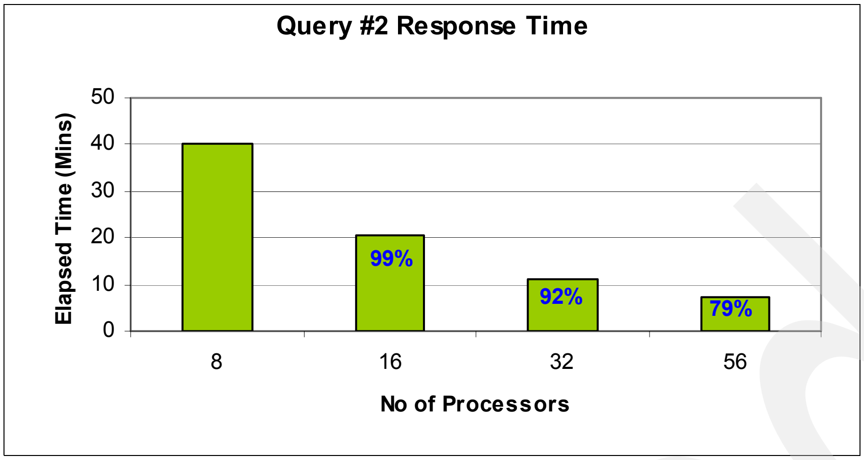


Figure 9-5 Query #2 response time against multiple processors

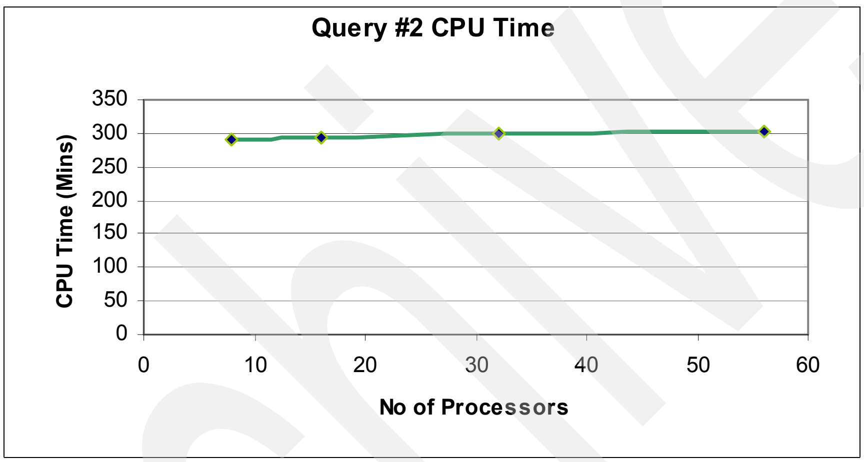


Figure 9-6 Query #2 CPU time against multiple processors

9.3.4 Observations

Impressive processor scalability performance was seen for the single query measurements. There was a consistent and proportional reduction in the response time with an increase in the number of processors and a constant CPU time measurement for the same query. As expected, for the same query executing against any number of processors, the total CPU time for executing the query should be consistent.

The key here is the proportional reduction in response time, resulting in a scalable system indeed. For business critical situations requiring the use of the enterprise BI solution for important queries, excellent scalability can be achieved through the additional CPUs available. Responses for business critical BI workloads can be controlled effectively, allowing businesses to determine importance and assign an increased amount of critical resources, that is, processors, to these BI workloads for a better and faster response time.

As the number of processors are increased, the SMP effect described in 9.2.4, “Observations” on page 174 is visible. As we increase the number of processors, there is an overhead added because of the need of processing capacity to control and orchestrate the interaction between various resources. As we see, for processor intensive BI workloads, a highly reduced SMP effect is observed with excellent response time and a consistent CPU time.

9.4 Data scalability

Data scalability is the ability of the system to respond appropriately to the growing enterprise data. When the data for a data warehouse doubles for the same system capacity, the system should have the ability to continue processing queries in double the amount of time. If the data grows to four times the original configuration, the query performance is expected to get longer with the same system configuration. The important factor is that the system should have the ability to continue processing and provide accurate results, instead of giving unpredictable results. This is what would make the system data scalable.

As businesses grow, their data processing needs also grow. Over the years, businesses keep growing their data warehouses, moving to larger and larger data warehouses. Business ventures, such as mergers, acquisitions, new services, or new government regulations, can accelerate how quickly the warehouse grows. New customers and increasing orders also contribute to the growing warehouse, and in turn, a rapid change of requirements for BI and data. As this rapid growth in data occurs, enterprises need to be able to scale as per their data growth to execute their business successfully and sustain the existing business requirements along with developing new methodologies.

It is important for enterprises to understand the type of data growth that occurs within their organization and the impact it has to them. Not all data growth demands increased system resources and not all data growth impacts system performance. As an example, if an enterprise primarily accesses the last 6 months of data, but continues to accumulate years of data, the growth will not impact performance as the work done against their data warehouse is consistently only for the last 6 months. Thus, the historical data build up does not impact performance of the enterprises data warehouse environment. On the other hand, business ventures, such as a merger or an acquisition, that causes an increase in data that is accessed as part of the day to day operations, can cause a performance impact. In such a scenario, enterprises would have to determine their data scalability needs and put in place the right infrastructure and technology to support the corresponding growth.

To demonstrate this capability of DB2 for z/OS, tests have been executed to demonstrate data scalability. The test methodology, test data used, workload characterization and results are described in the sections that follow.

9.4.1 Test methodology

The methodology followed for our tests are consistent across measurements as described in 9.2.1, “Test methodology” on page 169. This is so that the measurements can be compared with each other for accuracy. Neither changes to setup or operation, nor unique tuning activities are done to favor any configuration.

The measurements for the data scalability tests were taken using a single query to run against the 50 TB database to retrieve varying amounts of data ranging from 1 TB to 50 TB.

For an accurate performance measurement, it is essential that all available CPU cycles are used to the maximum extent possible. This is so that CPU time is not wasted and consistency is maintained across measurements. During the data scalability measurements, we chose a to execute the measurements against a 16-way processor configuration across all the measurements. This allowed the CPU to be busy through each measurement, whether the query was executed to retrieve 1 TB data or whether the query was executed to retrieve 50 TB data.

Measurements were taken based on the start to finish methodology. Repeatability tests were executed for each of the measurements for consistency and reliability of results across measurements. We were looking to be repeatable within 2% between measurements and achieved it consistently within 1%. Steps for each measurement is described in 9.2.1, “Test methodology” on page 169.

Performance metrics

The CP time and the response time for each execution of the query against the 16-way configuration was measured from the start of query execution to the end of query execution. Section 9.4.3, “Performance results” on page 180 describes the results in detail.

9.4.2 Workload characterization

A single query was executed against a 16-way configuration to measure data scalability. The processing capacity and all other execution factors were kept constant, but the query was modified to retrieve data of various sizes, ranging from 1 TB to 50 TB from the 50 TB data warehouse. In such a situation, it is expected that the response time and processing time increases proportionately.

This test was executed with two different sets of queries specifically designed such that the data accessed is proportionately increased. The index and data access were proportionate so as to keep consistency and reliability between the amount of data accessed.

The degree of parallelism was set to ANY for the single query processor scalability measurements allowing the optimizer to determine and choose the best degree of parallelism for the respective measurement.

9.4.3 Performance results

In this section we discuss the results for the data scalability tests executed during the 50 TB benchmark. We executed the tests as described in 9.4.1, “Test methodology” on page 179 and 9.4.2, “Workload characterization” on page 180. User experiences may vary based on the actual environment in which the tests are executed.

Figure 9-7 and Figure 9-8 on page 181 display the results for the response time measurements (in minutes) and the CPU time measurements (in minutes) for query #1 against the 16-way configurations to retrieve data of sizes 1 TB, 5 TB, 10 TB, and 50 TB.

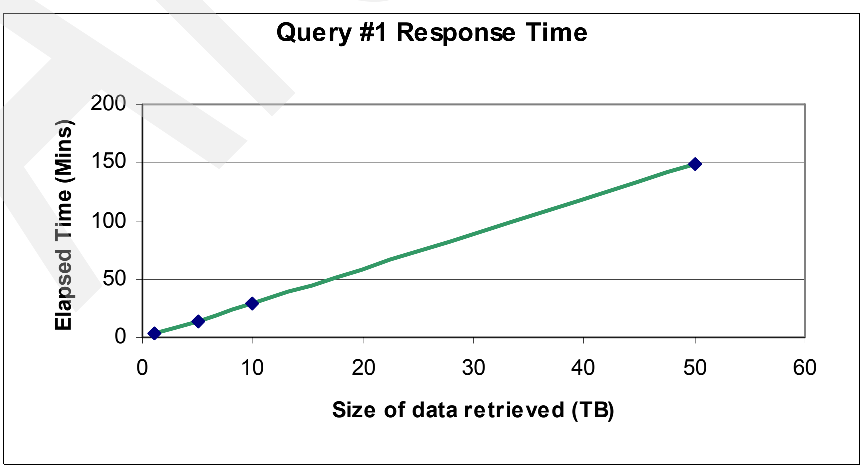


Figure 9-7 Query #1 response time for data scalability measurements

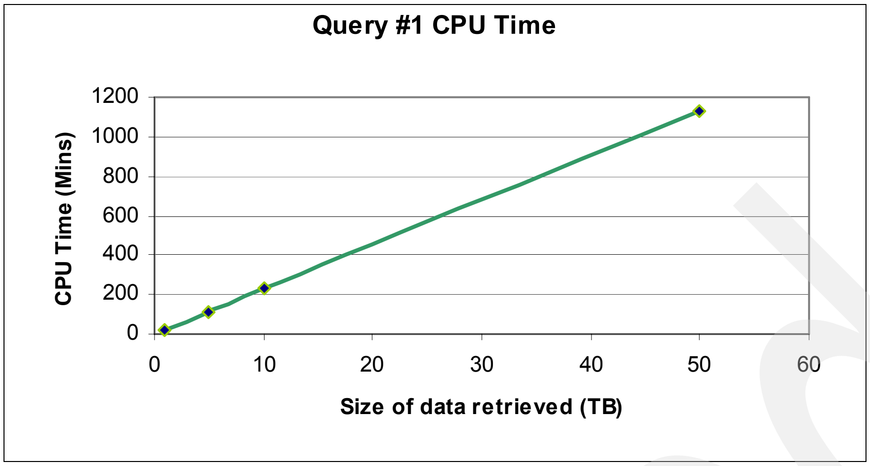


Figure 9-8 Query #1 CPU time for data scalability measurements

Figure 9-9 and Figure 9-9 display the results for the response time measurements (in minutes) and the CPU time measurements (in minutes) for query #2 against the 16-way configurations to retrieve data of sizes 1 TB, 5 TB, 10 TB, and 50 TB.

Figure 9-9 Query #2 response time for data scalability measurements

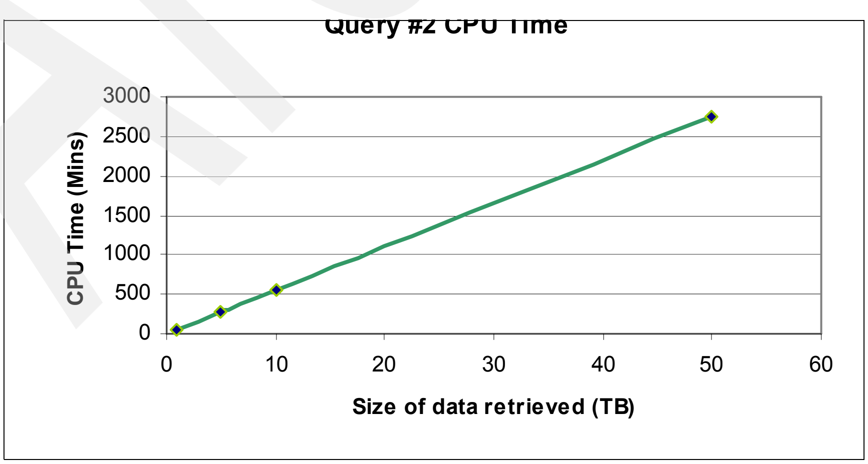


Figure 9-10 Query #2 CPU time for data scalability measurements

9.4.4 Observations

Impressive data scalability performance was seen for the single query measurements. For both queries used during the data scalability measurements, a near linear increase in response timeand CPU time was observed. There was a consistent and proportional increase in both the response time and CPU time with an increase in the amount of data that the same query retrieved. The CPU use was consistently high for all executions, ensuring accurate results across measurements.

Our measurements of up to 50 TB shows how well DB2 for z/OS scales for very large data warehouses. Enterprises need to review their data warehouses and their data needs, to determine how their data grows and what their processing needs are, as well as though the growing enterprise data is having an impact on their data processing and BI functions. Once it is determined that the growing data does indeed impact their data processing and BI requirements, these results can help them understand how DB2 for z/OS offers an excellent answer to their data scalability needs.

9.5 Conclusion

Rapid growth within each enterprise across the world is now a reality. Enterprises must look into their data warehousing capabilities and determine the right direction for their enterprise data warehouse. A long term view needs to be taken when making the important decision on the enterprise data warehouse. The ability to sustain growth and the ability to improve performance without an impact to existing processes or requiring major changes to the way business is executed is absolutely necessary.

Many factors need to be considered when determining the best solution for the enterprise, and scalability of their systems plays the most important role. Enterprises need to determine their growth path and have the right ratios available for databases of various sizes to determine how scalable their system really is.

The scalability results from the various processor and data scalability tests conducted during our benchmark answers important questions for our customers and enterprises across the world. DB2 for z/OS is the answer to the growing needs of enterprises. DB2 for z/OS is a leading enterprise data server, designed and tightly integrated with IBM System z to use the strengths of IBM System z.

We have seen impressive scalability results for DB2 for z/OS during our tests. The internal throughput ratios for a BI workload have shown excellent scalability and are much better than other workloads. Comparison of the BI workload ratios against the LSPR ratios for various configurations demonstrated the excellent scalability features of DB2 for z/OS BI workloads. BI customers can expect excellent throughput rates and response times using DB2 for z/OS, use fewer engines for their BI workload and thus reduce the cost of BI.

Customers should determine how data growth impacts them before committing to scale their systems. Depending on the type of workload and BI that is gathered, it may not be necessary to scale their systems. However, if it is necessary to have the ability to scale their systems as their data size grows, DB2 for z/OS is the answer to their growing data challenges.

10

Using materialized query tables

In this chapter we introduce materialized query tables (MQTs) and show how they can be used in the context of a very large database to reduce query response times for strategical and tactical analytical queries. MQTs, previously called summary tables, contain data that is typically aggregated from one or more source tables.

In the following sections, we explain the benefits of MQTs, how to define and build them, and how queries can benefit from MQTs. The MQTs defined in this book are used to demonstrate best practices and show performance figures achieved in our test environment.

10.1 Benefit of MQTs

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, 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. See Figure 10-1.

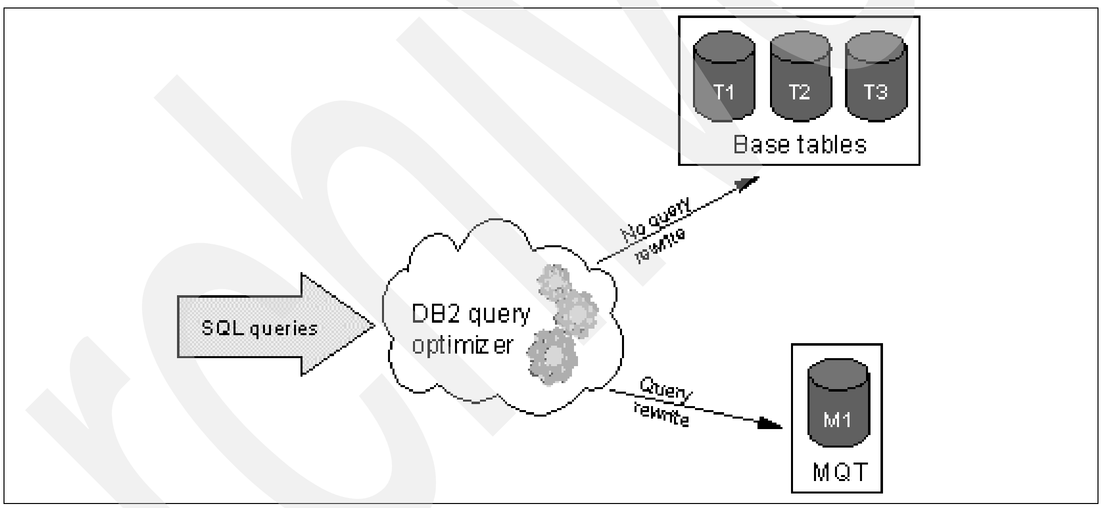


Figure 10-1 Query rewrite by the DB2 query optimizer

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 the query in Example 10-1 on page 185.

Example 10-1 Query for transactions between 2001 and 2008

```
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 increase. 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.

10.2 Anatomy of an MQT

This section examines the SQL syntax for the creation of the MQTs used in this study. Other sections discuss the syntax for MQT population, refresh, statistics gathering, index creation, and so on. See Figure 10-2 for an overview of the SQL syntax options.

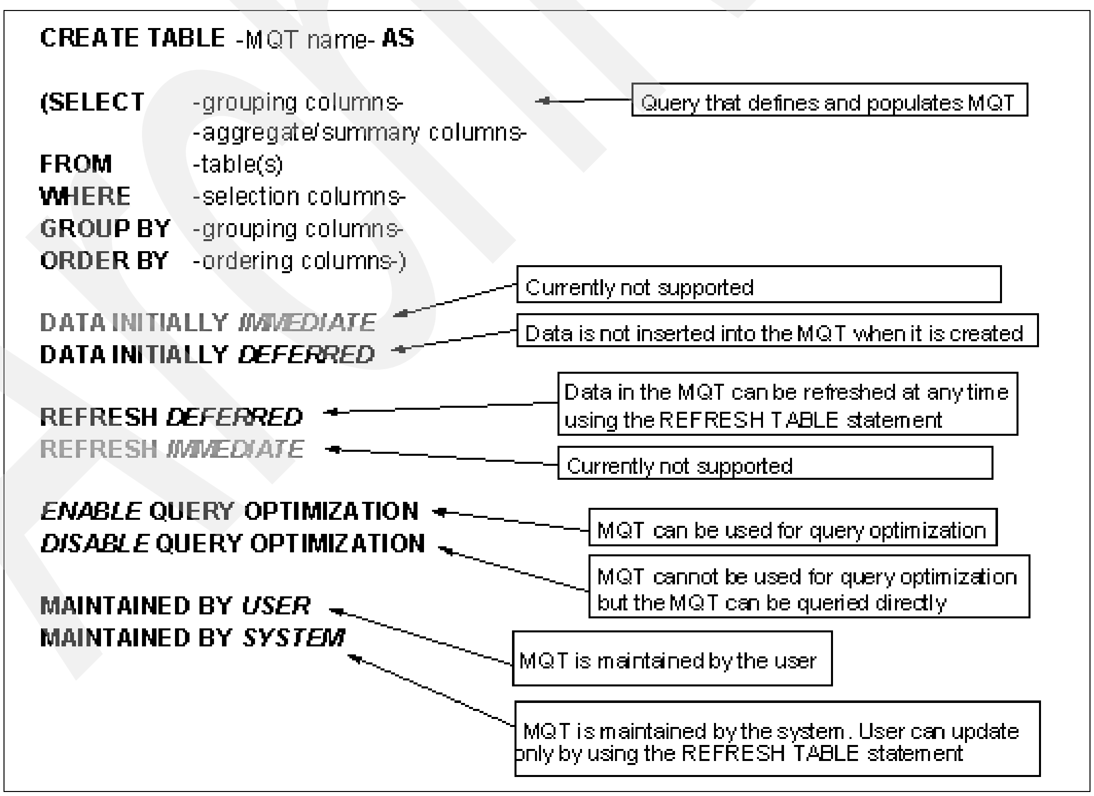


Figure 10-2 SQL syntax options for MQT creation

DATA INITIALLY DEFERRED clause

DB2 does not populate the MQT when you create the table. You must explicitly populate the MQT. For system-maintained MQTs, populate the tables for the first time by using the REFRESH TABLE statement. For user-maintained MQTs, populate the table by using the LOAD utility, INSERT statement, or REFRESH TABLE statement.

REFRESH DEFERRED clause

DB2 does not immediately update the data in the MQT when its base tables are updated. This delta represents the data latency. You can use the REFRESH TABLE statement at any time to update MQTs and maintain data currency with underlying base tables.

MAINTAINED BY SYSTEM clause

The MAINTAINED BY SYSTEM clause specifies that the MQT is a system-maintained MQT. You cannot update a system-maintained MQT by using the LOAD utility or the INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statements. You can update a system-maintained MQT only by using the REFRESH TABLE statement. BY SYSTEM is the default behavior if you do not specify a MAINTAINED BY clause.

MAINTAINED BY USER clause

The MAINTAINED BY USER clause specifies that the table is a user-maintained MQT. You can update a user-maintained MQT by using the LOAD utility, the INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statements, as well as the REFRESH TABLE statement.

ENABLE QUERY OPTIMIZATION clause

The ENABLE QUERY OPTIMIZATION specifies that DB2 can consider the MQT in automatic query rewrite. When you enable query optimization, DB2 is more restrictive of what you can select in the fullselect for a MQT.

DISABLE QUERY OPTIMIZATION clause

The DISABLE QUERY OPTIMIZATION clause specifies that DB2 cannot consider the MQT in automatic query rewrite, but still allows the MQT to be directly queried.

Note: When creating a user-maintained MQT, you should initially disable query optimization. Otherwise, DB2 might automatically rewrite queries to use the empty MQT. After you populate the user-maintained MQT, you can alter the table to enable query optimization.

10.3 MQT design

Creating both effective and efficient MQT definitions requires analyzing and understanding the underlying data, the response times of targeted queries, the query syntax, and the data model. Understanding these pieces helps answer questions:

- What queries should I target for MQT usage?
- Should my MQTs be specific to a query or reusable by many queries?
- How can column statistics help me design my MQTs?

Understanding data and workloads is often an iterative process. Therefore, MQT design is both proactive and reactive in nature. The data model-based approach is generally performed before you have detailed knowledge about the data. The workload-based approach is performed after gaining experience with the queries. It is important to keep in mind that implementing MQTs is not free. Besides the time and resources required to perform maintenance, the optimization time for a given query increases as the optimizer considers more and more MQTs. It is best to design a few MQTs that provide the widest coverage and the largest benefit. The following sections describe some best practices for developing an effective DB2 for z/OS MQT strategy in a VLDB environment.

10.3.1 Data considerations

Highly effective MQTs are those that can be employed across many different queries, often for grouping and aggregating large volumes of data along common dimensions. Moreover, MQTs provide the greatest benefit in queries that group and aggregate large volumes of data into a small number of groups. MQTs provide the most benefit when their underlying data sources are infrequently changing or when the MQT consumers are tolerant of data latency.

The effectiveness of an MQT can be determined by measuring the overall resource use savings that it provides. Because accessing large volumes of data can be resource intensive, MQTs should be designed to eliminate this I/O for the most frequently run and poorest performing queries, as shown in Figure 10-3. However, as the number and complexity of MQTs increase, so too does the maintenance cost (or data latency). Therefore, MQT definitions should also be generic enough that they can be applied across many queries.

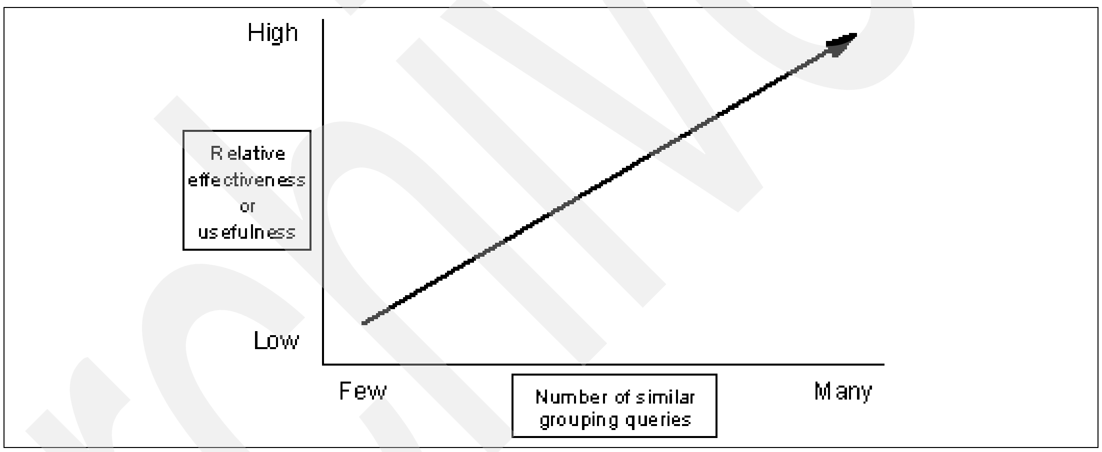


Figure 10-3 Increasing effectiveness with many similar grouping queries

Performing summarizations and aggregations over large volumes of data are often computationally expensive. MQTs should be designed to assist in low cardinality grouping/aggregation queries over large volumes of data. See Figure 10-4.

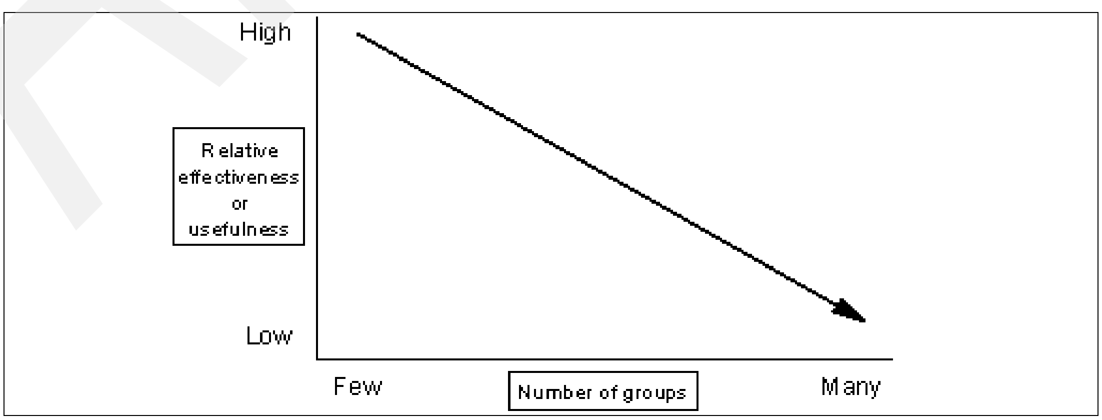


Figure 10-4 Decreasing effectiveness with many groups

The more often the MQT has to be refreshed, the less effective the MQT might be, as illustrated in Figure 10-5. Therefore, the tolerance of data latency between the base tables and the MQTs is an important consideration in designing an MQT strategy. If the MQTs require refreshing less often and there is an adequate window of time to perform the refreshes, more MQTs can be employed. For instance, MQTs are common in a data warehousing environment. One reason for this is that the data is updated much less frequently in a data warehousing environment than in an OLTP environment. Consequently, the MQT content is relevant for a longer period of time and the MQT can be refreshed less often.

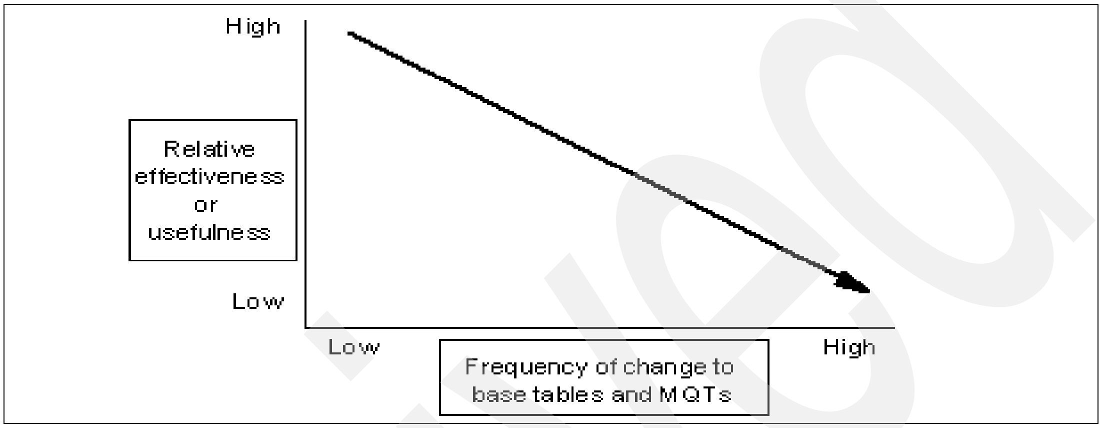


Figure 10-5 Decreasing effectiveness with many changes to the base tables and MQTs

10.3.2 Analyzing the application

Analyzing the data model and application reveals any requirements for grouping and aggregations, along with common and frequent join criteria. It is important to pay particular attention to implicit or explicit hierarchies represented in both the data model and the data itself. If there is a need for frequent aggregations along a hierarchy within a detailed table, MQTs can provide help. For example, a detailed transaction table has a time hierarchy of Year/Quarter/Month/Week/Day, and these periods are specified as grouping criteria on a periodic basis. One or more MQTs can be created to support the queries against the detailed data, especially if these reports run frequently.

The candidate SQL requests might be longer-running queries that use a lot of processing or I/O resources. By sorting and grouping the queries based on the tables accessed, as well as the aggregated columns and the grouping criteria, an MQT can improve performance.

Some application environments are good candidates for MQT creation and usage. BI and data warehousing (DW) environments lend themselves to the advantages of pre-summarized data. BI and DW applications normally store and query vast quantities of data. Decision-support queries might need to operate over terabytes of data, performing multiple joins and other complex SQL queries. BI and DW applications typically catalog and process data along hierarchies such as time and business subject areas. These hierarchies provide natural opportunities to create MQTs. Furthermore, BI and DW environments usually have clearly defined latency between the transaction data and the data warehouse data. For example, adding daily transactions to the data warehouse delivers a natural and consistent batch of data to the BI system on a periodic basis. Reviewing the hierarchies and the query requests within the BI environment yields a set of MQTs that can provide tremendous benefit and yet can also be maintained as part of the DW extract, transform, and load (ETL) process.

Star-schema or snowflake-schema data models are specific cases where MQTs can be employed. Traditionally, the fact table contains detailed facts, or measures, that are rolled up as sums, averages, and counts. The dimension tables contain the descriptive information and this information frequently defines a hierarchy. For example, the time dimension contains a time hierarchy (year/month/day), the product dimension contains a product hierarchy (category/product) and the location dimension contains a location hierarchy (country/region/territory). MQTs can be proactively defined to provide pre-aggregated data along the most commonly used levels.

In many cases, a pre-summarization process already exists and is in use. In these situations, the existing process can be left as is, or it can be modified to include the use of MQTs and the optimizer's query rewrite capability. By altering the existing summary tables to be MQTs, and modifying the queries to access the base tables, the optimizer can be relied upon to make the decision whether or not to use the base tables or the MQTs. This decision is based on the estimated runtime cost for each set of data queried. Using the query optimizer to make this decision allows more flexibility.

10.3.3 Designing MQT definitions

With any identified hierarchies or grouping criteria, laying out the MQTs can be straightforward. Using the previous example of a time hierarchy, assume 100 million rows in a detailed transaction table that represents three years of evenly distributed data, including the following distinct levels or groups:

- 3 years
- 12 quarters (3 years × 4 quarters)
- 36 months (3 years × 12 months)
- 156 weeks (3 years × 52 weeks)
- 1095 days (3 years × 365 days)

Executing a query that groups all 100 million rows by the most detailed level (day), results in 1095 distinct groups. Yet, continually processing all 100 million rows is time-consuming and resource-intensive. This is where designing an MQT is valuable.

By providing an MQT that represents all the data grouped by Year/Quarter/Month/Week/Day, the query optimizer can rewrite the query to use the MQT instead of the base transaction table. Rather than reading and processing 100 million rows, the database engine reads and processes only 1095 rows from the MQT, resulting in a significant boost in performance.

It is tempting to create MQTs for the other levels of this time hierarchy (for example, Year/Quarter), but in this case, there is little benefit to be gained. The query optimizer is able to use the MQT with Year/Quarter/Month/Week/Day and regroup the MQT rows to build aggregates for Year/Quarter. The query does not need to match the precompiled results exactly. Reading and processing 1095 rows to build 12 distinct groups is significantly faster than reading and processing all 100 millions rows of the transaction table. Yet accessing an MQT with 12 rows based on Year/Quarter is not that much faster than accessing an MQT with 1095 rows. In other words, the largest benefit is derived from pre-aggregating 100 million transactions down to 1095 groups.

If there is a requirement to maintain relatively static figures for each level of the hierarchy, MQTs can be created for each level. This is one way to take advantage of the data latency inherent in MQTs. In this case, building the lowest level first and using that level to build and maintain the next MQT is the preferred approach. This avoids reading and processing the detailed transactions, minimizing the time and resources required to build the next level of groups. Using the example of a time hierarchy, it is advantageous to create the most detailed level first (Year/Quarter/Month/Week/Day), and, at the appropriate time, use this table to create a higher level (for example, Year/Quarter/Month) at the appropriate time. This approach minimizes the time and effort to build or refresh the various levels of MQTs.

Another MQT benefit is the ability to persist join results to minimize or avoid the joining of rows. Because joining many rows together can result in high physical I/O operations and potentially long response times, full or partial denormalization of the data model can significantly increase query performance.

MQTs can be created from one base table or many base tables. When creating MQTs over many base tables, the MQT is used to denormalize the base tables. This denormalization of data minimizes or eliminates the need to join rows during query execution

MQTs can be created with local selection against one or more tables. In this case, the MQT is considered to be sparse and only reflects the data represented by the specified local selection. Because the MQT only contains some of the data from the base tables, its overall usefulness might be decreased.

10.4 MQT matching in DB2 for z/OS

This section discusses how MQT matching takes place in DB2 for z/OS. We also present some best practices for designing MQTs and for adjusting environmental settings to make MQT Automatic Query Rewrites (AQRs) functionally possible. Ultimately, the query optimizer will take into account environmental settings, the cost of using the MQT, MQT syntax, query syntax, table constraints, and other relevant information in determining whether to perform an AQR.

10.4.1 Performance considerations

The MQT matching algorithm of DB2 for z/OS will only attempt to perform a query rewrite for dynamic read-only queries. From a performance standpoint, the optimizer will compare the cost of running the original query to the cost of running the rewritten query. Rewrites are not considered for queries that are estimated to be short-running.

10.4.2 Definitional considerations

This section describes some aspects to consider when defining an MQT and ensuring proper usage by DB2 for z/OS.

ENABLE QUERY OPTIMIZATION

For an MQT to be considered for query rewrite, it must be defined as ENABLE QUERY OPTIMIZATION. An MQT that specifies DISABLE QUERY OPTIMIZATION can be queried directly but cannot be used in query rewrites. The ALTER TABLE clause can be used to enable or disable query optimization for an MQT.

CURRENT MAINTAINED TABLE TYPES

The CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register specifies a value that identifies the types of tables that can be considered when optimizing the processing of dynamic SQL queries and affects dynamic statement cache matching. The initial value is SYSTEM. Its value can be changed by the SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION statement.

- ALL: Indicates that all MQTs will be considered
- NONE: Indicates that no MQTs will be considered
- SYSTEM: Indicates that only system-maintained MQTs that are refresh deferred will be considered
- USER: Indicates that only user-maintained MQTs that are refresh deferred will be considered

CURRENT REFRESH AGE

CURRENT REFRESH AGE specifies a timestamp duration value with a data type of DECIMAL(20,6). This duration is the maximum duration because a REFRESH TABLE statement has been processed on a system-maintained REFRESH DEFERRED MQT such that the MQT can be used to optimize the processing of a query. This special register affects dynamic statement cache matching.

If CURRENT REFRESH AGE has a value of 99999999999999 (ANY), REFRESH DEFERRED MQTs are considered to optimize the processing of a dynamic SQL query. This value represents 9999 years, 99 months, 99 days, 99 hours, and 99 seconds.

The default for the initial value of CURRENT REFRESH AGE is 0 unless your installation has changed it to ANY by modifying the value of that field. You can change the value of the register by executing the SET CURRENT REFRESH AGE statement.

MQT indexing strategy

In general, it is important to create similar indexes on an MQT to those you would create on a base table. On base tables, defining unique indexes when columns are truly unique may help the optimizer perform additional rejoins or residual joins. Defining indexes over the MQT grouping columns and aggregation columns for MQTs with a single base table, or local selection and join columns for MQTs with multiple base tables can reduce refresh time significantly. Feedback mechanisms such as an index advisor can be used to aid in this effort.

Isolation level

An MQT will only be considered for query optimization if its isolation level is higher than or equivalent to the isolation level of the query with which the optimizer is attempting to match it. The isolation level of the MQT is based on the isolation level when the MQT was created or altered from the base table(s). To get an idea of when an MQT is not used, take a close look at the access plan and compare it with the catalog attributes of the MQT.

10.4.3 Table considerations

The following section describes some DB2 for z/OS syntactical considerations for creating effective MQTs. In general, the MQT definition should be broad enough that it can be used in multiple matching scenarios. Including restrictive clauses and column definitions can severely restrict the matching ability of the query optimizer. Additionally, not specifying enough information in the MQT definition can inhibit MQT matching. For instance, the GROUP BY clause in an MQT definition should include enough columns such that its result set can be regrouped to match the broader set of candidate queries. Rather than grouping by DAY, it is generally more effective to group by the natural time hierarchy of YEAR, QUARTER, MONTH, DAY, to allow the optimizer to match the MQT to a broader set of queries. Another best practice is to provide DB2 for z/OS with as much known constraint information as possible. By defining unique constraints and referential integrity constraints (enforced or not-enforced) whenever possible, DB2 for z/OS has an increased ability to match an MQT to a query in cases where required information is missing from the MQT.

Base table considerations

Referential integrity constraints should be specified whenever possible. Data warehousing environments tend to be predominantly read-only and have their data loaded through a rigorous ETL process. They also tend to be significantly denormalized. Therefore, enforced referential constraints may not be appropriate for such environments. However, MQT matching can be aided in such cases through the definition of informational constraints. Informational constraints are essentially hints to the DB2 optimizer to help it determine things like join fanout conditions. Informational constraints are implemented as check constraints or referential constraints, and are defined in the CREATE/ALTER TABLE statement with the NOT ENFORCED option.

While informational constraints are not enforced by the database manager during updates to a table, they can be used by the DB2 optimizer for potential query rewrites. For instance, if an MQT references a table not referenced in a given query, then the unreferenced table is examined to determine if it is a parent table in a referential integrity constraint. If the foreign key is non-nullable and the two tables are joined using a primary key to foreign key equal predicate, then the MQT can still be potentially used. Moreover, including foreign keys in the grouping clause of an MQT may allow the optimizer to perform lossless joins or lossless rejoins to tables to obtain additional required data from a parent table.

Referential constraints between base tables are also an important factor in determining whether a MQT can be used for a query. For instance, in a data warehouse environment, data is usually extracted from other sources, transformed, and loaded into data warehouse tables. In such an environment, the referential integrity constraints can be maintained and enforced by other means than the database manager to avoid the overhead of enforcing them by DB2. However, referential constraints between base tables in MQT definitions are important in a query rewrite to determine whether or not a MQT can be used in answering a query. In such cases, you can use informational referential constraints to declare a referential constraint to be true to allow DB2 to take advantage of the referential constraints in the query rewrite.

DB2 allows the user application to enforce informational referential constraints, while it ignores the informational referential constraints for inserting, updating, deleting, and using the LOAD and CHECK DATA utilities. Thus, the overhead of DB2 enforcement of referential integrity is avoided and more queries can qualify for automatic query rewrite using MQTs. So an RI constraint can be defined as NOT ENFORCED and can also be ENABLED for QUERY OPTIMIZATION.

Defining unique indexes can allow the optimizer to determine if joins are lossless. This can allow the optimizer to eliminate unneeded information from an MQT or to perform additional lossless joins to gather additional required information.

Base table columns should be defined as NOT NULL whenever possible. When a column allows NULL values (nullable), aggregations such as COUNT(column_name) and COUNT(*) will behave differently because the former does not count NULL values.

MQT considerations

Include aggregate functions strategically in the fullselect of a MQT definition:

- Include COUNT(*) and SUM(expression).
- Include SUM(expression*expression) only if you plan to query VAR(expression), STDDEV(expression), VAR_SAMP(expression), or STDDEV_SAMP(expression).
- Include COUNT(expression) in addition to COUNT(*) if expression is nullable.
- Include MIN(expression) and MAX(expression) if you plan to query them.
- Do not include AVG(expression), VAR(expression), or STDDEV(expression) directly if you include either of the following parameter combinations:
- SUM(expression), SUM(expression*expression), and COUNT(*)

- SUM(expression), SUM(expression*expression), and COUNT(expression)
- DB2 can derive AVG(expression), VAR(expression), and STDDEV(expression) from SUM(expression), SUM(expression*expression), and the appropriate COUNT aggregate function.

Include commonly used, non-nullable foreign keys in the GROUP BY and SELECT list of a MQT definition to give the optimizer the ability to perform lossless rejoins to parent tables to gather additional required information.

Include all the higher-level columns of a hierarchy in the MQT. For example, if you include CITY in a GROUP BY clause, also include STATE and COUNTRY in the clause. Similarly, if you include MONTH in the GROUP BY clause, also include YEAR in the clause.

Do not use the HAVING clause in your MQTs. A MQT with a HAVING clause in its definition has limited usability during query rewrite.

Avoid using local selection predicates that limit the effectiveness of the MQT matching algorithm. If local selection predicates must be used, ensure that those predicates can be subsumed by predicates in the matching queries. For instance, a query predicate such as MONTH in ('JAN', 'FEB') will subsume the MQT predicate MONTH in ('JAN', 'FEB', 'MAR') and still allow matching. A query predicate such as AGE > 21 will subsume the MQT predicate AGE > 17 and still allow matching.

10.4.4 MQT matching debug

Use the following checklist to help determine why an MQT was not used:

- Check if CURRENT REFRESH AGE = ANY.
- Check if CURRENT MAINTAINED TABLE TYPES = ALL | SYSTEM | USER.
- Check if the query is read-only.
- Check if MQT is enabled for query optimization.
- Check if MQT has been refreshed at least once if it is system-maintained.
- Check the isolation level of the MQT and the query.
- Check if the query is short-running.
- Check if MQT is available.
- Check if a more effective MQT was used.
- Check all matching conditions described in 10.4, “MQT matching in DB2 for z/OS” on page 190.

10.5 MQT setup

This section identifies best practices for ensuring proper MQT usage and performance.

10.5.1 MQT partitioning

A partitioned table space is used to store a single table. DB2 divides the table space into partitions. The partitions are based on the boundary values defined for specific columns. Utilities and SQL statements can run concurrently on each partition.

In a partitioned table space, you can think of each partition as a unit of storage. For each partition that you specify in the CREATE TABLESPACE statement, DB2 creates a separate data set. You assign the number of partitions (from 1 to 4096), and you can assign partitions independently to different storage groups.

The maximum number of partitions in a table space depends on the data set size (DSSIZE parameter) and the page size. The size of the tables space depends on the data set size and on how many partitions are in the table space.

Partitioned table spaces share the following characteristics:

- You can use separate jobs for mass update, delete, or insert operations instead of using one large job; each smaller job can work on a different partition. Separating the large job into several smaller jobs that run concurrently can reduce the elapsed time for the whole task. Common partitioning schemes for MQTs in a BI environment include partitioning on DATE, GEOGRAPHY (Country, Region, Zip Code), KEY VALUE, and so forth.
- You can put frequently accessed data on faster devices. Evaluate whether table partitioning or index partitioning can separate more frequently accessed data from the remainder of the table. You can put the frequently accessed data in a partition of its own. You can also use a different device type.
- You can rebalance data among the partitions by redefining partition boundaries with no impact to availability. You can also add a partition to the table and to each partitioned index on the table; the new partition becomes available immediately.
- You can spread a large table over several DB2 storage groups or data sets. All the partitions of the table do not need to use the same storage group.
- Partitioned table spaces let a utility job work on part of the data while allowing other applications to concurrently access data on other partitions. In that way, several concurrent utility jobs can, for example, load all partitions of a table space concurrently. Because you can work on part of your data, some of your operations on the data might require less time.
- You can take advantage of parallelism for certain read-only queries. When DB2 determines that processing is likely to be extensive, it can begin parallel processing of more than one partition at a time. Parallel processing (for read-only queries) is most efficient when you spread the partitions over different disk volumes and allow each I/O stream to operate on a separate channel.
- You can take advantage of query parallelism. Use the Parallel Sysplex data sharing technology to process a single read-only query across many DB2 subsystems in a data sharing group. You can optimize Parallel Sysplex query processing by placing each DB2 subsystem on a separate central processor complex.
- DB2 opens more data sets when you access data in a partitioned table space than when you access data in other types of table spaces.
- You can enable partitioned table spaces for extended addressability (EA), a function of DFSMS. The term for table spaces and index spaces that are enabled for extended addressability is EA-enabled. You must use EA-enabled table spaces or index spaces if you specify a maximum partition size (DSSIZE) that is larger than 4 GB in the CREATE TABLESPACE statement.
- Both EA-enabled and non-EA-enabled partitioned table spaces can have only one table and up to 4096 partitions.

10.5.2 Loading and refreshing MQTs

In order for a system-maintained MQT to be used by DB2 for z/OS, the user must trigger a populate of the MQT. This is either done using a REFRESH TABLE statement or by creating an MQT by altering an existing table. Refreshing a system maintained MQT will cause all of the data in the MQT to be deleted and cause the MQT to be re-populated. Moreover, various bookkeeping activities will take place, including updating the REFRESH_TIME and CARD fields in the catalog for the MQT. These processes all take place with logging turned on. Additionally, any dynamic cache entry that references the refreshed MQT or references a dependant table within the MQT will be invalidated. By issuing a REFRESH TABLE statement, the MQT will be locked until the refresh is complete. During this time, the MQT is not eligible for use by the optimizer in query rewrites. Conversely, a user-maintained MQT can be used by DB2 for z/OS upon creation.

For these reasons, it is advised that a user-maintained MQTs query optimization setting should be disabled while it is populated. Once populated, the table can be altered to allow query optimization. For system-maintained MQTs, MQT refresh should be done at times with low query volumes. Currently, DB2 for z/OS does not support incremental refresh of MQTs using the SET INTEGRITY statement. However, user-driven incremental refresh can be achieved by issuing INSERTS/UPDATES/DELETES against a user-maintained MQT or by employing triggers on the base tables.

The alternative to refreshing system maintained MQTs in DB2 for z/OS with the REFRESH statement is to use DB2 for z/OS utilities to maintain the MQT. This can be achieved by first altering the table to disable query optimization. Using the Load/Replace utility will allow the MQT to be loaded without logging turned on. Any dependant indexes should be rebuilt or dropped/recreated. The table can then be altered to enable query optimization, and the RUNSTATS utility should be run to refresh statistics against the newly populated data. When you run the REFRESH TABLE statement, the only statistic that DB2 updates is the cardinality statistic. To ensure that the catalog statistics for the MQT are up to date, the RUNSTATS utility should be run any time the MQT data has changed significantly.

10.5.3 Considerations for very large databases

While MQTs could become crucial to get reasonable response times for certain complex BI queries against very large databases, building them could become a challenge due to the potentially required resource demands.

In 10.7, “MQTs used in this book” on page 201, we show various options to build MQTs, including one example where we use a system-maintained MQT and just leave it to DB2 to load data from the referenced table and load it into the MQT.

In general, however, you might encounter a situation where large databases require more planning to consider the various ways to build MQTs. Whenever resulting demands for sort space and work files exceed what your system setup can handle, you should consider user maintained MQTs and load the data explicitly into the MQT. This gives you the option to reduce and control the amount of data handled in one step. In many cases, loading an MQT in a set of steps and defining the steps by a time value does not only result in less resource consumption but also gives you the option to incrementally update an existing MQT with more recent data. We demonstrate this approach in 10.7.1, “MQT for line item revenue” on page 201.

If you go for user-maintained MQTs, Example 10-2 on page 196 shows an excerpt of a job where we use the LOAD utility to efficiently load data from a cursor that defines a subset of data (based on the first half of the year 1992). The LOAD utility can directly load the output of a dynamic SQL SELECT statement into a table. The dynamic SQL statement can be executed on data at a local server or at any DRDA-compliant remote server. This functionality is called the DB2 family cross-loader function.

Example 10-2 Job to LOAD first half of 1992 data with the cross-loader function

```
//LD92H1   EXEC PGM=DSNUTILB,
...
//SYSIN    DD  *
EXEC SQL
  DECLARE 1992H1 CURSOR FOR
  SELECT L_SHIPDATE, L_DISCOUNT, L_QUANTITY,
  SUM(L_EXTENDEDPRICE * L_DISCOUNT) AS REVENUE
  FROM TPCH50TB.LINEITEM
  WHERE L_SHIPDATE BETWEEN '1992-01-01' AND '1992-06-30'
  GROUP BY L_SHIPDATE, L_DISCOUNT, L_QUANTITY
  ORDER BY L_SHIPDATE, L_DISCOUNT, L_QUANTITY
ENDEXEC
LOAD DATA
  INTO TABLE TPCH50TB.MQTLINE PART 1 REPLACE INCURSOR(1992H1)
  LOG NO NOCOPYPEND
```

10.5.4 zParm settings impacting MQTs

The following section describes zParm settings, which can affect the ability of DB2 for z/OS to perform MQT rewrites.

Special register CURRENT REFRESH AGE

CURRENT REFRESH AGE specifies a timestamp duration value with a data type of DECIMAL(20,6). This duration is the maximum duration because a REFRESH TABLE statement has been processed on a system-maintained REFRESH DEFERRED MQT such that the MQT can be used to optimize the processing of a query. This special register affects dynamic statement cache matching. If CURRENT REFRESH AGE has a value of 99999999999999 (ANY), REFRESH DEFERRED MQTs are considered to optimize the processing of a dynamic SQL query. This value represents 9999 years, 99 months, 99 days, 99 hours, and 99 seconds. The default for the initial value of CURRENT REFRESH AGE is 0 unless your installation has changed it to ANY by modifying the value of that field. You can change the value of the register by executing the SET CURRENT REFRESH AGE statement.

Special register CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

This special register specifies a VARCHAR(255) value. The value identifies the types of MQT that can be considered in query rewrite. Specify the default value for the CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register, when no value is explicitly set, by using the SQL statement SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION. Accepting the default value allows query rewrite using system-maintained MQTs (SYSTEM) when CURRENT REFRESH AGE is set to ANY. Alternatively, specifying USER allows query rewrite by using user-maintained MQTs when CURRENT REFRESH AGE is set to ANY. Specifying ALL allows query rewrite by using both system-maintained and user-maintained MQTs.

10.5.5 Sort and work file considerations (DSNDB07)

A sort operation is invoked when a cursor is opened for a SELECT statement that requires sorting. The maximum size of the sort work area allocated for each concurrent sort user depends on the value that you specified for the SORT POOL SIZE field on installation panel

DSNTIPC. The default value is 2 MB. The work files that are used in sort are logical work files, which reside in work file table spaces in your work file database (which is DSNDB07 in a non data-sharing environment).

The sort begins with the input phase, when ordered sets of rows are written to work files. At the end of the input phase, when all the rows have been sorted and inserted into the work files, the work files are merged together, if necessary, into a single work file that contains the sorted data. The merge phase is skipped if only one work file exists at the end of the input phase. In some cases, intermediate merging might be needed if the maximum number of sort work files has been allocated.

DB2 uses the buffer pool when writing to the logical work file. Only the buffer pool size limits the number of work files that can be used for sorting. A sort can complete in the buffer pool without I/Os. This ideal situation might be unlikely, especially if the amount of data being sorted is large. The sort row size is actually made up of the columns being sorted (the sort key length) and the columns that the user selects (the sort data length). Having a large buffer pool for sort activity can help you to avoid disk I/Os.

Improving the performance of sort processing

Many factors affect the performance of sort operations, but you can take measures to reduce I/O contention and minimize sort row size. For any SQL statement that initiates sort activity, the OMEGAMON SQL activity reports provide information about the efficiency of the sort that is involved. Setting the sort option to YES enables IFCID 95 and 96 to see sort information at the SQL level.

To minimize the performance impacts of sort/merge processing, consider the following approaches:

- Increase the size of the sort pool. The larger the sort pool, the more efficient the sort is.
- Allocate additional physical work files in excess of the defaults, and put those work files in their own buffer pool. Segregating work file activity enables you to better monitor and tune sort performance. It allows DB2 to handle sorts more efficiently because these buffers are available only for sort without interference from other DB2 work. For a given query, the recommended number of work file disk volumes to have is one-fifth the maximum number of data partitions, with five as a minimum and 50 as a maximum. For concurrently running queries, multiply this number by the number of concurrent queries.
- Increase the amount of available space for work files. To estimate sort data size and required workfile data set size, divide workfile Getpage as follows:
  - 2 if the number of merge passes = 0
  - 4 if the number of merge passes = 1 and then multiply by 4 KB

  Example: 100,000 row sort with 4000 workfile Getpage
  Sort data size = 4000*4KB/4 = 4MB
  DASD space requirement estimate = 4MB*1.3
- Minimize I/O contention on the I/O paths to the physical work files, and make sure that physical work files are allocated on different I/O paths and packs to minimize I/O contention. Using disk devices with Parallel Access Volumes (PAV) support is another way to significantly minimize I/O contention.
- Set the buffer pool sequential steal threshold (VPSEQT) to 100% unless a sparse index is used to access the work files. The default value, which is 80%, allows 20% of the buffers to go unused. A value of 99% prevents space map pages, which are randomly accessed, from being overwritten by massive prefetch.
- Increase the buffer pool deferred write threshold (DWQT) or data set deferred write threshold (VDWQT) values. If the DWQT or VDWQT are reached, writes are scheduled. For a large sort using many logical work files, this is difficult to avoid, even if a large buffer pool is specified.

10.6 Feedback mechanisms

This section describes the feedback mechanisms that can be used to validate MQT usage and to understand why an MQT was (or was not) used by DB2 for z/OS.

10.6.1 Validate query rewrite with Optimization Expert

To validate that a given query is rewritten and now uses an MQT, we use DB2 Optimization Expert. The same could be achieved by explaining the statement directly in DB2 for z/OS.

DB2 Optimization Expert for z/OS, V1.2 supports monitoring and tuning of SQL statements that run as a single statement or as part of a workload on your DB2 for z/OS subsystem, V8 and DB2 9. In addition to EXPLAIN options, DB2 Optimization Expert provides a graphical rendering of query access plans, advanced statistics, query and index advisors with “what if” and table-based priority options. Both experienced DBAs and application programmers will find DB2 Optimization Expert V1.2 easy to use and customize with powerful filters, views, and search options.

For this book, we created a new project in Optimization Expert called VLDB50-Query and used the application environment options in the project’s context tab (Figure 10-6) to set the relevant parameters for MQT rewrite:

- Current refresh age = **ANY**
- Current materialized table types = **ALL**

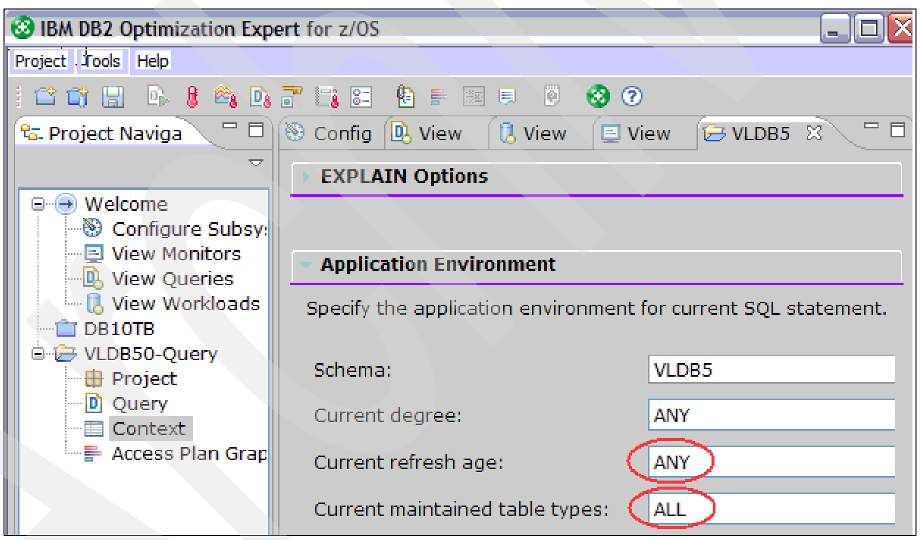


Figure 10-6 Context setting in Optimization Expert for z/OS

Next, we specified the SQL statement of the query we want to validate (Figure 10-7).

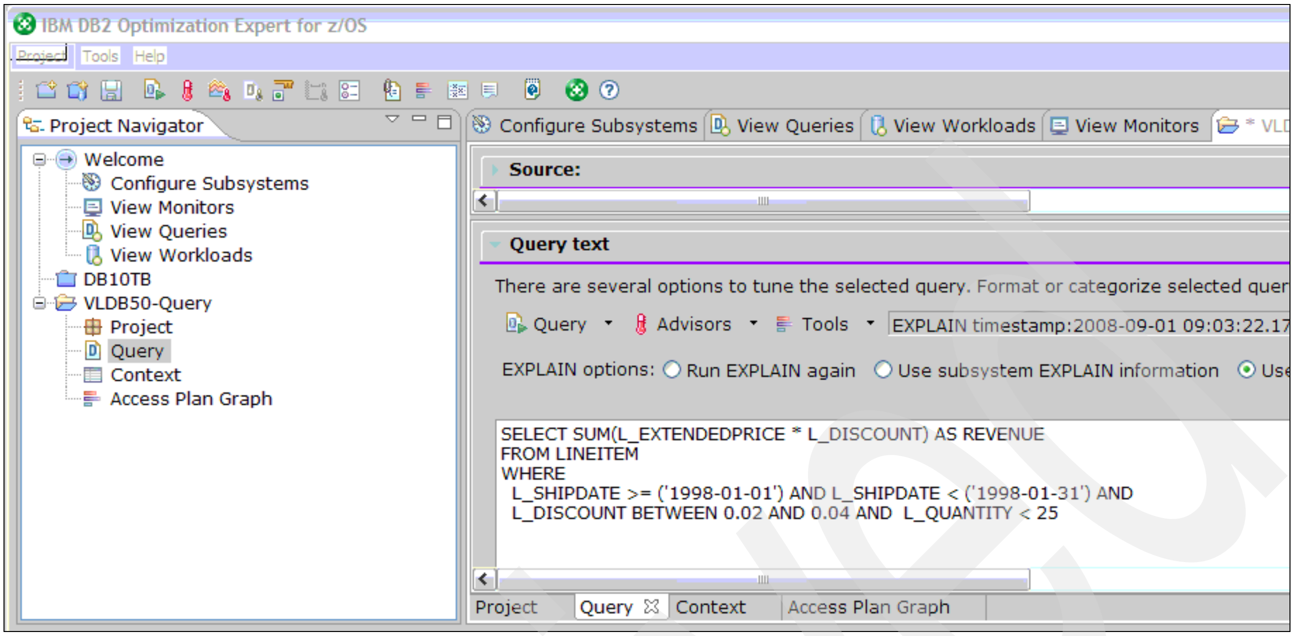


Figure 10-7 Query text in Optimization Expert for z/OS

Select **Tools** → **Access Plan Graph** to run explain and create the access plan graph. Figure 10-8 shows a graph that includes a query rewrite using an MQT (MQTLINE).

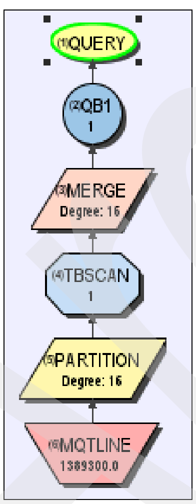


Figure 10-8 Example of a visual access graph with query rewrite

Optimization Service Center provides less powerful capabilities compared to Optimization Expert, but can be used for access plan graph creation in a similar way. It also works with DB2 V8 if the prereqs are installed. Visit the following Web page to download the Optimization Service Center:

http://www.ibm.com/software/data/db2imstools/db2tools/opti-expert-zos/

For more details on Optimization Expert for z/OS and Optimization Service Center, see the following publications:

- *IBM DB2 9 for z/OS: New Tools for Query Optimization*, SG24-7421
- *News on DB2 Optimization Expert for z/OS V1.2 and DB2 Optimization Service Center Version 1.1*, TIPS0673

10.6.2 Catalog information

Type M for MQTs is stored in the following DB2 tables:

- SYSIBM.SYSTABLES
- SYSIBM.SYSVIEWS
- SYSIBM.SYSVIEWDEP
- SYSIBM.SYSPLANDEP
- SYSIBM.SYSPACKDEP
- SYSIBM.SYSVTREE
- SYSIBM.SYSVLTREE

Both SYSIBM.SYSTABLES and SYSIBM.SYSROUTINES include the column NUM_DEP_MQTS, which contains the information about how many MQTs are dependent on a table or a table UDF respectively. Table SYSIBM.SYSVIEWS has six new columns that contain information related to MQTs:

- REFRESH

  D for deferred refresh mode; or blank, which means the row does not belong to an MQT.
- ENABLE

  Y or N for QUERY OPTIMIZATION enablement, or blank for a view.
- MAINTENANCE

  S for system-maintained, U for user-maintained or blank for view.
- REFRESH_TIME

  only used by system-maintained MQTs. It indicates the time stamp of last REFRESH TABLE statement.
- ISOLATION

  Isolation level when MQT is created or altered from a base table.
- SIGNATURE

  Contains an internal description of the MQT.

The ENFORCED column in the catalog table SYSIBM.SYSRELS describes whether the referential integrity constraint entry belongs to an informational RI constraint or an enforced RI constraint.

10.7 MQTs used in this book

We build three MQTs with different characteristics and requirements with respect to definition, building, and loading data. See Table 10-1.

Table 10-1 MQTs used in this book

MQT name	Rows	Comment
MQTLINE	1,389,300 rows	Illustrates incremental building of an MQT for a large base table
MQTTOP10	45,640 (1,467) rows	Shows how to use the rank() function with DB2 9 for z/OS (smaller number is data for one day)
MQTSUPP	108,191,247 rows	Demonstrates a system managed MQT and loading through the **refresh table** command.

> **Note:** In some cases, we limited sort and work file requirements as well as build times by restricting data to a certain time frame. Be aware that you might have to take different approaches and get to different build times in your environment.

10.7.1 MQT for line item revenue

We want to improve the response time of a query, such as in Example 10-3, where the revenue over a year for a certain range of discount and quality values is calculated. This could be a BI query to support strategic decision-making. It may be requested with different values of ship date, discount, and quantity.

Example 10-3 Query to get revenues per line item

```
SELECT SUM(L_EXTENDEDPRICE * L_DISCOUNT) AS REVENUE
FROM LINEITEM
WHERE
  L_SHIPDATE >= (‘1994-01-01’) AND L_SHIPDATE < (‘1994-01-01’) + 1 YEAR AND
  L_DISCOUNT BETWEEN 0.03 - 0.01 AND 0.03 + 0.01 AND
  L_QUANTITY < 25
```

We decide to build an MQT named MQTLINE to support this query. The MQT in Example 10-4 on page 202 uses table partitioning and is partitioned by year (seven partitions total for the years 1992 to 1998 respectively) using L_SHIPDATE as a partitioning key. We used option MAINTAINED BY USER to load data into the table on our own. We set DISABLE QUERY OPTIMIZATION to avoid undesired query rewrites while the table is still empty.

Example 10-4 MQT definition for line item revenue (MQTLINE)

```
CREATE TABLE MQTLINE AS (
   SELECT L_SHIPDATE, L_DISCOUNT, L_QUANTITY,
   SUM(DECIMAL(L_EXTENDEDPRICE,31,2) * L_DISCOUNT) AS REVENUE
   FROM LINEITEM
   GROUP BY L_SHIPDATE, L_DISCOUNT, L_QUANTITY)
 DATA INITIALLY DEFERRED
 REFRESH DEFERRED
 MAINTAINED BY USER
 DISABLE QUERY OPTIMIZATION
 IN VLDB50.TSMQT1
 PARTITION BY RANGE(L_SHIPDATE)
   (PARTITION 0001 ENDING AT ('1992-12-31')
   ,PARTITION 0002 ENDING AT ('1993-12-31')
   ,PARTITION 0003 ENDING AT ('1994-12-31')
   ,PARTITION 0004 ENDING AT ('1995-12-31')
   ,PARTITION 0005 ENDING AT ('1996-12-31')
   ,PARTITION 0006 ENDING AT ('1997-12-31')
   ,PARTITION 0007 ENDING AT (MAXVALUE));
```

To load data into the MQT, we use an iterative time-based approach. The assumption is that new line items added to the LINEITEM table come with a newer ship date. Therefore, computation of already aggregated older line items does not have to be repeated. It is sufficient to just compute the delta of the most recent changes.

Figure 10-9 illustrates this approach. Base table LINEITEM has 4000 partitions in our environment and is partitioned by order number. For a certain time interval (we use two months), we create ten interim tables (MQTLINEA, MQTLINEB,... MQTLINEJ) with aggregates from LINEITEM and restrictions to 400 partitions each. Table layout for the interim tables is identical to the final MQTLINE table. Restricting the interim tables to both partitions and time frame makes them smaller, faster to build and easier to handle.

The ten interim tables are combined and the total aggregates are added to MQTLINE. Depending on your sort and work file limitations, this approach can consider more or fewer interim tables or time intervals.

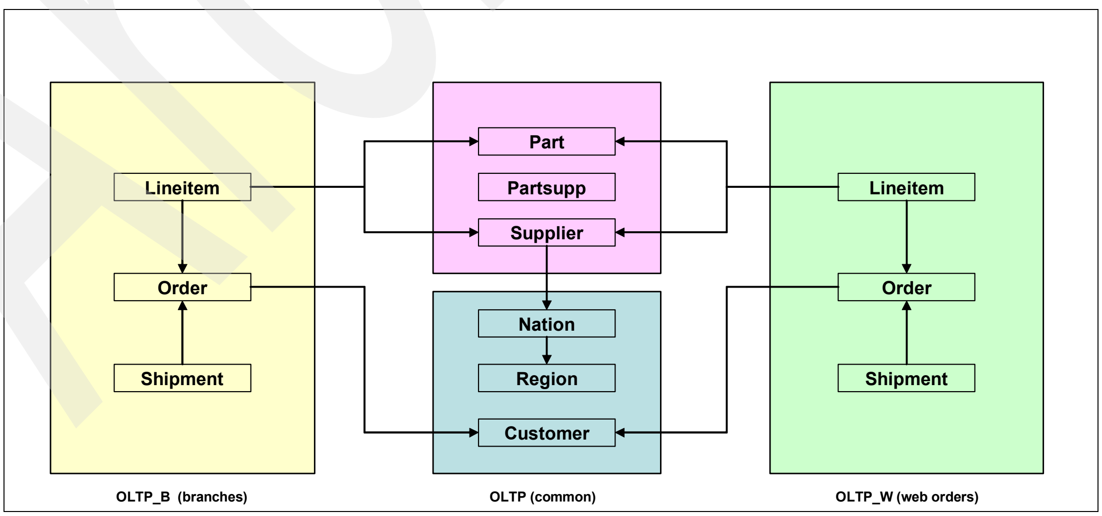


Figure 10-9 Incremental build of MQTLINE

The SOF procedure in Example 10-5 shows how we build MQTLINEA for a given time frame and how we restrict data selection to the first partitions of LINEITEM by adding the predicate L_ORDERKEY <= 30000000000.

Example 10-5 SOF procedure to build interim table MQTLINEA

```
&JOB YEAR FROMMO TODAY  TOMO TODAY TABLE
&*   1998 01     01    06   30    A

  INSERT INTO VLDB50.MQTLINE&5
     SELECT L_SHIPDATE, L_DISCOUNT, L_QUANTITY,
        SUM(L_EXTENDEDPRICE * L_DISCOUNT) AS REVENUE
     FROM VLDB50.LINEITEM
     WHERE L_SHIPDATE BETWEEN '&0-&1-&2' AND '&0-&3-&4
     AND L_ORDERKEY <= 30000000000
         GROUP BY L_SHIPDATE, L_DISCOUNT, L_QUANTITY
         ORDER BY L_SHIPDATE, L_DISCOUNT, L_QUANTITY;
```

The SQL statement in Example 10-6 shows how the interim tables could be combined and aggregated to insert the total aggregates to MQTLINE. This approach requires consideration of distinct time frames to avoid duplicate keys and wrong results in MQTLINE.

Example 10-6 SQL statement to insert interim table content into MQTLINE

```
INSERT INTO MQTLINE
  SELECT
   INTERIM.L_SHIPDATE, INTERIM.L_DISCOUNT, INTERIM.L_QUANTITY,
   SUM(INTERIM.REVENUE) AS REVENUE
   FROM
  (
    SELECT * FROM MQTLINEA
    UNION ALL
    SELECT * FROM MQTLINEB
    UNION ALL
    ....
    SELECT * FROM MQTLINEJ
   ) AS INTERIM
   GROUP BY INTERIM.L_SHIPDATE, INTERIM.L_DISCOUNT, INTERIM.L_QUANTITY;
```

Before we enable MQTLINE for query rewrites, we use Optimization Expert (see 10.6.1, “Validate query rewrite with Optimization Expert” on page 198) to investigate the original access plan for the query in Example 10-3 on page 201). The graph in Figure 10-10 on page 204 shows, as expected, a quite expensive table scan on LINEITEM.

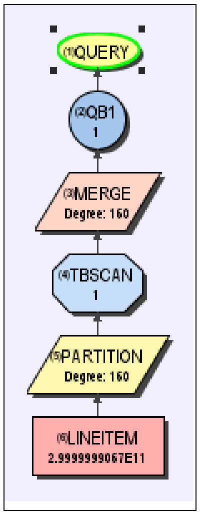


Figure 10-10 Original access plan graph before enabling MQTLINE

We now enable query optimization for MQTLINE to make it eligible for query rewrite. See Example 10-7.

Example 10-7 Turing query optimization on for MQTLINE

```
ALTER TABLE VLDB50.MQTLINE
  ALTER MATERIALIZED QUERY
  SET ENABLE QUERY OPTIMIZATION;
```

The new access plan graph for the same query now shows that MQTLINE is considered by the optimizer. See Figure 10-11.

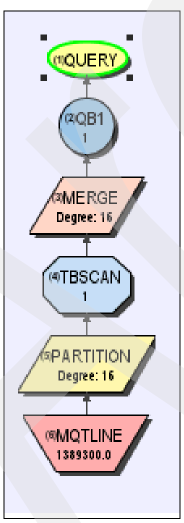


Figure 10-11 New access plan graph with MQTLINE

It is no surprise that the new access plan leads to reduce response times significantly, as shown in Table 10-2. Even though the query can run fairly well parallel without the MQT, the MQT gives better response time by orders of magnitude.

Table 10-2 Query response times without and with MQT query rewrite

	Without MQT	With MQT query rewrite
Query response times as per job log	302.82 minutes CPU time 7.3 minutes elapsed time	0.00 minutes CPU time 0.0 minutes elapsed time (responds immediately)

10.7.2 Top 10 sold parts in a region

To provide customers with additional recommendations for certain parts, an operational BI application could provide a list of parts that are most frequently sold in a certain geographic location. To compile this list, an application might run a query like Example 10-8.

Example 10-8 Query to return “top 10” parts for a customer’s geography

```
SELECT
  M.GEO, M.PARTKEY, M.SALESRANK
FROM
  MQTTOP10 M, CUSTOMER C
WHERE
  M.GEO = C.C_GEOKEY AND
  M.QUARTERCAPTION = ’1998Q1’ AND
  C.C_CUSTKEY = 111110
ORDER BY M.SALESRANK;
```

Note that that this sample query references the MQT directly. Normally the reference would be in a table in the database. The context for the query is determined by a certain customer (identified by C_CUSTKEY=111110). For this customer’s geographic location, the MQT provides a list of part keys along with its corresponding sales rank in the sales period (first quarter of 1998 in our example).

For our database, the result looks like Example 10-9 and the operational BI application could propose customer 111110 to consider part 4059167954, because it has been sold frequently in his location.

Example 10-9 Top 10 ranked sales for parts in a location

```
GEO         PARTKEY              SALESRANK
----------- -------------------- --------------------
        121           4059167954                    1
        121           9005809853                    2
        121           4371736930                    3
        121           3581321901                    4
        121           9890450909                    5
        121           9700364861                    6
        121           3951979877                    7
        121           5451831769                    8
        121           6691904684                    9
```

Because MQTTOP10 is rather small (10 ranked parts for 28 quarters and 163 geographies = 45,650 rows), we do not use table partitioning for the definition. See Example 10-10. Note however, that we use the OLAP function rank() which has been introduced in DB2 9 for z/OS.

Example 10-10 MQT definition for MQTTOP10

```
CREATE TABLE MQTTOP10 AS (
  SELECT * FROM (
    SELECT
       C.C_GEOKEY AS GEO,
       A.A_CALENDARQUARTERCAPTION AS QUARTERCAPTION,
       L.L_PARTKEY AS PARTKEY,
            SUM(L.L_EXTENDEDPRICE) AS SUMPRICE,
            RANK() OVER(PARTITION BY C.C_GEOKEY
                   ORDER BY SUM(L.L_EXTENDEDPRICE) DESC) AS SALESRANK
         FROM
            VLDB50.LINEITEM L,
            VLDB50.ORDERS O,
            VLDB50.CUSTOMER C,
            VLDB50.ALLTIME A
         WHERE
           O.O_CUSTKEY = C.C_CUSTKEY AND
           O.O_ORDERKEY = L.L_ORDERKEY AND
           O.O_ORDERDATE = A.A_CALENDARDT
         GROUP BY
           C.C_GEOKEY, A.A_CALENDARQUARTERCAPTION, L.L_PARTKEY
        ) AS TEMP
        WHERE SALESRANK <= 10
)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY USER
DISABLE QUERY OPTIMIZATION
IN VLDB50.TSMQT2;
```

Ranking is done for the sum of extended prices in a given area. To improve build times and reduce the amount of sort space and work files required, we use an optimized approach with an interim table named GEOPARTSALES. See Example 10-11. This table contains aggregated data before it becomes ranked and filtered.

Example 10-11 Definition of interim table GEOPARTSALES

```
CREATE TABLE GEOPARTSALES
 (
    PARTKEY BIGINT,
    GEOKEY INTEGER,
    ORDERDATE DATE,
    SUMPRICE DECIMAL(12,2)
 )
 IN TPCH50TB.TSGEOSAL
  PARTITION BY RANGE(PARTKEY)
   (PARTITION 0001 ENDING AT ( 1000000000)
   ,PARTITION 0002 ENDING AT ( 2000000000)
   ,PARTITION 0003 ENDING AT ( 3000000000)
   ,PARTITION 0004 ENDING AT ( 4000000000)
   ,PARTITION 0005 ENDING AT ( 5000000000)
   ,PARTITION 0006 ENDING AT ( 6000000000)
   ,PARTITION 0007 ENDING AT ( 7000000000)
   ,PARTITION 0008 ENDING AT ( 8000000000)
   ,PARTITION 0009 ENDING AT ( 9000000000)
   ,PARTITION 0010 ENDING AT (10000000000));
```

Example 10-12 shows the SQL statement that we use to populate data in GEOPARTSALES. We could also use an approach similar to the one described in 10.7.1, “MQT for line item revenue” on page 201 to total up data for a certain time frame. In a real world application, you might be interested only in sales over the past quarter, week, or day. Our example shows how to insert data based on orders for a certain day (1998-01-01).

Example 10-12 Inserting data into GEOPARTSALES

```
INSERT INTO GEOPARTSALES
  SELECT
        L.L_PARTKEY AS PARTKEY,
        C.C_GEOKEY AS GEO,
        O.O_ORDERDATE AS ORDERDATE,
        SUM( L.L_EXTENDEDPRICE ) AS SUMPRICE
     FROM
        CUSTOMER C,
        LINEITEM L,
        ORDERS O
     WHERE
        O.O_ORDERDATE = '1998-01-01' AND
        O.O_ORDERKEY = L.L_ORDERKEY AND
        L.L_CUSTKEY = C.C_CUSTKEY
     GROUP BY
        L.L_PARTKEY, C.C_GEOKEY, O.O_ORDERDATE;
```

In Table 10-3, we show the build times to build GEOPARTSALES. The query runs highly parallel. Most of the required time is used to scan, sort, and merge the large LINEITEM and ORDER tables.

Table 10-3 Build times for GEOPARTSALES

	CPU time (mins)	Elapsed time (mins)
Building of GEOPARTSALES	693.57	110.0

We insert data from GEOPARTSALES to MQTTOP10 using the rank() function. See Example 10-13.

Example 10-13 Inserting data from GEOPARTSALES to MQTTOP10

```
INSERT INTO MQTTOP10
 SELECT
   GEOKEY AS GEO,
   A_CALENDARQUARTERCAPTION AS QUARTERCAPTION,
   PARTKEY, SUMPRICE,
   SALESRANK
 FROM
 (
   SELECT
     GEOKEY, PARTKEY, ORDERDATE, SUMPRICE,
     RANK()
       OVER(PARTITION BY GEOKEY ORDER BY SUMPRICE DESC)  AS SALESRANK
   FROM
    TPCH50TB.GEOPARTSALES
  ) AS R,
  TPCH50TB.ALLTIME A
 WHERE R.ORDERDATE = A_CALENDARDT AND R.SALESRANK < 10;
```

The build times in Table 10-4 show CPU and elapsed times for building the MQT. Note that the statement in Example 10-13 on page 207 did not run in parallel; therefore elapsed time is higher than CPU time.

Table 10-4 Build times for MQTTOP10

	CPU time (mins)	Elapsed time (mins)
Building of MQTTOP10	5.99	8.1

10.7.3 Parts sold by suppliers

At the end of a time period, different reports should be created, comparing number of sold parts per supplier. To support these reports, an MQT is built that aggregates sales in a certain time frame and groups them by suppliers, nationkey, and orderdate. Example 10-14 shows the definition of MQTSUPP. To accommodate to sort space and work file limitations, we restrict the definition again to a day (1998-01-01).

Example 10-14 MQT definition for MQTSUPP

```
CREATE TABLE MQTSUPP AS (
 SELECT
   S.S_SUPPKEY AS SUPPKEY,
   S.S_NATIONKEY AS NATIONKEY,
   O.O_ORDERDATE AS ORDERDATE,
   COUNT(O.O_ORDERKEY) AS NUMBERORDERS
 FROM
   SUPPLIER S, ORDERS O, LINEITEM L
 WHERE
   O.O_ORDERDATE = '1998-01-01' AND
   O.O_ORDERKEY = L.L_ORDERKEY AND
   L.L_SUPPKEY = S.S_SUPPKEY
 GROUP BY S.S_SUPPKEY, S.S_NATIONKEY, O.O_ORDERDATE )
 DATA INITIALLY DEFERRED
 REFRESH DEFERRED
 MAINTAINED BY SYSTEM
 DISABLE QUERY OPTIMIZATION
 IN VLDB50.TSMQT4
  PARTITION BY RANGE(SUPPKEY)
   (PARTITION 0001 ENDING AT ( 50000000)
   ,PARTITION 0002 ENDING AT (100000000)
   ,PARTITION 0003 ENDING AT (150000000)
   ,PARTITION 0004 ENDING AT (200000000)
   ,PARTITION 0005 ENDING AT (250000000)
   ,PARTITION 0006 ENDING AT (300000000)
   ,PARTITION 0007 ENDING AT (350000000)
   ,PARTITION 0008 ENDING AT (400000000)
   ,PARTITION 0009 ENDING AT (450000000)
   ,PARTITION 0010 ENDING AT (MAXVALUE));
```

We defined our MQT as MAINTAINED BY SYSTEM. With this definition, data cannot be inserted by insert or load statements. However, the refresh table function (Example 10-15 on page 209) can be used to automatically load data into the MQT, based on the MQT definition.

Example 10-15 Refreshing MQT table content

```
SET CURRENT DEGREE = 'ANY';
REFRESH TABLE MQTSUPP;
```

Table 10-5 shows the build time for MQTSUPP. The underlying definition executes in parallel.

Table 10-5 Build times for MQTSUPP

	CPU time (mins)	Elapsed time (mins)
Building of MQTTOP10	832.26	39.0

10.7.4 MQT join compensation example

Recall the definition of our MQT MQTSUPP, like Example 10-16.

Example 10-16 Definition of materialized query MQTSUPP

```
SELECT
   S.S_SUPPKEY AS SUPPKEY,
   S.S_NATIONKEY AS NATIONKEY,
   O.O_ORDERDATE AS ORDERDATE,
   COUNT(O.O_ORDERKEY) AS NUMBERORDERS
 FROM
   SUPPLIER S, ORDERS O, LINEITEM L
 WHERE
   O.O_ORDERDATE BETWEEN '1998-01-01' AND '1998-01-31' AND
   O.O_ORDERKEY = L.L_ORDERKEY AND
   L.L_SUPPKEY = S.S_SUPPKEY
 GROUP BY S.S_SUPPKEY, S.S_NATIONKEY, O.O_ORDERDATE
```

The query in Example 10-17 shows an extension where the additional table PARTSUPP is added to the query expression to retrieve the corresponding available quantity.

Example 10-17 Additional table PARTSUPP in query

```
SELECT
   S.S_SUPPKEY AS SUPPKEY,
   S.S_NATIONKEY AS NATIONKEY,
   O.O_ORDERDATE AS ORDERDATE,
   PS.PS_AVAILQTY,
   COUNT(O.O_ORDERKEY) AS NUMBERORDERS
 FROM
   SUPPLIER S, ORDERS O, LINEITEM L, PARTSUPP PS
 WHERE
   O.O_ORDERDATE = '1998-01-01'  AND
   O.O_ORDERKEY = L.L_ORDERKEY AND
   L.L_SUPPKEY = S.S_SUPPKEY   AND
   S.S_SUPPKEY = PS.PS_SUPPKEY
 GROUP BY S.S_SUPPKEY, S.S_NATIONKEY, O.O_ORDERDATE, PS.PS_AVAILQTY;
```

The access plan in Figure 10-12 shows that even with the modified query, the MQT can be used.

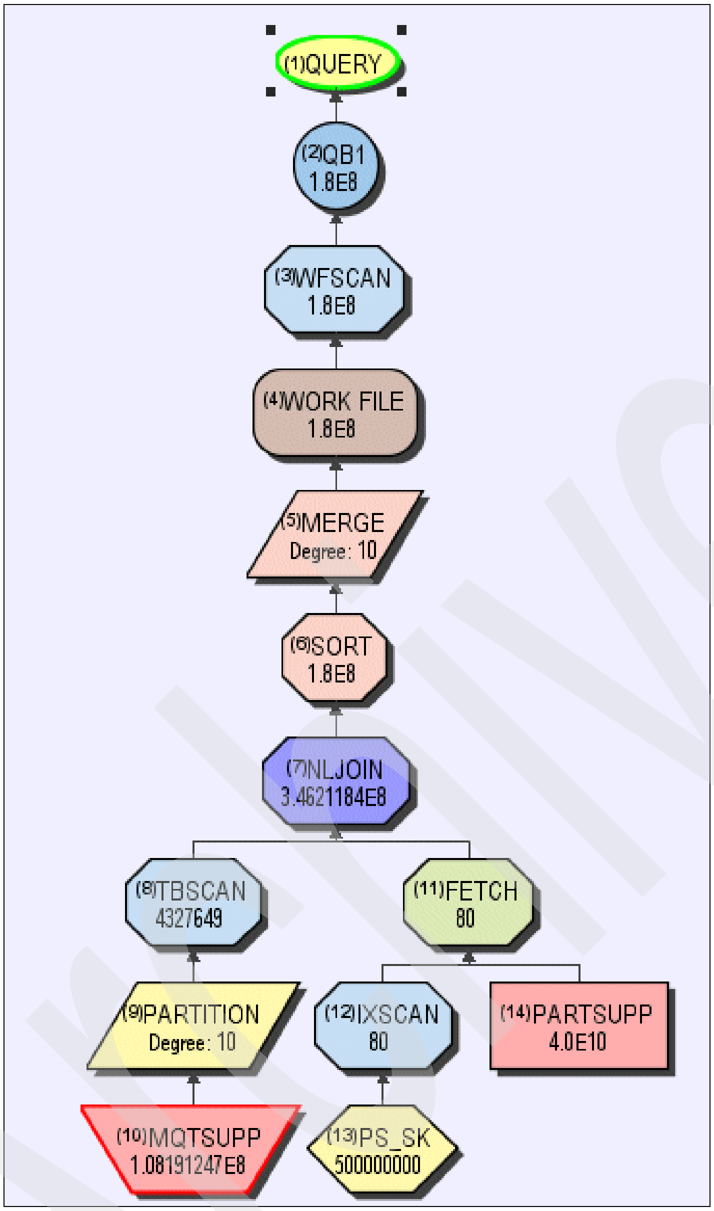


Figure 10-12 Access plan for query with additional table

11

Workload management

The purpose of this chapter is to help a DB2 Administrator understand how WLM works and how it can impact the way that applications receive resources from the system. It would also help the DBA to obtain business requirements and translate them to a Service Level Agreement, in discussion with those in charge of managing the z/OS WLM definitions.

This chapter is organized in the following manner:

- Background on how WLM works. With a DBA in mind, this section describes the principles of how WLM works and its components. We also list some reference materials.
- What data collect and how to collect it. Presentation of tools for analysis and reporting of a WLM policy and how it impacts workloads.
- Discussion of a process for WLM policy verification and changes.
- Discussion of how to audit a WLM policy and guidelines on how to adapt a policy to business requirements, based on a customer example.
- Sample lab scenarios designed to understand how to use WLM in a Data Warehouse environment. Controlled scenarios where you will see how the main features of WLM can be implemented, and how to obtain and understand relevant data.

11.1 Introduction

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 function that makes this possible is dynamic workload management, which is implemented in the Workload Manager component of the z/OS operating system.

The idea of z/OS Workload Manager (WLM) is to make a contract between the applications and the operating system. The installation classifies the work running on the z/OS operating system in distinct service classes, and defines goals for them that express the expectation of how the work should perform. WLM uses these goal definitions to manage the work across all systems of a sysplex environment.

The evolution of Data Warehousing (DW) 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 concerns in the DW design.

11.2 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 will adjust dispatch priorities and resource allocations to meet the goals of the service class definitions. It will do 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
- IBM redpaper: *Workload Management for DB2 Data Warehouse*
- WLM presentations

  http://www.ibm.com/servers/eserver/zseries/zos/wlm/presentations/

- WLM tools

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

11.2.1 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 11-1 shows the hierarchical relation between the WLM components that are described in the sections that follow the figure.

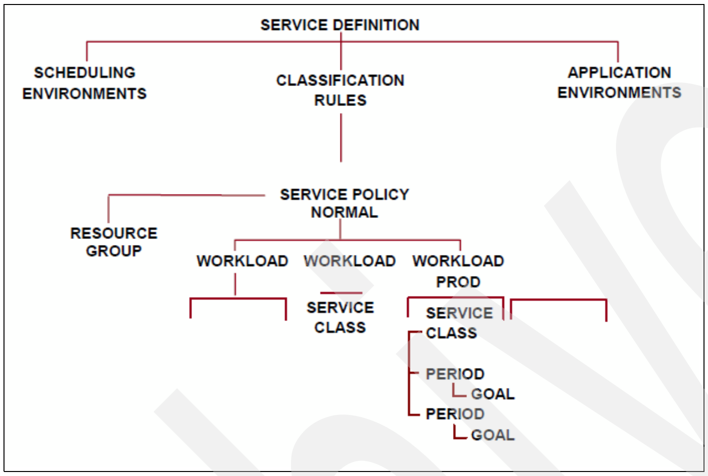


Figure 11-1 WLM components relationship

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 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.

- SYSOTHER

  This functions as the default service class for non-STC address spaces when no classification rules exists for the subsystem type. It is assigned a discretionary goal.

Report classes

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 zOS. 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/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% CPU 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). In addition, discretionary goals are associated with an important of 6 and are always considered as donors. Workload in a discretionary goal receive resources only when all the other service classes are achieving their goals.

Table 11-1 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 type 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.

Figure 11-2 represents how incoming workload is classified into a specific service class by the classification rules.

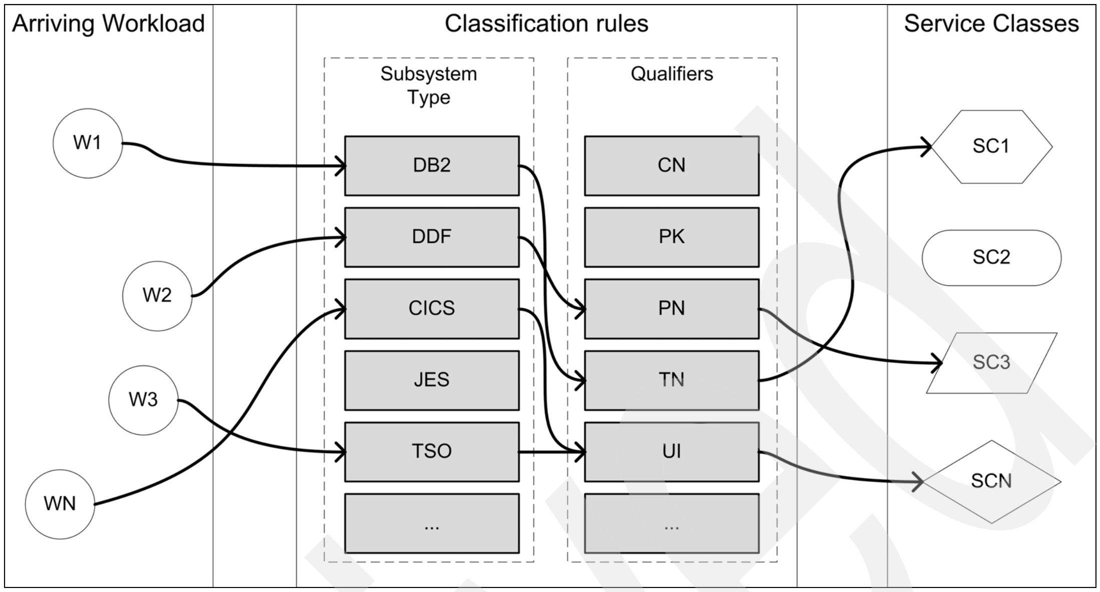


Figure 11-2 WLM workload classification

For DB2 specific performance information for WLM, see the DB2 Administration Guide for your version of DB2.

Table 11-2 shows an extract of the IBM-defined subsystem and the Table 11-3 on page 217 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 11-2 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 11-3 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 to make easy classification and it 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 CPU service for one or more service classes.

11.2.2 The performance index

WLM maintains a performance index (PI) for each service class period to measure 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.

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 time 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 11-1.

  Example 11-1 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 11-2.

  Example 11-2 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 11-3 shows how this PI is calculated.

  Example 11-3 PI calculation for RT goals with percentile

  ```
  PI = Percentile Actual / Percentile Goal
  ```

Figure 11-3 on page 219 show 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 goals achievement.

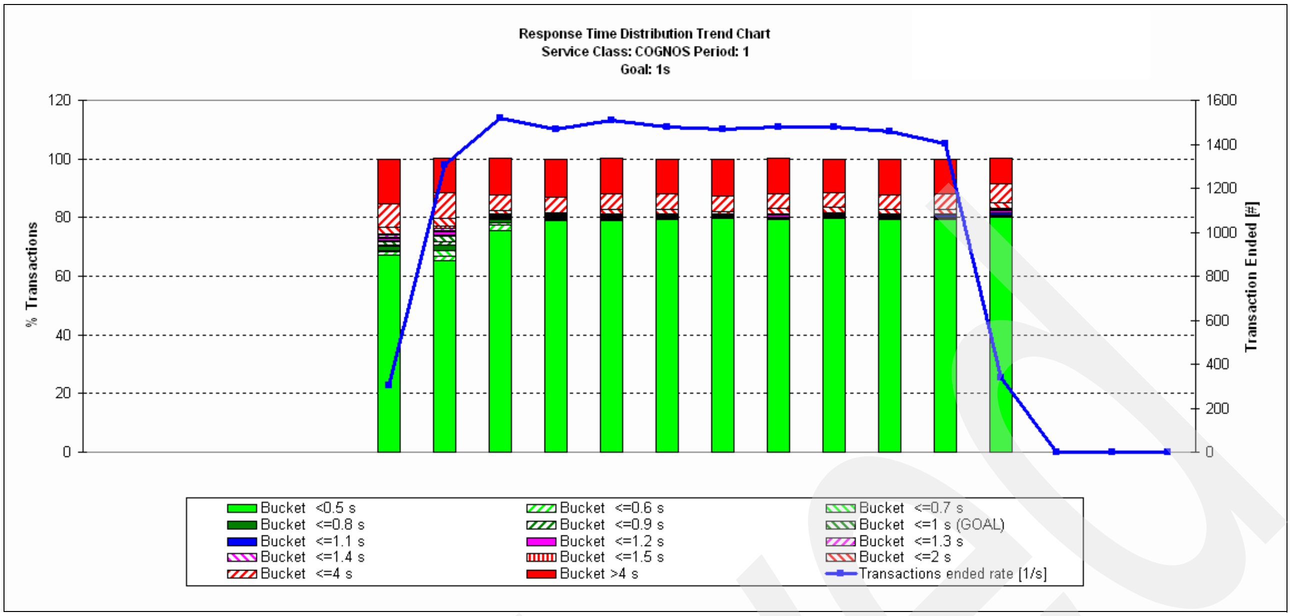


Figure 11-3 RMF response time distribution trend chart

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 an universal donor with a fixed PI=0.81. It receives resources only when all other service classes are achieving its goals.

11.2.3 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 is 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, like CPU 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) specify 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 CMSTAT=INACTIVE (and the thread cannot become pooled), DDF creates one enclave for the life of the thread. Response time goals and multiple periods should not be used.

You need to classify each request. If you do not classify your DDF transactions into service classes, they are assigned to the default class discretionary goal. It is a good idea to have different goals for mission-critical DDF work (higher goals), and batch-like DDF activity (lower goals).

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.

For a detailed description and samples of how to classify DB2 stored procedures, refer to IBM Redbooks publication *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083.

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 it 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 the 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 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.

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 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 11-4 shows the relation between stored procedure, WLM application environments, and procedures definitions.

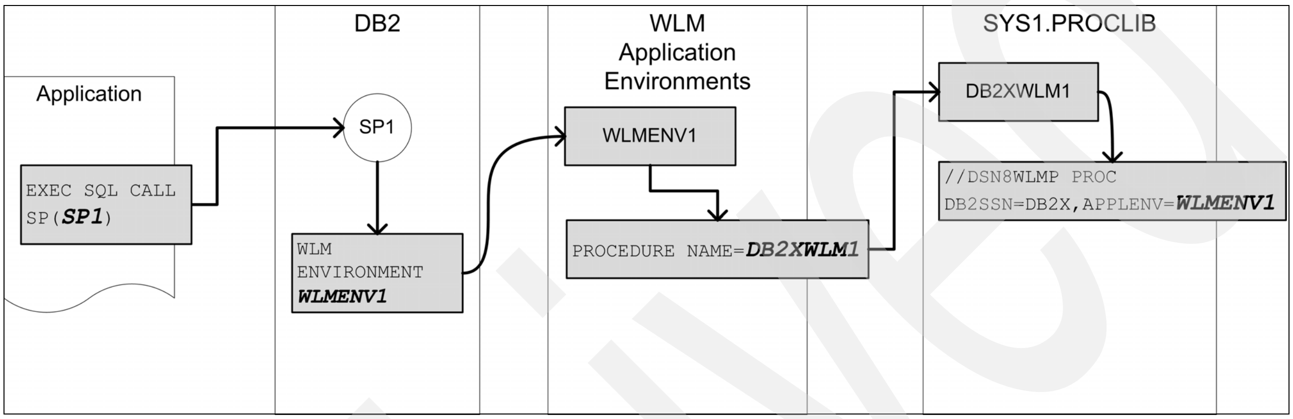


Figure 11-4 Stored procedures definitions relationship

Table 11-4 lists the recommended basic application environments that you typically will define in your environment. There could be more, depending on your environment's requirements.

Table 11-4 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 prevent 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.

11.2.4 Goal guidelines for mixed data warehouse workloads

We suggest the use of multiple service classes to differentiate users and applications that have different levels of importance to the business. Consider the following guidelines for goals for mixed DW workload. Within each of the service classes, consider utilizing multiple periods.

- Consider percentile response time goals for early periods that have frequent completions of shorter consumption work.
- Consider velocity goals for later periods containing work having less-frequent completions and larger, perhaps more varying, resource consumption characteristics.
- Potentially utilize a discretionary goal for the last period.

For example. Operational BI queries are typically numerous and short consumers. Therefore they should have response time goals and fall into early periods. On the other hand, data mining activity might be less frequent, long-running, and have wide variability in resource consumption. It would therefore likely be targeted for velocity goals and later periods.

> **Important:** If you need to change a goal, change the velocity between 5 and 10%. Velocity goals do not translate directly to priority. Higher velocity tends to have higher priority, but this is not always the case.

The importance of defining realistic goals

It is important to define goals that are realistic and in line not only with the business requirements, but also with the current machine capabilities. For instance, consider the graph shown in the Figure 11-5 on page 224. It shows that the goals for the service class COGNOS are not being achieved.

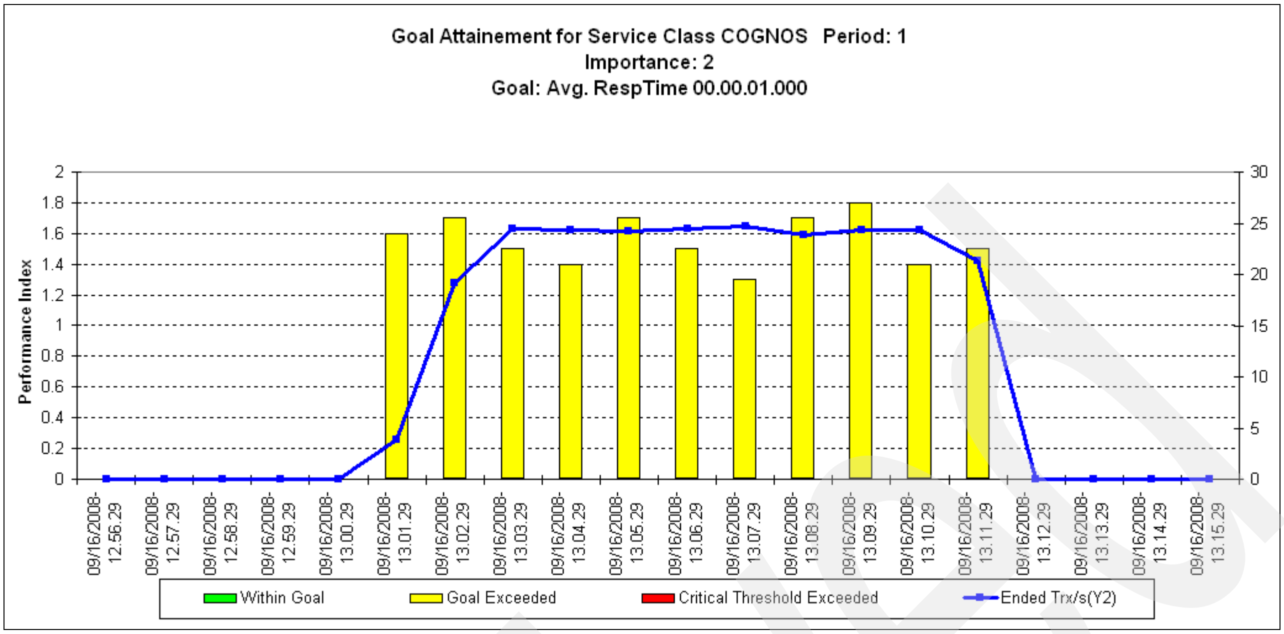


Figure 11-5   Goals not achieved

As shown in the graph, the goal for this service class is an average response time of 1 second. This service class is used by COGNOS OLTP like activity and this response time may be a business requirements for this tasks. Example 11-4 shows a portion of a RMF batch report showing the workload activity for this service class during the period on which the goals were not achieved. This report describes the goal as response time 1 sec avg. If you look at the execution velocity (EX VEL%) column, you will read that this service class was receiving CPU resources 99.1% of the time during this period, which is exceptionally high. A response time of 1 second is not achievable for this kind of transactions on the machine on which the test was executed.

Example 11-4   RMF batch reporting sample

```
GOAL: RESPONSE TIME 000.00.01.000 AVG

            RESPONSE TIME EX   PERF  AVG   ------ USING% ----- ------------ EXECUTION DELAYS % ------------
---DLY%-- -CRYPTO%-    %
   SYSTEM    HHH.MM.SS.TTT VEL% INDX ADRSP   CPU  AAP  IIP  I/O  TOT
UNKN IDLE  USG  DLY QUIE

   P17       000.00.01.529 99.1  1.5  41.6   1.5  N/A  N/A  0.8  0.0
97.7  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        1070            1070        72.9        72.9
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   <= 00.00.00.600        1072               2        73.0         0.1  >
   <= 00.00.00.700        1073               1        73.1         0.1  >
   <= 00.00.00.800        1076               3        73.3         0.2  >
   <= 00.00.00.900        1078               2        73.4         0.1  >
   <= 00.00.01.000        1089              11        74.2         0.7  >
   <= 00.00.01.100        1090               1        74.3         0.1  >
```

```
<= 00.00.01.200        1103            13         75.1         0.9  >
<= 00.00.01.300        1112             9         75.7         0.6  >
<= 00.00.01.400        1127            15         76.8         1.0  >
<= 00.00.01.500        1138            11         77.5         0.7  >
<= 00.00.02.000        1187            49         80.9         3.3  >>
<= 00.00.04.000        1305           118         88.9         8.0  >>>>>
>  00.00.04.000        1468           163          100        11.1  >>>>>>
```

11.2.5 The role of the database administrator

The database administrator (DBA) is typically in close contact with the application developers and usually is in one of the first to be contacted in case of performance degradation. This situation makes the DBA the ideal link between the business requirements and the system.

The WLM on System z provides sophisticated workload management capabilities. These capabilities should be applied to serve the business requirements.

The DBA is in the ideal position to match the application requirements to the WLM definitions. To achieve this, the DBA must have the following types of knowledge:

- How the application works and which workload is the most important
- Understand how WLM works

Service Level Agreement

The human view of a system's performance is often subjective, emotional, and difficult to measure. Because systems were created to meet the business needs of users, however, the concept of the Service Level Agreement (SLA) was introduced to match specified business requirements with subjective perceptions. The SLA is a contract that objectively describes and enforces measurements, including the measurements that follow:

- Average transaction response time for network, I/O, CPU, or total
- The distribution of these response times (for example, 90% TSO trivial transactions at less than 200 milliseconds)
- System availability metrics

As a general guideline, it is not suggested that you include the external throughput rate (ETR) in an SLA, because ETR includes variables not under control of the system administrators, such as the number of users and user think time.

11.3 Analyzing DW workloads

This section describes some of the tools that can be used to explore the WLM definitions and how these definitions are actually working. Emphasis was placed on describing the tools that would help with a quick view and understanding of the system definition and behavior.

11.3.1 WLM Service Definition Formatter

The WLM Service Definition Formatter is a tool that assists you in displaying your WLM service definition. To use the tool, download the WLM service definition to your workstation and load it into the spreadsheet. Use the various worksheets to display parts of your service definition to get a better overview of your WLM definitions.

The tool is not a service definition editor. All modifications to the WLM service definition must be entered through the WLM Administrative Application.

Tip: You can get this software for free from the following Web page:

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

You need to print the WLM definitions and transfer it to your workstation before using this tool.

Example 11-5 shows the option 5 PRINT from the WLM ISPF definition menu. Use this option for printing the WLM definitions. See the previous section for guidelines on how to get access to this menu.

Example 11-5 Print WLM definitions from WLM ISPF

```
File  Utilities  Notes  Options  Help
 EssssssssssssssssssssN ---------------------------------------------------
 e 5  1. New          e        Definition Menu        WLM Appl LEVEL019
 e    2. Open         e ____________________________________________________
 e    3. Save         e
 e    4. Save as      e . : none
 e    5. Print        e
 e    6. Print as GML e . . TPCH50TB  (Required)
 e    7. Cancel       e . . Service Definition - 50TB Study
 e    8. Exit         e
 DssssssssssssssssssssM
 following options. . . . . ___  1.   Policies
                                 2.   Workloads
                                 3.   Resource Groups
                                 4.   Service Classes
                                 5.   Classification Groups
                                 6.   Classification Rules
                                 7.   Report Classes
                                 8.   Service Coefficients/Options
                                 9.   Application Environments
                                 10.  Scheduling Environments
```

When the process is done, you will receive the confirmation message shown in Example 11-6.

Example 11-6 WLM written to data set confirmation message

```
Output written to ISPF list dataset. (IWMAM001)
```

To get the printed definitions into your workstation, perform the following steps. Exit the ISPF until you get to the confirmation panel shown in Example 11-7. Be sure to select option 4 KEEP DATA SET NEW under the LIST DATA SET menu.

Example 11-7 Exit ISPF confirmation panel

```
Specify Disposition of Log and List Data Sets
 Command ===>
                                                   More:     +
 Log Data Set (VLDB12.SPFLOG2.LIST) Disposition:
 Process Option . . . . 2  1. Print data set and delete
                           2. Delete data set without printing
                           3. Keep data set - Same
                              (allocate same data set in next session)
```

```
                          4. Keep data set - New
                             (allocate new data set in next session)
Batch SYSOUT class . .
Local printer ID or
writer-name  . . . . .
Local SYSOUT class . .

List Data Set (VLDB12.SPF6.LIST) Disposition:
Process Option . . . . 4  1. Print data set and delete
                          2. Delete data set without printing
                          3. Keep data set - Same
                             (allocate same data set in next session)
                          4. Keep data set - New
                             (allocate new data set in next session)
Batch SYSOUT class . . A
Local printer ID or
```

Take note of the List Data Set file name, VLDB12.SPF6.LIST on this example. Exit ISPF completely, you will receive a confirmation that the list data set was saved as shown in Example 11-8.

Example 11-8 List data set saved confirmation message

```
  VLDB12.SPFLOG2.LIST has been deleted.
  VLDB12.SPF6.LIST has been kept.
READY
```

Logging back into ISPF you can browse this data set as shown in Example 11-9. This file contains the WLM definitions in a readable format and you can explore it for understanding your WLM setup.

Example 11-9 Browsing printed WLM definitions in ISPF

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
 EDIT       VLDB12.SPF6.LIST                                Columns 00001 00072
 Command ===>                                                  Scroll ===> CSR
 ****** ***************************** Top of Data ******************************
 000001 1 * Service Definition TPCH50TB - Service Definition - 50TB Study
 000002
 000003     8 workloads, with 13 service classes
 000004     0 resource groups
 000005     7 service policies
 000006     20 classification groups
 000007     7 subsystem types
 000008     60 report classes
 000009     13 application environments
 000010     0 scheduling environments
 000011     0 resources
 000012
 000013     CPU coefficient of 1.0
 000014     IOC coefficient of 0.5
 000015     MSO coefficient of 0.0001
 000016     SRB coefficient of 1.0
 000017
 000018     I/O priority management           Yes
 000019     Dynamic alias tuning management   Yes
```

You need to get this data set into your workstation for processing using the WLM Service Definition Formatter. Transfer this file and store it into your workstation. Figure 11-6 shows how to start it from your workstation.

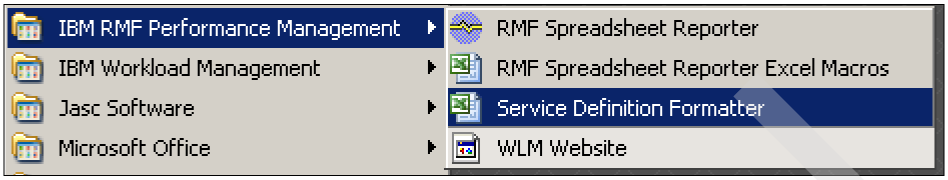


Figure 11-6 Starting the WLM service definition formatter

Figure 11-7 shows the panel you will receive when the software is started. This product is a spreadsheet application. From this panel, select the printed definitions file you stored in your workstation by clicking **READ SERVICE DEFINITION**. You can then explore the service definitions by using the spreadsheet tabs.

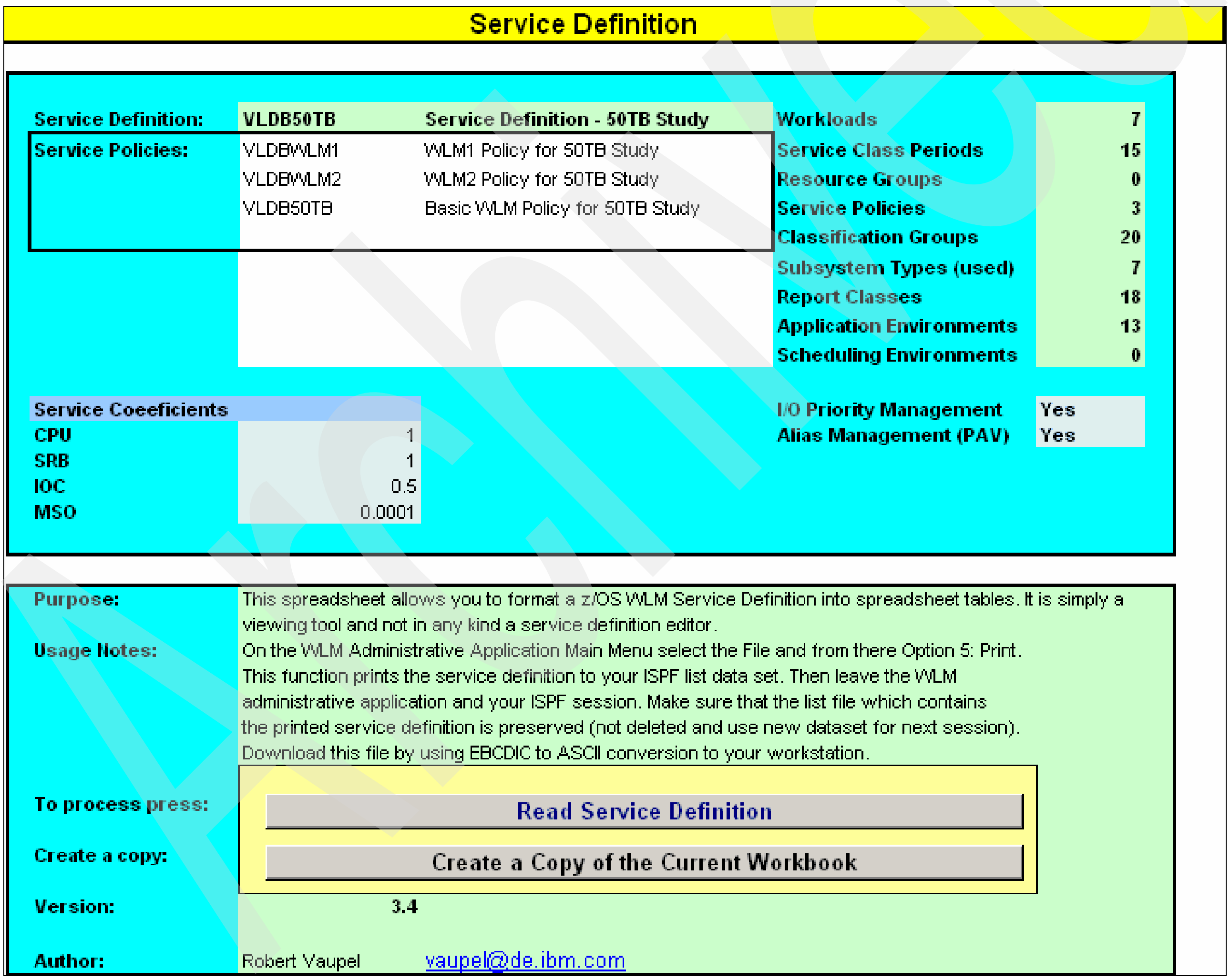
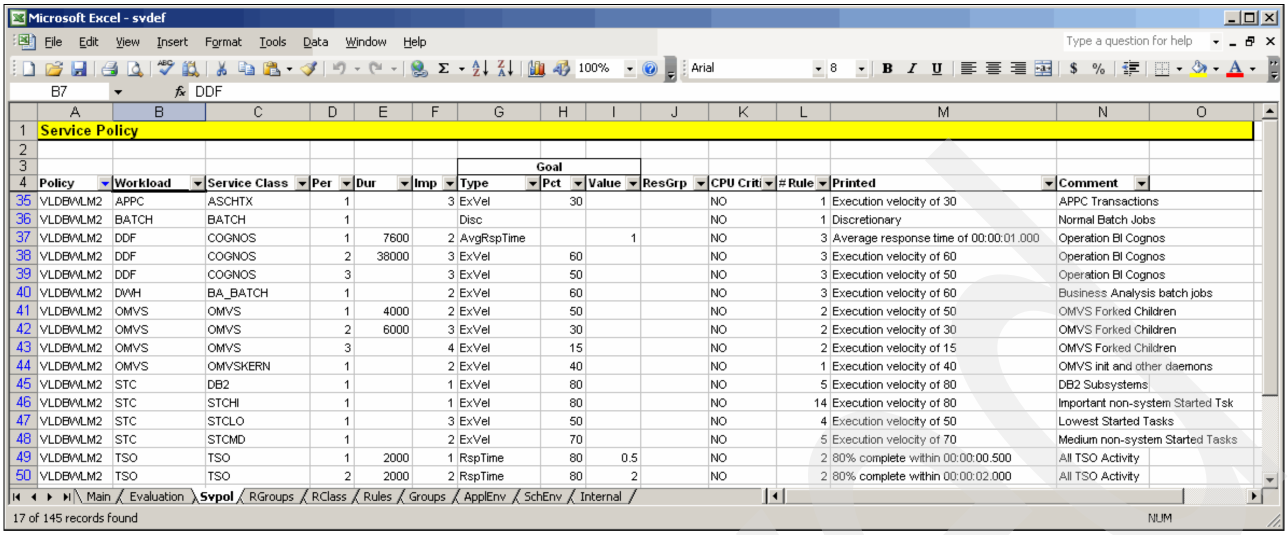


Figure 11-7 WLM Service Definition Formatter main window

Figure 11-8 shows an example of exploring a service policy definition.



Service Policy

	Policy	Workload	Service Class	Per	Dur	Imp	Goal Type	Goal Pct	Goal Value	ResGrp	CPU Criti	# Rule	Printed	Comment
35	VLDBWLM2	APPC	ASCHTX	1		3	ExVel	30			NO	1	Execution velocity of 30	APPC Transactions
36	VLDBWLM2	BATCH	BATCH	1			Disc				NO	1	Discretionary	Normal Batch Jobs
37	VLDBWLM2	DDF	COGNOS	1	7600	2	AvgRspTime		1		NO	3	Average response time of 00:00:01.000	Operation BI Cognos
38	VLDBWLM2	DDF	COGNOS	2	38000	3	ExVel	60			NO	3	Execution velocity of 60	Operation BI Cognos
39	VLDBWLM2	DDF	COGNOS	3		3	ExVel	50			NO	3	Execution velocity of 50	Operation BI Cognos
40	VLDBWLM2	DWH	BA_BATCH	1		2	ExVel	60			NO	3	Execution velocity of 60	Business Analysis batch jobs
41	VLDBWLM2	OMVS	OMVS	1	4000	2	ExVel	50			NO	2	Execution velocity of 50	OMVS Forked Children
42	VLDBWLM2	OMVS	OMVS	2	6000	3	ExVel	30			NO	2	Execution velocity of 30	OMVS Forked Children
43	VLDBWLM2	OMVS	OMVS	3		4	ExVel	15			NO	2	Execution velocity of 15	OMVS Forked Children
44	VLDBWLM2	OMVS	OMVSKERN	1		2	ExVel	40			NO	1	Execution velocity of 40	OMVS init and other daemons
45	VLDBWLM2	STC	DB2	1		1	ExVel	80			NO	5	Execution velocity of 80	DB2 Subsystems
46	VLDBWLM2	STC	STCHI	1		1	ExVel	80			NO	14	Execution velocity of 80	Important non-system Started Tsk
47	VLDBWLM2	STC	STCLO	1		3	ExVel	50			NO	4	Execution velocity of 50	Lowest Started Tasks
48	VLDBWLM2	STC	STCMD	1		2	ExVel	70			NO	5	Execution velocity of 70	Medium non-system Started Tasks
49	VLDBWLM2	TSO	TSO	1	2000	1	RspTime	80	0.5		NO	2	80% complete within 00:00:00.500	All TSO Activity
50	VLDBWLM2	TSO	TSO	2	2000	2	RspTime	80	2		NO	2	80% complete within 00:00:02.000	All TSO Activity

Figure 11-8 WLM Service Definition Formatter service policy view

11.3.2 WLM Service Definition Editor

The WLM Service Definition Editor is a workstation-based tool that helps maintain your WLM service definition. It integrates a FTP download/upload mechanism that allows you to transfer WLM service definitions in ISPF format between host and workstation. The WLM Service Definition Editor provides different views to analyze, edit, and print your workload management definitions.

A DBA would not normally be responsible for updating the WLM service definitions, but this tool can be used as a graphical and accessible way of exploring its definitions.

11.3.3 Using RMF for monitoring workloads

There are various products and features that allow you to translate RMF statistics into usable information.

The RMF spreadsheet reporter

The RMF Spreadsheet Reporter serves as a front-end to the RMF postprocessor on your z/OS system. With its graphics capabilities, RMF Spreadsheet Reporter allows you to analyze z/OS performance data through powerful graphical charts right from your workstation.

The main advantage of this product is its easy utilization. You may be able to get a clear view of the system behavior in a short period of time. This tool was extensively used during the examples of this chapter.

> **Tip:** You can get this software for free from the following Web page:
>
> `http://www.ibm.com/servers/eserver/zseries/zos/rmf/tools/#spr_win`

Figure 11-9 shows the process involved in getting data from RMF to a spreadsheet report. The needed steps are described in the following paragraphs.

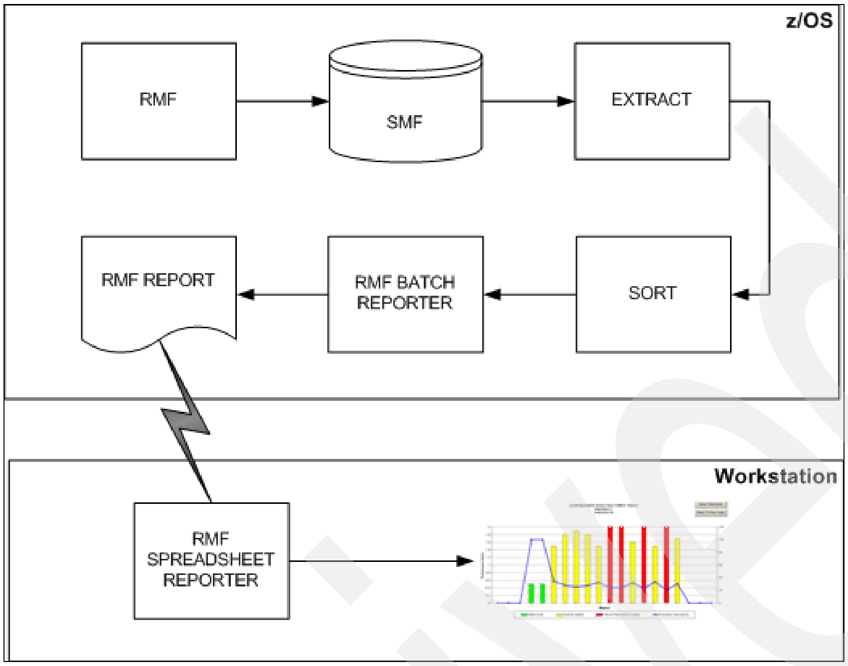


Figure 11-9 RMF spreadsheet reporter data flow schema

> **Important:** During the example shown in this chapter, the RMF interval was set to 1 minute to get a high level of granularity during the analysis. This is not the default for RMF. Verify with your installation the parameter being used if you need a good detail level.

Example 11-10 Locating an SMF dump data set

```
Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY SMFDP17  STC17793  DSID   101 LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> CSR
********************************* TOP OF DATA **********************************
IFA010I SMF DUMP PARAMETERS
IFA010I END(2400) -- DEFAULT
IFA010I START(0000) -- DEFAULT
IFA010I DATE(1900000,2099366) -- DEFAULT
IFA010I OUTDD(DUMPOUT,TYPE(0:255)) -- SYSIN
IFA010I INDD(DUMPIN,OPTIONS(ALL)) -- SYSIN
IFA020I DUMPOUT  -- VLDB.SMF.P17.D080911.T115314
IFA020I DUMPIN   -- SYS1.P17.MANY
IFA018I SMF DATA SET DUMPIN   HAS BEEN CLEARED.
                                           SUMMARY ACTIVITY REPORT
      START DATE-TIME  09/11/2008-11:30:25                       END DATE-TIME
      RECORD       RECORDS         PERCENT      AVG. RECORD    MIN. RECORD   MAX.
        TYPE          READ        OF TOTAL           LENGTH         LENGTH
           2             0
           3             0
```

RMF generates SMF records 70:79. The Workload Activity Reports are contained in the SMF 72. To save processing time and resource usage, you can extract only these records from SMF. However, these records alone may be not enough for exploiting the full reporting capabilities of the RMF Spreadsheet Reporter. Example 11-11 shows the JCL used for extracting the SMF records belonging to a specific partition. If your installation cumulates data of several partitions in a single data set, this method can be used for reducing the data that need to be treated.

Example 11-11 Extracting SMF records for a specific partition

```
//CRISEXTR JOB CLASS=A,
//    MSGLEVEL=(1,1),MSGCLASS=H,NOTIFY=&SYSUID
//*-----------------------------------------------------------*
//EXTRACT  EXEC   PGM=IFASMFDP
//INDD1    DD     DISP=SHR,DSN=VLDB.SMF.P17.D080911.T115314
//OUTDD1   DD DSN=VLDB12.SMF.EXTRACT,
//         DISP=(NEW,CATLG),SPACE=(CYL,(155,155),RLSE)
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
         INDD(INDD1,OPTIONS(ALL))
         SID(P17)
/*
```

You may encounter errors during this process due to SMF records not in order, as shown in Example 11-12. To avoid this problem, you can sort the SMF data using a JCL like the one in Example 11-13.

Example 11-12 RMF error due to unordered SMF data

```
ERB486I PPS: CROSS SYSTEM REPORTS OPTIONS IN EFFECT
ERB486I PPS:   SYSRPTS(WLMGL(POLICY)) -- SYSIN
ERB486I PPS:   SYSRPTS(WLMGL(SYSNAM(P17))) -- SYSIN
ERB103I PPS: OPTIONS IN EFFECT
ERB103I PPS:   SUMMARY(TOT)  -- DEFAULT
ERB103I PPS:   SUMMARY(INT)  -- DEFAULT
ERB103I PPS:   NODELTA  -- DEFAULT
ERB103I PPS:   NOEXITS  -- DEFAULT
ERB103I PPS:   MAXPLEN(50)  -- DEFAULT
ERB103I PPS:   ETOD(0000,2400)  -- DEFAULT
ERB103I PPS:   STOD(0000,2400)  -- DEFAULT
ERB103I PPS:   PTOD(0000,2400)  -- DEFAULT
ERB103I PPS:   RTOD(0000,2400)  -- DEFAULT
ERB103I PPS:   SYSOUT(A)  -- DEFAULT
ERB103I PPS:   DATE(01011958,12312057) -- DEFAULT
ERB478I PPS: SMF RECORDS NOT SORTED
```

Example 11-13 Sort SMF data JCL example

```
//CRISSORT JOB CLASS=A,
//    MSGLEVEL=(1,1),MSGCLASS=H,NOTIFY=&SYSUID
//*-----------------------------------------------------------*
//RMFSORT  EXEC PGM=SORT
//SORTIN   DD DISP=(SHR),DSN=VLDB12.SMF.EXTRACT
//SORTOUT  DD   DISP=(OLD),DSN=VLDB12.SMF.SORTED
//SORTWK01 DD   DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10))
//SORTWK02 DD   DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10))
//SORTWK03 DD   DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10))
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   *
  SORT FIELDS=(11,4,CH,A,7,4,CH,A),EQUALS
  MODS E15=(ERBPPE15,36000,,N),E35=(ERBPPE35,3000,,N)
```

> **Important:** The process can be done against SMF dump data sets or against an extract of SMF data. The latter option can save processing time and resource consumption

Once you have a selection of SMF data, you can create the RMF batch reports using a JCL as shown in Example 11-14.

Example 11-14 JCL sample for the creation of a RMF batch report

```
//CRISISPX JOB CLASS=A,
//     MSGLEVEL=(1,1),MSGCLASS=H,NOTIFY=&SYSUID
//*-----------------------------------------------------------*
//STEP01 EXEC PGM=ERBRMFPP,REGION=0M
//MFPINPUT DD DISP=(SHR),DSN=VLDB12.SMF.SORTED
//PPRPTS DD SYSOUT=*
//SYSIN DD *
  SYSRPTS(WLMGL(SYSNAM(P17)))
  SYSRPTS(WLMGL(SCPER,WGROUP,RCLASS,RCPER,POLICY))
/*
```

The batch reports can be used for analyzing how WLM is working in your system. You get a more powerful view by using the spreadsheet reporter. Save the batch report and transfer it to your workstation. Start the software and select **Working Set** from the Create menu as shown in Figure 11-10. Select the Report Listing folder to get this option available.

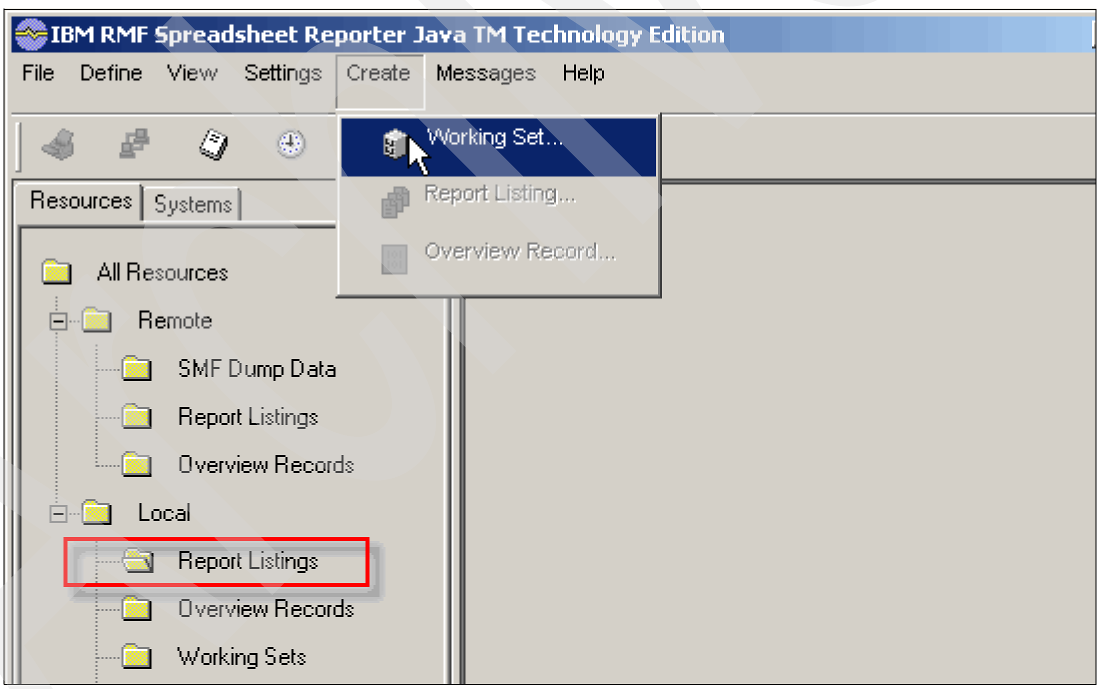


Figure 11-10 Creating a working set

You will get the Create Working Set panel shown in Figure 11-11 on page 233. You can provide a meaningful name to the working set under the Name section. Press **Run** to start the working set creation. This process could take some time if the report is big.

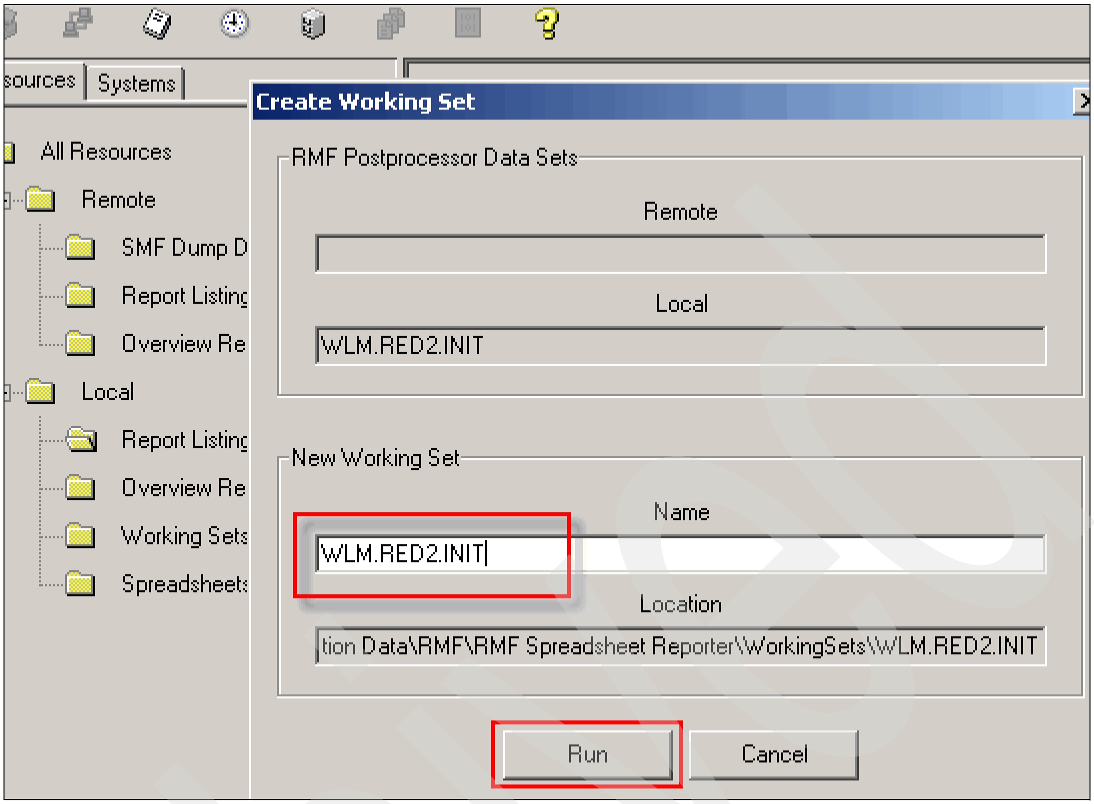


Figure 11-11 Create working set panel

Once you get confirmation of the working set, you can go to the Spreadsheet folder and select the **Workload Activity Trend Report**, for instance. This process is shown in Figure 11-12.

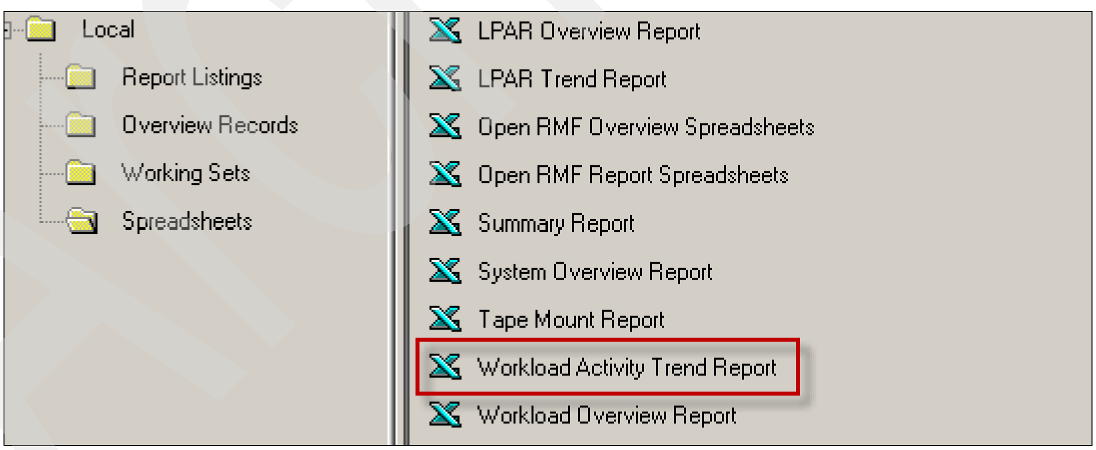


Figure 11-12 Select Workload Activity Trend Report

You will get the report main panel from which you can select a specific working set. Refer to Figure 11-13 for an example.

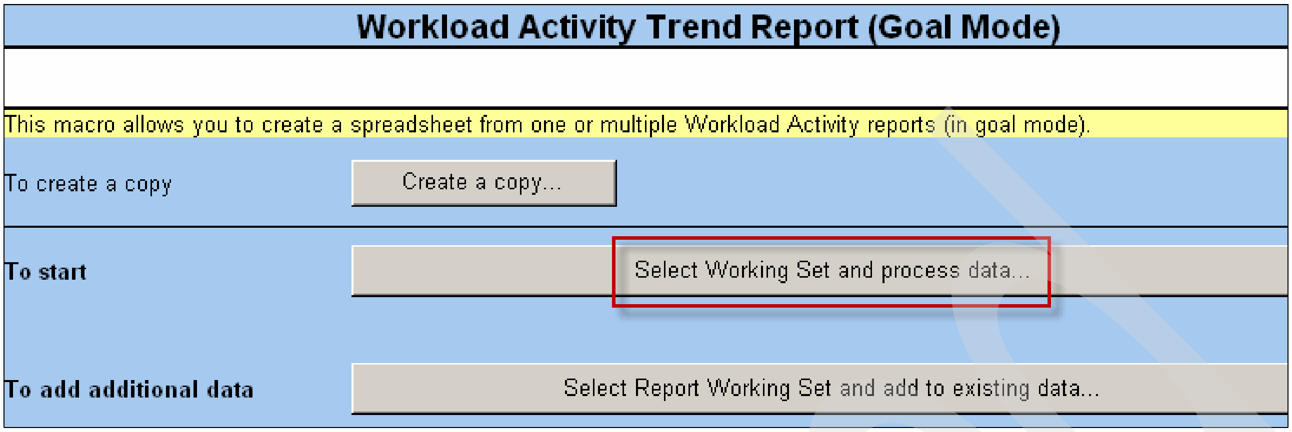


Figure 11-13 Workload activity main panel

Figure 11-14 shows the panel that will allow you to select a working set among the ones available in your system. Here you will find the name you entered under the Name section in the working set creation panel in Figure 11-11 on page 233.

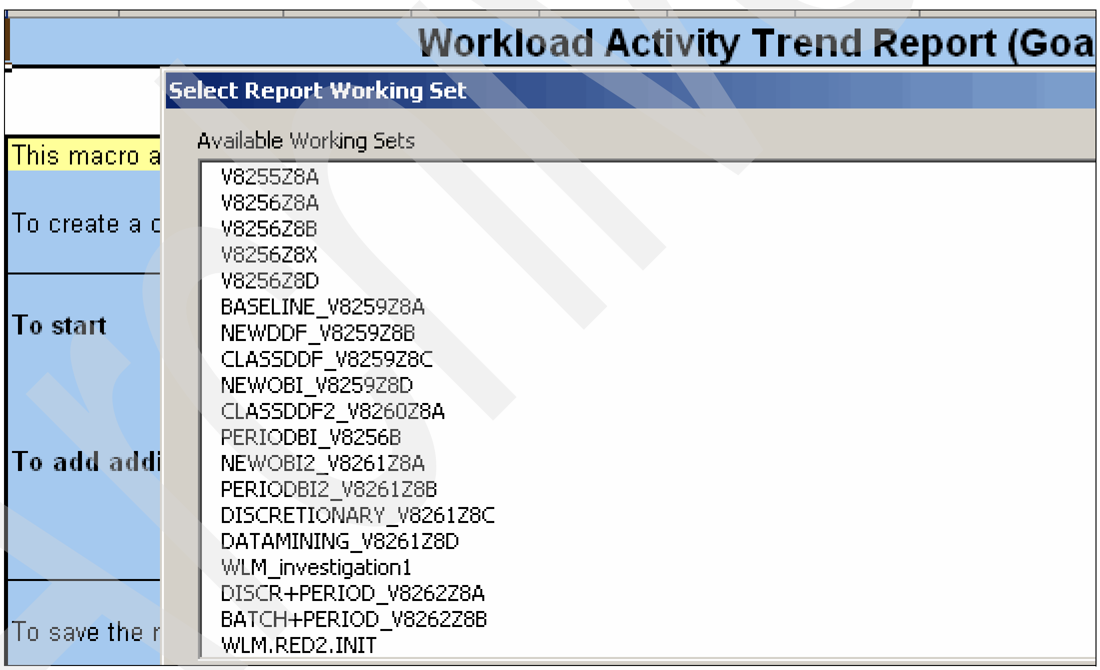


Figure 11-14 Selecting a working set

You will get the option of selecting some reporting periods for analysis. The period duration depends on the setting for the RMF interval of your installation (typically 15 minutes). For the purposes of this book we set this interval to 1 minute. If you choose to do the same for your installation, consider potential performance impacts if you plan this for production. Figure 11-15 on page 235 shows this panel. Normally you select all the periods for reporting.

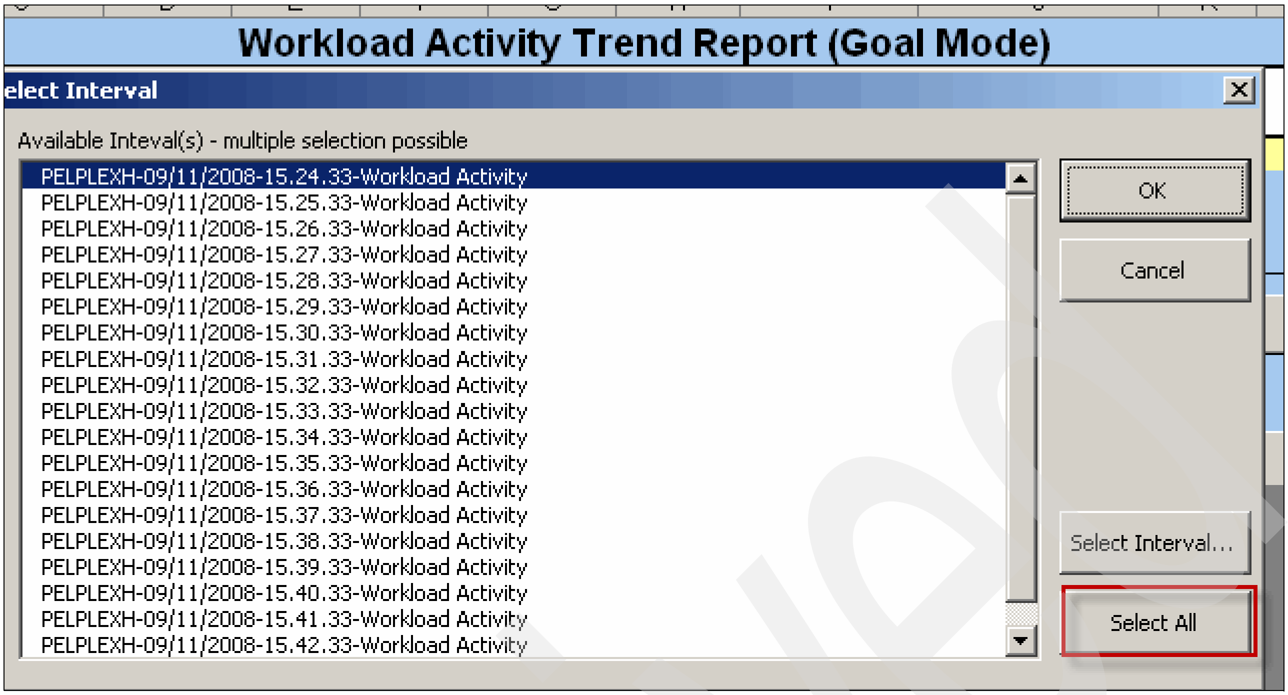


Figure 11-15 Selecting working set reporting periods

Online RMF reports

You can also use the online RMF ISPF interface for getting a view of how the WLM goals are behaving for your system. From the RMF main panel, see Example 11-15, select option 3 MONITOR III.

Example 11-15 RMF main panel

```
RMF - Performance Management                      z/OS V1R9 RMF
 Selection ===> 3

 Enter selection number or command on selection line.


   1 Postprocessor     Postprocessor reports for Monitor I, II, and III      (PP)
   2 Monitor II        Snapshot reporting with Monitor II                    (M2)
   3 Monitor III       Interactive performance analysis with Monitor III     (M3)

   U USER              User-written applications (add your own ...)          (US)

   R RMFPP             Performance analysis with the Spreadsheet Reporter
   P RMF PM            RMF PM Java Edition
   N News              What's new in z/OS V1R9 RMF

                                 T TUTORIAL    X EXIT

   RMF Home Page:      http://www.ibm.com/servers/eserver/zseries/rmf/

          5694-A01 Copyright IBM Corp. 1994, 2007. All Rights Reserved
                     Licensed Materials - Property of IBM
```

Under the RMF MONITOR III primary menu, select the option S SYSPLEX, as shown in Example 11-16.

Example 11-16 RMF Sysplex reports

```
RMF Monitor III Primary Menu                  z/OS V1R9 RMF
 Selection ===> s

 Enter selection number or command on selection line.


   S SYSPLEX        Sysplex reports and Data Index                         (SP)
   1 OVERVIEW       WFEX, SYSINFO, and Detail reports                      (OV)
   2 JOBS           All information about job delays                       (JS)
   3 RESOURCE       Processor, Device, Enqueue, and Storage                (RS)
   4 SUBS           Subsystem information for HSM, JES, and XCF           (SUB)

   U USER           User-written reports (add your own ...)                (US)


                    O OPTIONS    T TUTORIAL    X EXIT

        5694-A01 Copyright IBM Corp. 1986, 2007. All Rights Reserved
                      Licensed Materials - Property of IBM
```

In the next panel, select the option 1 SYSSUM, as in Example 11-17.

Example 11-17 RMF Sysplex performance summary menu

```
RMF Sysplex Report Selection Menu
 Selection ===> 1

 Enter selection number or command for desired report.


  Sysplex Reports
          1 SYSSUM   Sysplex performance summary                 (SUM)
          2 SYSRTD   Response time distribution                  (RTD)
          3 SYSWKM   Work Manager delays                         (WKM)
          4 SYSENQ   Sysplex-wide Enqueue delays                  (ES)
          5 CFOVER   Coupling Facility overview                   (CO)
          6 CFSYS    Coupling Facility systems                    (CS)
          7 CFACT    Coupling Facility activity                   (CA)
          8 CACHSUM  Cache summary                               (CAS)
          9 CACHDET  Cache detail                                (CAD)
         10 RLSSC    VSAM RLS activity by storage class          (RLS)
         11 RLSDS    VSAM RLS activity by data set               (RLD)
         12 RLSLRU   VSAM LRU overview                           (RLL)

  Data Index
          D DSINDEX  Data index                                   (DI)
```

Figure 11-16 show an example of this report. You can get near-real-time information about WLM goals, actual velocities, and performance indexes, among other useful information. Refer to the RMF documentation for more details.

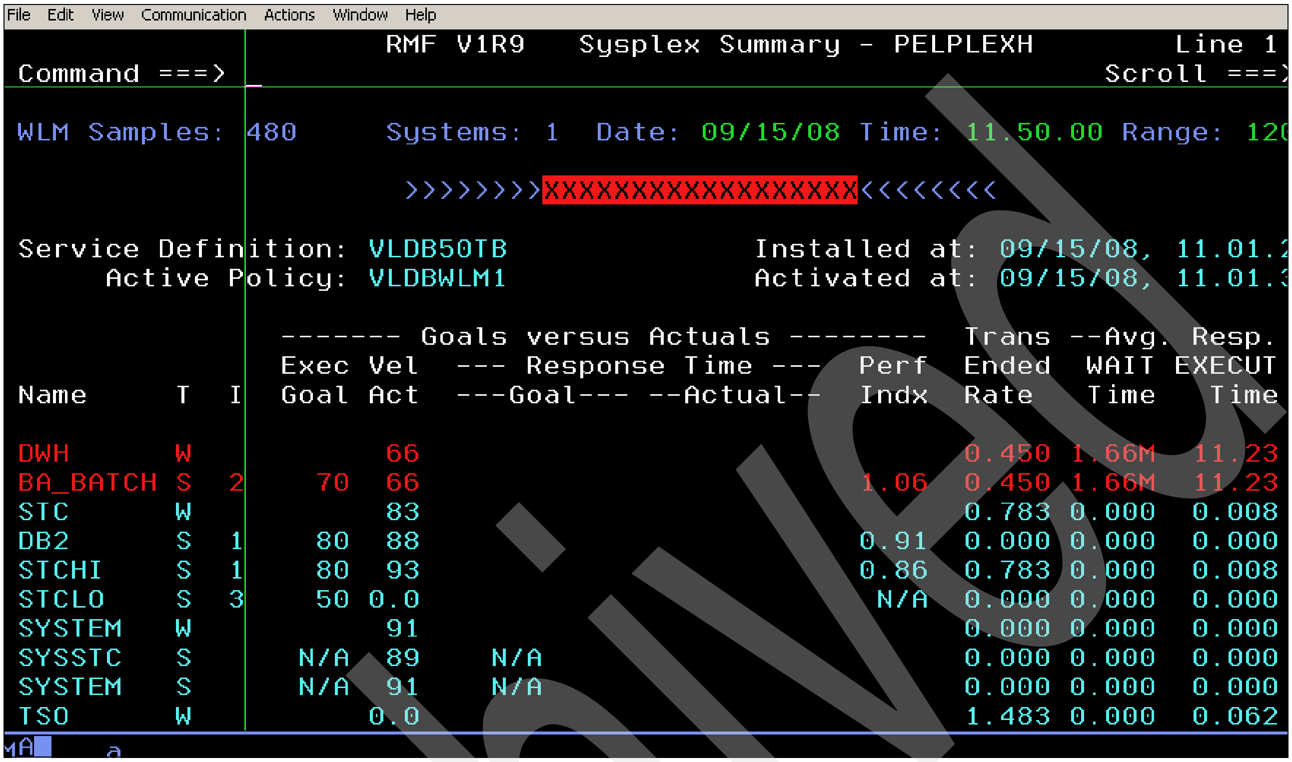


Figure 11-16 RMF sysplex summary

11.3.4 DB2 accounting

The DB2 accounting report examples in this chapter are based on the table DB2PMFACCT_GENERAL of the DB2 Performance Expert Performance Warehouse. For a DW system, where most of the workload activity is done in DB2, this information is useful for analyzing how workloads interact, and to assess the impact of changes on a WLM policy.

> **Important:** You need to consider other workloads in the system, that are not included in the DB2 accounting, to get the complete view of total CPU usage

Figure 11-17 shows the data flow from DB2 accounting to a spreadsheet report as the ones used for the analysis of the examples exposed on this book.

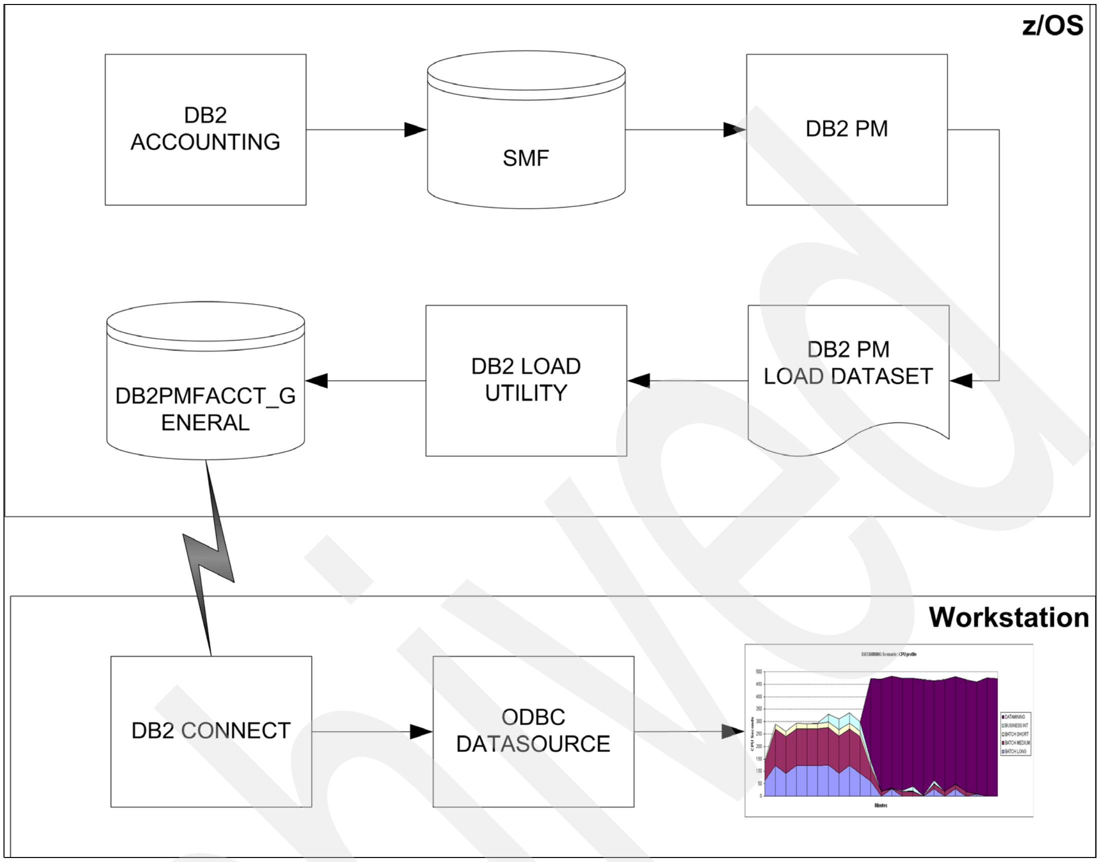


Figure 11-17 Creating CPU reports from the DB2 accounting records

We create a load file from the SMF records using the JCL shown in Example 11-18.

Example 11-18 Creating a load file from SMF using DB2PM

```
//CRISPM   JOB CLASS=A,
//     MSGLEVEL=(1,1),MSGCLASS=H,NOTIFY=&SYSUID
//*-----------------------------------------------------------*
//PEV410  EXEC PGM=DB2PM
//INPUTDD  DD  DSN=VLDB12.SMF.SORTED,DISP=SHR
//ACFILDD1 DD  DSN=VLDB12.DB2PM.ACCLOAD4,
// DISP=(NEW,CATLG,CATLG),
// UNIT=SYSDA,SPACE=(4000,(2000,2000),,,ROUND)
//DPMLOG   DD  SYSOUT=A
//SYSOUT   DD  SYSOUT=A
//SYSIN    DD  *
ACCOUNTING
            REPORT                    /*Example default accounting rpt*/
                     DDNAME(ACRPTDD)
            file
                     DDNAME(ACFILDD1)
EXEC
```

This file is loaded into the table DB2PMFACCT_GENERAL, which is part of the DB2PM performance data warehouse. We used a JCL like the one shown in Example 11-19.

Example 11-19 Loading the DB2PM performance data warehouse

```
//CRISLOAD JOB CLASS=A,
//    MSGLEVEL=(1,1),MSGCLASS=H,NOTIFY=&SYSUID
//*----------------------------------------------------------*
//STEP1 EXEC DSNUPROC,UID='CRIS.LOAD',
// UTPROC='',
// SYSTEM='VLD9'
//SYSREC DD DISP=OLD,DSN=VLDB12.DB2PM.ACCLOAD4
//SYSUT1 DD DSN=&&SYSUT1,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(40,(20,20),,,ROUND)
//SORTOUT DD DSN=&&SORTOUT,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(40,(20,20),,,ROUND)
//SYSIN DD *
 LOAD DATA INDDN(SYSREC)
   RESUME YES  LOG YES
   INTO TABLE VLDB12.DB2PMFACCT_GENERAL
   WHEN (9:9) = ' '
  (DB2PM_REL             POSITION(3) SMALLINT,
......
   CLASS3_PLOCK_PSET     POSITION(1997) DECIMAL
                            DEFAULTIF(CLASS3='N'),
   PLOCK_PAGE_SUSP       POSITION(2005) INTEGER
                            DEFAULTIF(CLASS3='N'),
   CLASS3_PLOCK_PAGE     POSITION(2017) DECIMAL
                            DEFAULTIF(CLASS3='N'),
   PLOCK_OTHER_SUSP      POSITION(2025) INTEGER
                            DEFAULTIF(CLASS3='N'),
   CLASS3_PLOCK_OTHER    POSITION(2037) DECIMAL
                            DEFAULTIF(CLASS3='N'),
   DCL_TEMP_TABLE        POSITION(2177) INTEGER,
   PAR_ONEDB2_DCLTT      POSITION(2181) INTEGER,
   CREATE_SEQUENCE       POSITION(2185) INTEGER,
   ALTER_SEQUENCE        POSITION(2189) INTEGER,
   DROP_SEQUENCE         POSITION(2193) INTEGER,
   DB2V8                 POSITION(323) CHAR(1))
//
```

To provide an easier way of analyzing the test scenarios, we used a view like the one shown in Example 11-20.

Example 11-20 View on accounting DB2 table

```
CREATE VIEW VLDB12.ACC_WLM_SCENARIOS
AS
(
SELECT
       DATE(TIMESTAMP) AS DATE
      ,HOUR(TIMESTAMP) AS HOUR
      ,MINUTE(TIMESTAMP) AS MINUTE
      ,CLASS1_ELAPSED  AS ELAPSED
      ,CLASS2_CPU_TOTAL AS CPU
      ,CORRNAME
      ,SUBSTR(CORRNAME,1,1) AS TYPE
      ,PRIMAUTH
```

```
      ,CASE
          WHEN (SUBSTR(CORRNAME,1,1) = 'L')
            THEN 'BATCH LONG'
          WHEN (SUBSTR(CORRNAME,1,1) = 'M')
......
         ELSE SUBSTR(CORRNAME,1,1)
      END
      AS QUERY_TYPE
     ,CASE
         WHEN (TIMESTAMP >= '2008-09-15-17.10.00.000000'
           AND TIMESTAMP <= '2008-09-15-17.30.00.000000')
           THEN 'BASELINE - V8259Z8A'
......
          WHEN (TIMESTAMP >= '2008-09-19-19.30.00.000000'
            AND TIMESTAMP <= '2008-09-19-20.00.00.000000')
            THEN 'NEW SCEN7A - V8263Z8B'
          ELSE 'NOT QUALIF'
       END
       AS TESTRUN
FROM VLDB12.DB2PMFACCT_GENERAL
)
;
```

You need to catalog the DB2 for z/OS containing the DB2PM performance data warehouse in your workstation using DB2 Connect. During this process, define the DB2 subsystem as a ODBC data source. In this way, you can read the DB2 accounting information directly from the spreadsheet and you can exploit this data for plotting the CPU utilization, number of executions per time unit, etc.

11.4 Verification of a working WLM policy

WLM and its capacity of managing heterogeneous workloads in the same system show most of its value when its definitions reflect the business requirements. One of the most important roles of a DBA in the context of WLM is to be the bridge between the business objectives and the WLM policy.

There are several reasons why you should review your current WLM policy and check it validity with regards of the business expectations. Some of these reasons are listed here:

- Business evolves and objectives can changes with time. An unimportant workload may become critical and vice versa.
- Infrastructure evolves and changes on number of CPUs, for instance, can have an impact on how goals are achieved needed a service class review.
- Workload increases. The behavior of a system changes with utilization. Almost any WLM policy would work in a system running with a lot of spare capacity. WLM shows most of its value during busy moments, but if workload increases and your machine utilization is higher, you may need to review the WLM definitions.
- Business integration. The fusion of applications or business into the same LPAR would most probably justify a review of the current WLM definitions.
- New software. The addition of new software into a partition generally comes with specific requirements in terms of resources availability and workload priority.
- WLM evolves. New version of WLM an z/OS incorporate more capabilities into WLM that you may be interested on explore and exploit.

You can follow the following steps as a guide for the review of your current WLM policy. You should plan for a periodic review of the WLM policy against business SLAs.

> **Important:** The review of the WLM policy and the definition of goals and its attainment should be a continous process. You may consider to verify how WLM is working in your environment on a regular basis.

Figure 11-18 on page 242 shows the steps that you may use as a guide for the WLM policy review process. The involved steps can be described as follows:

1. Start process.
2. Understand business requirements: Requirements can be formally documented in for or SLAs or you may need to meet the people that holds the knowledge of the business requirements in terms of importance. Inevitably all application is 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 doesn’t match the business requirements, you need to update these definitions. A change need 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: monitor of the goals achievement 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 application’s expectations, you have a well tuned system. You may plan for a recurrent review of this process, starting by reviewing if the business requirement have evolved.
8. If the goals are not being achieved, you may be facing a capacity problem: WLM cannot create extra CPU 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, as a new hardware configuration, for instance increase on CPU capacity but with fewer processors, can introduce changes on how the resources can be distributed among tasks.
10. Once service classes are in line with the business requirements and the goals are achieved, you just need to plan for a periodic review of this definition process.

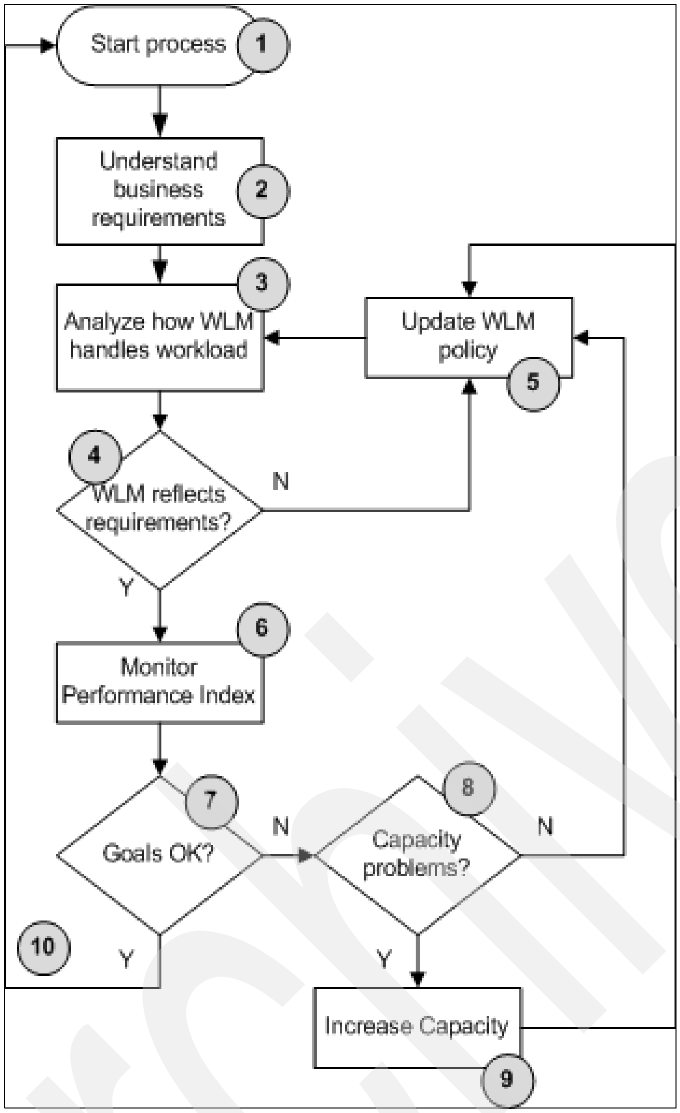


Figure 11-18 WLM policy review process

11.4.1 Example of understanding business requirements

As an example of WLM managing an application, let us consider a data warehouse system composed of a dedicated z/OS partition exploiting DB2 for z/OS as the data store. Table 11-5 summarizes the relevant workload and business requirements for this DW partition:

Table 11-5 Workload classification example: The business view

Importance	Workload	Details	Qualifier
Highest	Cognos critical user IDs	All the workload created by three specific user IDs is critical for the business and must receive the highest priority	The users are known: UNPAPC01, UVECPC02 and UDWHP2Q
High	Cognos reporting	Business intelligence	Cognos technical user ID
Medium	Other DDF reporting tools	Some queries can be resource intensive but should be prevented of impacting the more important DDF reporting	SqlView and MSACCESS

Importance	Workload	Details	Qualifier
Medium	Batch replication process	The Operational Data Store is ingested with replicated data from the production system in batch. Some reports have dependencies on the end of these process and they are executed during the business hours at regular intervals	Batch CORRID is identified: DP*
Lowest	Other batch	Miscellanies batch jobs	
Not relevant	Ad-how queries	Development is taking place sometimes in the partition	

Figure 11-19 shows the daily CPU profile for this data warehouse system. This graph was built using the DB2 accounting information.

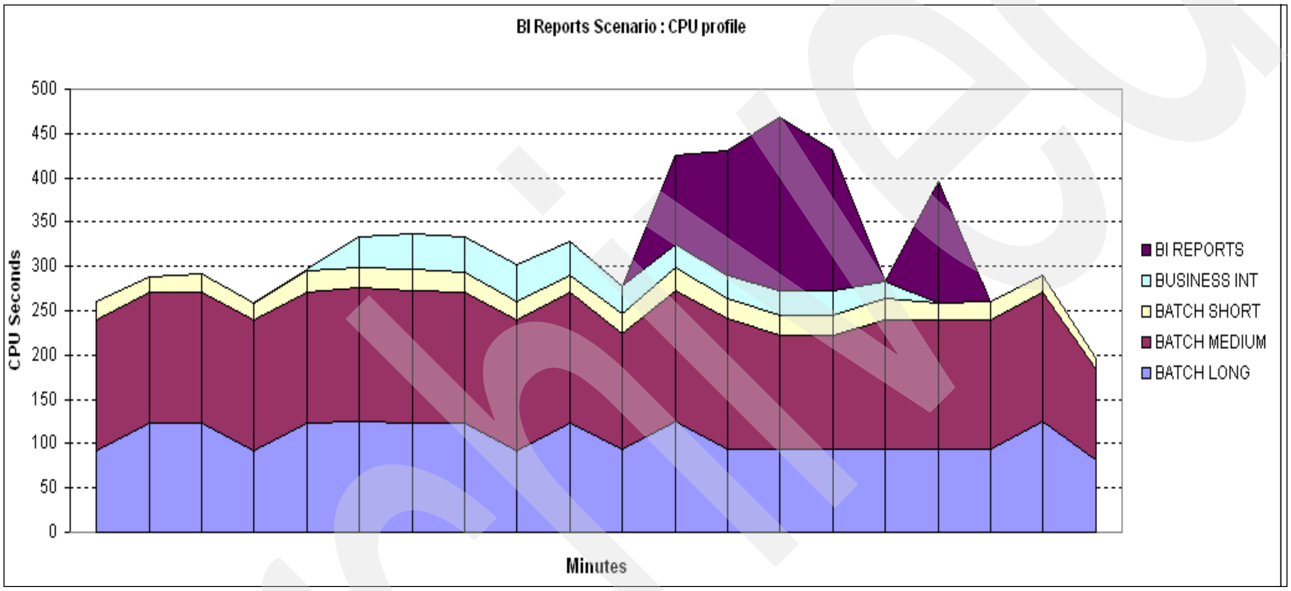


Figure 11-19 Example of daily CPU profile

> **Important:** Consider other workloads on top of the DB2 activity if present on the partition (for instance CICS, IMS, and so forth).

Example of exploring the WLM definitions in use

You may find it convenient to explore the current WLM policy definitions in a user friendly format to make the analysis easier. You can use one of the methods described in the previous sections (for instance the WLM policy formatter or the WLM policy editor).

Example 11-21 on page 244 shows the first part of a printed WLM policy. From this part of the document you can get a summary of what the policy contains in terms of WLM definitions.

Example 11-21 Service definition print example: summary

```
1 * Service Definition PRODZ180 - PROD z180 (version 555-555-01)

    8 workloads, with 36 service classes
    5 resource groups
    2 service policies
    14 classification groups
    12 subsystem types
    932 report classes
    256 application environments
    22 scheduling environments
    23 resources
...

    I/O priority management           Yes
    Dynamic alias tuning management   No
```

During your review of the definition, check for good practices. For instance:

- Validate that not too many service classes are active at the same time
- Ensure that there are defaults defined for all the workloads
- Check that the system uses reporting classes efficiently to enable helpful reports.

Refer to IBM Redbooks publication *System Programmer's Guide to: Workload Manager*, SG24-6472 for more guidelines.

Table 11-6 summarizes the most relevant service classes for this sample policy.

Table 11-6 Review WLM policy: Relevant service classes

Workload	Service Class	Period	Importance	Goal
DDF	SC600	1	2	Velocity 10
DDF	SC650	1	3	Velocity 20
DDF	SC251L	1	2	Velocity 45
JES	SC600	1	2	Velocity 10
TSO	SC300L	1	1	95% complete within 0.3 s
TSO	SC300L	2	1	95% complete within 1.0 s
TSO	SC300L	3	2	95% complete within 15.0 s
TSO	SC300L	-	-	Class assigned to resource group DWHLIMTD

The next step is to verify how the business requirements are being supported by WLM. Is the current classification matching the business expectations? How are the goals for the important workload?

Example 11-22 on page 245 shows an extract of the classification rules sections. In this example there is no default service class or report class for the Subsystem Type DB2 workload. This is not a good practice and should be changed to provide a default service class. Unclassified workload will use the SYSOTHER service class, which has a discretionary goal. This may not be in line with the requirements of the business. You should check that the service class by default does not have a too aggressive goal or importance defined, as unexpected work may take resources from the system affecting more important work. You may define a by-default reporting class that is not used by any other work and you may decide to inspect this utilization to classify the workload in a specific service and reporting class.

In this example the DDF workload has the service class SC650 defined as the default and DDFXXXXX is the default reporting class.

Example 11-22 Service definition print example: Classification section

```
* Subsystem Type DB2 - Use Modify to enter YOUR rules

        Classification:

          There is no default service class.
          There is no default report class.
.....
        ================================================================

      * Subsystem Type DDF - DDF queries

        Classification:

          Default service class is SC650
          Default report class is DDFXXXXX
```

You need a way to identify in which service class the work is executed. You can use, for instance, the following sources of information:

- Your DB2 activity monitor should provide the service class on which a workload is executed. This information is available online
- The DB2 accounting field QWAWLME. This field provides the WLM service class. This information is normally stored in SMF. Your installation may have a performance data warehouse where this information is ready available
- For batch, this information is provided in the JESLOG, as shown in Example 11-23.

Example 11-23 Service class in JES job log

```
   J E S 2  J O B  L O G  --  S Y S T E M X X X X --  N O D E  Y Y Y Y Y Y

10.44.33 JOB28971 ---- SATURDAY,  11 OCT 2008 ----
10.44.33 JOB28971  IRR010I  USERID XXXX IS ASSIGNED TO THIS JOB.
10.44.38 JOB28971  ICH70001I XXXX LAST ACCESS AT 10:44:13 ON SATURDAY, OCTOBER 11, 2008
10.44.38 JOB28971  $HASP373 DPD0035  STARTED - WLM INIT  - SRVCLASS SC600    - SYS XXXX
10.44.38 JOB28971  IEF403I DPD0035 - STARTED - TIME=10.44.38
11.28.51 JOB28971  -                                        --TIMINGS (MINS.)--
----PAGING 11.28.51 JOB28971  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  CLOCK
SERV  PG   PAGE 11.28.51 JOB28971  -DPD0035  STP010   DB2STEP     00    489   5.93    .00
44.21 70189K   0      0 11.28.51 JOB28971  IEF404I DPD0035 - ENDED - TIME=11.28.51
11.28.51 JOB28971  -DPD0035  ENDED.  NAME-                    TOTAL CPU TIME=  5.93  TOTAL
ELAPSED 11.28.51 JOB28971  $HASP395 DPD0035  ENDED
------ JES2 JOB STATISTICS ------
```

Use the WLM classification rules to evaluate on which service class a workload would be executed. This process is useful when you are updating the classification rules of a WLM policy to confirm if WLM would classify workloads as expected. A quick and easy way of implementing this procedure is by using the RMF Service Definition Formatter. This tool allows you to discover which default service class will be used for a certain workload.

Figure 11-20 shows a view of this section of the tool. You fill the columns Subsystem, Rule, and Value for at least one rule. Up to four levels are supported.

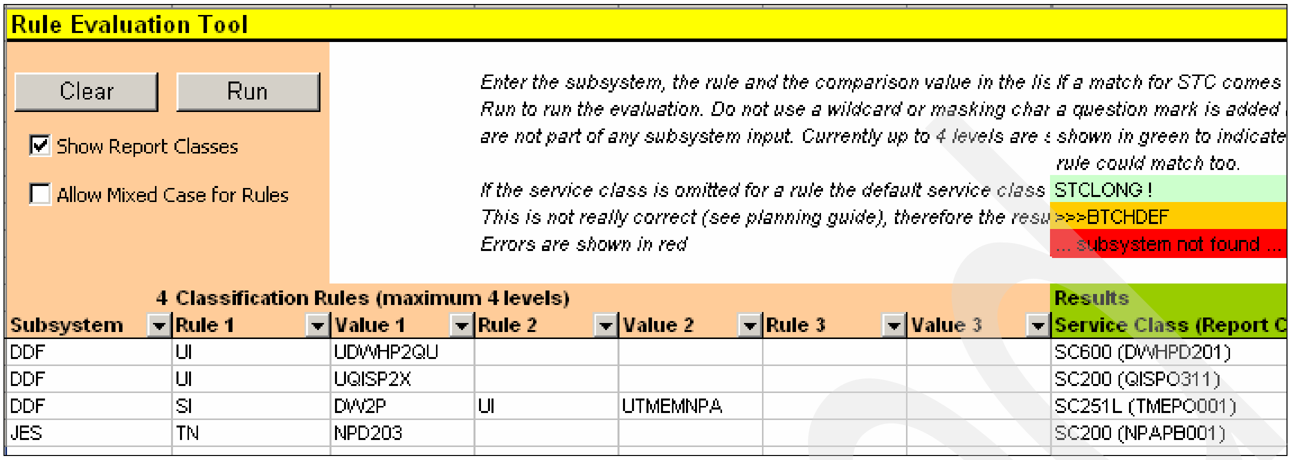


Figure 11-20 Using the Rule evaluation tool of the Service Definition Formatter

Remember that Table 11-7 on page 247 showed some of the IBM-defined subsystem types used in the classification rules. Table 11-3 on page 217 showed an extract of the work qualifiers. Not all work qualifiers are valid for every subsystem type. They are subsystem dependent. For details on which qualifiers are valid for which subsystems, and a full list of qualifiers see *MVS Planning: Workload Management*, SA22-7602.

It is important to remember that the classification rules can be composed of more than one criteria. Example 11-24 shows a portion of classification rules for this example policy.

Example 11-24 WLM Service definition extract: DDF classification rules

```
* Subsystem Type DDF - DDF queries

        Classification:

          Default service class is SC650
          Default report class is DDFXXXXX

          Qualifier  Qualifier      Starting       Service  Report
          # type       name           position       Class    Class
          - ---------- -------------- ---------      -------- --------
          1 UI         UDWHP2QU                      SC600    accountingPD201
          1 UI         UQISP2X                       SC200    QISP0311
          1 SI         DW2P                          SC650    DDFXXXX1
          2 . UI       . UTMEMNPA                    SC251L   TCCP0001
....
```

Table 11-7 shows an example of how you could summarize the information provided by the business requirements and the current service classes and classification rules.

Table 11-7 Business importance and workload classification summary

Importance	Workload	Details	Service class
Highest	Cognos critical user IDs	All the workload created by three specific user IDs is critical for the business and must receive the highest priority	SC600
High	Cognos reporting	Business intelligence	SC600, SC650 and SC251L
Medium	Batch replication process	Some reports have dependencies on the end of these process. They are executed during the business hours	SC600
Medium	Other reporting	Some queries can be resource intensive	SC650
Low	Other tools	Includes change management done on some objects during business hours. These process are controlled and monitored by the DBAs and can be resource intensive	SC300
Low	Other batch		SC600
Low	Ad-hoc queries	Development is taking place sometimes in the partition	SC650

The analysis of this table shows some indications that a review is needed. For instance, the most important work is sharing the same service class (SC600) as the less important tasks. A new classification is needed, and eventually new service classes may be required.

You can exploit the RMF spreadsheet reporter to illustrate how goals are being achieved. For instance, Figure 11-21 shows the goals for the service class SC600 during the day. This service class is shared by several workloads, including a mix of important and unimportant workloads.

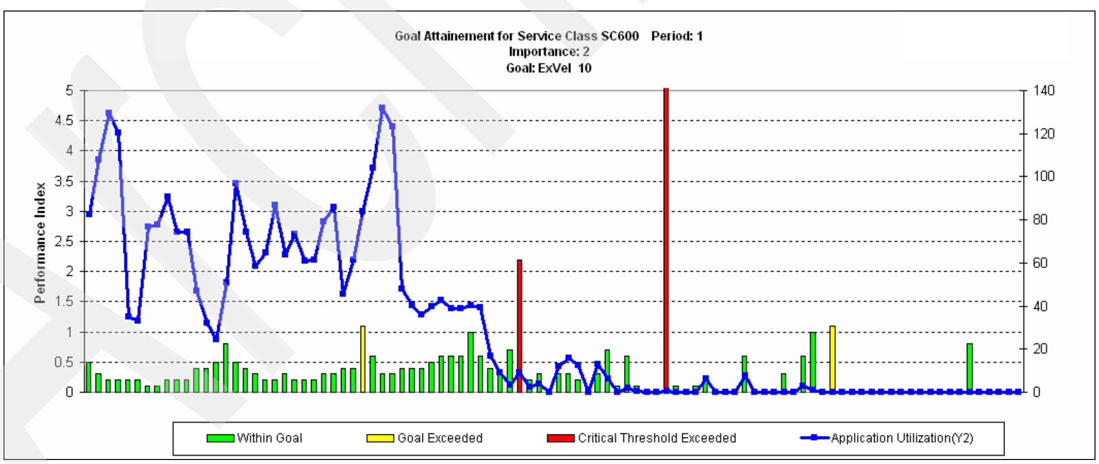


Figure 11-21 Goal achievement for service class SC600

Figure 11-22 shows the goals for the service class SC650. The goal achievement for this service class is rather low during the morning. This will affect the Cognos reporting classified on it.

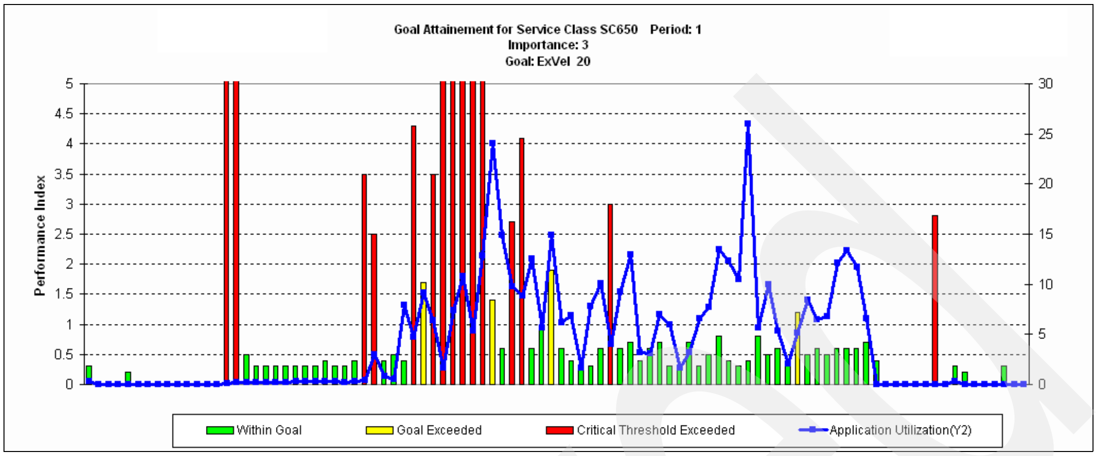


Figure 11-22 Goal achievement for service class SC650

If you plan a WLM policy for busy hours, you might eventually end up with two policies: one for the night and other for the day, if the requirements are clearly different. The RMF spreadsheet reporter allows plotting all the active service classes goals for a specific period of the day. Figure 11-23 shows such a report for the middle of the morning in a typical day. Note that the service class SC600, where the most critical work is classified, is achieving its goals. It may explain why the users do not complain. The important work is going well, but the policy may be adapted to distribute better the available resources and delay a capacity upgrade, for instance. Note that the service class SC650, where the Cognos reporting are qualified, is clearly missing its goals.

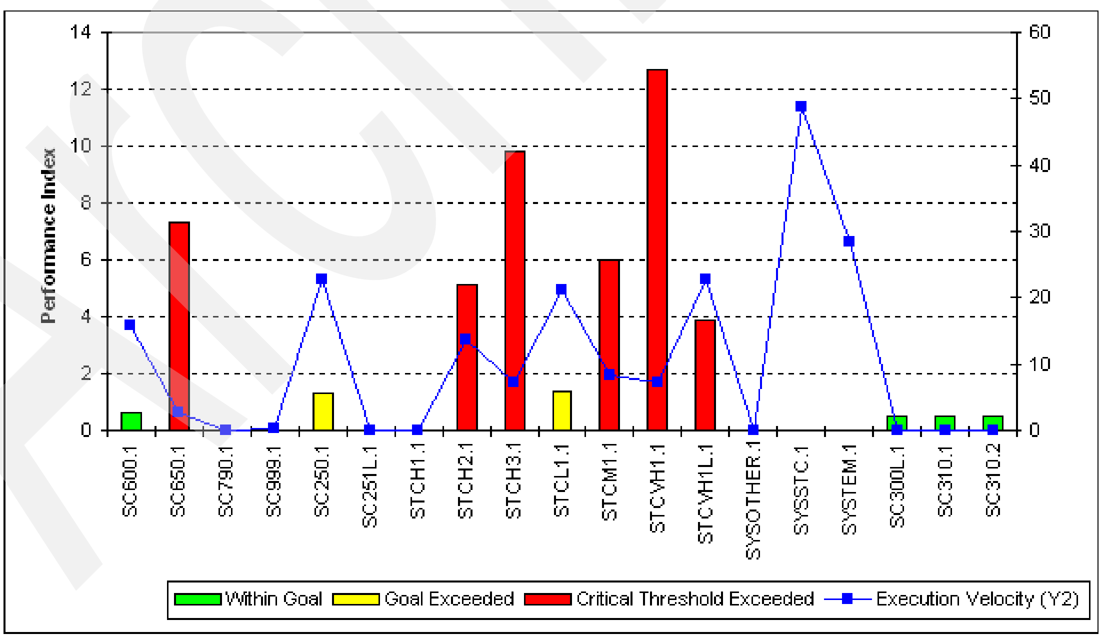


Figure 11-23 Performance index for busy period

11.5 Analysis and implementation examples

To illustrate the WLM components and tools described on this chapter, several examples of analysis and implementation were executed. They are summarized in Table 11-8.

Table 11-8 Summary of implementation example

Scenario	Description	Value
Mixed batch	Initial workload representing a traditional batch Business Analysis workload, including short, medium and long queries some of which exploit parallelism. The SLA is not aggressive	This is a baseline workload. This example is used for showing the monitoring and analysis tools that you can use for following up your DW environment from a WLM point of view
New DDF	A new Cognos application is added into the system. This application is resource intensive[a] and no WLM preparation was done before its implementation. The batch workload is impacted with service degradation	Show impact of new workload when there is not adequate WLM preparation
Classify DDF	To protect the more important batch workload, the DDF Cognos application is classified in a less important service class	Shows how to classify workload in a proper Service Class. Explore the value of classification rules for protecting important workload.
New OBI	New Operational BI workload is introduced in the system using Cognos. This workload is OLTP like and the response time observed is not acceptable.	Show how OBI processing, an OLTP like workload, would require WLM adjustments in a traditional DW environment.
Period goals	Introduction of a period goal for the Cognos workload. This change provides acceptable response times to the OBI users	Show the value of Period goals in a DW environment
New Data Mining	Without WLM preparation, a Data Mining batch process, resource intensive, is added to the system affecting the service provided to the other workloads in the DW.	Exhibit what happens with large consumption work in system without prioritization
Discretionary	The Data Mining processing doesn't have fast response time requirements and a Discretionary period is added to the WLM definitions for this process.	Exhibit how WLM Discretionary goal can be used to efficiently utilize spare cycles for work that doesn't have immediate RT requirements

a. This particular Cognos application consisted of large resource consumption analytics.

These examples are not isolated. They can be seen as the evolution of a DW from a traditional batch workload to an Operational BI system. These examples are intended to show you the value of WLM for administering heterogeneous workloads in the same partition and to provide practical examples that you could use in your own system.

Our test system is described in Chapter 4, "Configuration" on page 57.

11.5.1 Mixed batch workload

This scenario represents a baseline DW workload. The workload in the partition for this scenario is composed of batch queries representing a traditional DW. The queries can be classified as:

- Short queries
- Medium queries
- Long queries

These queries are implemented as dynamic SQL and the system is driven to a utilization of about 60% in CPU terms. The RMF interval was set to one minute to obtain a small interval for accurate analysis of the system response to WLM changes. There is no DDF activity on the system.

The batch queries are classified by WLM into the service class BA_BATCH. Table 11-9 shows the most relevant WLM definitions for this scenario.

Table 11-9 Relevant WLM definitions during mxed batch workload scenario

Workload	Service Class	Importance	Goal
DDF	DDF	1	Velocity 80
DWH	BA_BATCH	2	Velocity 60

Note that even is there is no DDF workload in the system, a service class called DDF is defined. This service class is more important than the service class for the batch queries in WLM terms. Its importance is equal to 1, while the service class BA_BATCH was defined with importance equal to 2.

As a suggested good practice, the system will not use the default SYSOTHER service class for unclassified DDF. A new incoming DDF workload will be classified to the DDF service class. For improved clarity, it would be better to use a service class named in another way than the workload, DDF for both in this example.

During this test scenario, all the batch queries were running under the same service class and therefore no priority was given to any of the queries. All of them were achieving their objectives. It could happen that short queries would be more important than medium or long queries and it could be necessary to reclassify short queries to a more important service class.

Observations

Figure 11-24 shows the CPU profile for the test as provided by the DB2 accounting traces. This figure shows the CPU distribution for Long, Medium and Short queries. We used these reports during the tests to obtain a DB2 view of resources allocation.

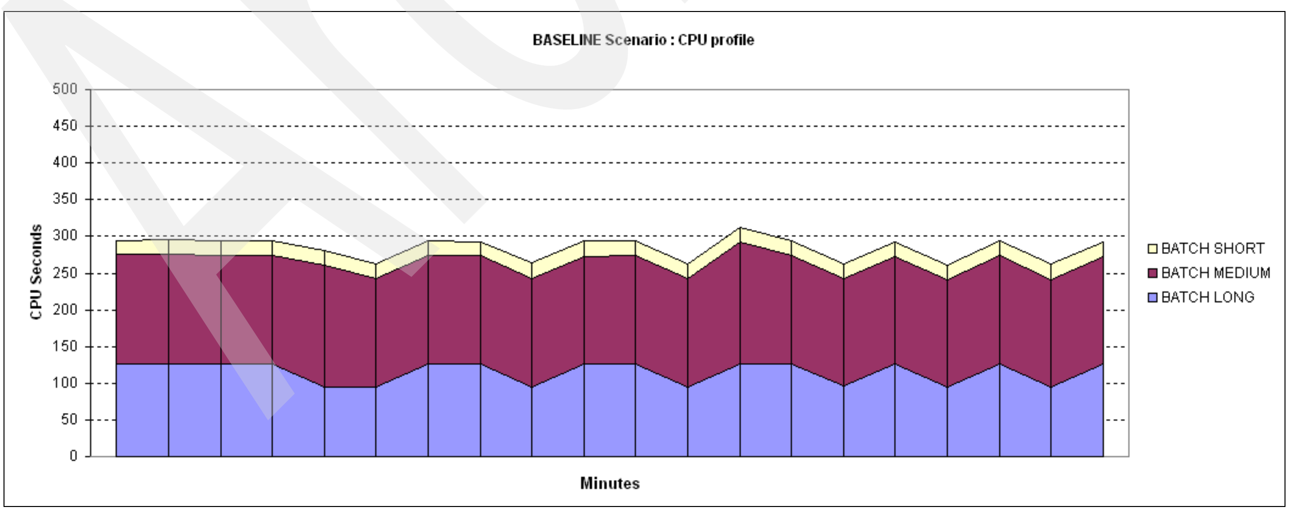


Figure 11-24 Baseline scenario: DB2 accounting CPU profile

Figure 11-25 shows the LPAR utilization during the test. This report was created with the RMF spreadsheet reporter. The machine utilization, in terms of CPU, was stable around 70% during all the duration of the test.

It is important to mention that almost any WLM policy will work well when the machine is not busy. The biggest value of WLM comes into play during busy periods. You should plan your WLM policy design for the busier hours.

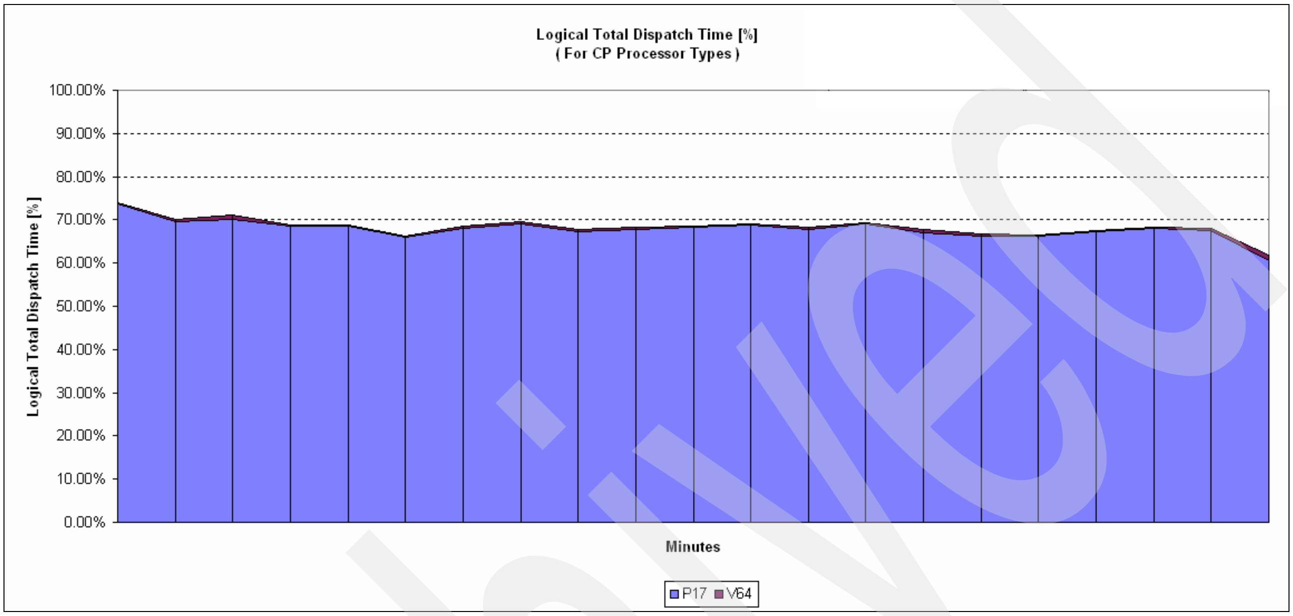


Figure 11-25 Baseline scenario: LPAR dispatch time

Figure 11-26 on page 252 shows a RMF report containing the goal attainments for the BA_BATCH service class on which the batch queries were classified. The goals for this service class were defined as execution velocity of 60%, meaning that this workload was expected to receive resource for 60% of the time. This graph shows that the execution velocity was a little lower than 70% during the execution of the test. The graph shows also the performance index, which was always <1, indicating that the objective was exceeded during all the execution of the test. As described above, we used a RMF interval of 1 minute to get a granular view of the tests, but you may find a bigger interval in your installation. This report shows the service class goal definition during the observed period.

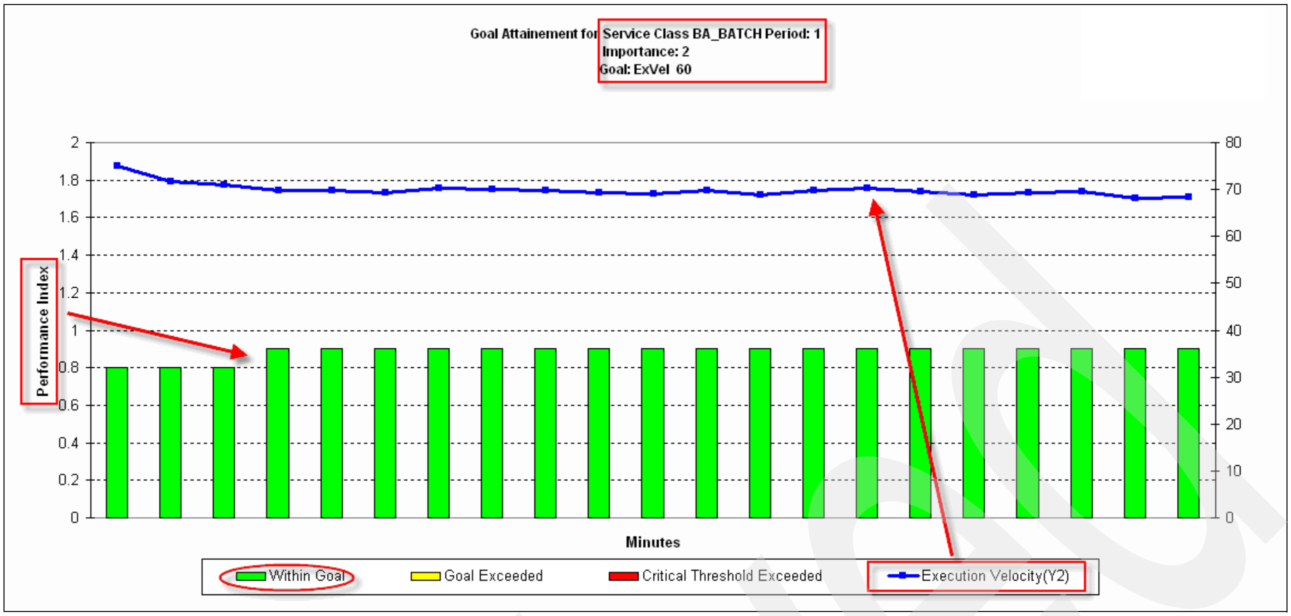


Figure 11-26 Baseline scenario: Goals

11.5.2 Introduction of new DDF workload

The objectives of this scenario are:

- Show the potential impact on existing workload of a new workload when there is no adequate WLM preparation.
- Show the added value of report classes for workload analysis.

Description

This scenario represents a batch DW system on which a new DDF workload is injected. This could be a new Cognos application being deployed during business hours without any specific WLM preparation. In this case, the Cognos application consisted of large resource consumption analytics. On this system, the DDF workload is classified into a higher priority service class than the existing batch queries.

Table 11-10 shows the relevant service classes that were defined in the system during the execution of this scenario. Note that all the DDF workload is classified in the DDF service class. If you do not define a by-default service class for DDF work, it will be classified as SYSOTHER, which is a discretionary service class.

Table 11-10 New DDF scenario: Relevant WLM service classes definitions

Workload	Service Class	Importance	Goal
DDF	DDF	1	Velocity 80
DWH	BA_BATCH	2	Velocity 60

Because two different workloads will be present in the system and will need different analyses, report classes were introduced to provide a better reporting. These report classes are described in the Table 11-11 on page 253. The report class DWH is used for reporting the batch queries, while the report class DDFRPT will report the DDF workload. The classification rules were updated to classify the appropriate workloads into the desired report classes.

Table 11-11 New DDF scenario: Relevant WLM report classes definitions

Reporting class	Qualifying
DWH	Mix batch workload
DDFRPT	DDF workload

The test scenario was executed in the following phases:

- The baseline workload was executed for several minutes
- A resource-intensive Cognos report was introduced in the system. This reporting activity was executed for approximately five minutes
- After the end of the Cognos activity, the system continued to be used for baseline batch queries

Observations

Figure 11-27 shows the CPU profile created from the DB2 accounting for this scenario. The three stages of the test can be easily identified. The value of this graph is to identify that the DDF workload was getting most of the resources during its execution. The batch queries were greatly impacted during the Cognos reporting period. When the Cognos report ended, the baseline workload continued to develop normally. The machine on which the tests were done was an 8 processors unit. You can deduce that the computer was at almost 100% of utilization and that almost all the workload was on DB2 from this graph: 8 CPUs can deliver at most 480 CPU seconds per minute (8 CPUs times 60 seconds/minute) and the graph shows nearly this value during the DDF activity.

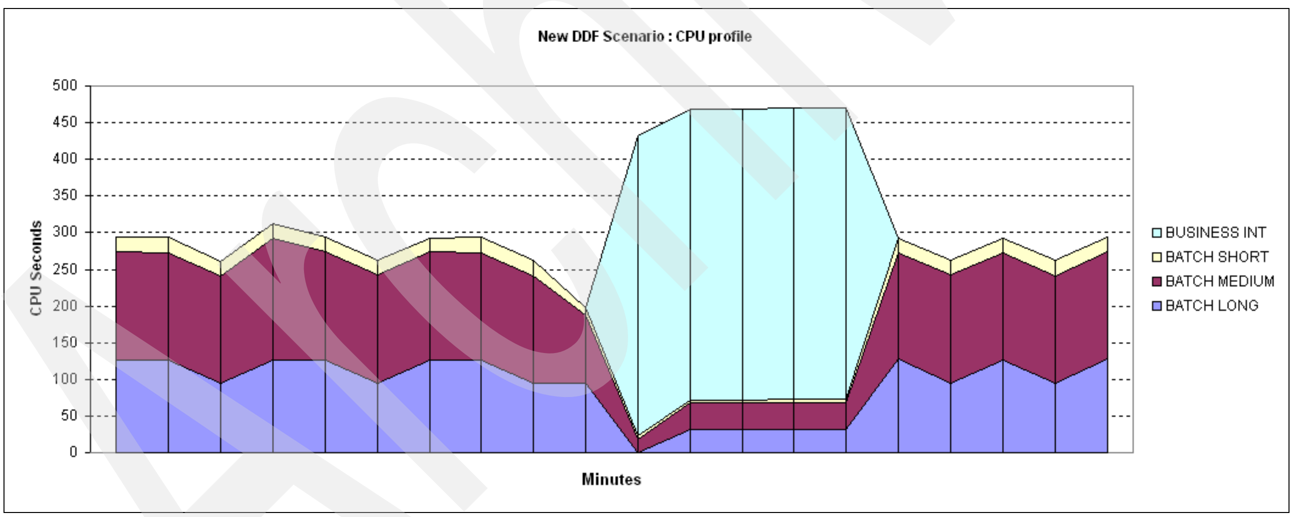


Figure 11-27 New DDF scenario: DB2 accounting CPU profile

It is possible to graph the number of execution of batch queries during the test based on the DB2 accounting. This graph will show that the number of executions was dramatically reduced during the busy period. A long-running batch executing on the BA_BATCH service class will report high Class 3 not accounted values during the DDF execution.

Figure 11-28 on page 254 shows the RMF report of service times for the WLM policy during the test scenario. It shows a huge increase in system utilization during the period on which the Cognos report was active.

As this report contains the service times consumed by all the service classes in the policy, a more detailed view would provide a better understanding of the impact of the new DDF workload.

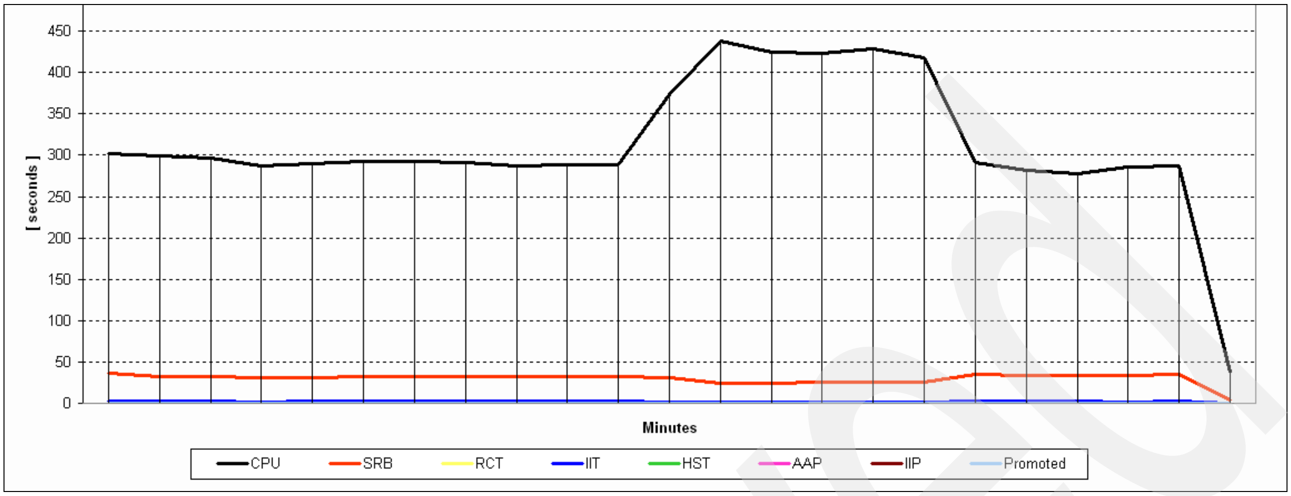


Figure 11-28   New DDF scenario: Policy service times

Figure 11-29 shows the service times for the workload DWH, which is used for grouping the batch queries. Figure 11-31 on page 255 shows the DDF workload, which contains the DDF Cognos report activity.

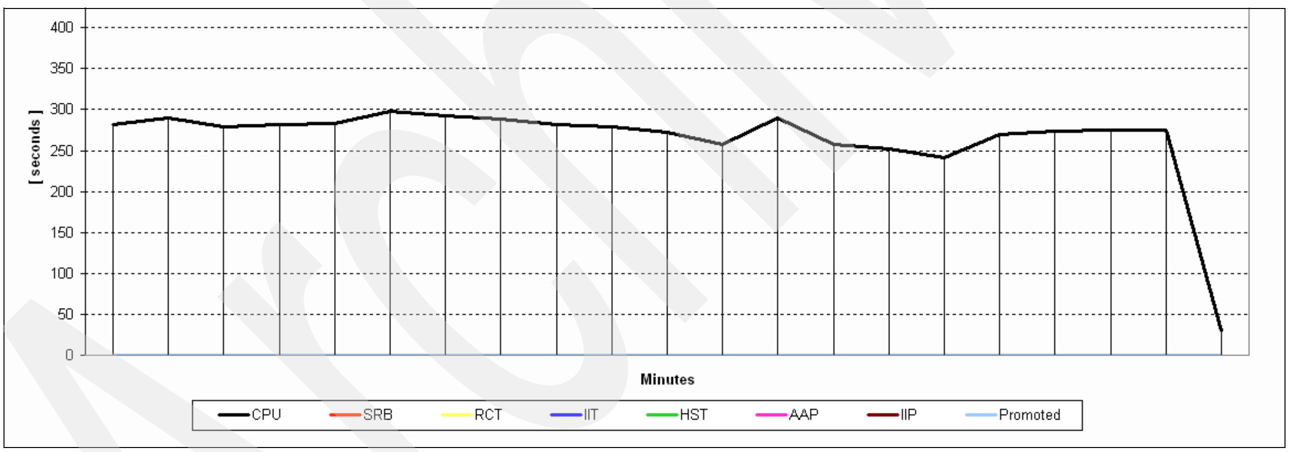


Figure 11-29   New DDF scenario: Service times for the DWH workload

Both graphs provide a better understanding of the activity for each workload. The same reports could be done based on the reporting classes that were defined for this test scenario.

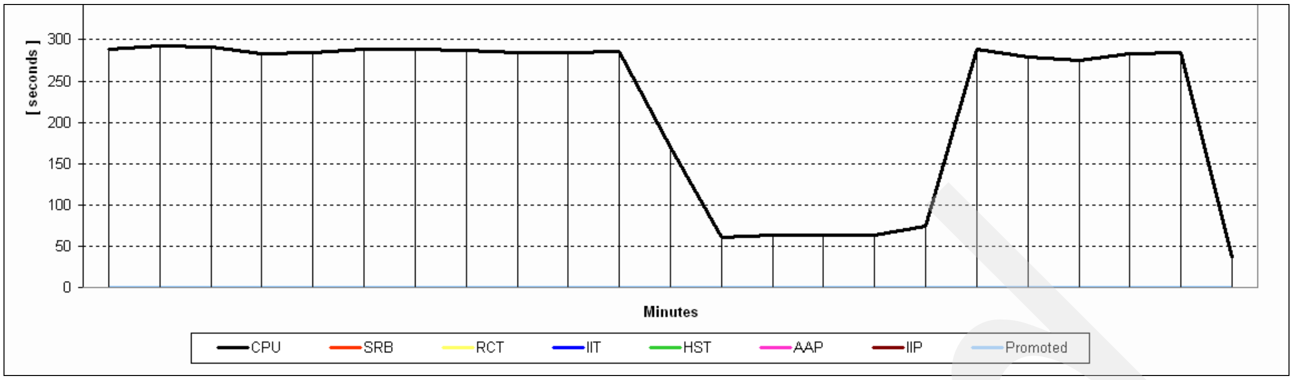


Figure 11-30 New DDF scenario: Service times for the DDF workload

These reports are intended to show that you may study a performance problems from several information sources, such as the RMF reporting or the DB2 accounting.

It is clear that the Cognos reporting affected the delivery of work done by the batch queries. This is translated in WLM terms as goals not being achieved. It is possible to observe this at execution time. You can use the RMF ISPF interface, as shown in Figure 11-31.

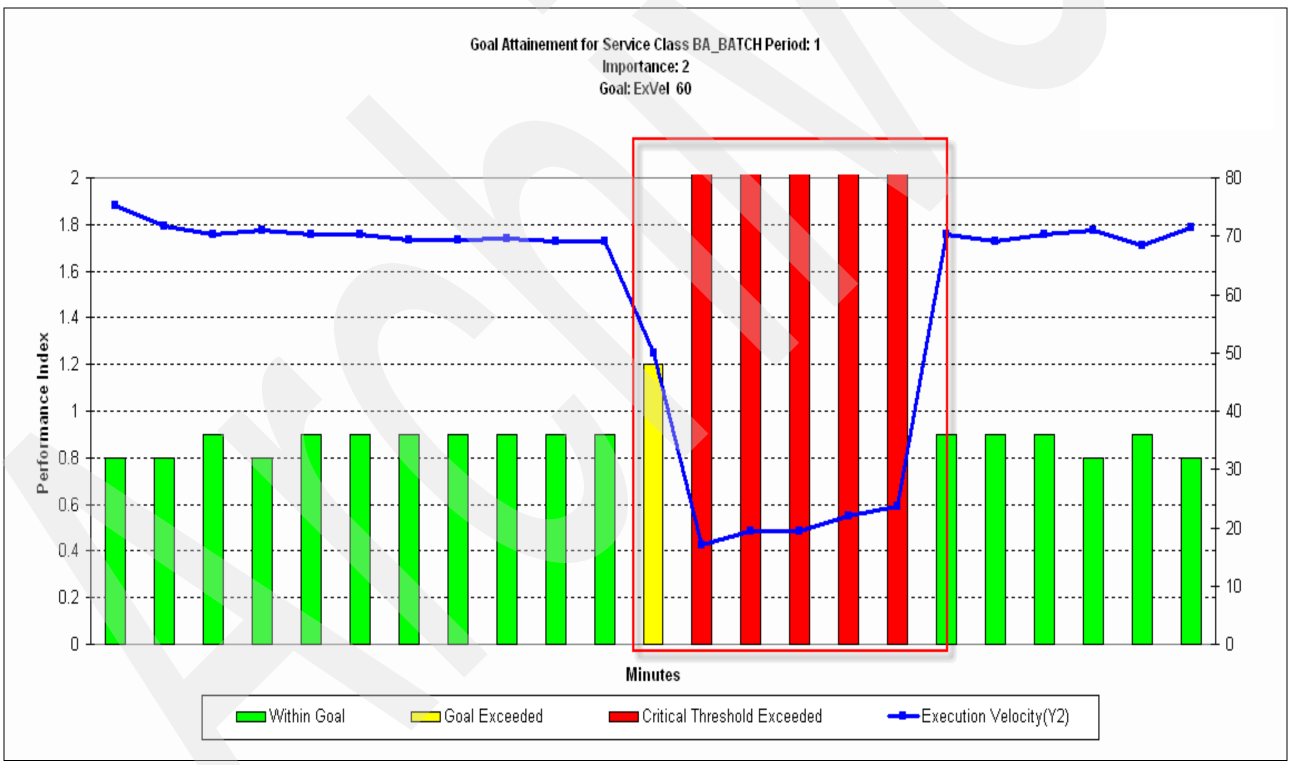


Figure 11-31 New DDF workload: Real time RMF monitoring

This panel shows the goal definitions for the service classes DDF and BA_BATCH and the actual execution velocity. This panel also reports the performance index for both service classes. Figure 11-31 shows that the service class DDF is achieving a performance index of 1, while the service class BA_BACTH is missing it goals with a performance index of 3.7.

It is possible to change the service class for a batch process manually at execution time from SDSF or by the use of the RESET MVS command, for instance. If you decide to do so to solve an emergency situation and provide resources to a specific process to unblock a production problem, you can use this panel to observe the effects in near real-time.

Figure 11-32 shows the goals for the service class BA_BATCH during the test. The box in the graph represents the period in which the Cognos reports were executed. This graph was produced from RMF data. The scale for the performance index was fixed to a value that increased visibility but was lower than the values achieved during the busy period. The execution velocity was dramatically impacted during this period. This graph shows how WLM can report the fact that the batch queries are not achieving their objectives when a more important workload (and resource intensive) is present in the system.

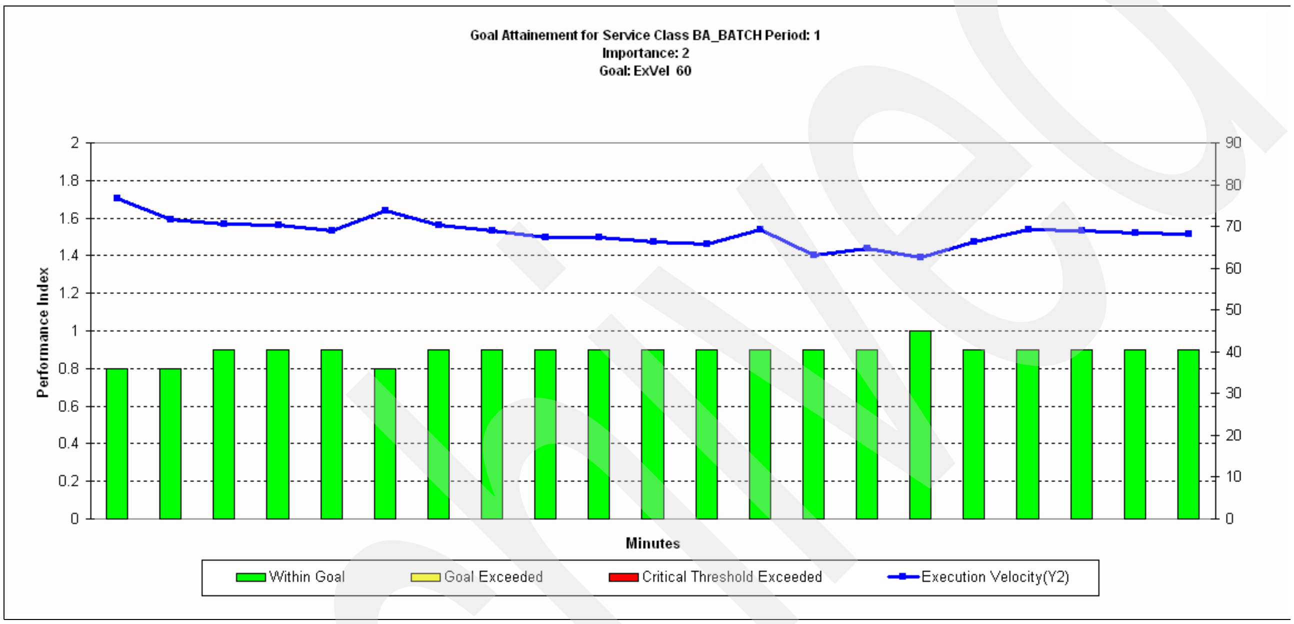


Figure 11-32 New DDF workload: Goals for service class BA_BATCH

Figure 11-33 on page 257 shows the goals for the service class DDF. It is evident that, being a more important work in WLM terms, it has received the needed resources to achieve it goals. The graph shows the resource utilization during the test and the performance index for this service class.

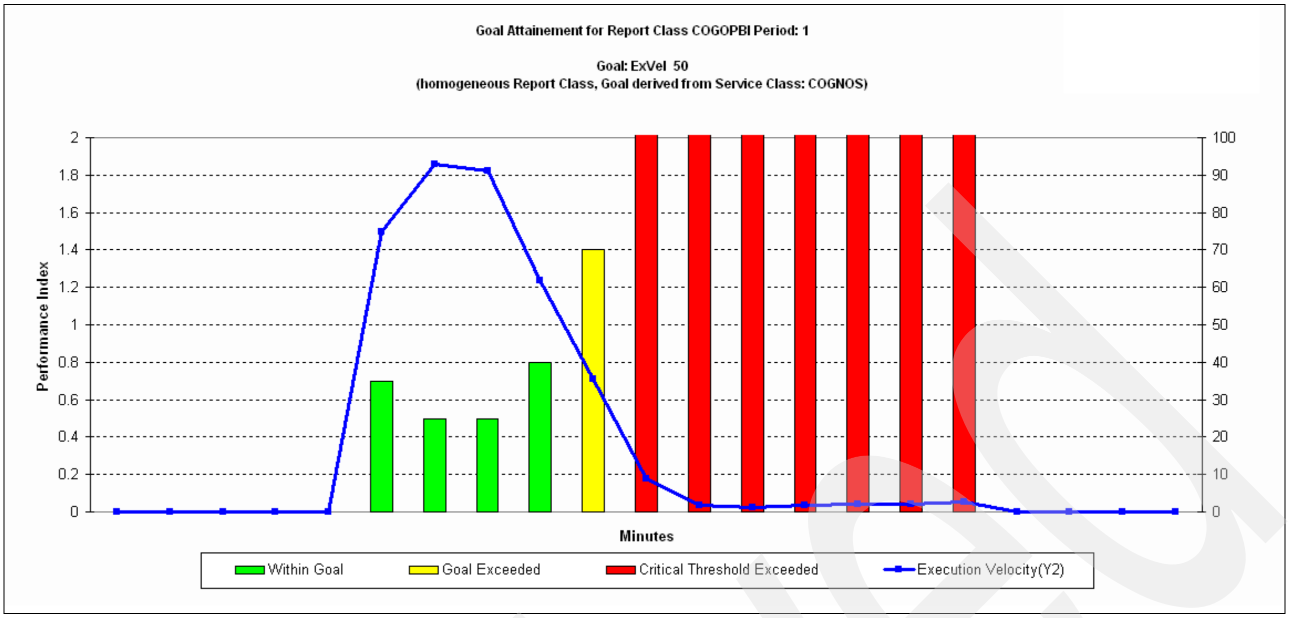


Figure 11-33 New DDF workload: Goals for service class DDF

Analysis and conclusions

The introduction of an unplanned workload, from a WLM point of view, can produce a dramatic impact on current workload if the adequate WLM definitions are not implemented. This example shows how WLM has allocated resources to the most important workload, but this may not be appropriate for the business requirements. It is necessary to match the business priorities and SLAs with the WLM definitions. The understanding of the business requirements is of capital importance to this process.

Attention: You should build your policy based on what your current system capacity can handle not on what would be ideal. WLM cannot create capacity. However, it is possible to delay a capacity upgrade if the WLM policy is well tuned and the business requirements are met.

11.5.3 Classification of DDF workload

The introduction of a new Cognos report in the system has had a dramatic impact on the important batch query processing. The introduction of this workload was done without WLM preparation to reflect the business requirements.

The main objectives of this scenario are as follows:

- Show how to classify a DDF workload in an appropriate service class
- Exhibit how an accurate WLM classification can protect an important workload

Description

For the purposes of this example, let us consider that the query processing is defined as more important for the business and that WLM will be changed to protect this workload.

After the observation of the previous example and following good practice recommendations, the Cognos reporting workload is classified in a specific service class named COGNOS. This is achieved by an update of the classification rules that distinguished DDF from COGNOS workload. Following the business requirements, this service class is defined with an importance lesser than the importance of the service class on which the batch processing is classified.

Important: To provide priorities to workloads, use the importance parameter instead of more aggressive goals within the same importance.

Table 11-12 summarizes the most relevant services classes for this example. The objective is to protect the most important workload, the batch queries, from the COGNOS activity. The effects of these changes need to be validated during periods of high activity: WLM shows its value most during busy moments.

Table 11-12 Relevant Service classes during classification of DDF workload scenario

Workload	Service Class	Importance	Goal
DDF	COGNOS	3	Velocity 50
DWH	BA_BATCH	2	Velocity 60

Table 11-13 shows the relevant reporting classes for this scenario. Note that it is possible to get a view of the COGNOS workload independently of other DDF activity. This would be also be possible even if the service class for DDF and COGNOS were defined the same It is possible to classify workload in different reporting classes even if the service class is the same.

Important: Do not use service classes for reporting purposes. Use reporting classes instead.

Table 11-13 Relevant reporting classes during classification of DDF workload scenario

Reporting class	Qualifying
DWH	Mix batch workload
COGNORPT	New DDF workload, COGNOS reports
DDFRPT	DDF workload

Observations

As in the previous test, the batch workload was run for some minutes to get a stable running system and to eliminate border effects. The COGNOS reporting was started and the workload injected in the system was the same as in the previous example.

Figure 11-34 shows the CPU profile build from the DB2 accounting records.

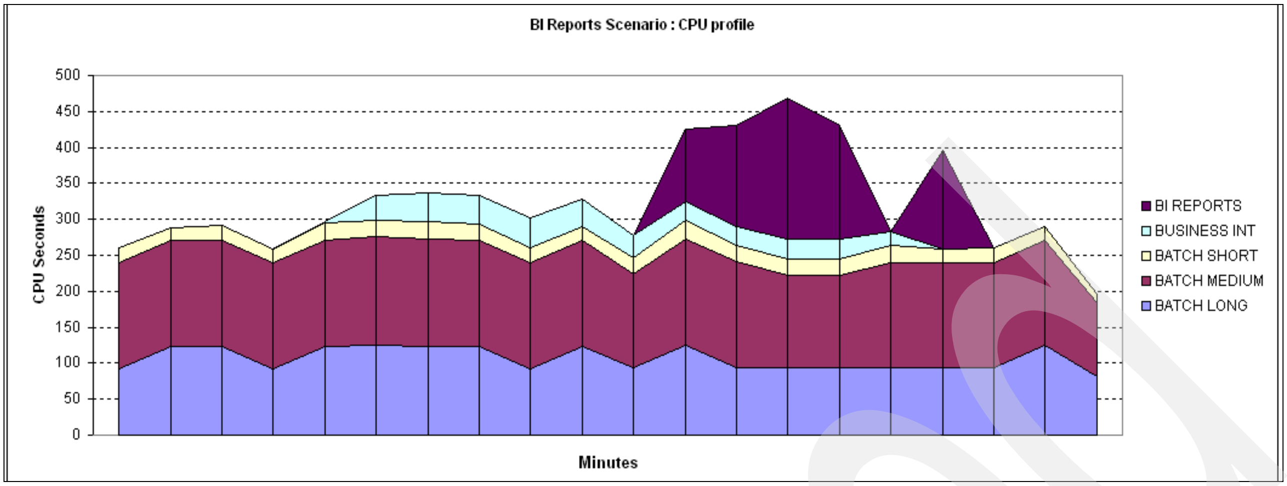


Figure 11-34 Classify DDF scenario: DB2 accounting CPU profile

During this scenario, the batch query processing was not impacted by the COGNOS activity, and the COGNOS workload was executed in a longer elapsed time. This proves the ability of the new WLM definitions to protect the more important batch workload. This was achieved during busy moments, the LPAR was driven to a 100% utilization, as it can be seen in the Figure 11-35.

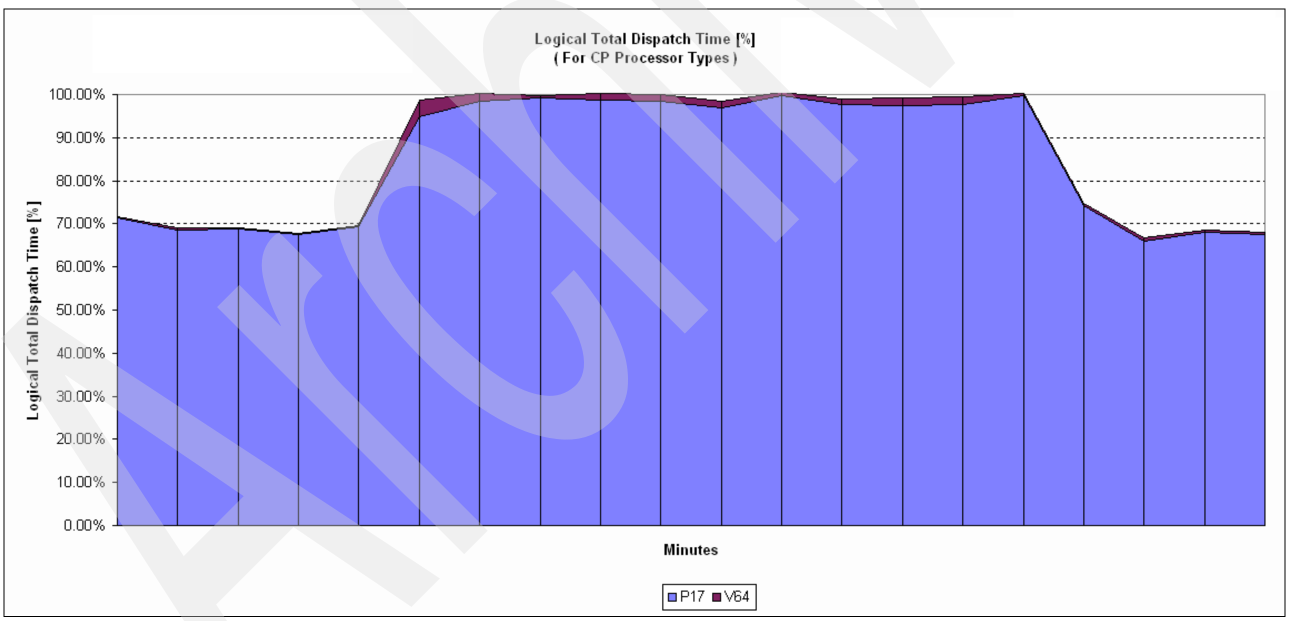


Figure 11-35 Classify DDF scenario: LPAR Dispatch Time

A better confirmation that the batch queries were executing within its objectives can be obtained from the observation of the performance index for the BA_BATCH service class. You can find this information in Figure 11-36 on page 260. The performance index for this workload was always less than 1, even during the period on which the COGNOS reporting were active. This report can be compared to the one exposed in the Figure 11-32 on page 256, the equivalent report for the previous example.

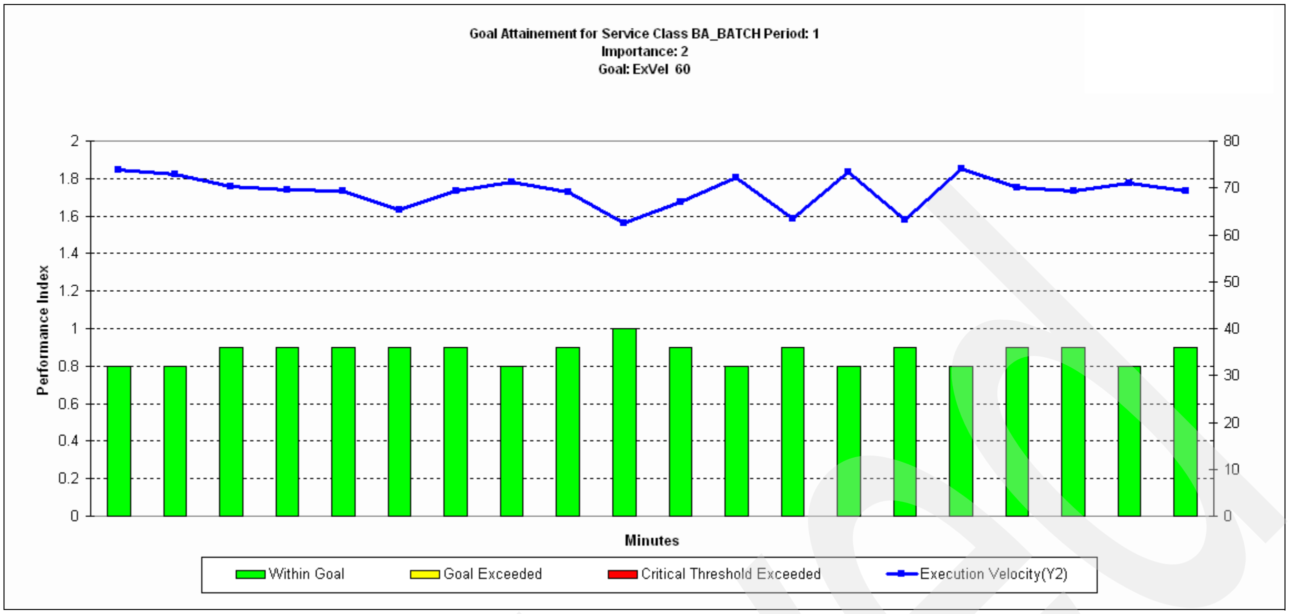


Figure 11-36 Classify DDF workload: Goals for service class BA_BATCH

However, the service class COGNOS was missing its objectives during almost all of the test scenario, as Figure 11-37 shows. Apart from a short adaption period at the beginning of the reporting, the performance index was higher than 1.

This can be due to two reasons:

- The objective for the service class is too aggressive. A lower goal within the same importance could have met the objectives. You can try to adapt the service class definitions.
- There is a capacity problem. If the business requirements for the COGNOS workload are strictly reflected on this service class definition, you need to increase the capacity of this LPAR. WLM cannot create CPU cycles or more capacity.

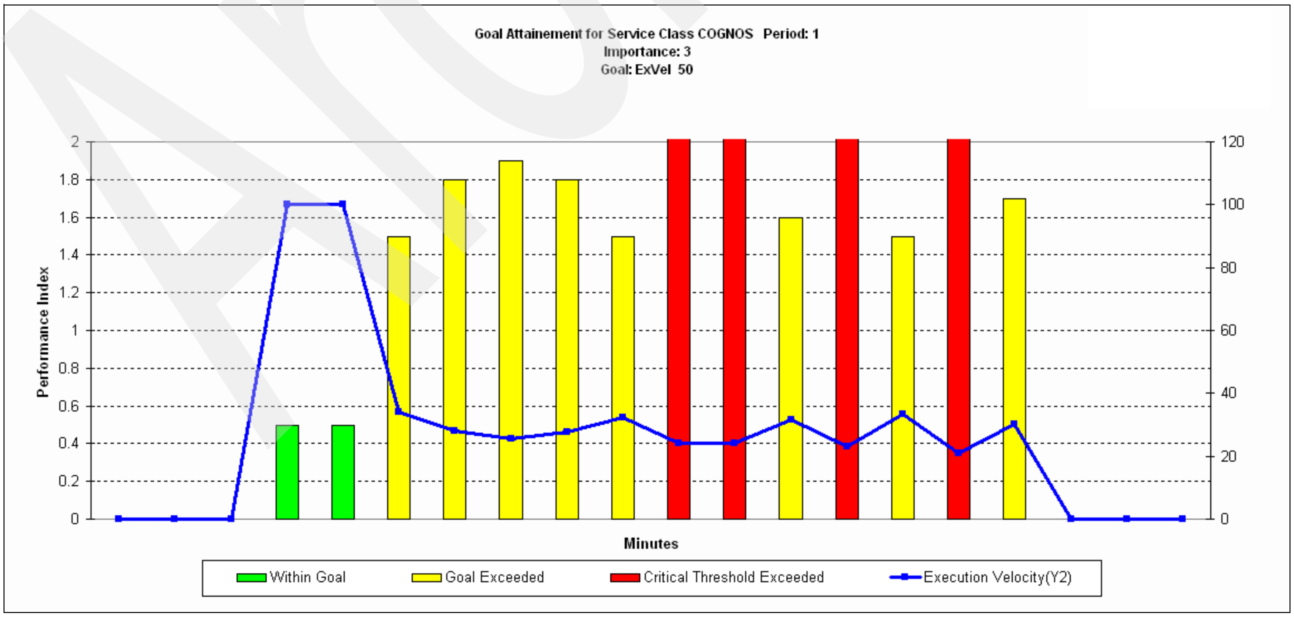


Figure 11-37 Classify DDF workload: Goals for service class COGNOS

Analysis and conclusions

Figure 11-38 compares the CPU profile before and after applying changes in the WLM policy to protect the more important batch workload.

A proper service class definition complement with a correct classification rules can protect important work but cannot create capacity that the system does not have but is needed to cope with all the workload requirements.

WLM provides most of this valuable help during busy moments. You need to plan your policy with rush hours as objective.

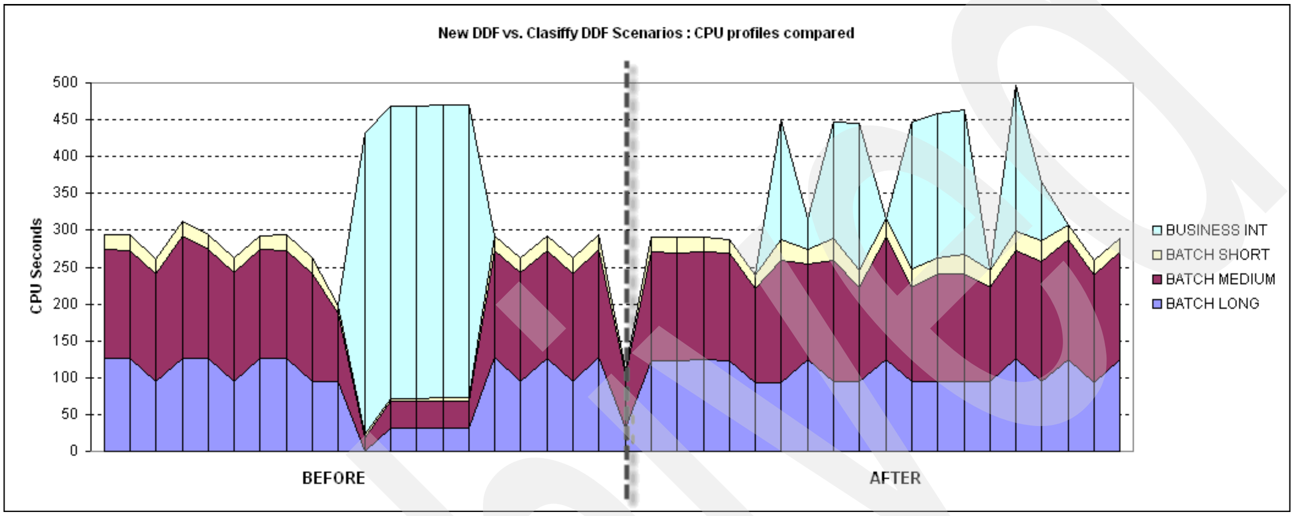


Figure 11-38 Classify DDF workload: Before and after CPU profile

11.5.4 Introduction of new Operational BI workloads

WLM can be used for protecting important work even from impacts that are less dramatic than the one shown in shown in the previous example, even from workload classified in the same service class. In this scenario, a batch DW workload is present during the complete duration of the test. This workload is classified into the service class BA_BATCH.

An Operational BI workload is classified into a less important service class COGNOS. The characteristics of the queries executed on this service class are OLTP-like, of short duration, and not intensive on resources utilization. From the same origin and in the same service class, BI reporting is introduced in the system. These reports are more like traditional DW workload. That is, of long duration and resource intensive.

Table 11-14 summarizes the characteristics of these service classes. Note that the service class on which the batch queries are classified, BA_BATCH, is more important than the service class used by the Cognos OLTP and reporting work.

Table 11-14 Relevant service classes during introduction of OBI workload scenario

Workload	Service Class	Importance	Goal
DDF	COGNOS	3	Velocity 50
DWH	BA_BATCH	2	Velocity 60

For this scenario we can make the assumption that the mix of Operation BI and Cognos reporting in the same service class was a known fact for the system administrator. Under these circumstances, one may decide to keep a shared service class but implement specific reporting classes to evaluate the actual impact of each of these workloads in the system. Table 11-15 summarizes some relevant reporting classes used during this exercise. A distinction is made between the Cognos Operation BI (OLTP) and the Cognos reporting. These workloads are classified in different reporting classes.

Table 11-15 Relevant reporting classes during introduction of OBI workload scenario

Reporting class	Qualifying
DWH	Mix batch workload
COGNORPT	COGNOS reports
COGOPBI	Cognos Operation BI workload (OLTP)

Objectives

The objectives of this scenario are as follows:

- Show how OBI processing, an OLTP-like workload, would require WLM adjustments in a traditional DW environment.
- Show the value of reporting classes for reporting and problem analysis when looking for a better workload classification.

Observations

Figure 11-39 shows the evolution of the CPU utilization from the DB2 accounting. The BI work is affected during the execution of the BI reports, while the batch queries, which were defined as more important, are not. Using the DB2 accounting, it is also possible to show that the number of BI OLTP transactions is lower during the time that the BI reports were executed.

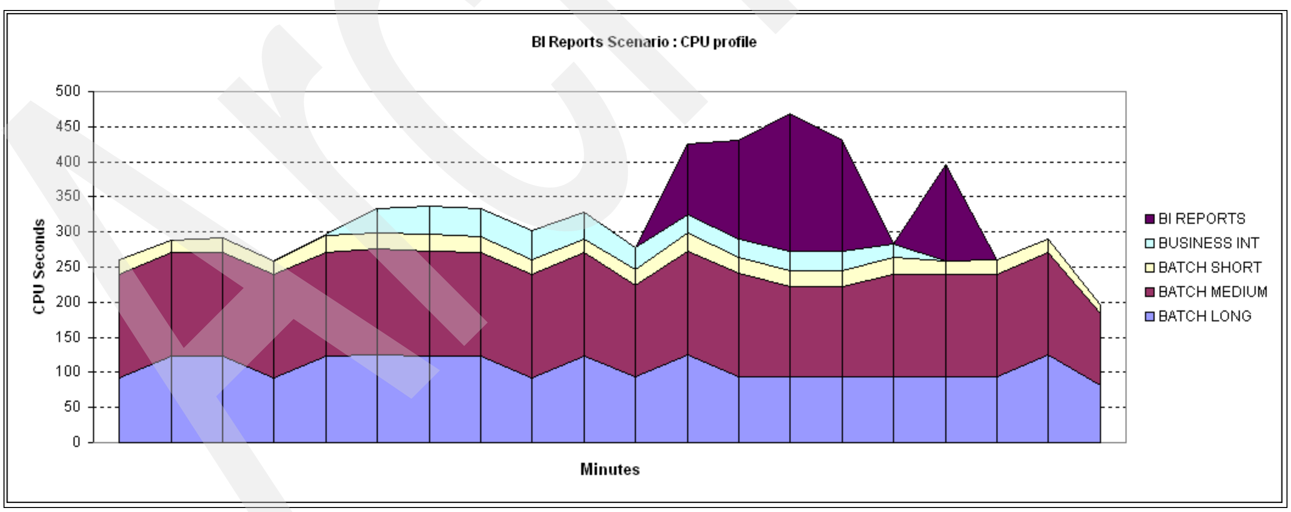


Figure 11-39 New OBI work scenario: CPU profile from DB2 accounting

Figure 11-40 on page 263 illustrates the machine utilization during the test. The z/OS LPAR was driven to 100% only during the period on which the Cognos reporting was active. This partition is represented as P17 on this figure. Partition V64 is Linux for System z containing the Cognos application.

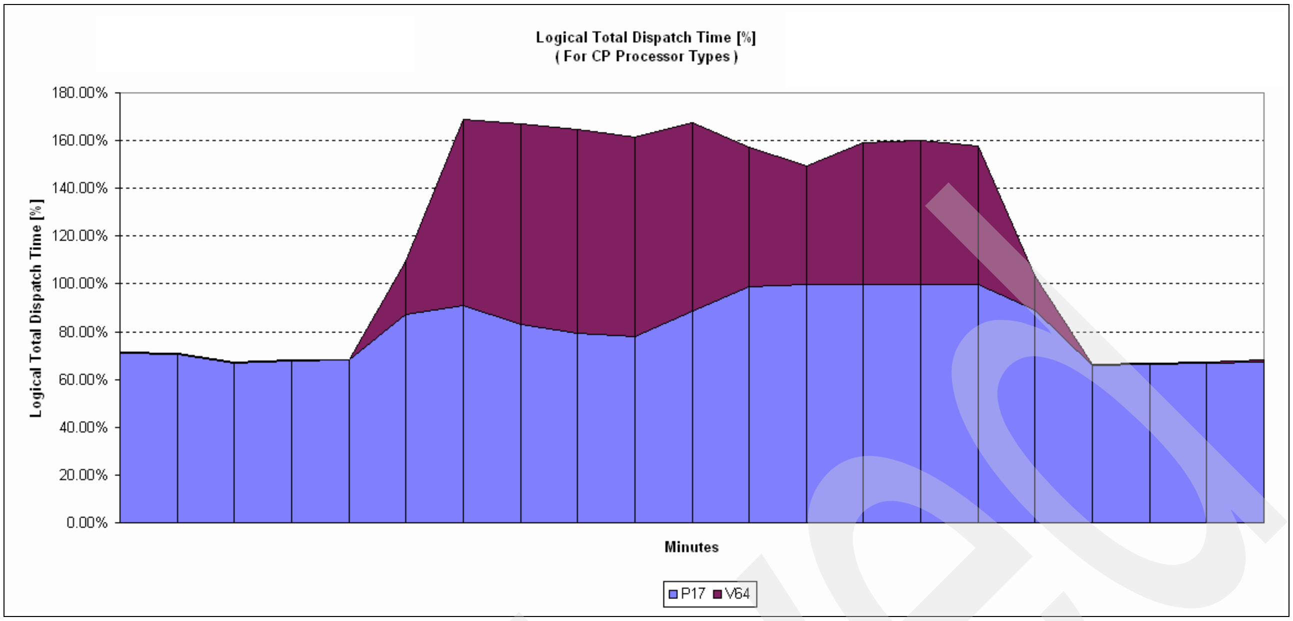


Figure 11-40 New OBI work scenario: LPAR Dispatch Time

WLM has protected the workload under the service class BA_BATCH and assigned to it the needed resources for achieving its goals, as shown in Figure 11-41.

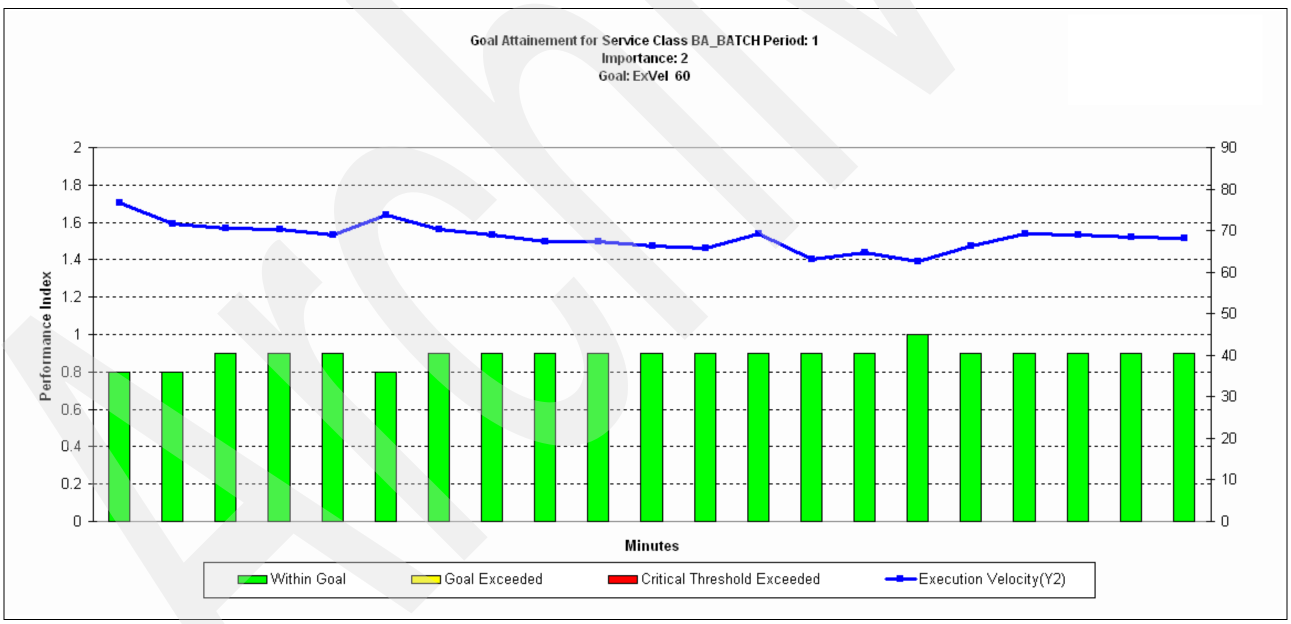


Figure 11-41 New OBI work scenario: Goal attainments for batch queries

The Operational BI workload received enough resources to achieve its goals at the beginning of the test. The introduction of the BI Reporting reduced the CPU that was available for this work, so the goals were missed, as shown in Figure 11-42 on page 264. The scale of the performance index was locked at a lower value for better plotting, but the goals were severely missed.

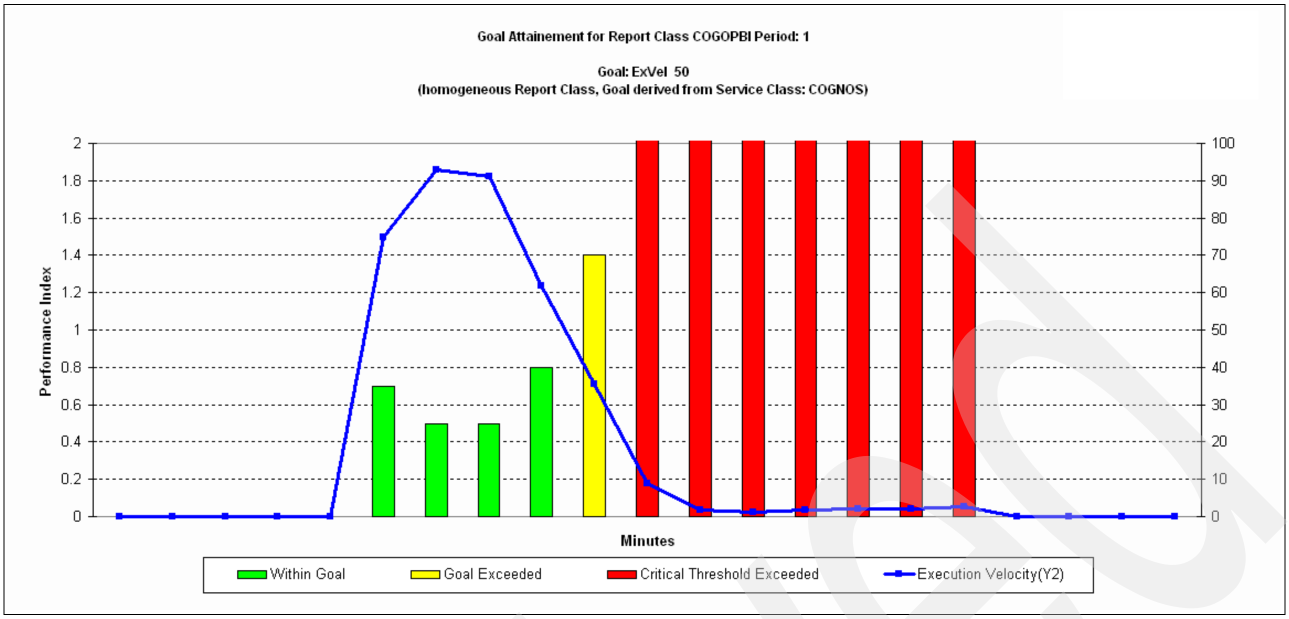


Figure 11-42 New OBI work scenario: goal attainments for Operational BI

Figure 11-43 shows the goal attainments for the BI reporting workload. Even if the goals were not achieved, the performance index was less affected than the one for the Operational BI queries.

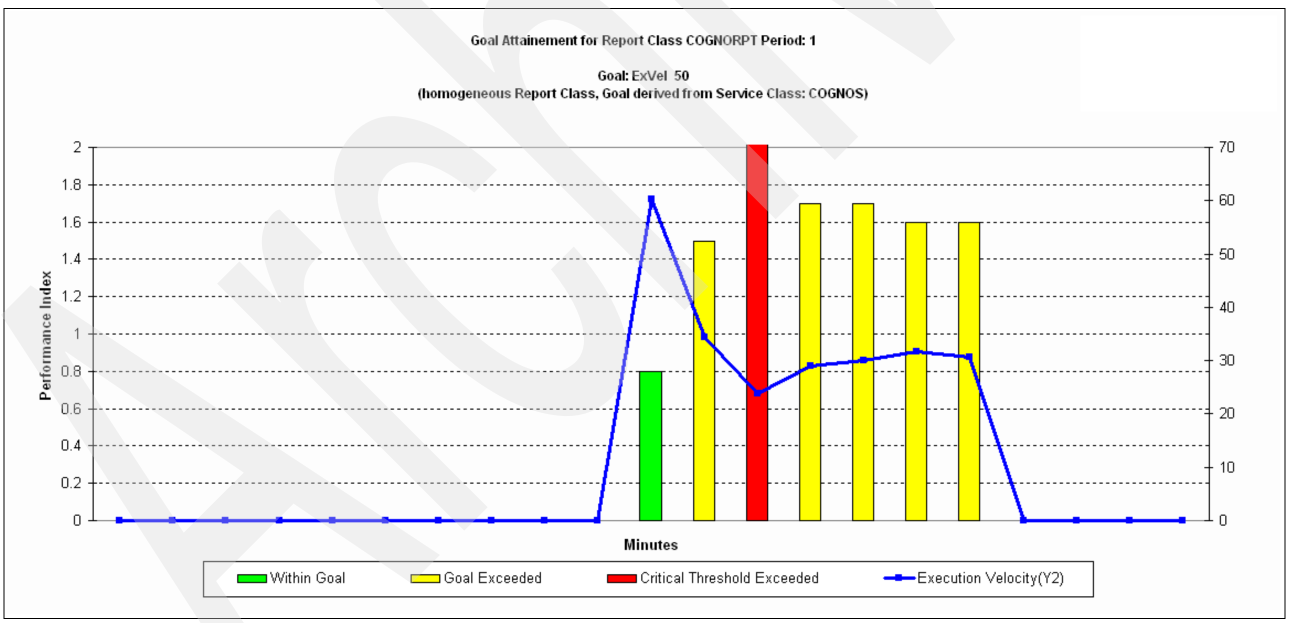


Figure 11-43 New OBI work scenario: goal attainments for BI reporting

Analysis and conclusions

WLM is able to manage a higher priority workload and protect it by assigning needed resources. A proper use of reporting classes can be of great help for problem investigation and for describing how resources are distributed inside a single service class.

The introduction of BI reporting can severely affect the goals of operation BI queries in a DW system if defined in the same service class. A way of solving this issue can be found by the implementation of period goals.

11.5.5 Period goals

Because some work may have variable resource requirements, you can define multiple periods for a service class. Periods are a way of defining different goals for work depending on the amount of resources the work consumes. Typically, periods are used to give shorter transactions more aggressive goals and to give longer running work of the same type less aggressive goals. If you have multiple periods, each period except the last has a duration. Duration is the amount of resources, in service units, that the work consumes before it is switched to the goals of the next period.

Table 11-16 summarizes the most relevant service classes for this scenario.

Table 11-16 Relevant service classes during period goals scenario

Workload	Service Class	Period & duration	Importance	Goal
DDF	COGNOS	1 - 7600	2	AvgRT 1 sec
DDF	COGNOS	2 - 38000	3	Velocity 60
DDF	COGNOS	3 - N/A	3	Velocity 50
DWH	BA_BATCH	1	2	Velocity 60

Note the introduction of periods for the COGNOS service class. You can imagine this as a sort of internal to the service class workload classification in function of the resource consumption. The first period will receive a high importance during 7600 service units, a query consuming more than this quantity will fall in the second period, where it will be affected by an importance 3 goal. After consuming the 38000 service units of this period, the transaction will fall into the period 3, where the goal is less aggressive, with a lower velocity.

Response time goals are not reserved only for Operational BI workloads, but can be appropriated for any work that executes frequently with a small resource utilization, such as OLTP for instance. As a guide, consider the utilization of a response time goal if you observe at least 1 completion every 10 seconds.

The first period of this service class was defined with a response time goal. There are two types of response time goal:

- Average response time
- Percentile response time

The response time represents the time that WLM is aware of the unit of work. For a batch job, response time is the time when the job has finished the conversion step until it is completed and the result is placed on the output queue. For TSO, response time is the time from when the user presses **Enter** until the result is returned to the terminal. Note that the response time measures different things depending on the subsystem that is measured, and it is not an end-to-end response time.

In percentile response time goals, you set a response time target for a given percentage of transactions (for example, 90% of transactions within 1 second). WLM tries to ensure that the selected percentage of transactions finishes within the specified time. If there are a few transactions that last much longer than most, they do not affect the management of the majority.

You might use average response time goals if the transaction response times of a workload are extremely homogeneous. Unfortunately, this is often not the case.

The value for the durations were calculated based on the average CPU consumption per type of work extracted from the DB2 accounting with the following objectives:

- First period

  This period should contain all the OLTP queries. They will be executed at a high priority and the goal is response time based. A RT-based goal is the most appropriate for an OLTP transaction and it is easier to translate a complete business requirement to a response time goal.

- Second period

  Only BI reports should fall on this period after consuming the resources of period one. Average BI reports are expected to complete its processing during this period. The assigned goal is less aggressive than the one for the first period. This is a way of protecting OLTP transactions from less critical BI reporting.

- Third period

  Only unexpected longer-than-average BI reports should be executed on this period. This is a way of protecting the system from long running processes taking too many resources longer than expected. This period has no duration and its goal is the lowest of the relevant service classes for this exercise.

The use of service units guaranties that even in the event of a CPU change or upgrade, the periods will be still valid because service units are a CPU model independent representation of quantity of work done.

Example 11-25 shows an extract of the WLM print listing of the active policy during this test. This extract shows how you can read on this file the description of the periods for a service class.

Example 11-25 Service class periods description in WLM listing

```
==============================================================
        * Workload DDF - Distributed DB2
          1 service class is defined in this workload.
            * Service Class COGNOS - Operation BI Cognos

                #  Duration   Imp  Goal description
                -  ---------  -    ----------------------------------------
                1  7600       2    Average response time of 00:00:01.000
                2  38000      3    Execution velocity of 60
                3             3    Execution velocity of 50
          ==============================================================
        * Workload DWH - Data Warehouse
          1 service class is defined in this workload.
            * Service Class BA_BATCH - Business Analysis batch jobs

                #  Duration   Imp  Goal description
                -  ---------  -    ----------------------------------------
                1             2    Execution velocity of 60
```

Table 11-17 shows the relevant reporting class for this scenario.

Table 11-17 Relevant Reporting classes during Period goals scenario

Reporting class	Qualifying
DWH	Mix batch workload
COGNORPT	COGNOS reports
COGOPBI	Cognos Operation BI workload (OLTP)

Objectives

The main objective of this scenario is to show the use and advantages of period aging within service classes.

Observations

Figure 11-44 shows the CPU profile created from the DB2 accounting traces. The batch and BI work were not impacted from the presence of the BI reporting. The WLM changes introduced were able to manage the workload as expected even if the system was running at 100% during the BI reporting period.

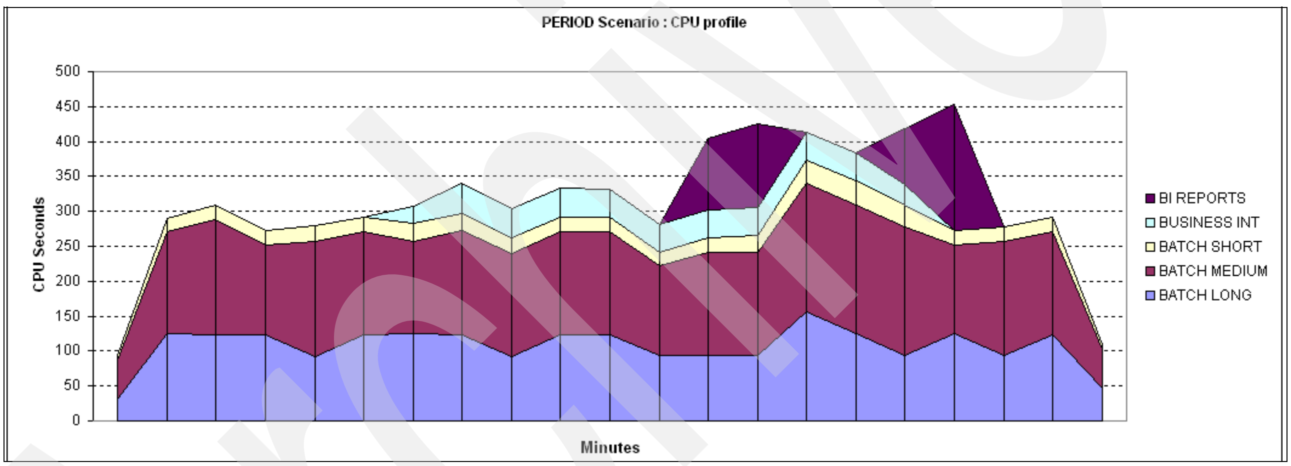


Figure 11-44 Period goals scenario: CPU profile from DB2 accounting

The goals for the batch queries were always achieved, as shown in Figure 11-45 on page 268, even if the batch queries and the first period of the COGNOS service class shared the same goal.

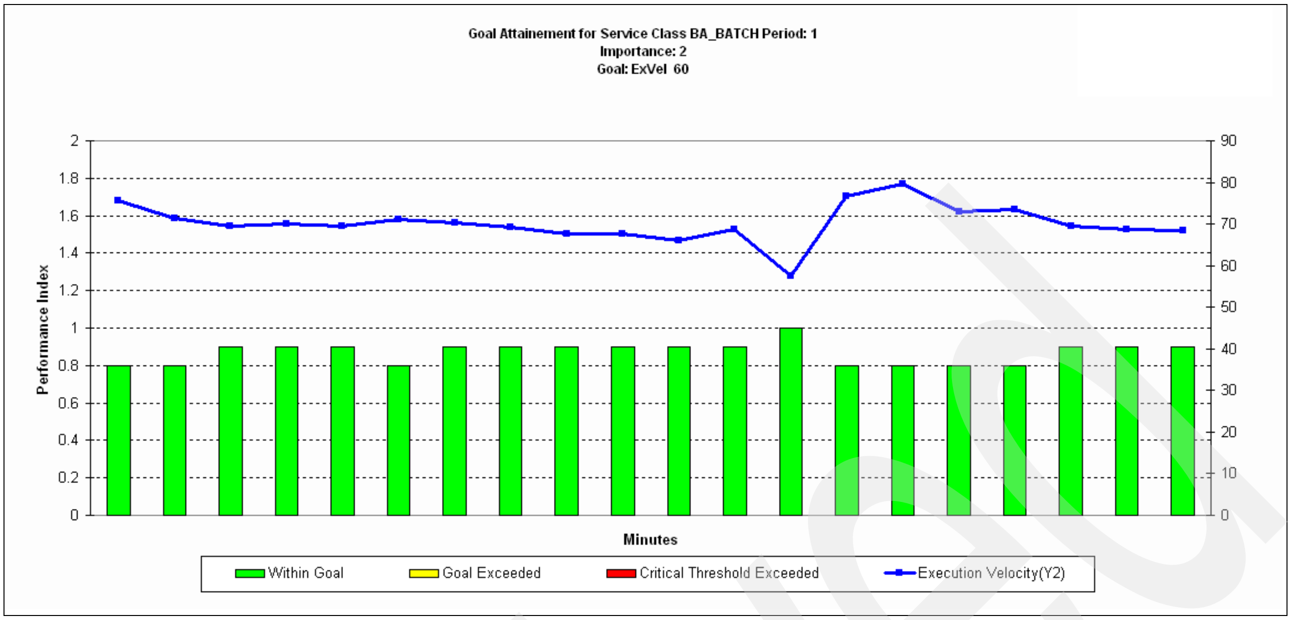


Figure 11-45 Period goals scenario: goal attainments for batch queries

All the Operational BI OLTP transactions were classified on the COGOPBI reporting class. This reporting class shows that the amount of resources assigned to these critical processes were not impacted by the presence of BI reports in the system.

Because the first period of the COGNOS service class has a response time-based goal, you can exploit the RMF spreadsheet reporter RT distribution trend chart report. This report is shown in Figure 11-46 for the COGNOS service class.

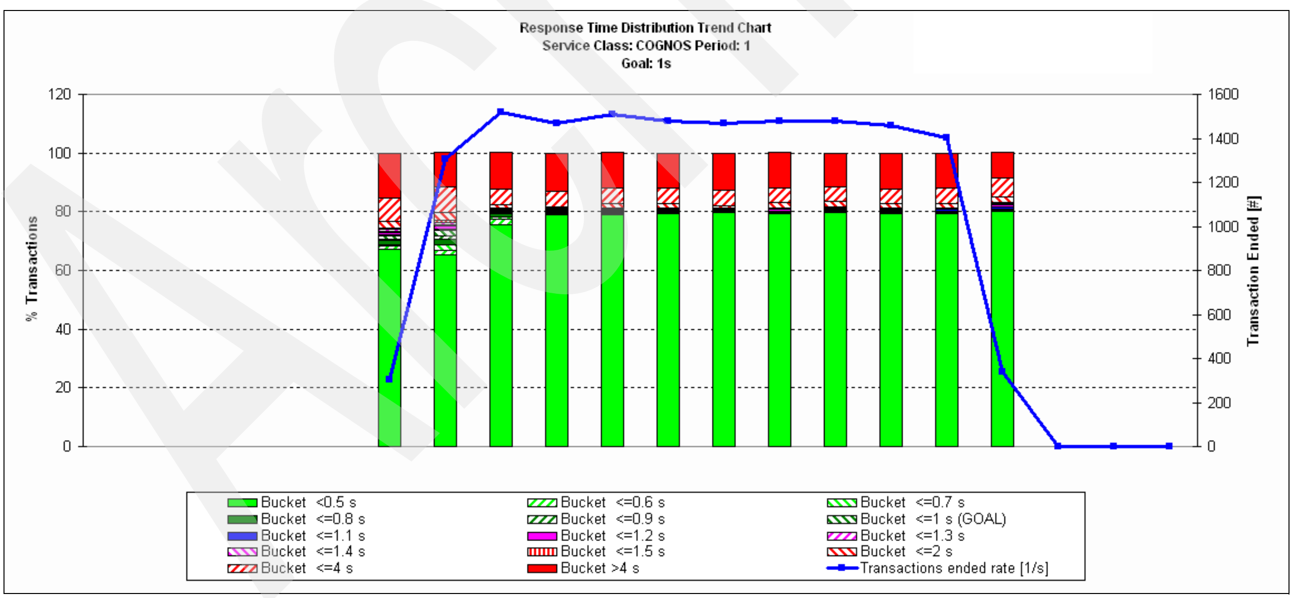


Figure 11-46 Period goals scenario: RT distribution chart

You need to consider that this chart contains information about all the transactions of the service class. Thus it includes not only the OLTP BI queries but also the BI reports that are expected to fall in the second period. It is normal then to verify that some executions were not finished within period one. This graph also shows the transaction ended rate was constant during all the test. You may refer to the DB2 accounting also for confirming that the OLTP queries were not affected during the BI report period by verifying the number of ended transactions per unit of time, for instance.

Because the service class was defined with three periods, we need to analyze the goals for each of them. You will get three reports, one for each period, from the RMF spreadsheet reporter for this service class.

Figure 11-47 shows the goals attainment trends for the first period. The goals were underachieved during all the period, representation of an overly aggressive definition. There is no execution velocity information in the graph due to the fact that this period was defined with a response time goal. The level of achievement remained constant for the duration of the test for this period, meaning that there was no negative impact of the BI reporting for the workload classified on this period.

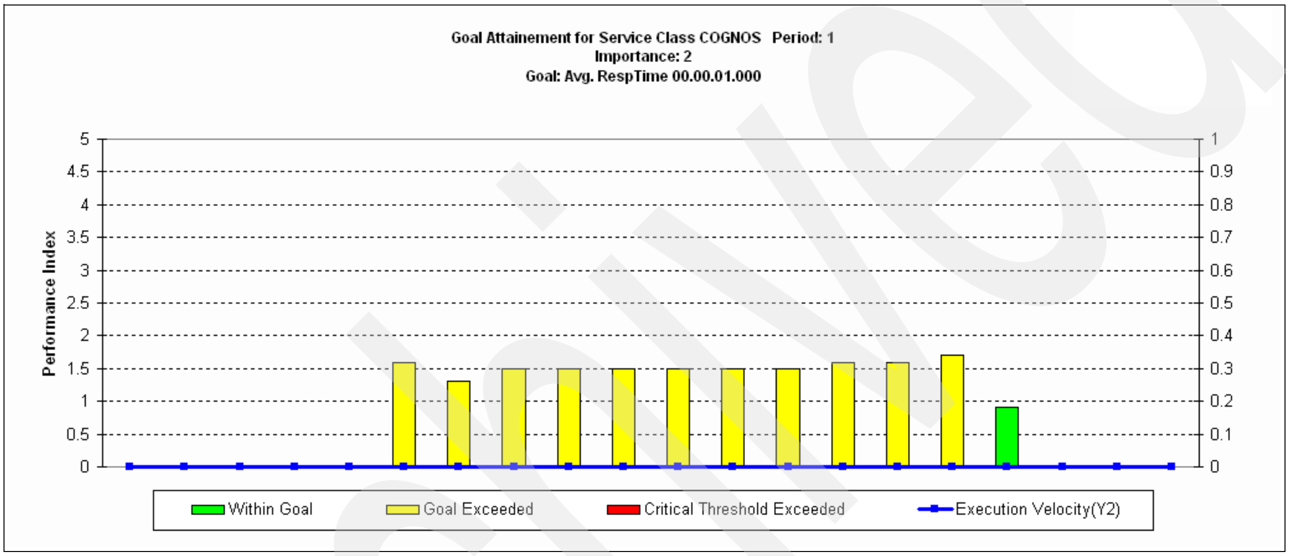


Figure 11-47 Period goals scenario: SC COGNOS goals for period 1

Figure 11-48 on page 270 shows the goal evolution for the second period. Notice that there is no reported activity on this period before the introduction of the BI reports. This is a signal that all the OLTP queries were completed in the first period, as expected, before the start of the BI reporting. The goal attainment for this period is generally acceptable.

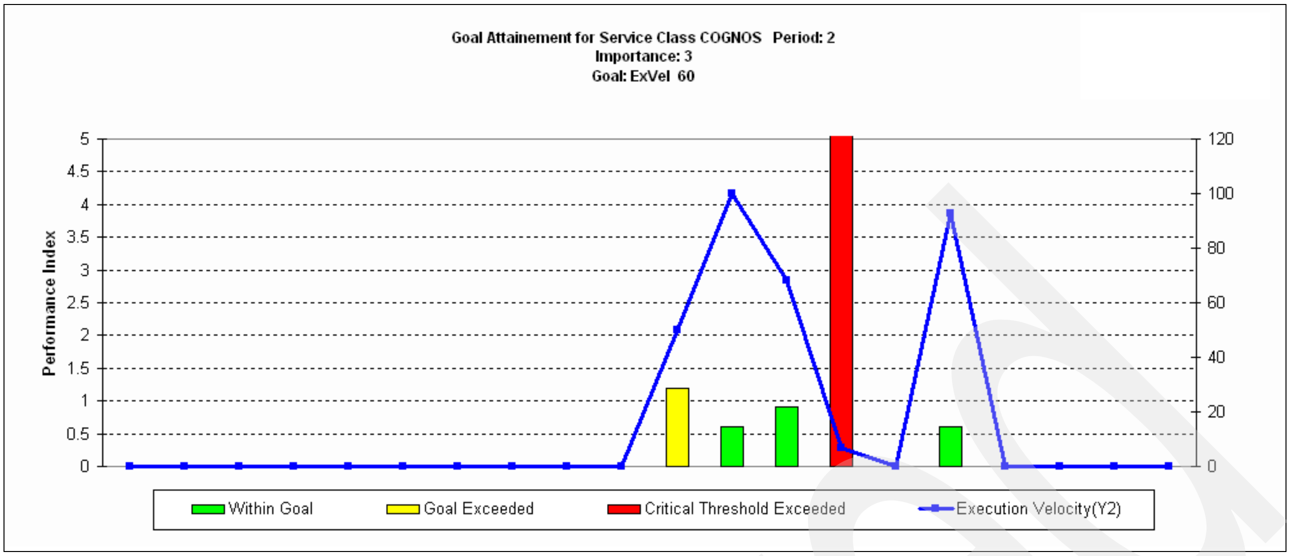


Figure 11-48 Period goals scenario: SC COGNOS goals for period 2

Figure 11-49 shows the goals for the third period. This period contains the BI reports that consumed more resources than expected.

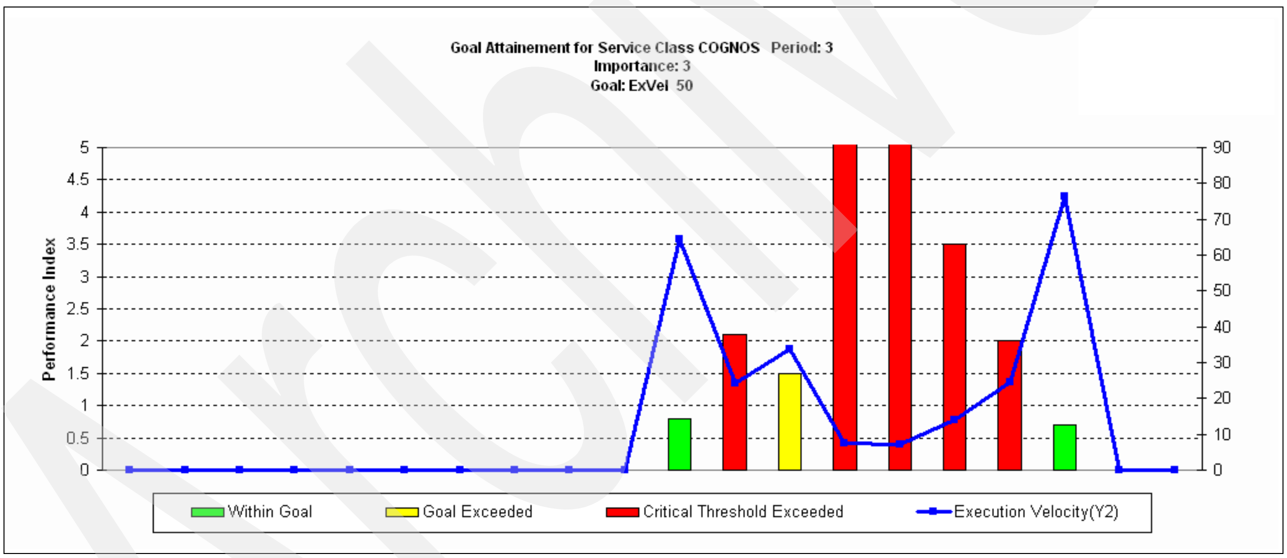


Figure 11-49 Period goals scenario: SC COGNOS goals for period 3

Analysis and conclusions

Defining service classes with periods can help you achieve the following objectives:

- To classify work inside a single service class
- To be selective on goals and importance based on resource consumption
- To prevent unexpected long running workload to absorb too much resources for a long time and impact negatively other work in the same system

11.5.6 Introduction of new data mining workload

Without WLM preparation, a data mining batch process, resource intensive, is added to the system affecting the service provided to the other workloads. This new workload is not critical for the business.

The original workload is composed of a mix of critical batch queries and OLTP-like operation BI queries coming from COGNOS. No specific service class preparation was done for this new workload. It is sharing the same service class as the batch queries. The data mining processing was classified into a specific reporting class for better monitoring.

Table 11-18 summarizes the most relevant service classes involved in this test.

Table 11-18 Relevant Service classes during Data Mining scenario

Workload	Service Class	Period	Importance	Goal
DDF	COGNOS	1	2	AvgRT 2 sec 70%
DWH	BA_BATCH	1	2	Velocity 60

Table 11-19 list the most important reporting classes for this scenario.

Table 11-19 Relevant reporting classes during data mining scenario

Reporting class	Qualifying
DWH	Mix batch workload
COGOPBI	Operation BI workload
DMINE	Data Mining workload

Objectives

The objective of this scenario is to show what happens with a large consumption work in DW systems without prioritization.

Observations

The mix of batch queries was running in the system for several minutes when the operational BI workload was introduced. The system continues to execute work without impact on any of the processes. After several minutes, a single data mining batch process was started. This batch job shared the service class with the original batch mix of queries. The system was driven to 100% utilization.

Figure 11-50 shows the CPU profile for the complete test based on the DB2 accounting.

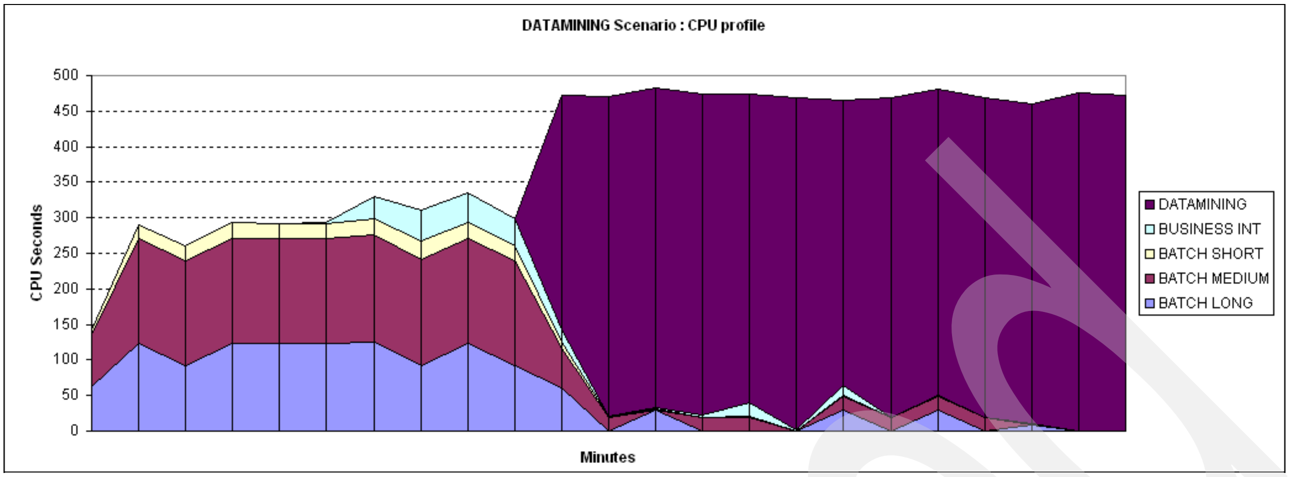


Figure 11-50 New data mining workload: CPU profile from DB2 accounting

The impact of the new data mining workload is dramatic on the other workloads.

Figure 11-51 shows the evolution of the goal achievement for the BA_BATCH service class. It is clear that the introduction of the data mining process has impacted the goal attainments for the more important workload batch mix of queries.

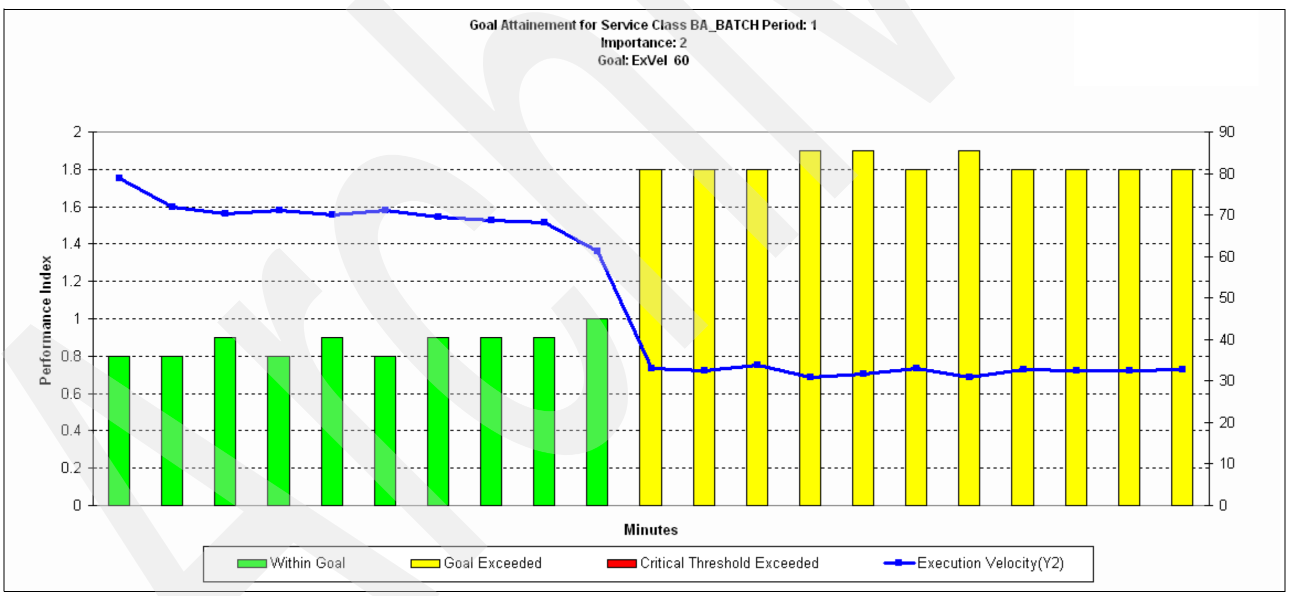


Figure 11-51 New data mining workload: goal attainments for batch service class

Figure 11-52 on page 273 shows the goals attainment evolution for the COGNOS reporting class. This workload was negatively impacted by the new data mining process. This is not in line with the business requirements, as the OLTP BI is more critical for the business.

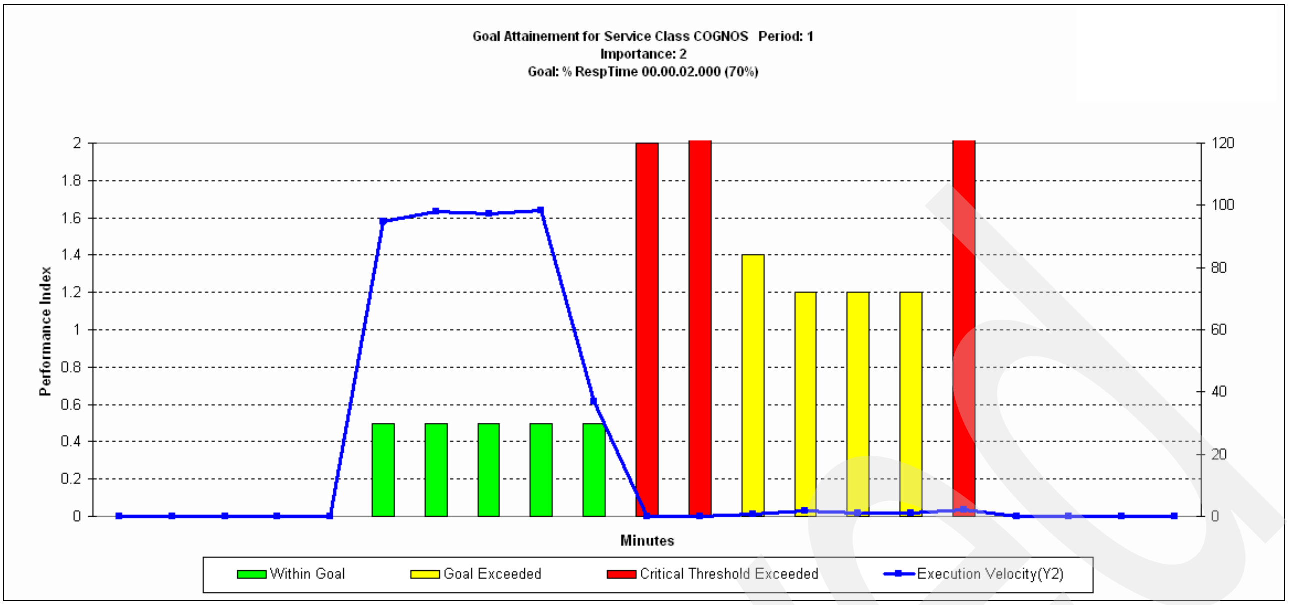


Figure 11-52 New data mining workload: goal attainments for COGNOS reporting class

Analysis and conclusions

This scenario shows the dramatic effects that an incorrectly classified workload can have in a running DW system. An example of these workloads can be a data mining process. It is necessary to find a way to allow such processes to be executed but without negatively impacting the other work in progress. A good example of this solution is the use of discretionary goals.

11.5.7 Discretionary goals

The data mining processing does not have fast response time requirements. A discretionary period is added to the WLM definitions for this process. The goal is to allow the data mining processing to complete by using spare capacity that is not needed by the more important batch mix and OLTP BI processing.

The BA_BATCH service class was modified to introduce two periods:

- First period

  Present the same goals and importance as in the original service class, and it duration was calculated on such a way all the batch mix queries will end during this period

- Second period

  No duration and goal is discretionary. A discretionary goal is equal to importance 6 and is the lowest a workload can receive. A discretionary service class will receive system resources only if all the other service classes are achieving their objectives. A discretionary goal is always a resource donor and will use only spare resources not claimed by any other process in the system.

Table 11-20 on page 274 summarizes the most relevant service class definitions for this test scenario.

Table 11-20 Relevant service classes during discretionary scenario

Workload	Service Class	Period	Importance	Goal
DDF	COGNOS	1	2	AvgRT 2 sec 70%
DWH	BA_BATCH	1	2	Velocity 60
DWH	BA_BATCH	2	6	DISCRET

Table 11-21 list the reporting classes of relevance for this test.

Table 11-21 Relevant Reporting classes during Discretionary scenario

Reporting class	Qualifying
DWH	Mix batch workload
COGOPBI	Operation BI workload
DMINE	Data Mining workload

Objectives

The objective of this scenario is to exhibit how the WLM discretionary goal can be used to utilize spare cycles efficiently for work that does not have immediate RT requirements.

Observations

After having a stable system composed by the batch mix queries and the OLTP BI workload, the data mining job was started. Figure 11-53 shows the CPU profile for this test build from the DB2 accounting information.

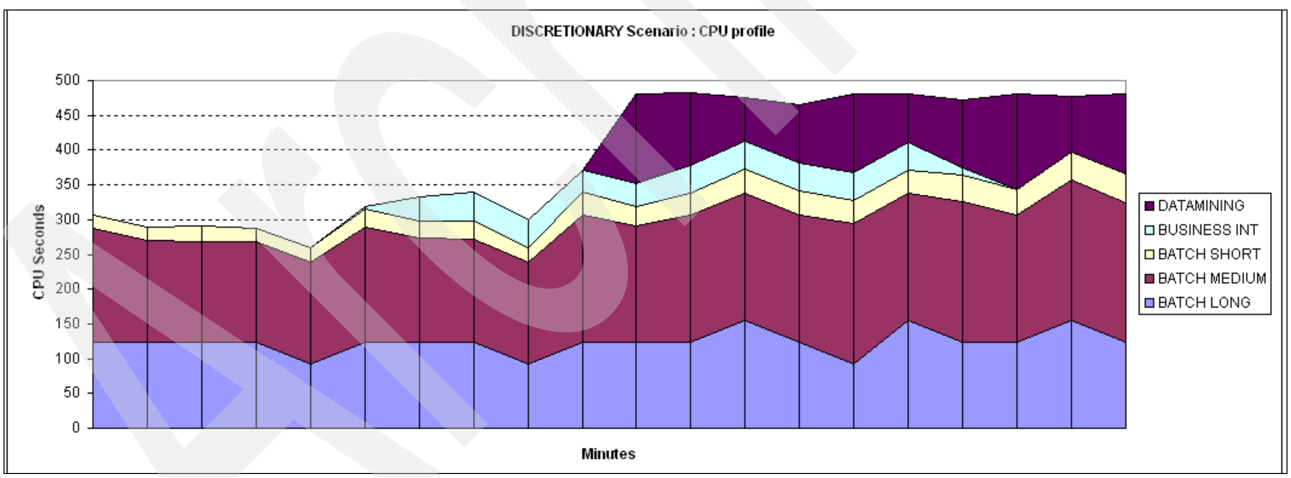


Figure 11-53 Discretionary scenario: CPU profile from the DB2 accounting

This report shows that there is no impact on the batch mix queries or OLTP BI workloads while the data mining report is being executed. This is in line with the business requirements and working as desired. The data mining process is not critical and must not impact the other workloads in the system.

During this test, the partition was driven to 100% utilization, as shown in Figure 11-54.

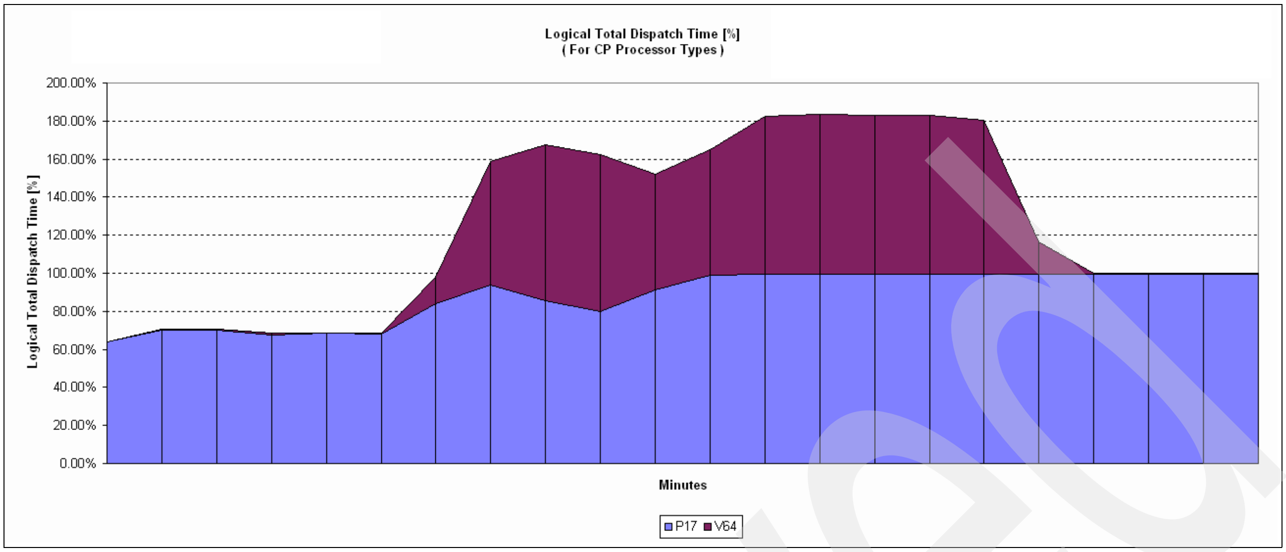


Figure 11-54 Discretionary scenario: LPAR dispatch time

The goal attainment for the BA_BATCH service class, period 1, shown in Figure 11-55 confirms that this workload was not impacted by the execution of the data mining job.

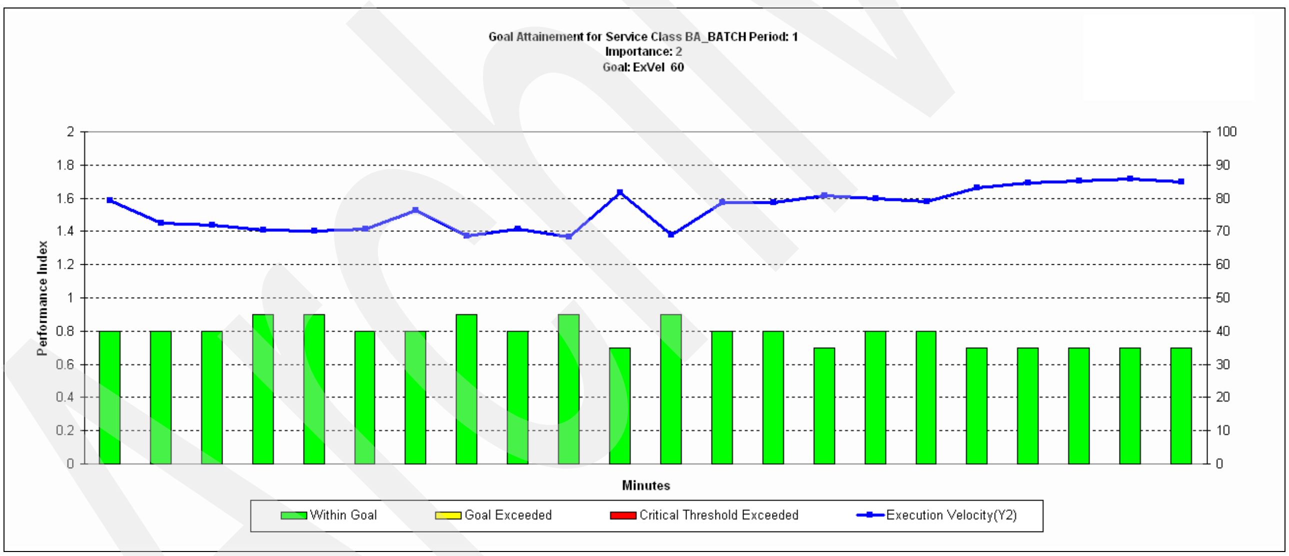


Figure 11-55 Discretionary scenario: goal attainments for BA_BATCH service class

The OLTP BI activity was unaltered as expected. This is shown in Figure 11-56.

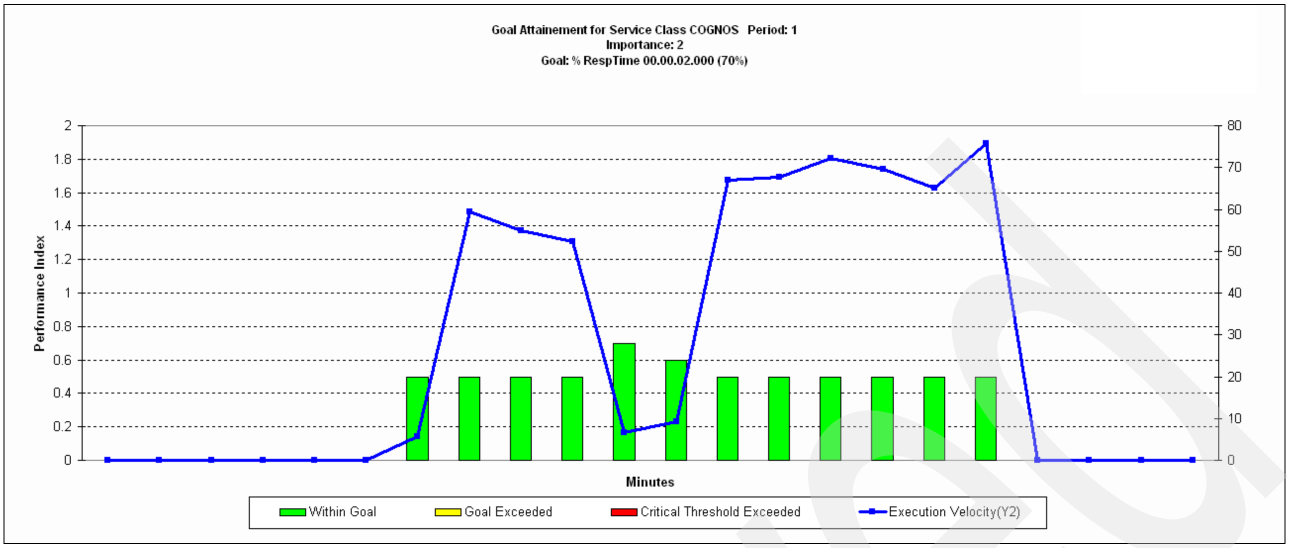


Figure 11-56 Discretionary scenario: Goal attainment for COGNOS service class

Figure 11-57 shows the goal attainment chart for the second period of the BA_BATCH service class. Only the data mining job executed in the second period of this service class. Notice that there is no performance index represented in the graph. A discretionary workload does not have a goal by definition. It will use spare cycles from the system. Note that the execution velocity is quite low compared to the previous workloads.

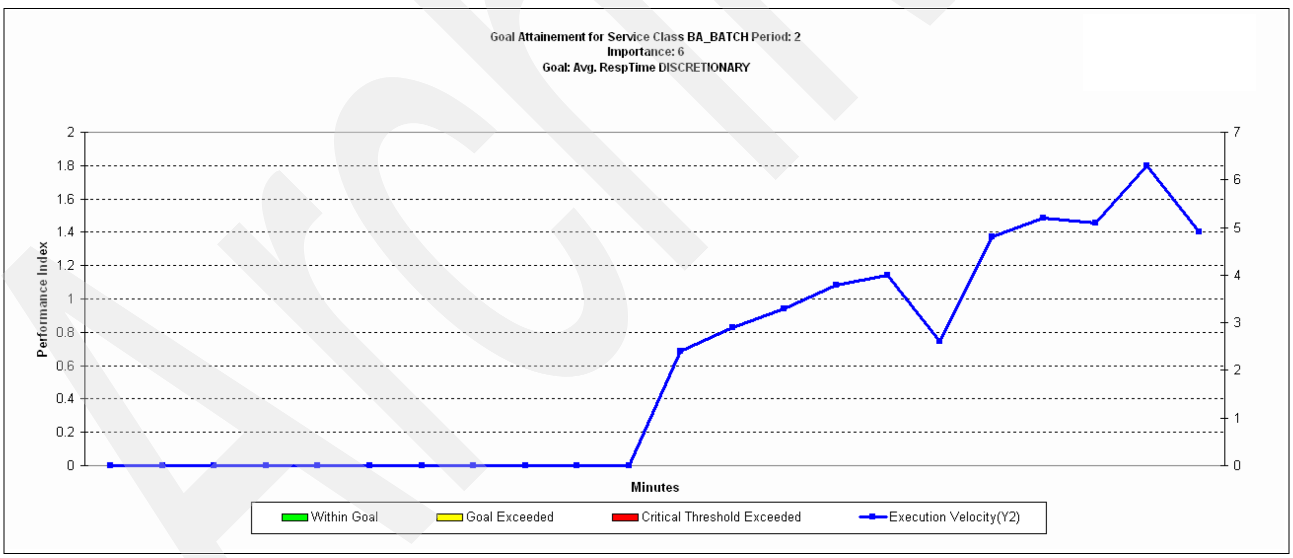


Figure 11-57 Discretionary scenario: goal attainment for data mining workload

Analysis and conclusions

Discretionary goals provide a way of getting work done by taking advantage of spare capacity without impact on other workloads. Discretionary work can make it easier to manage an extremely busy system. A z/OS system can run at 100% CPU 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.

The WLM definitions applied during this section may appear simplistic in a production system. A more mature implementation of WLM policies could look like the definitions shown in Table 11-22 for batch processing and Table 11-23 for BI workload.

Table 11-22 BI analysis, batch processing

Period	Duration	Importance	Goal
1	80000	2	90% RT < 5s
2	700000	3	Velocity = 60
3	1000000	4	Velocity = 40
4			Discretionary

Table 11-23 Business Intelligence

Period	Duration	Importance	Goal
1	5000	2	90% RT < 2s
2	50000	2	90% RT < 10s
3	100000	3	Velocity = 60
4	1000000	4	Velocity = 40
5			Discretionary

12

Extract, transform & load

Extract, transform, and load (ETL) was not within the scope of the implementation part for this book. However, ETL is a important part of a complete data warehouse solution and typically included as a dedicated architecture and system component.

This chapter discusses general approaches for ETL on System z as well as tools and techniques for extracting data from various sources, transforming data to the appropriate format, and loading the transformed data into the data warehouse.

12.1 An overview of ETL

ETL is a term that refers to the processes that extract information from an online transaction processing (OLTP) system and other sources, transform it according to the needs of the data warehousing environment (this can also include quality requirements), and load this data into the data warehouse.

The extraction process takes the data from various sources. Each of these source systems may use a different data organization, such as relational databases, hierarchical databases, network databases, flat files, and even unstructured data. Depending on the source of the data, you may need different techniques for extraction. The extraction process then converts the data into a format that the transformation process can use.

The transformation process applies various rules and functions to the extracted data to change the source data into a form that is useful for the DW. This may include converting to the required data structure, aggregation, converting code, joining data from multiple sources to derive new values, and so forth. The transformation process may also include cleansing of data, such as removing duplicates and standardizing data formats.

In the load process, the transformed data is stored into the target, the DW. When the DW is built for the first time, data needs to be extracted from various sources, transformed, and loaded one time. This is called the initial load. After the initial load, the DW has to be kept relevant by updating the data from the all the sources. This process is known as data refresh.

Figure 12-1 shows where ETL processing in a data warehouse takes place. First, data from (operational) data sources are loaded into the enterprise data warehouse. Second, data marts are derived from data in the enterprise data warehouse.

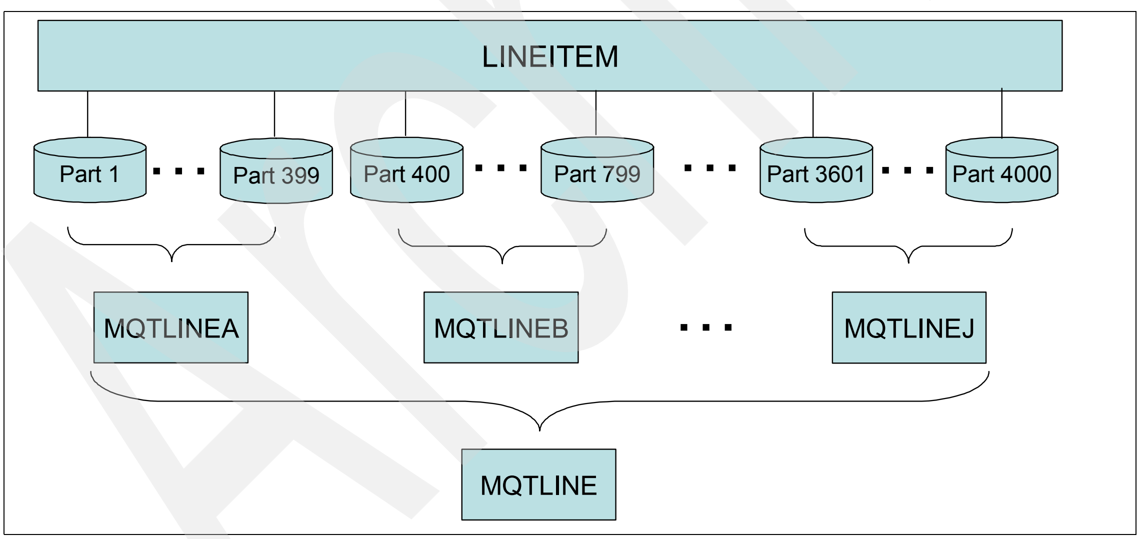


Figure 12-1 ETL processing in an enterprise data warehouse

Traditionally, for data refresh, data warehouses extract data from the operational storage through batch processes executed within maintenance windows. The batch extract jobs are run perhaps once a month, but could be run as often as once a week, depending on the business requirement and batch window availability. However, the need to obtain the most recent data is becoming more and more important.

Another challenge in the batch update is the volume of data. A monthly data refresh will have to handle a large volume of data. This will require a much longer batch window. A continuous trickle feed may be a better option in this situation. As soon as new data is generated in the OLTP system, it is captured and passed to the ETL process in near real time. Using the trickle feed approach, the information in a DW is kept as recent as possible and you avoid a batch window bottle neck.

Some of the data extracted from the OLTP may not need a complex transformation. It can be loaded directly to the DW with only minor modifications. In this case, data replication may be another option to consider. Data replication captures data from the data source as soon as it is generated, performs minor modifications, and then inserts it into the target database.

In most data warehouse environments, the ETL process is a combination of batch ETL, near real time ETL, and replication solutions, depending on the business needs.

The locations where you run the ETL processes are an important consideration. The extract process preferably runs on the platform where the source data is located. The compute-intensive transformation process could run on a different platform, but this implies moving a large amount of data between platforms. Finally, if load processing runs on a different platform from the target system or where transformation took place, data has to be moved again. There are many techniques available for data movement. Whichever technique you employ for data movement, this is one of the biggest challenges in a DW environment. Because most of your source data comes from the OLTP environment, we strongly suggest hosting your data warehouse on the same platform and running the ETL process on that platform.

ETL processing can be quite complex. There are many operational challenges, such as meeting the service level agreements (SLAs), scaling to the data volumes, and completing within the available batch window. Organizations can develop in-house applications for ETL processing.

IBM offers a rich set of tools for ETL processing on System z. These tools support most of the data organizations you will find today. In addition, they support complex data transformations and cleansing requirements. The tools exploit DB2 LOAD or parallel INSERT capability to load a large volume of data efficiently. The following sections introduce some of these tools.

12.2 Information Server for System z

The IBM InfoSphere Information Server suite of products provides a framework for services around data transformation, data quality assurance, metadata management, SOA services, and other operations. Figure 12-2 on page 282 depicts the four key integration functions in Information Server.

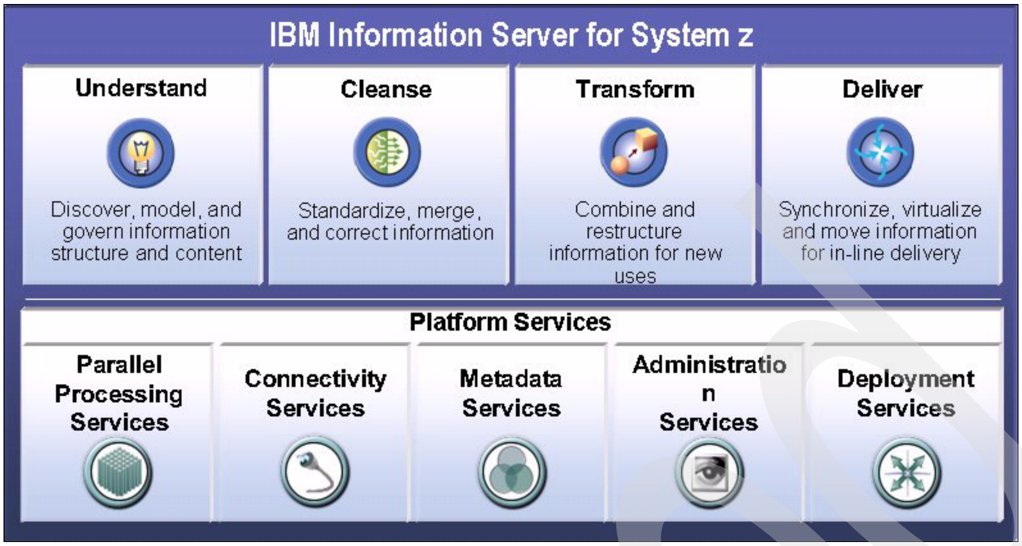


Figure 12-2 IBM Information Server for System z

The following list explains how these components can be used in a data warehouse environment:

- Understand data

  This function is about analyzing data to determine the meaning and relationships of information. Web-based tools can be used to define and annotate fields of business data. Monitoring functions can be used to create reports over time. A metamodel foundation (and the IBM Metadata Server) helps in managing changes in the transactional (OLTP) data model and their implication in the warehouse model and derived reports.

- Cleanse data

  In a data warehouse environment, multiple data sources maintained by different applications are to be integrated. Consequently, the data values in the transactional system may be stored in different formats and independent systems are likely to duplicate data. The QualityStage component in Information Server supports consolidation, validation, and standardizing of data from these multiple data sources and thereby helps to build a consistent, accurate, and comprehensive warehouse model.

- Transform data

  Transforming data in the context of an ETL process is a major requirement for a comprehensive data warehouse solution. The DataStage component in Information Server provides a variety of connectors and transformation functions for this purpose. High speed join and sort operations as well as the parallel engine help in implementing high-volume and complex data transformation and movement. A comprehensive set of tools are offered to monitor, manage, and schedule ETL jobs.

- Deliver data

  In addition to the components mentioned before this section, Information Server comes with many ready-to-use native connectors to various data sources, both located on distributed or mainframe systems.

Information Server 8.0.1 comes with a server and a runtime component. For a System z-based implementation, the server component may be installed on Linux for System z. The client components may be installed on a Windows system and are used to model data and design ETL jobs for DataStage and QualityStage.

Note: There is also an Information Server version available for z/OS (running under z/OS UNIX), as well as an MVS version of the product. Both are currently based on older product levels than the one we used on Linux for System z. Particularly when running multiple jobs concurrently, the available product version for Linux for System z may have advantages.

12.3 InfoSphere Classic Federation

A data warehouse may collect, consolidate and aggregate data from various sources and various source types. Not all sources are relational data sources, as DB2 for z/OS is. Some data might reside on System z in the following sources:

- IMS
- VSAM data sets
- ADABAS
- IDMS
- DATACOM
- Flat files

This data (or at least aggregates of it) needs to be integrated into the data warehouse so that all information becomes available in a single place.

InfoSphere Classic Federation can access this data in a way that makes it unnecessary to set up a specialized extraction process for each of the different data sources (such as hierarchical or relational databases).

The InfoSphere Classic Federation Data Server is installed where the data sources reside. A connector is configured and started for each data source that needs to be accessed. These connectors are used by the data server to access the source data from IMS, VSAM, ADABAS, or other supported sources. The data server maps the various source data structures to a relational structure. So all data accessed through the InfoSphere Classic Federation Data Server looks like one relational database, even if the source is structured hierarchically.

Mapping between source and target definitions is done with the Classic Data Architect, which is an Eclipse-based workstation tool. It allows importing IMS DBD or COBOL copybooks to obtain the structure definitions of source data. The user can select the information to include in a target table, which is simulated as a relational structure by the data server. No further configuration or coding is necessary.

An ETL server may access this data through JDBC or ODBC to move it to the data warehouse in the form of staging tables or aggregates. The ETL Server and the WS Classic Federation Data Server are primarily used for either an initial load or a full reload of the DW. Data in legacy data sources can be updated as well, but this is usually not required within a data warehouse environment. See Figure 12-3 on page 284 for an overview of components that come with and interact with InfoSphere Classic Federation Server.

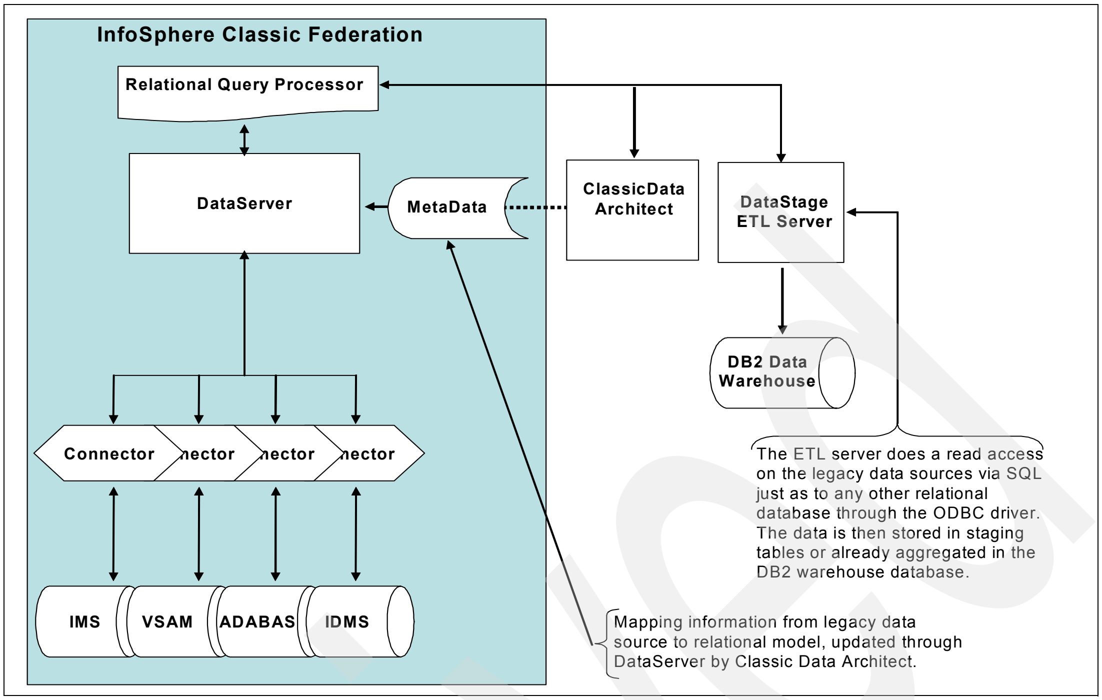


Figure 12-3 InfoSphere Classic Federation

12.4 IBM InfoSphere Change Data Capture

IBM InfoSphere Change Data Capture and IBM InfoSphere Change Data Capture for Oracle Replication can help gain the following advantages:

- Load real-time information into a data warehouse or operational data store
- Eliminate the batch window by providing continuous ETL of data into target systems
- Integrate real-time inventory and sales information with other commercial sites
- Integrate product and customer data across multiple branches, outlets, and offices
- Keep production systems running 24x7 for worldwide operations

12.5 InfoSphere Classic Data Event Publisher

Once the data is loaded into the data warehouse, it usually must be updated incrementally. It is not necessary to replace the entire content. InfoSphere Data Event Publisher and WS Classic Data Event Publisher, respectively, are used to detect the changes in DB2 sources and legacy data sources and to provide the information about the changes to the ETL server.

The Classic Data Event Publisher and the Classic Federation have much code in common. If you installed and configured Classic Federation before, you can extend this configuration to allow event publishing. You must map legacy data structures to relational tables by using Classic Data Architect. You can even configure the server so that this information is used for federation and event publishing at the same time, with just one running data server.

Instead of using a connector to directly access the legacy data sources, a change capture agent is used to detect all manipulations to the data source. In some cases, this is a logger exit (such as IMS, for example). In other cases it is a log reader. The Change Capture Agent sends the change information to the correlation service, which maps the legacy data structure to a relational structure that is designed with Classic Data Architect. Distribution Service and Publisher are used to send the information about the changes through InfoSphere MQ to the DataStage data integration server. The DataStage server is able to read the MQ messages directly from the queue, just as they are transmitted. These events can then be stacked up in staging tables and applied to the data warehouse in a composite update that is run in a batch window (see Figure 12-4).

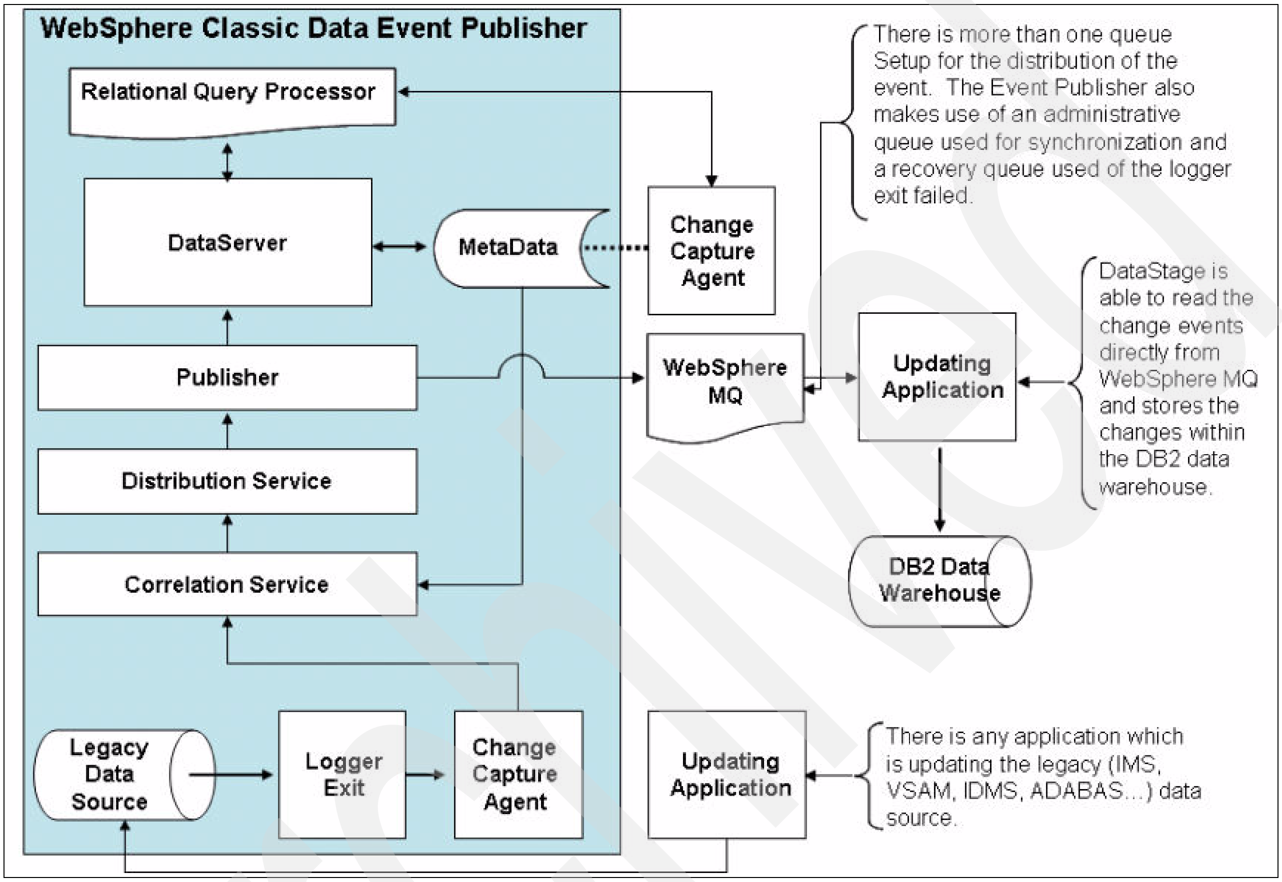


Figure 12-4 InfoSphere Classic Data Event Publisher

12.6 InfoSphere Data Event Publisher for z/OS and Q Replication Server

InfoSphere Data Event Publisher for z/OS offers efficient changed-data solutions for DB2 for z/OS databases.

A Q Capture program reads the DB2 log files for transactions containing changes to the sources in your publications. It places messages containing those transactions or messages containing individual changes on queues. You provide the destination for the messages. You can publish XML messages to a web application, use them to trigger events, and more. You can either start the Q Capture program with the default settings or modify those settings for your environment.

Figure 12-5 on page 286 shows a simple configuration for Q capturing and event publishing. A queue capture control program reads change information from DB2 for z/OS logs and (based on definition in control tables) makes these changes available in a queue.

With InfoSphere Data Event Publisher for z/OS, you can propagate data changes to DataStage and other integration solutions for populating a data warehouse, data mart, or operational data store with the changes.

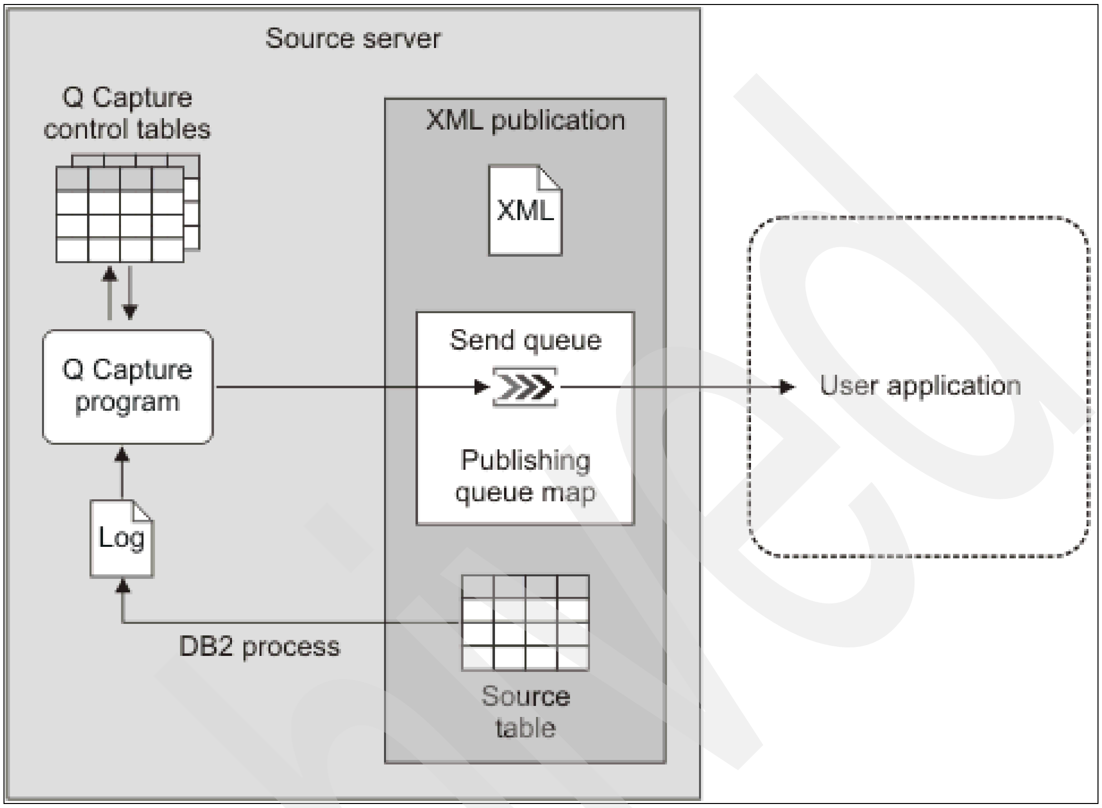


Figure 12-5 Event publishing and Q capturing

12.7 InfoSphere Data Architect

IBM InfoSphere Data Architect (formerly Rational Data Architect or RDA) is a collaborative data design solution to discover, model, relate, and standardize diverse and distributed data assets. It helps data professionals to:

- Facilitate an enterprise-wide understanding of heterogeneous data assets
- Improve data quality, enforce data governance, and ensure enterprise consistency
- Analyze and enforce compliance to enterprise standards
- Promote cross team, cross role and cross organization collaboration
- Discover, explore, and visualize the structure of data sources and their relationships for rapid understanding of current data assets
- Simplify and accelerate integration design for BI, Master Data Management and Service Oriented Architecture initiatives

For more information, see the following Web page:

http://www.ibm.com/software/data/studio/data-architect/

13

Summary—observations and conclusions

The 50 TB Data Warehouse Benchmark on IBM System z describes our experiences in operating a 50 terabyte data warehouse. Our team executed this study from June until October, 2008 at the IBM Worldwide Mainframe Benchmark Center, in Poughkeepsie, New York.

Poughkeepsie New York is one of five mainframe (System z) benchmark centers worldwide. The mission this center is to provide customers with on-demand benchmark capability across eServer™ and TotalStorage solutions. The benchmark center engagements include scalability, performance, and proof-of-concept studies. The engagement team usually comprises the customer (client, ISV, or both), IBM benchmark, and extended teams. Marketing and technical documents and presentations are often written during the engagement.

13.1 Summary

The objective of the 50 TB study was to deliver the proof points of System z scalability in a BI environment and to develop best practices of managing large data warehouses. It showcases the unique value of the IBM System z to perform the following tasks:

- Scale to larger volumes
- Manage a mixed workload, such as operational BI (a large number of users with smaller queries), using Workload manager
- Lower the cost of a BI solution using IBM System z Integrated Information Processors (zIIPs)
- Minimize compression overhead by using hardware compression

13.2 Our test environment

Tests were performed on a 64-way IBM System z10 with 32 four gigabits per second (Gbps) Fiber Connectivity (FICON) channels, using three IBM System Storage DS8300 systems with a total of 345 TB of raw and 267 TB of usable RAID 5 capacity space. See Figure 13-1.

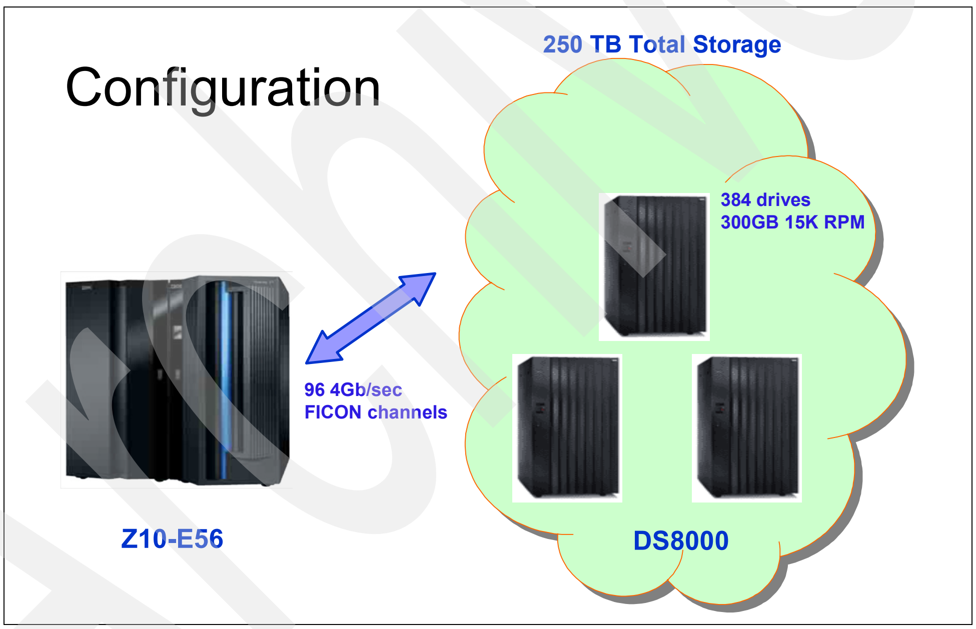


Figure 13-1 CPU and storage

13.2.1 Data placement

Careful planning of the disk configuration was important for such a large environment. An initial study was done to determine the most efficient and effective way of configuring the three DS8000 storage subsystems that would be used during the benchmark. Some of the factors considered were as follows:

1. Size of the largest DB2 object, including the partitioned table space and the partitioned and non-partitioned secondary indexes.
2. Maximum number of sort work data sets that can be allocated in DFSORT. (DB2 calls DFSORT when running utilities such as REBUILD INDEX that require sorting.)

We implemented DFhSMS for this study. The various types of files were categorized and assigned to four main storage groups that contained all table spaces and indexes, DB2 work files, DFSORT work files, and load input data files.

A number of volumes were carved up to meet the space requirements for each file category. The disk volumes within a STORGRP were carefully defined, making sure they were located across multiple control units, clusters and even storage boxes. Figure 13-2 illustrates this strategy. The number of volumes required for each STORGRP was determined and calculated based on the results of a preliminary study conducted before the benchmark's execution.

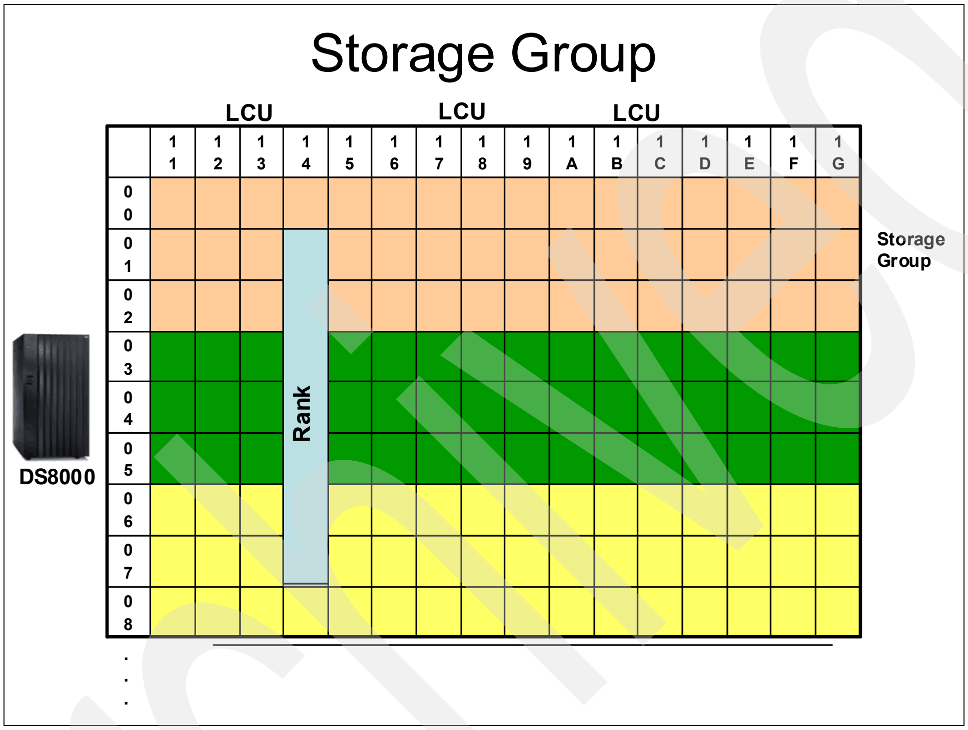


Figure 13-2 Storage Group

Understanding the type of data and size of the largest data set has lead us to define two data classes, COMPRESS and DB2EXAD. The first one allowed hardware compression for files created, resulting in a reduced disk space requirement and sometimes shorter elapsed times and less CPU time. DB2EXAD allows extended addressability on a single data set, a requirement when creating a file larger than 4 GB.

Going through such planning is important, especially when building a very large data warehouse. This enables ease in tracking the growth of data. It helps avoid logical volume contention. It enables random allocation with an even data distribution. This approach is suitable for single or multiple DS8300 storage subsystem configurations.

13.2.2 Database design

The 50 TB database consisted of nine tables. The largest had a total of 300 billion rows in 4000 partitions, while most of the other tables had 2000 partitions. The table spaces were created with a DSSIZE of 8 GB and 16 GB and a PAGESIZE of mostly 8 K and 16 K. The tables were partitioned by keys and not by time. The table data were compressed, but the indexes were not compressed.

13.2.3 Database build strategy

Building a database of this magnitude can take a lot of resources (time, machine, people) if not planned for properly. As part of our preliminary study, we took the CPU time and elapsed time required to load a partition for each table. Based on that information, we made up a strategy on how we should populate the tables using the LOAD utility. We determined the number of load jobs to run concurrently and estimated the amount of time it will take to load all the data for each table and the whole database.

DB2 data loading is CPU intensive. It takes about 1 CP to load a partition. To maximize the CPU cycles available, we ran multiple load jobs concurrently. Since we had 54 CPs available, we ran between 40 and 50 load jobs in parallel and shortened the total elapsed times to load all the data in a table. Using this strategy, we accomplished a load rate of four million rows per second or 1.6 TB per hour. The number of tasks (18) for rebuilding the index was constrained by the amount of workfiles available in the system.

Important considerations for data load and index build

1. Define only the data partitioned index prior to data load. This reduces the elapsed and CPU time, as well as work files and memory requirements.
2. Include online statistics when loading data. This saves time and eliminates the effort of setting up and running another set of jobs to collect catalog statistics.
3. Run one load job for each partition. This reduces load resource requirements per job, especially when partition size is large.
4. Create the non-partitioned secondary index (NPSI) prior to load with the DEFER YES option. Then run REBUILD INDEX after the data is loaded.
5. Use DB2-managed work file allocation. Large tables may require manual allocation. For the large nonpartitioned secondary indexes (NPSI), we implemented these choices:
   a. Disabled the use of hiperspace in DFSORT by coding HIPRMAX=0 under DFSPARM
   b. Specified DSNTYPE=LARGE to allocate the large sortwork file
   c. The preferred approach to sort work file allocation for index rebuilds is to let DB2 make the decision of how to allocate the files dynamically. This works well in most situations; however it fails during large index builds. In this study, we built indexes as large as 4 TB. When the number of keys passed to DFSORT is large, intermediate merges are triggered. This requires additional work space that DB2 does not anticipate, resulting in B37 abends. The solution is to manually allocate the sort work files in this situation. A sample "rebuild index" job is shown in Example 13-1.

   The number of tasks (18) for the index rebuild was chosen based on the number of workfiles available in the system. The number of sort work files per task (9) was because the indexes we built were very large, so we needed a large number of work files to support the build process.

Example 13-1 Sample JCL to rebuild an index

```
//SW01WK01 DD UNIT=SYSDA,SPACE=(CYL,(60000)),
//     DISP=(NEW,DELETE,DELETE),DSNTYPE=LARGE
  < lines skipped >
//SW18WK09 DD UNIT=SYSDA,SPACE=(CYL,(60000)),
//     DISP=(NEW,DELETE,DELETE),DSNTYPE=LARGE
//SORTDIAG DD DUMMY
//DFSPARM DD *
HIPRMAX=0
/*
//SYSIN    DD *
REBUILD INDEX (TPCH50TB.NW_L_CK)
STATISTICS KEYCARD
```

d. SPACE = (CYL,(60000)) was chosen because we wanted to ensure that we could use up all the space of a volume. If we had used primary and secondary quantities instead, we might have caused fragmentation and been unable to allocate more extents. Sort work file sizing is described in utilities manuals; see the shaded box for an excerpt.

e. For our study, we requested only KEYCARD for our REBUILD inline STATISTICS. This is because our TPCH database had uniform data distribution. For production sytems, additional statistics such as FREQVAL, NUMCOLS, and COUNT should be collected.

13.2.4 zIIP redirected processes

In the 50 TB study, several test case scenarios were designed and implemented to verify the effects of the zIIP engines in a data warehouse environment. For our experiments, we configured an LPAR with 13 zIIP engines and 19 general CPs. We ran a mixed workload simulating various operational BI queries, comprising of 384 large, 1872 medium, and 2800 small queries. Rational was used as the workload simulator tool, with the queries running from Cognos.

There are certain workloads eligible to be redirected to zIIP processors. These are parallel queries (including star joins), remote access to DB2 through TCP/IP using DRDA protocol, remote invocation of native SQL procedure (new in DB2 9), and index build and maintenance in utilities. When designing the test case scenarios to verify the effects of zIIP processors in a data warehouse environment, we considered the following points.

1. The zIIP processors offer significant hardware and software savings.
2. The number of zIIP processors cannot exceed the number of general processors in a physical server. However, an LPAR can be configured to contain more zIIPs than general processors.
3. A high percentage of parallel task activities are eligible to run on zIIP. However, the actual redirected percentage depends on the ratio of parallel to non-parallel work and the amount of available zIIP capacity.
4. RMF reports provide projections of redirected percentage prior to installation of the zIIP processors, but actual redirection is probably lower due to the z/OS dispatcher algorithm.
5. The benchmark with internal workloads indicates redirection between 50% and 70%.

Figure 13-3 shows no performance degradation when the workload was run with 19 zIIP processors and 13 general CPs.

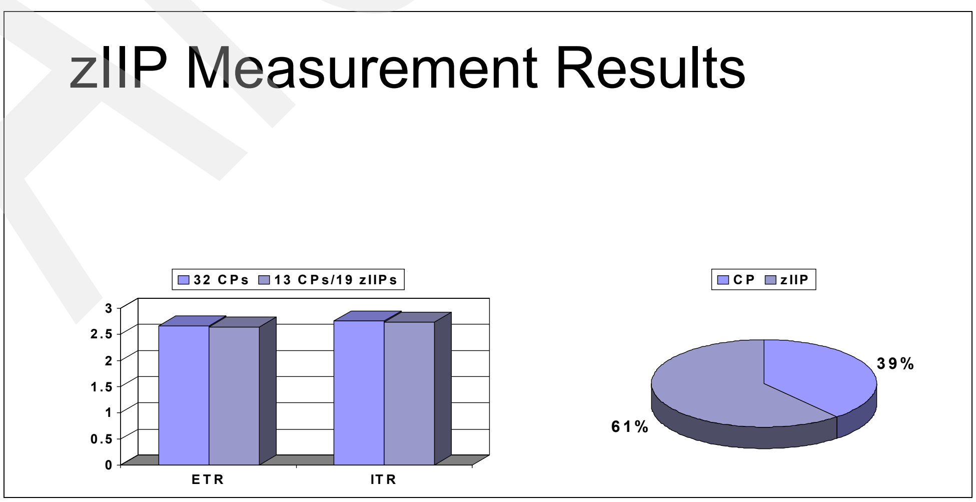


Figure 13-3 zIIP performance results

Based on the results of our 50 TB study, we suggest that you consider the following factors when planning for the number of zIIP processors required for a data warehousing environment:

1. There is a higher redirection percentage from query parallelism, which most data warehousing queries have.
2. A configuration with more zIIPs than CPs could yield higher redirection percentages. If upgrading an existing server, and the number of your zIIP CPs is less than half of the total CPs, you can obtain additional zIIPs as opposed to general CPs. You can then set up the data warehousing LPAR to have more zIIPs than GCPs. This will result in a lower cost for processing your DW workload.
3. Virtual storage management. In DB2 9, a significant amount of thread related storage is still below the virtual storage bar. A higher degree of parallelism increases virtual storage consumption proportionally. Virtual storage consumption per parallel task is proportional to the number of partitions of tables. To cap the maximum degree per query, set a DB2 system parameter PARAMDEG.
4. CPU monopolization A single query with high degree of parallelism can consume all the CPU capacity of a system. Tune your WLM policy to control parallel query CPU consumption.

13.2.5 Query performance

For this study, scalability and performance tests were performed for an organizational data warehouse built around a single large DB2 9 for z/OS instance containing 50 TB of raw data and approximately 20 TB of indexes. We ran various test case scenarios to measure query performance on a data warehouse environment. The query mix was designed to reflect the variety of query types and sizes that characterize typical data warehouse workloads.

13.2.6 CP scalability

Another objective of the 50 TB study was to determine the scalability of running a CPU-intensive query on different CP configurations. For this set of tests, we ran a single query on 8, 16, 32, and 56 CPs. We plotted the elapsed times of the query for each scenario. Figure 13-4 on page 293 shows elapsed time gains of 99%, 98%, and 95% by increasing the number of CPs from 8 to 16, then to 32, and to 56. The results proved that there is a proportional elapsed time gain each time we increased the number of CPs.

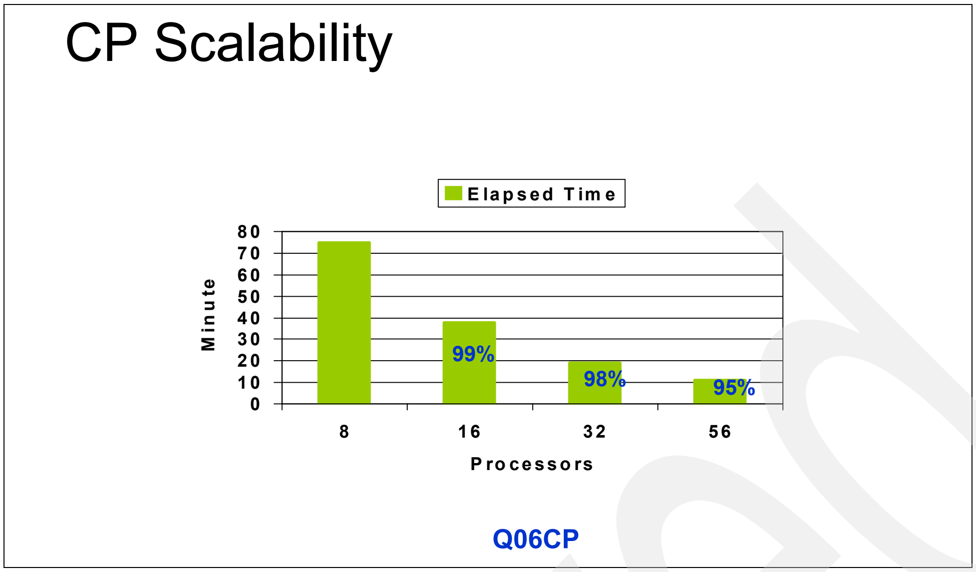


Figure 13-4 Proportional elapsed time gain

13.2.7 Data scalability

Our experiments showed linear data scalability as well. For the test case scenarios, the size of the database was varied from 1 to 50 terabytes. The results were linear, as shown in Figure 13-5.

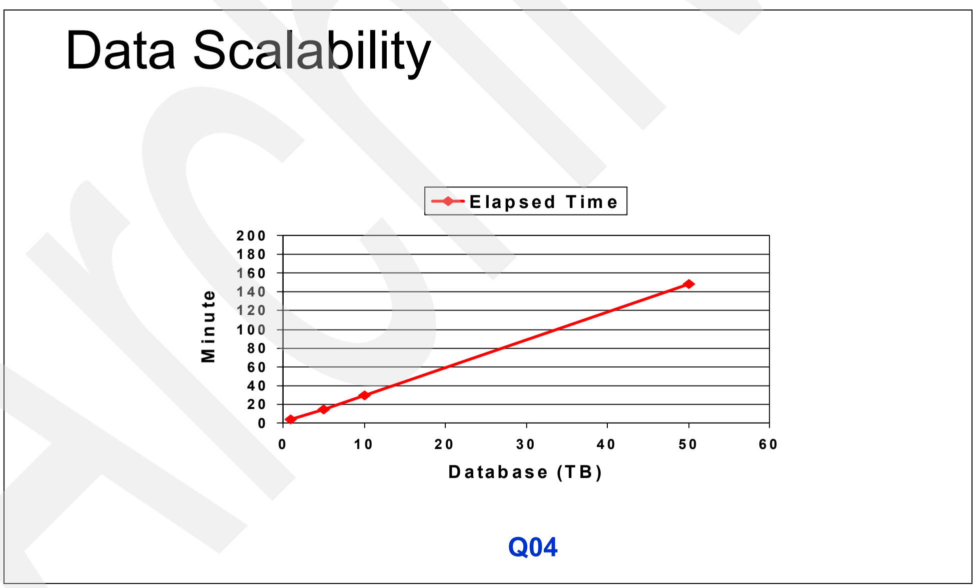


Figure 13-5 Linear scalability

13.2.8 Compression

DB2 compression should be strongly considered in a data warehousing environment. In our study, we observed a space savings of 50% or more when data compression was enabled. This means more tables can be located in the buffer pools. Data warehousing queries are dominated by sequential prefetch accesses which benefits greatly from DB2 compression. table space scans may have more overhead but elapsed time might be improved dramatically because of increased efficiency of data access.

It is important to note that previous System z processors used microcode for compression and decompression instructions, involving overhead. Newer generations of z processors implement instructions directly in circuits, yielding lower overhead. Their instruction design favors decompression over compression.

13.2.9 Materialized Query Tables

We sent our schema and queries to the IBM DB2 development lab in Toronto for advice. Once we defined the set of queries, we built the Materialized Query Tables (MQTs), using intermediate tables to reduce sorting requirements for large tables. See Figure 13-6. To avoid a complete refresh of the tables, we used LOAD with CURSOR.

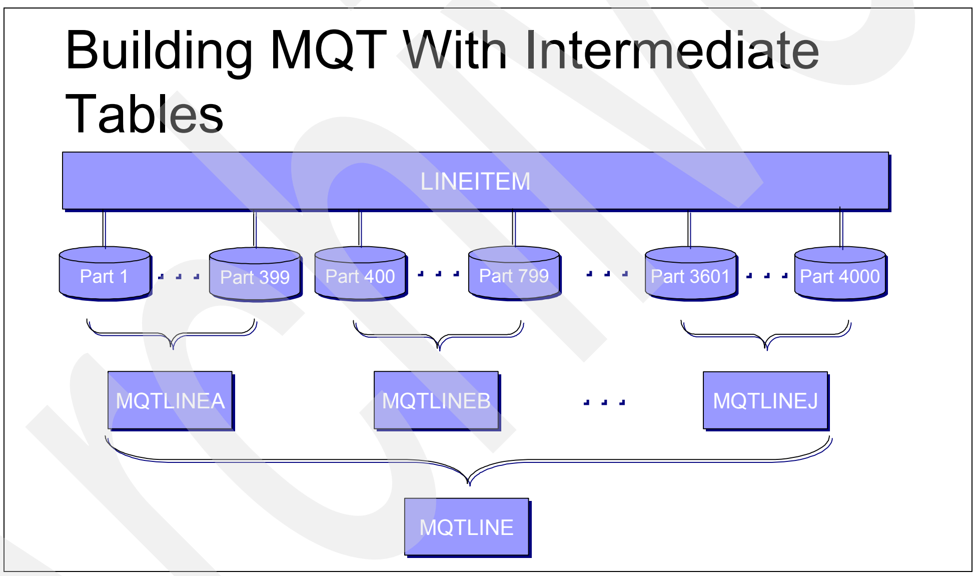


Figure 13-6

As expected, query response times were significantly reduced when using MQTs. Without MQTs, queries ran in parallel (degree of 160), but consumed a significant amount of CP time. In contrast, the use of MQTs drastically reduced response time to sub-second, and the CP time was reduce likewise. Figure 13-7 illustrates this contrast.

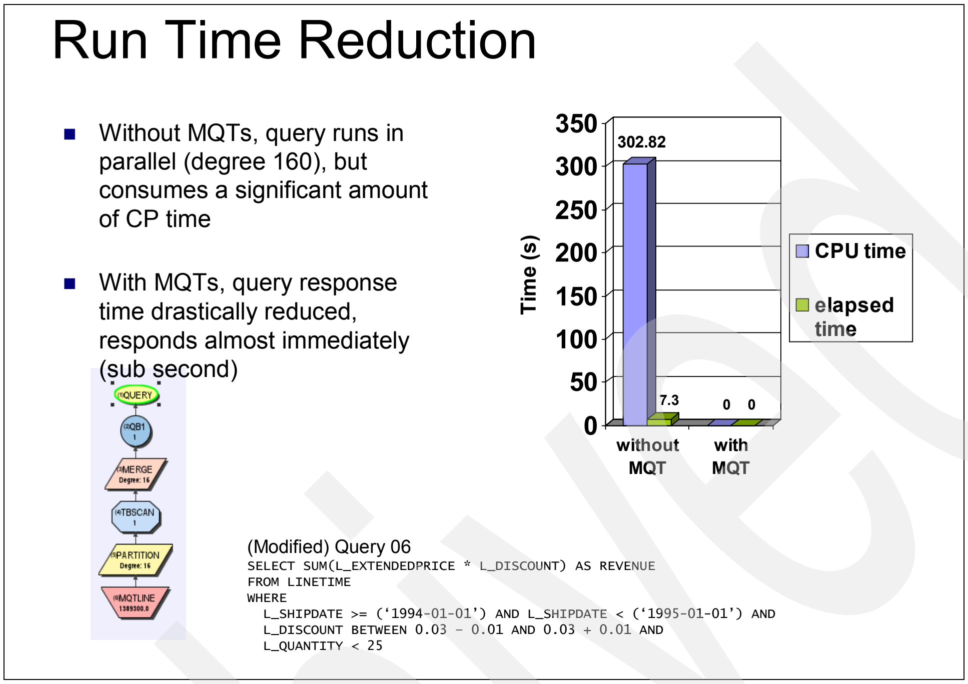
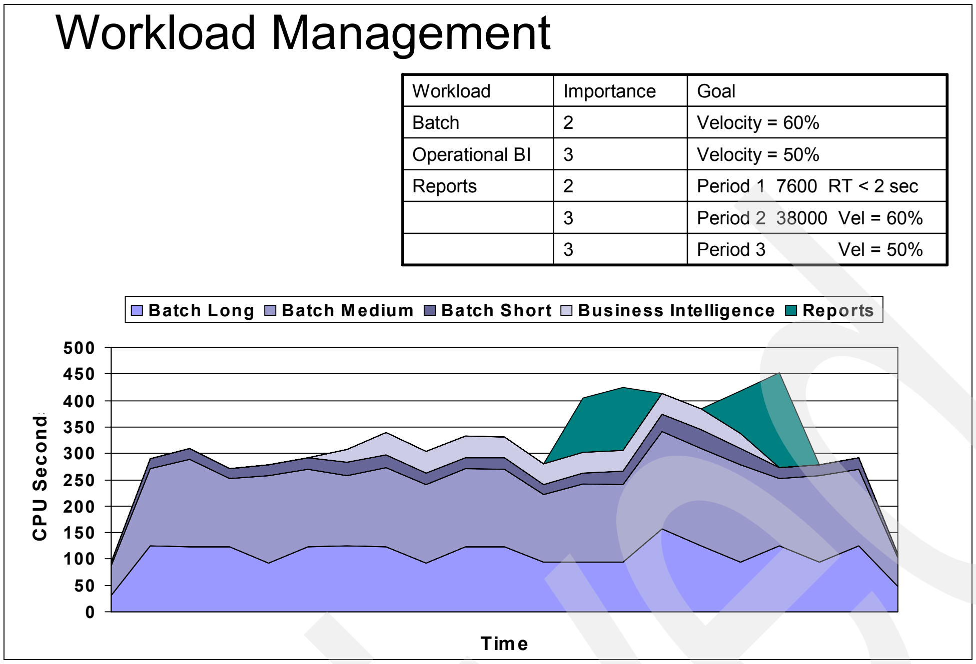


Figure 13-7 Response time reduction with MQTs

13.2.10 Workload management

A wide spectrum of people use BI applications in an enterprise. These diverse workloads consume different system resources and demand different priorities. It is therefore important to have a process in place to monitor and manage these workloads properly. IBM z/OS Workload manager (WLM) offers this capability.

For our study, we measured the effectiveness of WLM facilities in assigning priority to a subset of critical queries within a larger query workload. This was implemented by varying the definitions of certain service classes, running a consistent workload, and then comparing the performance results from various test scenarios. In the end, we had a WLM configuration that looks like Figure 13-8 on page 296.



Workload	Importance	Goal
Batch	2	Velocity = 60%
Operational BI	3	Velocity = 50%
Reports	2	Period 1 7600 RT < 2 sec
	3	Period 2 38000 Vel = 60%
	3	Period 3 Vel = 50%

Figure 13-8 Our WLM configuration

The following section describes some basic information and recommendations in setting up WLM for a data warehouse environment.

Basic WLM information and recommendations

- Measurement of system takes place every 250 milliseconds. Every 10 seconds, WLM summarizes the system state. WLM only makes 1 change during the 10 second interval.
- DDF creates an independent enclave for incoming requests. Define classification rules for the incoming DDF work; otherwise all enclaves will run under the default service class for low priority work.
- Response time goals apply to the time that the query spends in the server, not to end-to-end response time.
- Verify that service classes are being used to manage work and report classes to monitor work. Defining too many service classes could create performance degradation. Exploit report classes instead.
- Keep the overall number of active service class periods between 25 and 35.
- Ensure that the DB2 system parameter CMTSTAT is set to INACTIVE. This is required so that DB2 and WLM can accurately instrument the workloads based on "DB2 Commit" representing the end of a query (or transaction).

Guidelines and considerations

When evaluating a WLM policy for a data warehouse mixed workload, consider these points:

- In general, consider mix of response time, velocity, and discretionary goals using multi-period service classes for the DB2 portion of your BI workloads
- For queries that have similar processing resource requirements, moderate to high completion rates, and generally shorter response times (seconds to minutes), consider using response time periods. The WLM recommendation is a completion rate of at least 0.1 queries/second to have ample history for efficient management of response time

goals. This work might be more canned or traditional small resource consumer reporting, Operational BI or ODS access, though it could include high-concurrency ad hoc work.

- For queries that tend to be large with wide varying resource consumption profiles, velocity and discretionary goals might be more appropriate. These queries would tend to be more ad hoc in nature. Also these workloads have a higher likelihood of containing an occasional poorly written or ill-generated SQL that could consume large amounts of system resource.
- When it is not easy to differentiate (through classification rules) between the two different types of query workloads listed above, an approach would be to use one multi-period service class, using response time goals for the early periods, velocity goals for the higher periods, and ending with a discretionary goal. This will cover work that exceeds some reasonably anticipated amount of service.
- Consider using multiple service classes to manage various user groups according to their importance to the business. (such as executives, power users, business analysts, sales managers, VIP business clients, and general business clients). Note this does not necessarily require a different service class for all types of users. Work that has similar goals should use a common service class.
- Consider using multiple service classes to manage online query work, which is different from scheduled work. Online work refers to work associated with an end-user sitting at a terminal waiting on a response. Schedule work refers to work that is scheduled to run at a certain time of the day or dependent upon some other process; for example, a data warehouse refresh.
- Consider using multiple policies to change the goals for service classes/workloads whose importance to the business is time/date dependent (for example, a data warehouse refresh).
- For work that does not have a concrete goal to achieve, consider using a pure discretionary service class. This only run when all the other service classes are meeting their goals and there are still available resources. This is a good way to take care of the low priority work in the system.

13.2.11 Operational BI queries with Cognos

We recognize the importance of Operational BI in today's overall data warehousing solution. With this in mind, we developed a workload to represent this type of environment. We designed test case scenarios to understand the factors that affects the performance of operational BI queries. To simulate a production workload, we used Cognos, which is a software product for BI and corporate performance management.

Why is operational BI important?

Operational BI is about an organization's performance in the present, not the past. It provides the organization with near-time view using dashboards, matrixes, and reports. It helps link high level organization with the day to day tasks performed by operational employees.

Operational BI tools, such as Cognos, deliver BI to customer-facing humans and applications. They integrate the BI components with the operational systems and information portals. The applications use a massive number of queries, some of which are semi-aggregated data.

Figure 13-9 on page 298 is a snapshot of a Cognos report from our tests. When the customer number was entered as input, the tool created various charts and graphs to show business information about the customer. In this example, the report gives a customer data overview, an analysis of historical revenue with this customer, an order history, and an analysis of brands for sales to this customer.

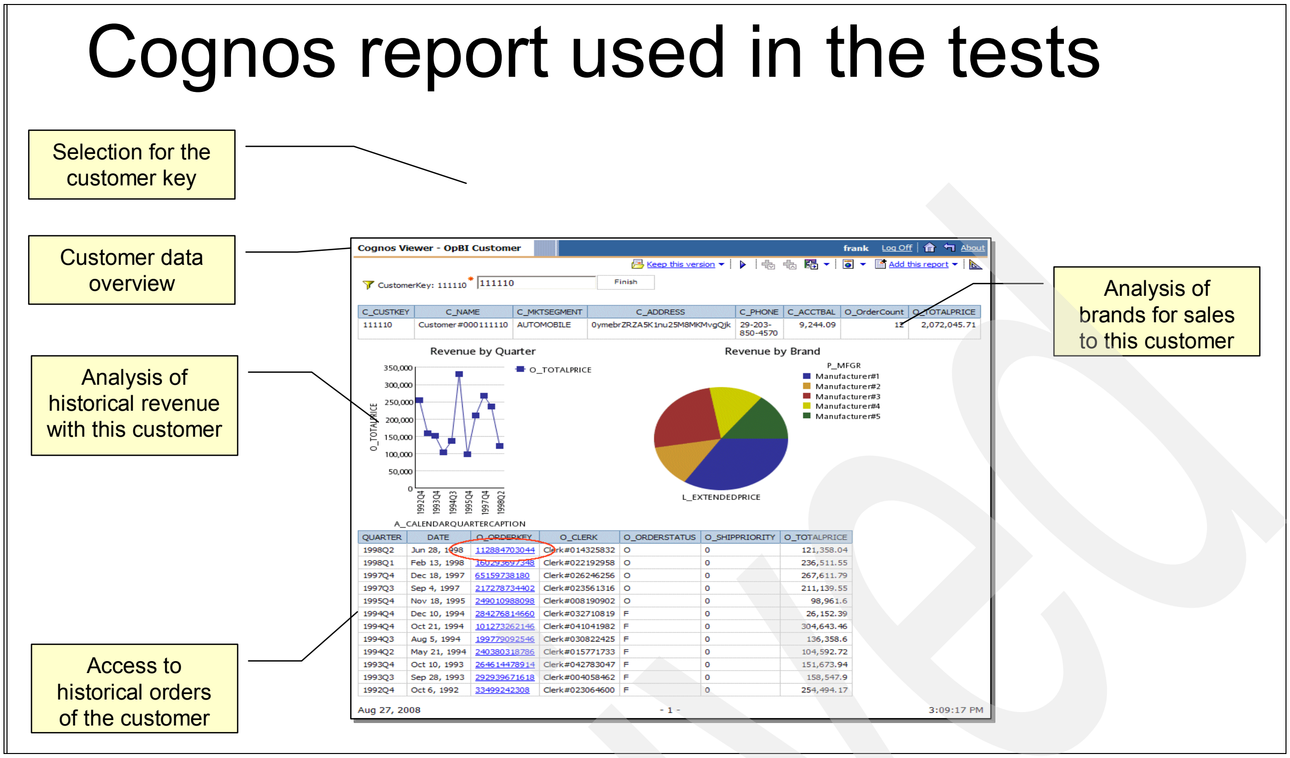


Figure 13-9 Cognos report

Based on the test results, we found that with 8 CPs each on z/OS (data warehouse) and Linux for System z (Cognos), a workload with 50, 200, and 400 users, giving an average transaction time of 4.02, 1.63, and 1.75 seconds respectively, is handled well.

We also observed that the CPU load on Linux was higher than z/OS. The number of configured report server processes (10, 14, 16) did not have any big impact for the tested scenarios, as shown in Figure 13-10.

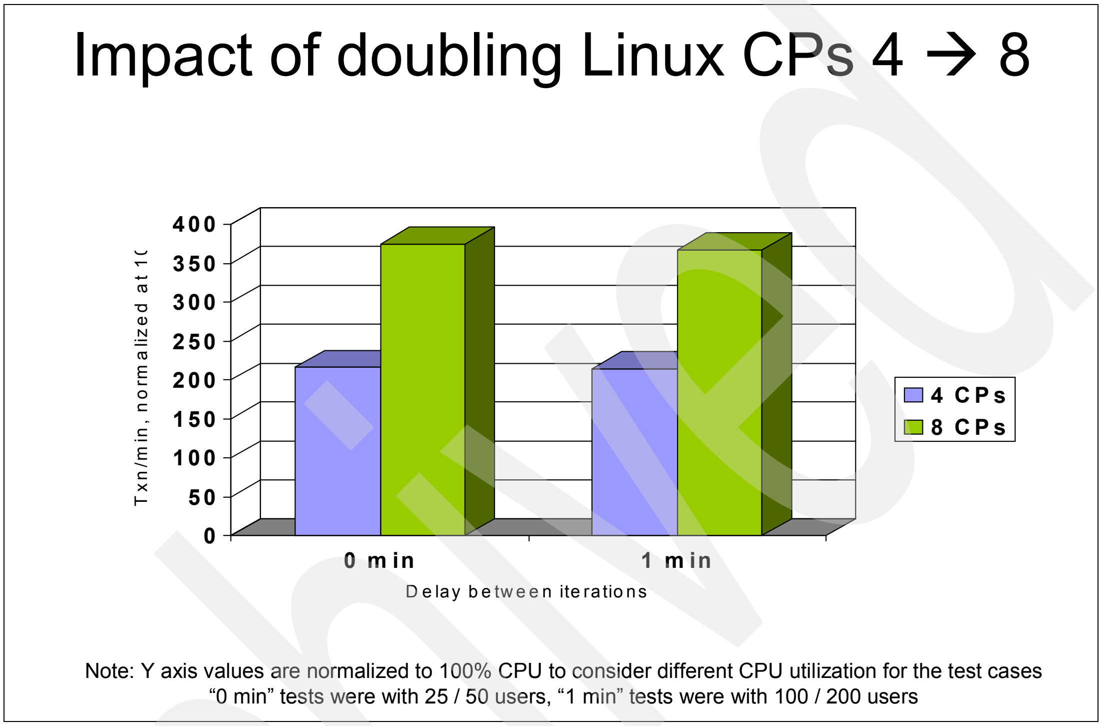


Figure 13-10 Doubling CPs for Linux on System z

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 301. Note that some of the documents referenced here may be available in softcopy only.

- *IBM DB2 9 for z/OS: New Tools for Query Optimization*, SG24-7421
- *News on DB2 Optimization Expert for z/OS V1.2 and DB2 Optimization Service Center Version 1.1, TIPS0673*
- *DB2 for OS/390 and Data Compression*, SG24-5261
- *Index Compression with DB2 9 for z/OS*, REDP-4345
- *Enterprise Data Warehousing with DB2 9 for z/OS*, SG24-7637

Other publications

These publications are also relevant as further information sources:

- "DB2 for z/OS Data and Index Compression", presentation by Willie Favero
- *DB2 9 for z/OS Performance Topics*, SG24-7473

Online resources

These Web sites are also relevant as further information sources:

- DB2 for z/OS

  http://www.ibm.com/software/data/db2/zos/
- Getting the most out of DB2 for z/OS and System z—Willie Favero's blog

  http://it.toolbox.com/blogs/db2zos

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, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

E

F

G

H

I

Q

R

S

50 TB Data Warehouse Benchmark on IBM System z

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

50 TB Data Warehouse Benchmark on IBM System z

Why use System z and IBM storage for Data Warehousing

Tips for building a very large database

Reporting and analysis with Cognos

Customers have a strong interest in the scalability of data warehousing solutions on the IBM System z platform using DB2 for z/OS. Because BI environments continue to grow, this IBM Redbooks publication explores the scaling and management of data in sizes beyond 50 TB. We explore the architectural software components that enable us to manage a data warehouse of this magnitude, including the Cognos product set.

The objective of our 50 TB study was to test System z scalability in a BI environment and to develop the best practices of managing large data warehouses. The IBM System z was proven to scale to larger volumes, manage a mixed workload, lower the cost of a BI solution using zIIPs, and increase performance by using hardware compression.

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

SG24-7674-00 0738432741