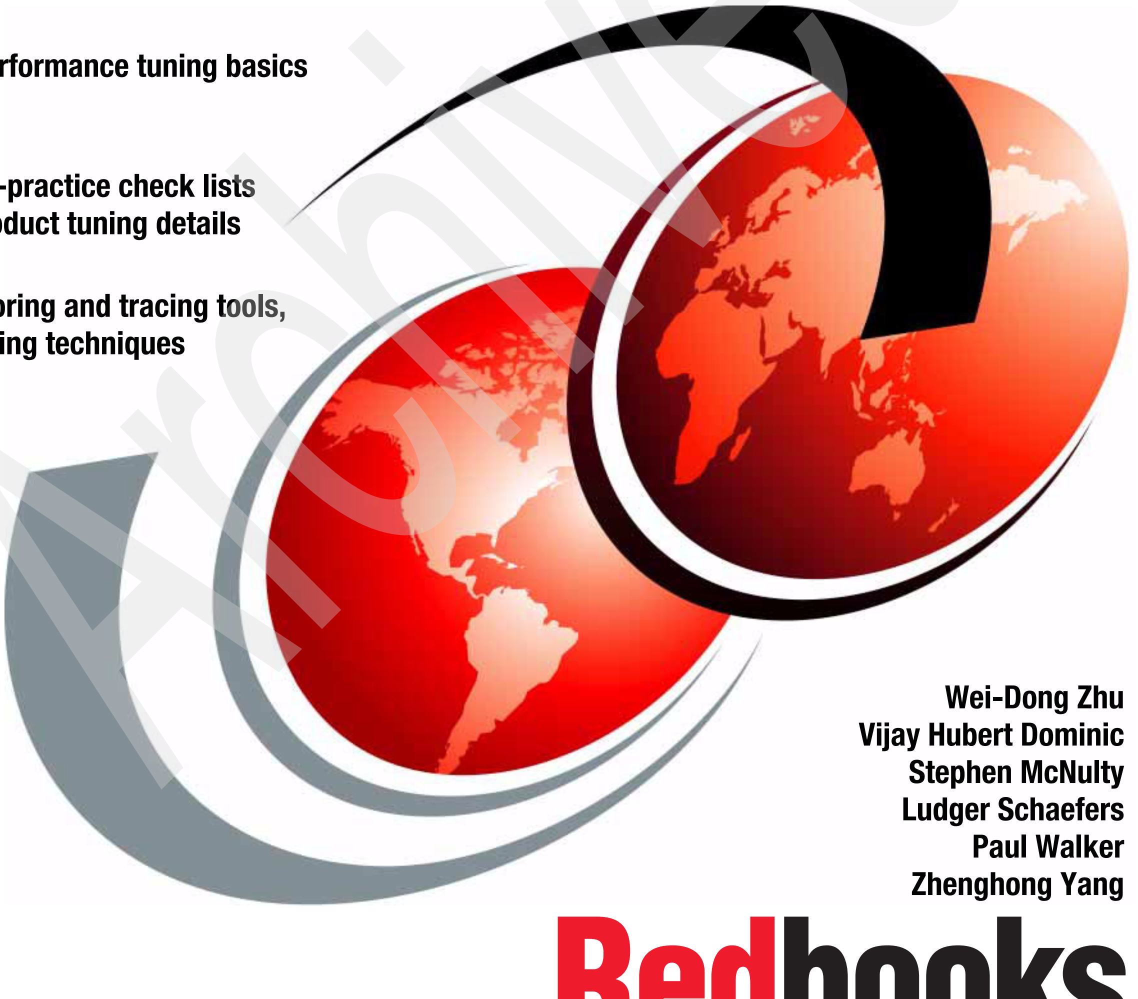
IBM

Performance Tuning for Content Manager



Introduce performance tuning basics

Provide best-practice check lists and base product tuning details

Cover monitoring and tracing tools, troubleshooting techniques

Wei-Dong Zhu
Vijay Hubert Dominic
Stephen McNulty
Ludger Schaefers
Paul Walker
Zhenghong Yang

Redbooks

ibm.com/redbooks

International Technical Support Organization

Performance Tuning forContent Manager

July 2006

SG24-6949-01

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

Second Edition (July 2006)

This edition applies to Version 8, Release 3 of IBM Content Manager for Multiplatforms (product number 5724-B19).

Contents

Notices

Trademarks

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

AIX®	IBM®	TotalStorage®
AIX 5L™	iSeries™	VideoCharger™
AS/400®	Lotus®	WebSphere®
DB2®	OS/2®	z/OS®
DB2 Connect™	Redbooks™	1-2-3®
DB2 Universal Database™	Redbooks (logo) ™	
Domino®	Tivoli®	

The following terms are trademarks of other companies:

Enterprise JavaBeans, EJB, Java, Java Naming and Directory Interface, JavaBeans, JavaMail, JavaServer, JavaServer Pages, JDBC, JMX, JSP, JVM, J2EE, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Excel, Internet Explorer, Microsoft, Windows NT, Windows Server, Windows, Win32, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Pentium, 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

This IBM® Redbook deals with performance tuning for IBM DB2® Content Manager Version 8 for Multiplatforms. Written for architects, developers, and system administrators, it is a guide to help plan, configure, and tune a Content Manager system for performance. Performance problems arise from poor planning, poor design, monitoring, and maintenance, so we want to make sure this book serves as a guide to help you from day one of system planning, and channels you through the right path for a successful system.

In Part 1, we start with an overview of Content Manager system architecture and an overview on the base products including DB2, WebSphere® Application Server, and Tivoli® Storage Manager. To achieve desired performance or troubleshooting performance issues for a Content Manager system, it is important to understand what base products make up the system.

In Part 2, we introduce performance tuning basics. We define what performance is, how it is measured, why we need to do it, and when to plan and tune your system for performance. We cover performance methodology, and describe the performance improvement process along with a set of general guidelines that you should use for planning a new system or maintaining and improving an existing system.

In Part 3, we summarize the best practices for tuning a Content Manager system for performance using check lists and a flowchart. For details, we cover the tuning of the operating system, DB2, WebSphere Application Server, and Tivoli Storage Manager for Content Manager. In most cases, for each parameter tuning, we describe what the parameter is used for, the default value if any, how to view, set, and update it, and our recommendation based on our experience.

In Part 4, we cover monitoring tools at the operating system level, monitoring and analysis tools for DB2 including techniques used with these tools, monitoring tools for WebSphere Application Server, and performance tracing for Content Manager. In addition, we describe performance-related troubleshooting techniques based on a collection of real-life scenarios and a case study.

Many performance-related books and articles have been published already, and this book is based heavily on some of these performance publications. Many thanks to the authors, especially the IBM Content Manager Performance Team for their contribution.

We have learned a lot through this experience, and we have tried to capture the knowledge we gained in our writing. By using this book, we hope you capitalize on our experiences as you work with Content Manager.

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.

Wei-Dong (Jackie) Zhu is a Project Leader in Content Management for the International Technical Support Organization at the Almaden Research Center in San Jose, California. She has more than 10 years of software development experience in accounting, image workflow processing, and digital media distribution. She holds a Master of Science degree in Computer Science from the University of Southern California. Jackie joined IBM in 1996.

Vijay Hubert Dominic is a Consultant for DB2 Content Manager and WebSphere Product Center in India Software Labs. He has three years of experience in the IT field. He earned a Master of Computer Applications (MCA) from St Joseph's College, Tiruchirappalli, Tamil Nadu, India. His areas of expertise include solution design, product installation, integration, development, troubleshooting, and technical enablement with Content Manager. He is a Sun™ Certified Java™ Programmer, a DB2 Certified Associate, and an Oracle Certified Associate.

Stephen McNulty is a a Senior Information Technology Specialist in the Portals and Content Management practice with IBM Canada. He holds a three-year Computer Programmer/Analyst degree from Georgian College. With 20 years of experience in the IT field, the past 10 years have been focused on image and content management applications. He is responsible for solution design, application installation, integration, development, and troubleshooting with Content Manager, Content Manager OnDemand, Commonstore for SAP, Commonstore for IBM Lotus® Domino®, and Commonstore for Microsoft® Exchange. He is a certified solution expert for IBM DB2 Content Manager OnDemand for Multiplatforms.

Ludger Schaefers is a Consulting IT Specialist with IBM Business Consulting Services in Walldorf, Germany. He received a PhD in Temporal Relational Databases from the Open University of Hagen, Germany. After joining IBM in 1991, he worked in the area of mobile data communication and mobile access to database systems. Since 1997, he has worked in various areas such as technical pre-sales consulting and solution development with a focus on database-centric environments. His areas of expertise include design, implementation, tuning, and maintenance of database systems.

Paul Walker is an IT Specialist working for Application Management Services in IBM Canada since 1988. He has worked with DB2 as a database administrator on the z/OS®, Windows®, OS/2®, and AIX® platforms providing application design guidance, database design guidance, and performance tuning. He has a bachelors degree in Business Management from Ryerson University, Toronto. He is an IBM Certified DB2 Database Administrator.

Zhenghong Yang is an advisory Software Engineer working for IBM DB2 Content Manager defect support at the IBM Silicon Valley Lab, California. He is an IBM Certified DB2 UDB Solution Expert, IBM Certified AIX Specialist, and IBM Certified DB2 Content Management Solution Expert with approximately eight years of experience in full life cycle application development and IBM product support. He holds a Master of Science degree in Computer Science from the State University of New York at Stony Brook. He has published three books about database management and programming.

Special thanks to the following people who co-authored the first version of this redbook:

Jairo Campos
Paulo E Folkmann
Charlie Rennie
Guirguis Youssef

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

Richard Heffel
Jesse Chen
Dave Royer
Bruce Smith
IBM Content Manager Performance Team, Silicon Valley Lab, USA

Ken Nelson
Glen Dinsmore
Zong Ling
Leonora Wang
Zhenghong Yang
IBM Software Group, Silicon Valliey Lab, USA

Ute Baumbach
Steve Rees
Calisto Zuzarte
IBM Germany and IBM Canada

Emma Jacobs
IBM ITSO, San Jose, USA

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll 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 redbooks to be as helpful as possible. Send us your comments about this or other redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

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

Summary of changes

This section describes the technical changes that were made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-6949-01
for *Performance Tuning for Content Manager*
as created or updated on July 28, 2006.

July 2006, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- Chapter 2, "Content Manager base products" on page 19 includes new performance-related concepts: query execution and access plan, indexes, data placement, clustering, multi-dimensional clustering, materialized query tables, system catalog statistics, table space types: SMS versus DMS, prefetching and I/O servers, coordinator agents, and locking. Also included is routine maintenance overview.
- Chapter 6, "Best practices for Content Manager system performance" on page 143 covers performance tuning best practice check lists for Content Manager.
- Chapter 7, "Tuning operating systems for Content Manager" on page 151 covers common performance tuning considerations for the operating systems including AIX, Windows, and the new platform, Linux®. Also included are the network tuning options.
- Chapter 8, "Tuning DB2 for Content Manager" on page 171 includes more in-depth information on tuning DB2, such as regular routine maintenance and buffer pool tuning. Also included is the SMS to DMS table space conversion.
- Chapter 11, "Performance monitoring, analysis, and tracing" on page 281 covers the baseline measurement, monitoring tools, and techniques to use for performance monitoring, analysis, and tracing in a Content Manager environment. DB2 monitoring tools covered include: DB2 Memory Visualizer, DB2 Health Monitor and DB2 Health Center, db2pd, DB2 Performance

Expert, DB2 Explain, and DB2 Design Advisor. WebSphere monitoring tools that are covered include: Tivoli Performance Viewer, WebSphere JVM™ verbosegc, and Java Virtual Machine Profiler Interface.

- Chapter 12, "Troubleshooting performance problems" on page 347 covers troubleshooting Content Manager performance problems with real-life scenarios. Performance issues we addressed include slow logon to the Content Manager system, slow search, slow import, slow migrator, and slow administration using the system administration client. Included also are the tools to work with logs for troubleshooting.
- Chapter 13, "Case study" on page 385 uses the Document Manager performance tuning as a case study to further show you how to troubleshoot system performance problems.
- Chapter 14, "Maintenance" on page 397 describes the maintenance tasks necessary to keep a Content Manager system up and running.

Changed information

- All other chapters have been updated to reflect the latest Content Manager Version 8.3.

Part 1

Architecture and technology overview

In this part we provide a Content Manager architecture overview and technology overview for its complementary software including DB2, WebSphere Application Server, and Tivoli Storage Manager.

If you are already familiar with Content Manager and the underlying software, we recommend that you skim through this part so that you can refresh your understanding and perhaps learn something new that can help you in Content Manager performance tuning.

1

Content Manager

In this chapter we introduce the IBM DB2 Content Manager (Content Manager) architecture. We provide an overview of various Content Manager components, how they interact with each other, and some of the key concepts that you should know related to Content Manager System.

1.1 Introduction

IBM DB2 Content Manager Enterprise Edition provides a flexible system that can be used to manage unstructured content according to business needs. Installable on Microsoft Windows, IBM AIX, Sun Solaris™, Linux, and IBM z/OS platforms, Content Manager provides a complete solution for storing and retrieving documents in almost any format. A companion product, Content Manager Standard Edition, provides similar functions for IBM iSeries™ users.

Content Manager features include:

- Support for multiple operating systems
- A Java-based tool for managing the system
- Client options including browser access
- Support for business documents of almost every kind
- Administration tools for defining users and user privileges, defining data models, and providing ways to manage your servers and data
- Efficient methods for keeping the system secure
- Document routing to manage the flow of work through the system

1.2 Content Manager system components

The Content Manager system is scalable to fit the needs of small, medium, and large-scale businesses. The main system components that make up a Content Manager system are:

- Library Server
- Resource Manager
- Clients

Content Manager *clients* include the system administration client, Client for Windows, Information Integrator for Content system administration client, and eClient. The clients running either in user desktops or in mid-tier application servers (such as WebSphere Application Server) invoke Content Manager services that are divided between a Library Server and one or more Resource Managers. The *Library Server* manages the content metadata and is responsible for access control to all of the content and interfaces to one or more Resource Managers. The *Resource Managers*, using WebSphere Application Server and Tivoli Storage Manager, manage the content objects. Contents can be stored on any devices supported by Tivoli Storage Manager, including DASD, optical, and tape.

Figure 1-1 shows the Content Manager system with its components.

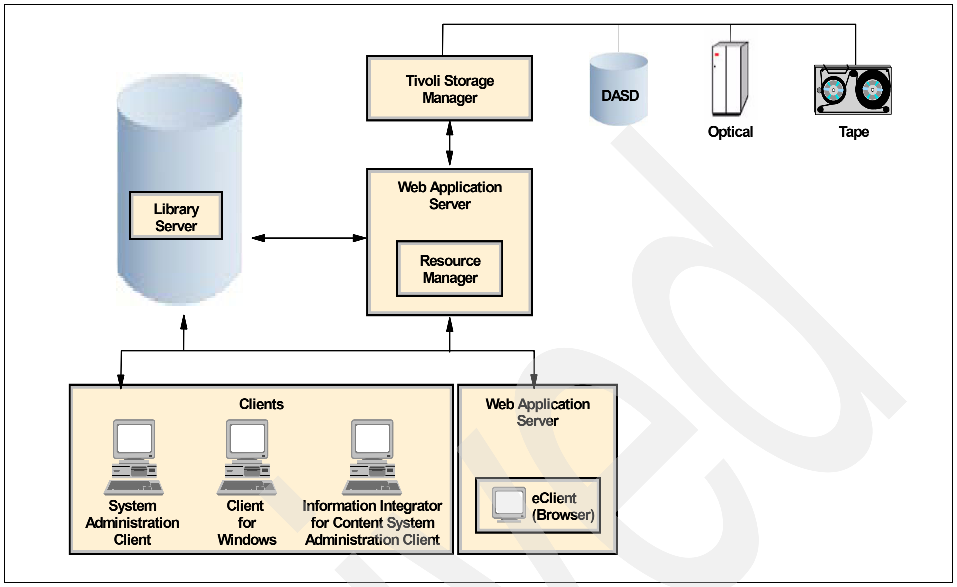


Figure 1-1 Content Manager system

All access to Library Server is through the database query language SQL. Results from the queries are passed back to the client with object tokens, locators for requested content that the user is authorized to access. The client then directly communicates with the Resource Manager using the Internet protocols HTTP, FTP, and FILE.

In the following sections, we explore the Content Manager system components, discuss the communication among the system components, and introduce the APIs used by Content Manager.

1.2.1 Library Server

The Library Server is the key component of a Content Manager system. It is where you define the information that you store in Content Manager.

The Library Server has the following functions:

- Storing, managing, and providing access control for objects stored on one or more Resource Managers. (For Resource Manager information, see 1.2.2, “Resource Manager” on page 7.)
- Processing requests (such as update or delete) from one or more clients.

- Maintaining data integrity between all of the components in a Content Manager system.
- Directly controlling user access to objects stored on any Resource Manager in the system.
- Managing the content and performing parametric searches, text searches, and combined (parametric and text) searches, using a relational database management system such as DB2 Universal Database™.

Figure 1-2 shows a Library Server component.

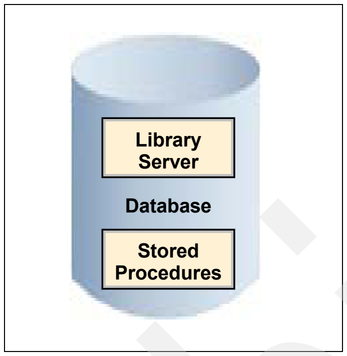


Figure 1-2 Library Server component

You can directly access the Library Server database using SQL (Structured Query Language) or a relational database client. A Library Server contains a Library Server database and is implemented as a set of stored procedures. All tasks carried out by Library Server are performed via stored procedures on the Library Server database.

Each Content Manager system can have only one Library Server, which can run on Windows, AIX, Linux, Solaris, and z/OS. Notice, if you use Oracle instead of IBM DB2, you cannot run the Library Server on Linux, Linux for s/390, and z/OS.

The Library Server requires the following software:

- IBM DB2 Universal Database or Oracle Database
- IBM DB2 Runtime Client or Oracle Client
- IBM DB2 NetSearch Extender or Oracle Text (optional)
- C++ Compiler (optional)

IBM DB2 Universal Database or Oracle Database

The IBM DB2 Universal Database (provided in the Content Manager package) or Oracle Database are required to run with the Library Server and must be installed on the same system or workstation as the Library Server. The Library

Server needs the database software to store document model definitions, document metadata, access control, and document routing definitions.

IBM DB2 Runtime Client or Oracle Client

The IBM DB2 Runtime Client, or Oracle Client, provides communication between the client workstation and the Library Server database. After the database client (DB2 or Oracle) is installed, the Library Server database must be cataloged on the client workstation. Cataloging establishes communication between the Library Server and the Content Manager Client for Windows application and the system administration client. Without this component, the Client for Windows and the system administration client cannot communicate with the Library Server (through stored procedure calls).

IBM DB2 NetSearch Extender or Oracle Text (optional)

Content Manager offers full-text searching of documents in the Content Manager database. This is an optional feature. To use it, you must install one of the following products, depending on the database you are using:

- DB2 Net Search Extender if you are using DB2 Universal Database
- Oracle Text if you are using the Oracle Database

If you do not require text search capabilities, you do not need to install DB2 Net Search Extender. If you currently do not need it, but anticipate using this feature in the future, it is best to install DB2 Net Search Extender before installing Content Manager. This is because the Content Manager installation program automatically enables the Library Server database to search for text if DB2 Net Search Extender is installed earlier.

C++ Compiler (optional)

The C++ Compiler is an optional component that can be used to define user exit routines.

1.2.2 Resource Manager

The Resource Manager stores objects for Content Manager. You can store and retrieve objects on the Resource Manager. When you request data, the request is first routed through the Library Server.

A single Library Server can support multiple Resource Managers and content can be stored on any of these Resource Managers. Figure 1-3 on page 8 shows the Resource Manager and its relationship to the Library Server.

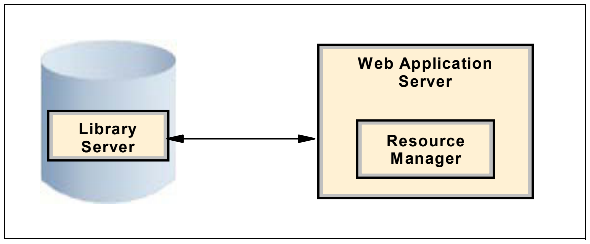


Figure 1-3 Resource Manager component and its relationship to the Library Server

The Resource Manager consists of a Resource Manager database and Resource Manager applications. Similar to the Library Server, the Resource Manager relies on a relational database to manage the location of stored objects and storage devices. The Resource Manager works closely with Tivoli Storage Manager (TSM) to define storage classes and migration policies.

The Resource Manager can run on Windows, AIX, Linux, Solaris, or z/OS. If you use Oracle instead of IBM DB2, you cannot run the Resource Manager on Linux, Linux for s/390, or z/OS.

The Resource Manager requires the following software:

- IBM DB2 Universal Database or Oracle Database
- WebSphere Application Server
- IBM HTTP Server
- Tivoli Storage Manager
- Tivoli Storage Manager Client

IBM DB2 Universal Database or Oracle Database

The IBM DB2 Universal Database (provided in the Content Manager package) or Oracle Database is required to run the Resource Manager database. Depending on the speed and storage requirements of your Content Manager system, you can install the Resource Manager database on the same machine as the Library Server database. The Resource Manager database server stores information pertaining to the objects being managed by the Resource Manager server.

WebSphere Application Server

The IBM WebSphere Application Server software provided in this package is required to run with the Resource Manager and must be installed on the same machine as the Resource Manager.

The IBM WebSphere Application Server provides the Java-enabled platform that enables Web clients to interact with Content Manager. Users and processes on a wide variety of operating systems can interact by using the facilities provided by WebSphere Application Server.

The Resource Manager is implemented as a Java 2 Enterprise Edition (J2EE™) Web application, and requires the WebSphere Application Server as its J2EE-compliant application server. WebSphere Application Server must be installed and configured before you begin the installation of the Content Manager Resource Manager component.

You can find more about WebSphere Application Server and about how it interacts with Content Manager by browsing through the documentation that is available through the WebSphere Application Server information center, which is located at:

`http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp`

IBM HTTP Server

Client requests to store or retrieve documents are communicated by HTTP to the Web server. The Web server then packages the request to be sent to the Resource Manager. Optionally, client configuration files (*.ini and *.property) may be published on the Web server so that all clients in the enterprise share a common and centrally maintained configuration.

Tivoli Storage Manager

Tivoli Storage Manager is a client/server product that provides storage management and data access services in a heterogeneous environment. It is provided so that you can store objects, long term, on a storage device other than the fixed disks that are attached to the Resource Manager. Tivoli Storage Manager supports a variety of communication methods and provides administrative facilities to schedule backup and storage of files.

Tivoli Storage Manager Client

The Resource Manager server uses the Tivoli Storage Manager Client APIs, which are installed with the Tivoli Storage Manager Client, to access the Tivoli Storage Manager server. These APIs provide connectivity between the Resource Manager server and the Tivoli Storage Manager server and can be programmed to centrally administer storage.

1.2.3 Content Manager clients

The Content Manager clients are the interface that users use to manage and work in the Content Manager system. Content Manager offers the following out-of-box clients:

- System administration client
- Client for Windows, a dedicated Windows client that runs on Windows 2000, Windows Server® 2003, or Windows XP

- eClient, a browser-based client that runs on Netscape or Internet Explorer®.

Your choice might be influenced by the portability of the Web client versus the functionality of the Windows client.

In addition to the out-of-box clients, Content Manager also offers the following possible choices of clients:

- Portal client
- Custom client

System administration client

The System administration client is one of the Content Manager clients. It oversees and manages the entire Content Manager system.

Through the system administration client, you can:

- Define your data model.
- Define users and their access to the system.
- Manage storage and storage objects in the system.

Figure 1-4 shows the system administration client and its relationship to the Library Server and the Resource Manager.

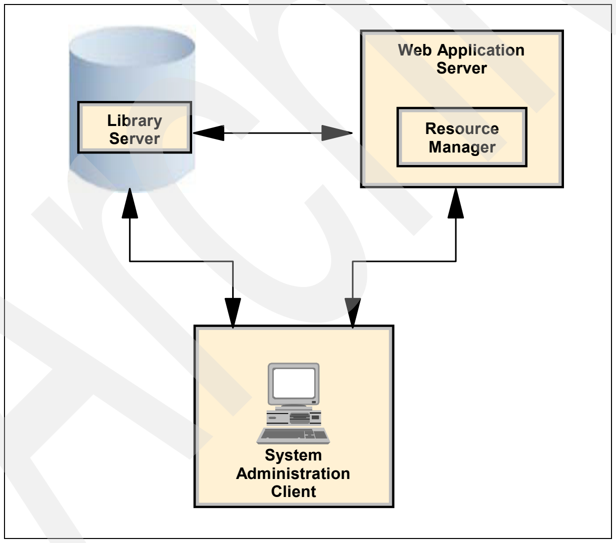


Figure 1-4 System administration client

The system administration client component can be installed on Windows, AIX, Solaris, or Linux. It can be installed where other components are installed or on a separate workstation. As a security measure, the system administration client communicates with the Resource Manager through the Secure Sockets Layer (SSL) interface.

Client for Windows

The Client for Windows is installed only on a Windows system. It enables users to import, view, store, and retrieve documents. The Client for Windows can also be run in a Terminal Server Edition (TSE) environment. The number of users who can be supported on any one TSE server depends on the memory, processing power, and other factors on the server, as well as on the amount of activity for each client user. All client actions are supported in this environment, except for scanning documents into the system (which must be done on a local workstation).

Figure 1-5 shows the relationship between the Content Manager Client for Windows and the other Content Manager components (Library Server, Resource Manager, and system administration client).

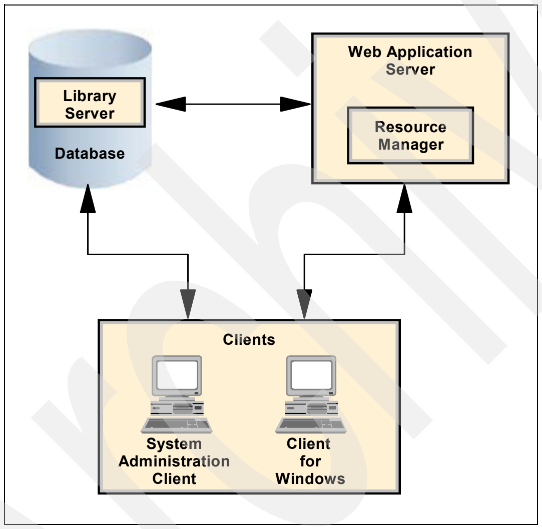


Figure 1-5 Client for Windows

eClient

eClient can be installed on a Windows, AIX, or Solaris system. eClient is a browser-based client that provides out-of-the-box capabilities for Content Manager systems, similar to that of the Windows client. Import and export can be done using the eClient wizards. Document and folder organization functions are available through eClient. The eClient viewer provides page navigation functions such as next, prev, last, goto, zoom in, and zoom out.

The application displays the first page of a document as soon as it is available without waiting for the entire document to download. This improves the response time when working with large documents. eClient also supports the same type of search capabilities as a Windows client.

Portal client

Portal client is an extensive and customizable front end to Content Manager. It is composed of Content Manager Portlets, a Web-based application, running on the WebSphere Portal environment.

Content Manager Portlets consists of two portlets, the Main portlet and the Viewer portlet. The *Main portlet* provides a single portlet-based functional replacement for the current Content Manager eClient. When deployed alone, the Main portlet will display documents in new browser windows for viewing.

Optionally, the *Viewer portlet* can be deployed on the same portal page as the Main portlet to provide document-viewing capability with a unified layout in an integrated fashion, instead of in separate browser windows. Click-2-Action (C2A) is also supported by the Viewer portlet. Other portlets, including the Main portlet, can communicate to the Viewer portlet using C2A to have documents displayed in the Viewer portlet.

Portlets are free. They are available from the WebSphere Portlet Catalog:

`http://catalog.lotus.com/wps/portal/workplace?NavCode=1WP100006`

Creating your own client

You can create custom Content Manager client applications using client APIs and user exit routines that are part of the Content Manager connectors. These connectors come with Content Manager Information Integrator for Content. You can use these APIs to:

- Access information in the Library Server and Resource Manager.
- Customize document processing.
- Design your own data model.

1.2.4 Content Manager component communication

When the Content Manager system is installed and running, the client application accesses the Library Server through an SQL request (such as create, retrieve, update, or delete). The Library Server, which contains metadata, processes the request by looking up, in its content index, where the data resides. The Library Server then gives the client a security token and a locator within the Resource Manager where the information can be found. The information is then returned to the client. The client uses the security token and locator to access the Resource Manager. The Resource Manager accepts the security token and returns the information to the client through HTTP.

1.2.5 Content Manager APIs

The Content Manager Application Programming Interface (API) provides communication through Content Manager and Information Integrator for Content connectors. The Content Manager V8 connector supports C++ APIs (on Windows and AIX) as well as Java. The Content Manager V8 connector also supports JavaBeans™, XML, and Portlets. Communication with the Library Server (through SQL queries) is through JDBC™ (Java Database Connectivity), or CLI (Command Line Interface). Communication with the Resource Manager is through HTTP request, FTP, and RTPS (Real-Time Publish Subscribe protocol).

1.3 Content Manager system configuration

Content Manager architecture is open, portable, and extensible. In earlier sections, we introduced the main Content Manager components, the Library Server, Resource Manager, and Content Manager clients.

Depending on your business requirements, Content Manager system can be build with two-tier or three-tier configuration. Two-tier configuration consists of Client for Windows or custom clients at the top tier, and the Library Server and the Resource Managers at the bottom. See Figure 1-6.

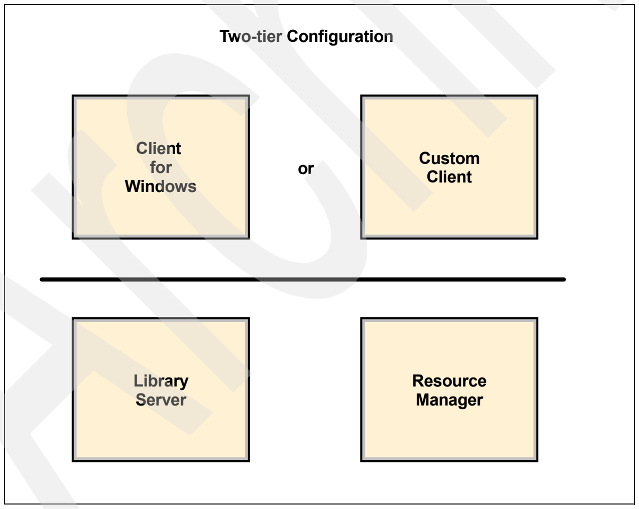


Figure 1-6 Content Manager two-tier configuration

Three-tier configuration (Figure 1-7 on page 14) consists of a browser or other types of clients that talk to the mid-tier server at the top tier, eClient at the mid-tier, and the Library Server and the Resource Managers at the bottom tier.

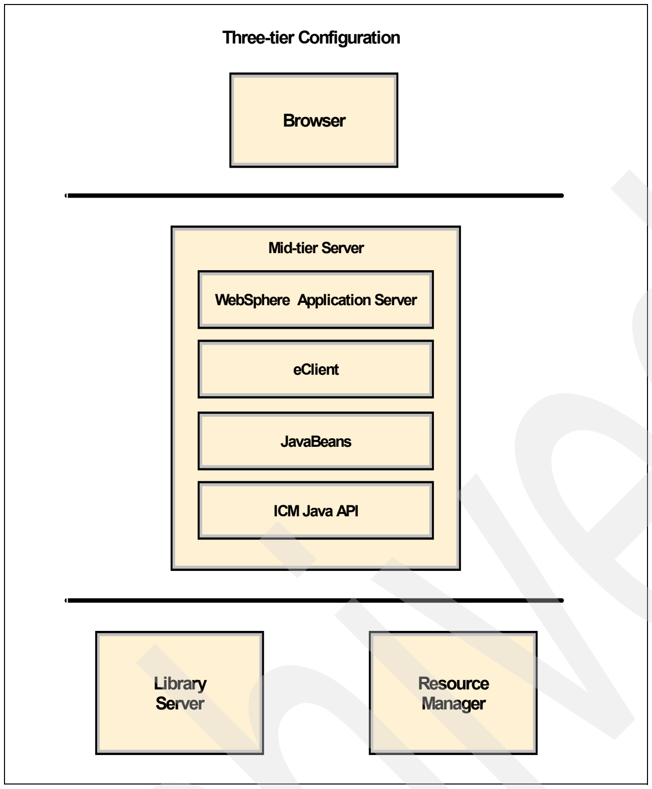


Figure 1-7 Content Manager three-tier configuration

Different Content Manager system configurations enable you to achieve different business requirements. System configuration plays an important role in overall system performance. In this section, we introduce several common Content Manager system configurations:

- All in one machine
- Separate servers on different machines
- Multiple Resource Managers
- Mixed platforms
- Multiple Content Manager systems with mixed platforms

All in one machine

Install all components (Library Server, Resource Manager, system administration client, Client for Windows, eClient) on a single machine as shown in Figure 1-8. This is a good option for the first prototype Content Manager system.

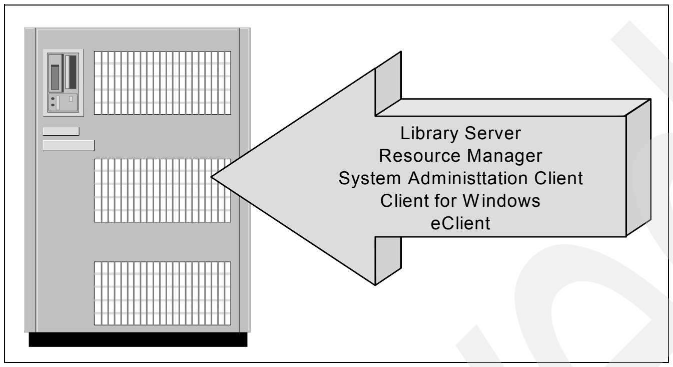


Figure 1-8 System configuration: all in one machine

Separate servers on different machines

Split the Content Manager's main components to different machines as shown in Figure 1-9. Install the Library Server on one machine, and the Resource Manager on another machine. Access the system using either Client for Windows or eClient. System administration client can be installed on the same machine as the client machine.

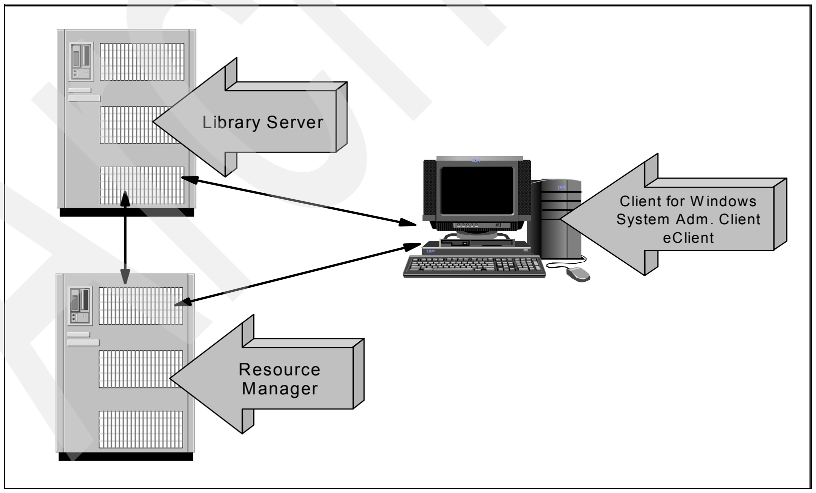


Figure 1-9 System configuration: separate servers on different machines

Multiple Resource Managers

Install multiple Resource Managers with a single Library Server running on the same platform. Figure 1-10 shows an example. Access the system using either Client for Windows, or eClient from a workstation. System administration client can be installed on the same machine as the client machine.

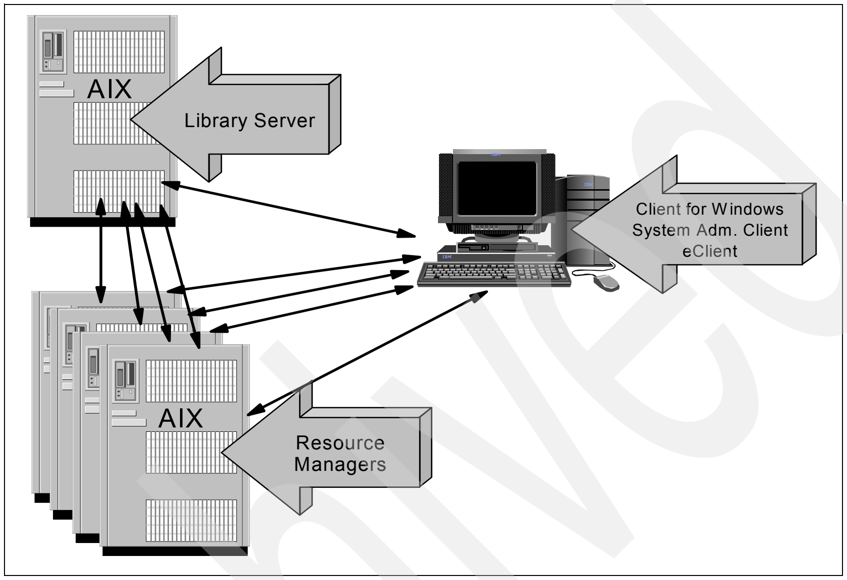


Figure 1-10 System configuration: multiple Resource Managers

Mixed platforms

Set up a Content Manager system with a single Library Server and multiple Resource Managers running on different platforms. Figure 1-11 shows an example. Access the system using either the Client for Windows or the eClient on a workstation. System administration client can be installed on the same machine as the client machine.

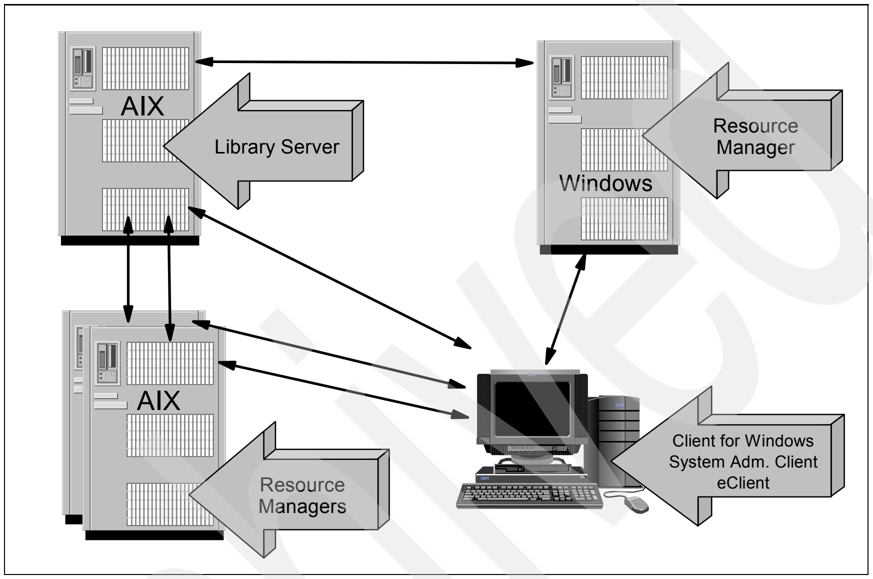


Figure 1-11 System configuration: mixed platforms

Multiple Content Manager systems with mixed platforms

Set up multiple Content Manager systems with multiple Library Servers and Resource Managers running on different platforms. Figure 1-12 shows an example. Access each system using one unique client, either a Client for Windows or an eClient. Note that each Content Manager system can have one and only one Library Server.

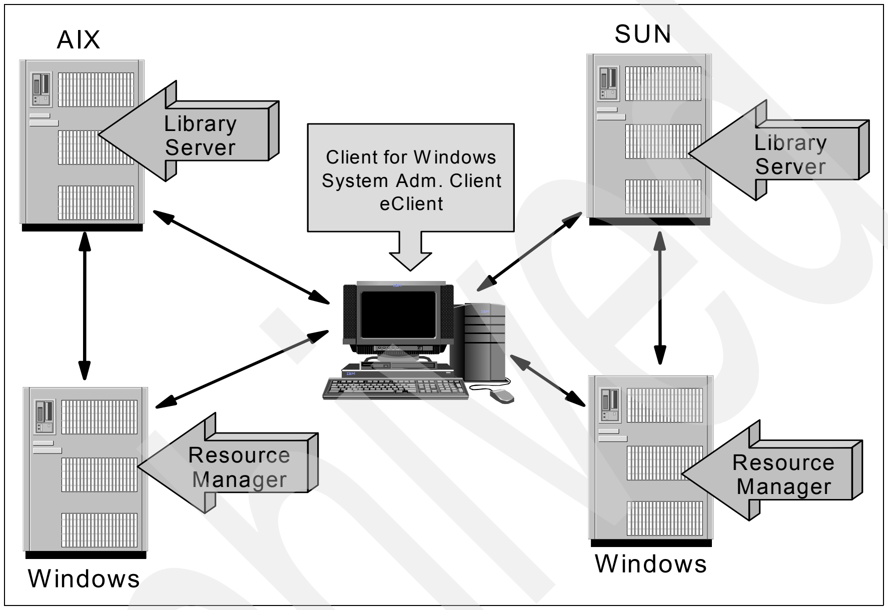


Figure 1-12 System configuration: multiple CM systems, mixed platforms

Summary

In summary, there are many ways to configure a Content Manager system. You can set up everything in one machine, or separate components in multiple machines, and across multiple platforms based on your business requirements. Remember, system configuration directly affects system performance. We discuss performance impact and performance tuning later in this book.

2

Content Manager base products

Content Manager Version 8 is built on the products IBM DB2 UDB, WebSphere Application Server, and Tivoli Storage Manager. Performance tuning a Content Manager system includes understanding, configuring, and tuning the underlying software for performance. In this chapter we provide an overview of these three base software components. For convenience, we use the term “Content Manager base product” or “base product” for short to denote any of these products in this context.

Note: Oracle is another database option that you can use with Content Manager. We do not discuss it in this chapter because it is beyond the scope of this book. We focus mainly on DB2 in this redbook.

2.1 DB2 Universal Database overview

Library Server uses a relational database, and Resource Manager also uses a relational database. IBM DB2 Universal Database (DB2) is a multimedia, Web-ready, relational database management system (RDBMS). It is available on z/OS, UNIX®, Windows, and AS/400® platforms. Understanding DB2 is critical to achieving success in performance tuning for Content Manager systems.

There are many well-written publications on DB2 that contain very useful information. Most of the material presented in the following sections is extracted from these publications:

- *IBM DB2 Universal Database Version 8.2 Administration Guide: Planning*, SC09-4822
- *IBM DB2 Universal Database Version 8.2 Administration Guide: Performance*, SC09-4821
- *IBM DB2 UDB V8 and WebSphere V5 Performance Tuning and Operation Guide*, SG24-7068

We include what we think is important to give you a quick start in the basic concepts of DB2, as well as aspects of performance tuning for the DB2 database system and routine maintenance that should be performed. For a more in-depth understanding of DB2, we recommend that you read these publications because they proved to be extremely helpful for us.

2.1.1 Basic DB2 database concepts

In this section, we introduce some of the basic relational database concepts. This includes an overview of the various database objects and the database configuration files.

Database objects

Database objects are the components that make up a DB2 relational database. Some of the more important database objects are: instances, databases, database partition groups, tables, views, indexes, schemas, system catalog tables, buffer pools, table spaces, containers, and stored procedures.

Figure 2-1 on page 21 displays the database objects relationship. The individual components of the database objects will be discussed throughout this section.

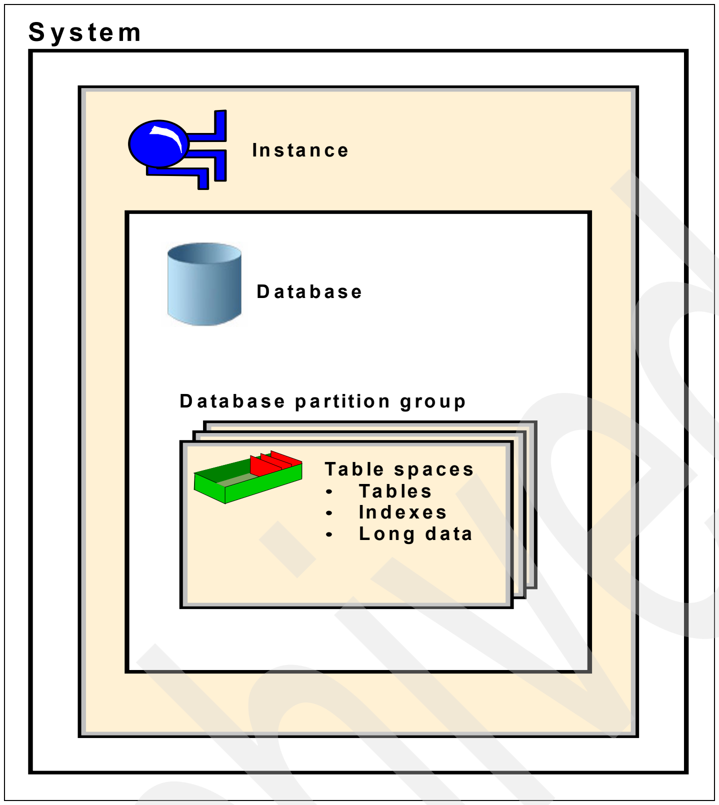


Figure 2-1 Database objects relationship

Instances (database managers)

An *instance* (also known as a database manager) is DB2 code that manages data. It controls what can be done to the data, and manages system resources that are assigned to it. Each instance is a complete environment. If the database system is parallel, it contains all of the database partitions. An instance has its own databases (which other instances cannot access), and all its database partitions share the same system directories. It also has separate security from other instances on the same machine or system.

Databases

A relational *database* presents data as a collection of tables. A *table* consists of a defined number of columns and any number of rows. Each database includes a set of system catalog tables that describe the logical and physical structure of the data, a configuration file containing the parameter values allocated for the database, and a recovery log with ongoing transactions and transactions to be archived.

Database partition groups

A *database partition group* is a set of one or more database partitions. When you want to create tables for the database, you first create the database partition group where the table spaces will be stored, then you create the table space where the tables will be stored. In earlier versions of DB2, database partition groups were known as nodegroups.

Note: For a Content Manager system, we do not recommend database partitioning.

Tables

A relational database presents data as a collection of tables. A table consists of data that is arranged logically in columns and rows. All database and table data is assigned to table spaces. The data in the table is logically related, and relationships can be defined between tables. Data can be viewed and manipulated based on mathematical principles and operations called relations.

Table data is accessed through *Structured Query Language* (SQL), a standardized language for defining and manipulating data in a relational database. A query is used in applications or by users to retrieve data from a database. The query uses SQL to create a statement in the form of:

```
SELECT <data_name> FROM <table_name>
```

Views

A *view* is an efficient way of representing data without needing to maintain it. A view is not an actual table and requires no permanent storage. A “virtual table” is created and used.

A view can include all or some of the columns or rows contained in the tables on which it is based. For example, you can combine a department table and an employee table in a view, so that you can list all employees in a particular department.

Indexes

An *index* is a set of pointers to rows in a table. An index allows more efficient access to rows in a table by creating a direct path to the data through pointers. The SQL optimizer automatically chooses the most efficient way to access data in tables. The optimizer takes indexes into consideration when determining the fastest access path to data.

Unique indexes can be created to ensure uniqueness of the index key. An index key is a column or an ordered collection of columns on which an index is defined.

Using a unique index will ensure that the value of each index key in the indexed column or columns is unique.

Schemas

A *schema* is an identifier, such as a user ID, that helps group tables and other database objects. A schema can be owned by an individual, and the owner can control access to the data and the objects within it.

A schema is also an object in the database. It may be created automatically when the first object in a schema is created. Such an object can be anything that can be qualified by a schema name, such as a table, index, view, package, distinct type, function, or trigger.

A schema name is used as the first part of a two-part object name. When an object is created, you can assign it to a specific schema. If you do not specify a schema, it is assigned to the default schema, which is usually the user ID of the person who created the object. The second part of the name is the name of the object. For example, a user named Smith might have a table named SMITH.PUBLICATIONS.

System catalog tables

Each database includes a set of *system catalog tables*, which describe the logical and physical structure of the data. DB2 creates and maintains an extensive set of system catalog tables for each database. These tables contain information about the definitions of database objects such as user tables, views, and indexes, as well as security information about the authority that users have on these objects. The system catalog tables are created when the database is created, and are updated during the course of normal operation. You cannot explicitly create or drop them, but you can query and view their contents using the catalog views.

Buffer pools

A *buffer pool* is an area of reserved main memory allocated into which user table data, index data, and catalog data from disk storage are temporarily stored or cached. The buffer pool plays a key role in overall database performance, because data accessed from memory is much faster than data accessed from a disk. The fewer times the database manager needs to read from or write to the disk, the better the performance. The configuration of the buffer pools is one of the most important aspects of performance tuning for a Content Manager system because you can reduce the delay caused by slow I/O.

Buffer pools can be defined with varying page sizes including 4 KB, 8 KB, 16 KB, and 32 KB. By default the Content Manager Library Server uses 4 KB and 32 KB page sizes for different parts of the database.

Table spaces

A database is organized into parts called *table spaces*. A table space is a place to store tables. When creating a table, you can decide to have certain objects such as indexes and large object (LOB) data kept separately from the rest of the table data. A table space can also be spread over one or more physical storage devices.

Table spaces reside in database partition groups. Table space definitions and attributes are recorded in the database system catalog.

Containers are assigned to table spaces. A *container* is an allocation of physical storage (such as a file or a device).

A table space can be either *system managed space* (SMS), or *database managed space* (DMS). For an SMS table space, each container is a directory in the file space of the operating system, and the operating system's file manager controls the storage space. For a DMS table space, each container is either a fixed-size pre-allocated file, or a physical device such as a disk, and the database manager controls the storage space. The page size of the table space must match the buffer pool page size.

The default user table space is called USERSPACE1. Any user-created tables that are not specifically defined to an existing table space will default to this table space. Always specify a user-defined table space when creating user tables and indexes to avoid having them created in USERSPACE1.

The system catalog tables exist in a regular table space called SYSCATSPACE. There is also a default system temporary table space called TEMPSPACE1, which is used to store internal temporary data required during SQL operations such as sorting, reorganizing tables, creating indexes, and joining tables.

Tables containing long field data or large object data, such as multimedia objects, exist in large table spaces or in regular table spaces. The base column data for these columns is stored in a regular table space, and the long field or large object data can be stored in the same regular table space or in a specified large table space.

Figure 2-2 on page 25 displays the different type of table spaces that might exist within a database.

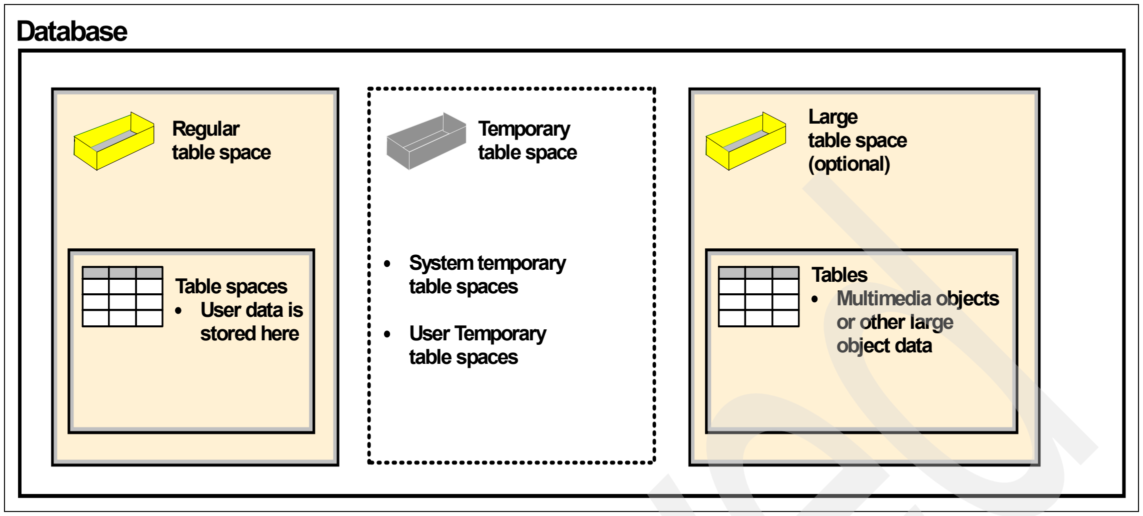

Database
Regular table space
Table spaces
• User data is stored here
Temporary table space
• System temporary table spaces
• User Temporary table spaces
Large table space (optional)
Tables
• Multimedia objects or other large object data

Figure 2-2 Table space types

The Content Manager product has predefined table spaces that are created when the product is installed.

Table 2-1 lists the default table spaces created for Library Server. Table 2-2 on page 26 lists the default table spaces created for Resource Manager.

Table 2-1 Library Server default table space setup

Buffer pool	Table space	Table spaces description
ICMLSMAINBP32	ICMLFQ32	Holds all large, persistent, frequently used tables – most notably, all item-type tables. When a new item type is defined, the system administration client defines the new tables in this table space. If you customize the DDL, this table space should be defined across multiple containers for performance and scalability.
	ICMLNF32	Holds the large but less frequently used tables, such as event logs, versions, and replicas. If you do not use any of these features, this table space will not need much space.

Buffer pool	Table space	Table spaces description
ICMLSVOLATILE BP4	ICMVFQ04	Holds the volatile tables, those whose size is usually small but which can vary widely, and for which both updates and deletes are very common (for example, the tables associated with document routing "in-progress" items and the checked-out items). Putting this on a separate physical disk should help performance.
ICMLSFREQBP4	ICMSFQ04	Holds all the small, frequently-used, but seldom updated tables, such as system information, user definitions, and ACLs, that do not take up much space but that are always needed in memory for good performance.
CMBMAIN4	CMBINV04	Holds the tables used for Information Integrator for Content federated administration.

Table 2-2 Resource Manager default table space setup

Buffer pool	Table space	Table spaces description
OBJECTPOOL	OBJECTS	Primary table space that holds the tables that grow very large, with information about all the objects stored on this Resource Manager.
SMSPOOL	SMS	Holds small but frequently read administrative information.
TRACKINGPOOL	TRACKING	Holds volatile tables related to transactions.
PARTSPOOL	PARTS	Holds tables used by the validation utility, which is infrequently used but can create a very large volume of data.
BLOBPOOL	BLOBS	Reserved for future use.

Containers

A *container* is a physical storage device. It can be identified by a directory name, a device name, or a file name. A container is assigned to a table space. A single table space can span many containers, but each container can belong to only one table space. We strongly recommend that multiple containers be used in order to promote the distribution of data across many disks and to increase input-output (I/O) parallelism. DB2 has a limit of 512 GB per table space using 32K rows.

Figure 2-3 shows how table space HUMANRES is spread across five containers, with each container being on a different disk.

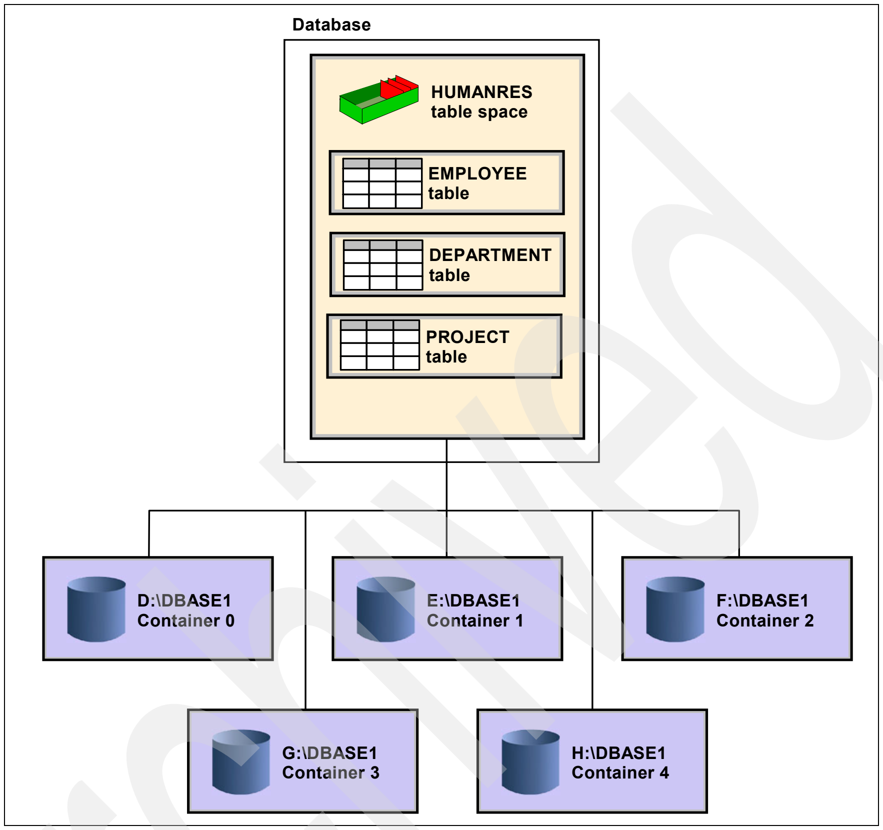


Figure 2-3 Table space spread over multiple containers

Stored procedures

Stored procedures permit one call to a remote database to perform a pre-programmed procedure in a database application environment in which many situations are repetitive. For example, for receiving a fixed set of data, performing the same set of multiple requests against a database is necessary.

Processing a single SQL statement for a remote database requires sending two transmissions: one request and one receive. An application might require many SQL statements to complete its work. This could result in many acts of transmission. A stored procedure that encapsulates those many SQL statements would enable the work to be done in one transmission.

A stored procedure is a *dynamically linked library* (DLL) or program that runs on the server and is called by the database engine in response to a call by a client.

Content Manager uses stored procedures for such functions as checking for access control privileges and executing user functions such as searching for an item. A stored procedure can call other DLLs as required.

Stored procedures usually run in a *fenced mode* where they are performed in processes separate from the database agents.

Configuration files and parameters

When a DB2 instance or a database is created, a corresponding *configuration file* is created with default parameter values. You can and should modify these parameter values to adapt the configuration to your environment and potentially improve performance.

Configuration files contain parameters that define values such as the resources allocated to the DB2 products and to individual databases, and the diagnostic level. There are two types of configuration files:

- The database manager configuration file for each DB2 instance
- The database configuration file for each individual database

Database manager configuration file

The *database manager configuration file* is created when a DB2 instance is created. The parameters it contains affect system resources at the instance level, independent of any one database that is part of that instance. Values for many of these parameters can be changed from the system default values to improve performance or increase capacity, depending on your system's configuration.

There is one database manager configuration file for each client installation as well. This file contains information about the client enabler for a specific workstation. A subset of the parameters available for a server are applicable to the client.

Database manager configuration parameters are stored in a file named db2systm. This file is created when the instance of the database manager is created. It is a binary file that can only be administered using DB2 tools and commands. In UNIX-based environments, this file can be found in the sqllib subdirectory for the instance of the database manager. In Windows environment, the default location of this file is the instance subdirectory of the sqllib directory. If the DB2INSTPROF variable is set, the file is in the instance subdirectory of the directory specified by the DB2INSTPROF variable.

In a partitioned database environment, this file resides on a shared file system so that all database partition servers have access to the same file. The configuration of the database manager is the same on all database partition servers.

Most of the parameters affect either the amount of system resources that will be allocated to a single instance of the database manager, or the setup of the database manager and the different communications subsystems based on environmental considerations. In addition, there are other parameters that serve informative purposes only and cannot be changed. All of these parameters have global applicability independent of any single database stored under that instance of the database manager.

Database configuration file

A *database configuration file* is created when a database is created, and resides where that database resides. There is one configuration file per database. Its parameters specify the amount of resource to be allocated to that database. Values for many of the parameters can be changed to improve performance or increase capacity. Different changes might be required, depending on the type of activity in a specific database.

Parameters for an individual database are stored in a binary configuration file named SQLDBCON. This file is stored along with other control files for the database in the SQL*nnnn*n directory, where *nnnnn* is a number assigned when the database was created. Each database has its own configuration file, and most of the parameters in the file specify the amount of resources allocated to that database. The file also contains descriptive information, as well as flags that indicate the status of the database.

In a partitioned database environment, a separate SQLDBCON file exists for each database partition. The values in the SQLDBCON file might be the same or different at each database partition, but we recommend that the database configuration parameter values be the same on all partitions.

As we mentioned earlier, for a Content Manager system, we do not recommend partitioning databases.

2.1.2 DB2 architecture overview

The previous section provides a brief overview of some of the basic relational database concepts. Now we examine the DB2 architecture and processes as shown in Figure 2-4 on page 30. Each client application, local or remote, is linked with the DB2 client library. Local clients communicate using shared memory and semaphores. Remote clients communicate using communication protocols such as TCP/IP, Named pipes, NetBIOS, or IPX/SPX. Activities on DB2 server are controlled by threads in a single process on Windows, or processes on UNIX. They are denoted with circles or elliptical shapes. Those processes with thick borders can be launched with multiple instances. For example, there might be one or more subagents, loggers, prefetchers and page cleaners running, but there is only one coordinator agent assigned for each client application. In the following section, we go over the key DB2 components and processes.

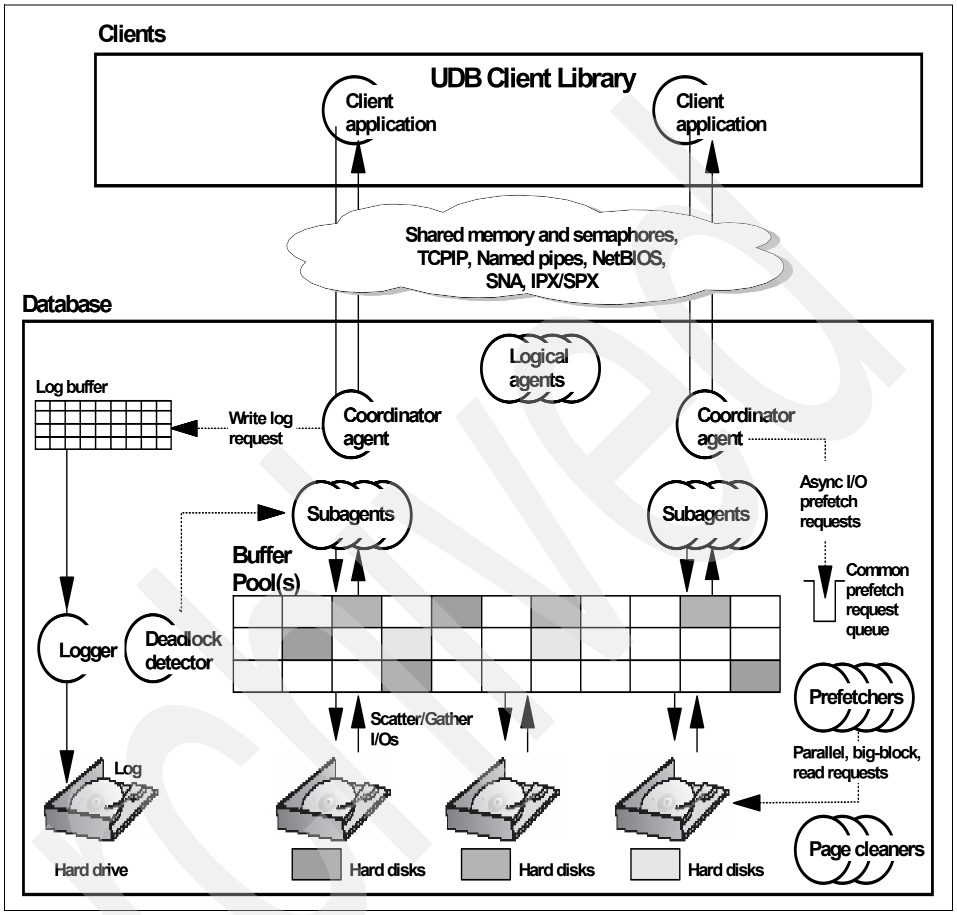


Figure 2-4 DB2 architecture and processes overview

DB2 agents (coordinator agent and subagents)

DB2 agents are the DB2 processes that carry out the bulk of SQL processing requests from applications. They include coordinator agents and subagents. DB2 assigns a *coordinator agent* for each client application. This agent coordinates the communication and processing for this application.

If intra-partition parallelism is disabled (this is the default), then the coordinator agent performs all of the application's requests. If intra-partition parallelism is enabled, then DB2 assigns a set of *subagents* to the application to process requests from the application. If the machine where the DB2 server resides has multiple processors, multiple subagents can also be assigned to an application.

The maximum number of agents that can be connected to a database instance can be controlled by the database manager configuration parameters such as MAXAGENTS， MAXCAGENTS, and MAX_COORDAGENTS.

The maximum number of applications that can be connected to a database can be controlled by the database configuration parameter MAXAPPLS, and the database manager configuration parameter MAX_AGENTS.

All agents and subagents are managed using a pooling algorithm that minimizes their creations and their destructions.

Prefetchers

Prefetchers retrieve data from disk and move them into the buffer pool before applications need the data. For example, applications needing to scan through large item-type tables would have to wait for data to be moved from disk into the buffer pool if there were no data prefetchers.

With prefetch, DB2 agents send asynchronous read-ahead requests to a common prefetch request queue. As prefetchers become available, they implement those requests by using big-block or scatter read input operations to bring the requested pages from disk to the buffer pool.

Having multiple disks for database data storage means that the data can be striped across the disks. This striping of data enables the prefetchers to use multiple disks at the same time to retrieve data.

Prefetchers are designed to improve the read performance of applications as well as utilities such as backup and restore, since they prefetch index and move data pages into the buffer pool, thereby reducing the time spent waiting for I/O to complete.

The number of prefetchers can be controlled by the database configuration parameter NUM_IOSERVERS.

Page cleaners

Page cleaners (also known as asynchronous page cleaners) make room in the buffer pool to enable agents and prefetchers to read pages from disk storage and move them into the buffer pool. They look for updated pages in the buffer pool and write them to disk so that these pages can be used for other data if needed. For example, if an application has updated a large amount of data in a table, many of the updated data pages in the buffer pool might not have been written to disk storage yet — such pages are called *dirty pages*. Because prefetchers cannot place fetched data pages from disk storage onto the dirty pages, these dirty pages must first be flushed to disk storage and become *clean* pages. (The clean pages remain in the buffer pool until a prefetcher or a DB2 agent “steals” them.)

Page cleaners are independent of the application agents that read and write to pages in the buffer pool. Page cleaners write changed pages from the buffer pools to disk before a database agent requires the space in the buffer pool. This process eliminates the need for the database agent to write modified pages to disk, thereby improving performance. Without the availability of independent prefetchers and page cleaners, database agents would have to do all of the reading and writing of data between the buffer pool and disk storage.

The configuration of the buffer pool, along with prefetchers and page cleaners, controls the availability of the data needed by the applications.

Note: Performance can improve if more page-cleaner agents are available to write dirty pages to disk. This is particularly true if snapshots reveal that there are a significant number of synchronous data-page or index-page writes in relation to the number of asynchronous data-page or index-page writes.

The number of page cleaners can be controlled by the database configuration parameter NUM_IOCLEANERS.

Proactive page cleaning

Starting with DB2 version 8.1.4, there is an alternative method of configuring page cleaning. This alternative method differs from the default behavior in that page cleaners behave more proactively in choosing which dirty pages get written out at any given time. This new method differs from the default page cleaning method in two major ways:

- Page cleaners will no longer react in response to the value of the CHNGPGS_THRESH configuration parameter.

  Instead, the alternate method of page cleaning provides a mechanism whereby the agents are informed of the location of good victim pages that have just been written out, so that agents do not have to search the buffer pool to look for a victim. When the number of good victim pages drops below an acceptable value, the page cleaners are triggered, and proceed to search the entire buffer pool, writing out potential victim pages, and informing the agents of the location of these pages.

- Page cleaners no longer respond to LSN gap triggers issued by the logger.

  When the amount of log space encompassing the log record that has updated the oldest page in the buffer pool and the current log position exceeds that allowed by the SOFTMAX parameter, it is said that the database is in an *LSN gap* situation. Under the default method of page cleaning, when the logger detects that an LSN gap has occurred, it will trigger the page cleaners to write out all of the pages that are contributing to the LSN gap situation. That is, it will write out those pages that are older than what is allowed by the

SOFTMAX parameter. Page cleaners will be idle for some period of time while no LSN gap is occurring. Then, when an LSN gap occurs, the page cleaners are activated to write a large number of pages before going back to sleep. This can result in the saturation of the I/O subsystem, which then affects other agents that are reading or writing pages. Furthermore, by the time an LSN gap is triggered, it is possible that the page cleaners will not be able to clean fast enough and DB2 might run out of log space.

The alternate method of page cleaning modulates this behavior by spreading out the same number of writes over a greater period of time. The cleaners do this by proactively cleaning not only the pages that are currently in an LSN gap situation, but also the pages that will come into an LSN gap situation soon, based on the current level of activity.

To use the new method of page cleaning, set the DB2_USE_ALTERNATIVE_PAGE_CLEANING registry variable to `ON`.

Log processing and loggers

All databases maintain log files that keep records of database changes. All changes to regular data and index pages are written as a log record into a log buffer. The data in the log buffer is written to disk by the logger process.

To optimize performance, neither the updated data pages in the buffer pool nor the log records in the log buffer are written to disk immediately. They are asynchronously written to disk by page cleaners and the *logger,* respectively. The logger and the buffer pool manager cooperate and ensure that an updated data page is not written to disk storage before its associated log record is written to the log. This ensures database recovery to a consistent state from the log in the event of a crash such as a power failure.

There are two logging strategy choices:

- *Circular logging*, in which the log records fill the log files and then overwrite the initial log records in the initial log file. The overwritten log records are not recoverable. For example, assume that a database has a circular logging strategy using five log files. When the fifth log file becomes full, logging would continue against the first log file, overwriting the file’s original contents.
- *Retain log records*, in which a log file is archived when it fills with log records. New log files are made available for log records. Retaining log files enables roll-forward recovery. Roll-forward recovery reapplies changes to the database based on completed units of work (transactions) that are recorded in the log. You can specify that roll-forward recovery is to the end of the logs, or to a particular point in time before the end of the logs.

Regardless of the logging strategy, log buffers are written to disk under the following conditions:

- Before the corresponding data pages are written to disk. This is called write-ahead logging.
- On a COMMIT, or after the number of COMMITS to group MINCOMMIT database configuration parameter is reached.
- When the log buffer is full. Double buffering is used to prevent I/O waits.

Note: The use of the log retain option can severely affect the functionality of your system if not properly maintained. With this strategy, log files are continually created as needed, filling up disk space in the process. When the file system or disk is full, DB2 will be unable to create log records and will start writing error messages continually into the diagnostics log file (db2diag.log) until the problem is resolved. Regular pruning of old log files must be done and is discussed in the subsequent section about Routine Maintenance.

Deadlock detector

A *deadlock* occurs when one application is waiting for another application to release a lock on data. Each of the waiting applications is locking data needed by another application. Mutual waiting for the other application to release a lock on held data leads to a deadlock. The applications can wait forever until one application releases the lock on the held data.

Because applications do not voluntarily release locks on data that they need, DB2 uses a background process called the *deadlock detector* to identify and resolve these deadlocks. The deadlock detector becomes active periodically as determined by the DLCHKTIME configuration parameter. When the deadlock detector encounters a deadlock situation, one of the deadlocked applications receives an error code, and the current unit of work for that application is rolled back automatically by DB2. When the rollback is complete, the locks held by this chosen application are released, enabling the other applications to continue.

Note: Selecting the proper interval for the deadlock detector ensures good performance. If the interval is too short, then the deadlock detector will become active too frequently, which will cause unnecessary overhead. If the interval is too long, then resolution of deadlock situations will increase, delaying processes unnecessarily and negatively affecting performance.

Memory usage

A primary performance tuning task is deciding how to divide the available memory among the areas in the database. An understanding of how DB2 organizes memory and the many configuration parameters that affect memory usage is essential.

Instance shared memory (also known as database-manager shared memory) is allocated when the database is started. All other memory is attached or allocated from the instance shared memory. When the first application connects or attaches to a database, database shared, application shared, and agent private memory areas are allocated. Instance shared memory can be controlled by the INSTANCE_MEMORY configuration parameter of the database manager. By default, this parameter is set to `automatic` so that DB2 calculates the amount of memory allocated for the instance.

Database shared memory (also known as Database Global memory) is allocated when a database is activated or connected to for the first time. This memory is used across all applications that might connect to the database. Database shared memory can be controlled by the DATABASE_MEMORY configuration parameter at database level. By default, this parameter is set to automatic so that DB2 calculates the amount of memory allocated for the database.

The following memory areas contained in the database shared memory are extremely important in performance tuning:

- Buffer pools
- Lock list
- Database heap (includes the log buffer)
- Utility heap
- Package cache
- Catalog cache

Although the total amount of database global memory cannot be increased or decreased while the database is active, all of the memory areas listed above can be changed dynamically while the database is running. Refer to Chapter 8, "Tuning DB2 for Content Manager" on page 171 for an in-depth look at the configuration parameter values and recommendations.

A system administrator needs to consider overall balance of memory usage on the system. Different applications running on the operating system might use memory in different ways.

For the Content Manager application, the majority of system memory should not be given to DB2 when using AIX. Only about 20% of available memory should be assigned to DB2 and the rest should be used for disk cache. As you will see often in this book, tuning is mandatory and should be performed on a regular basis to maintain optimal system performance.

Figure 2-5 shows different portions of memory that the database manager allocates for various uses.

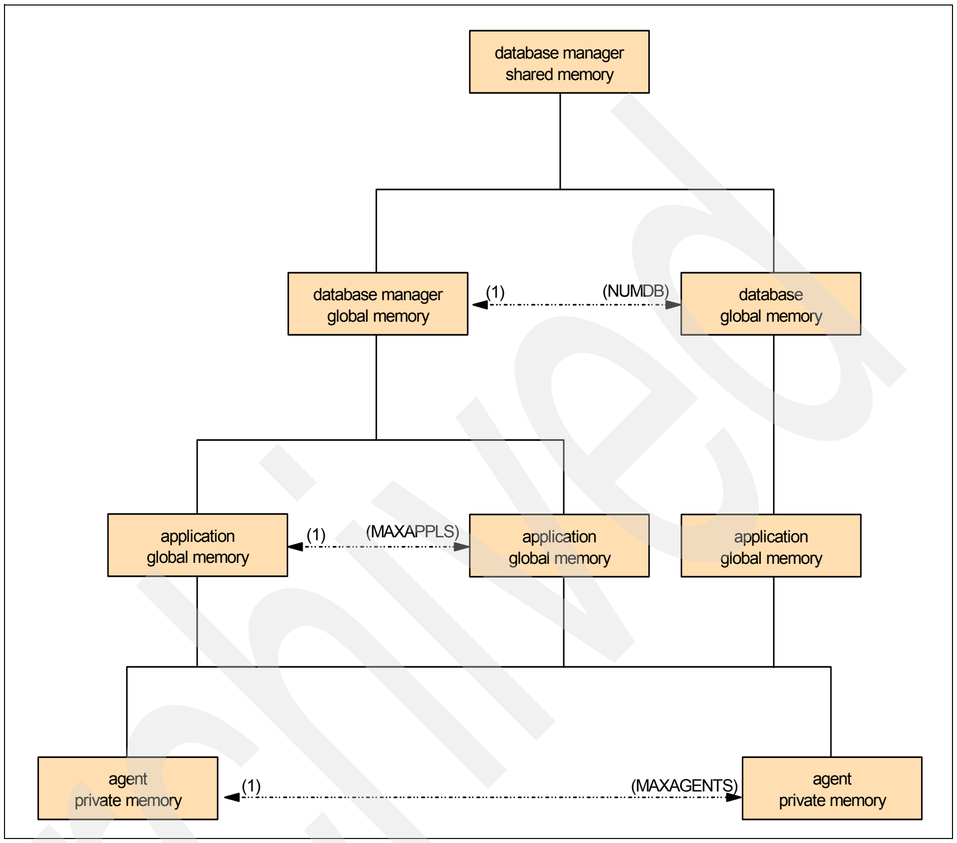

database manager shared memory
database manager global memory
(1)
(NUMDB)
database global memory
application global memory
(1)
(MAXAPPLS)
application global memory
application global memory
agent private memory
(1)
(MAXAGENTS)
agent private memory

Figure 2-5 DB2 memory model

When an application connects to a database in a partitioned database environment, in a non-partitioned database with the database manager intra-partition parallelism configuration parameter (INTRA_PARALLEL) enabled, or in an environment in which the connection concentrator is enabled, multiple applications can be assigned to application groups to share memory. Each application group has its own allocation of shared memory. In the application-group shared memory, each application has its own application control heap, but uses the share heap of the application group.

The following three database configuration parameters determine the size of the application group memory:

- The APPGROUP_MEM_SZ parameter, which specifies the size of the shared memory for the application group

- The GROUPHEAP_RATIO parameter, which specifies the percent of the application-group shared memory allowed for the shared heap
- The APP_CTL_HEAP_SZ parameter, which specifies the size of the control heap for each application in the group

The performance advantage of grouping application memory use is improved cache and memory-use efficiency.

The figure also lists the following configuration parameter settings, which limit the amount of memory that is allocated for each specific purpose. In a partitioned database environment, this memory is allocated on each database partition:

- NUMDB

  This parameter specifies the maximum number of concurrent active databases that different applications can use. Because each database has its own global memory area, the amount of memory that might be allocated increases if you increase the value of this parameter. When you set NUMDB to automatic, you can create any number of databases and memory consumption grows accordingly. Set this parameter to the number of databases within the instance.
- MAXAPPLS

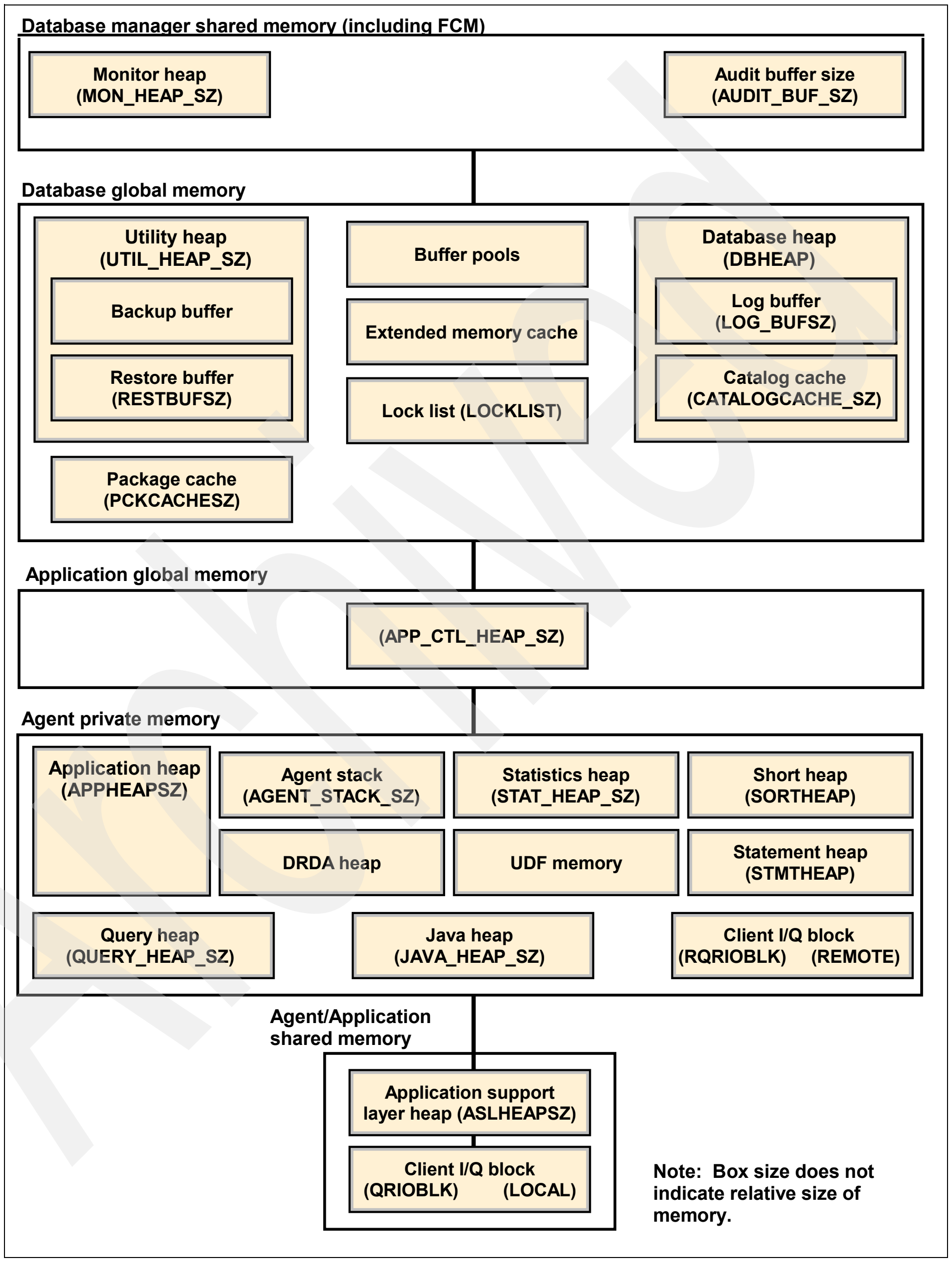
  This parameter specifies the maximum number of applications that can simultaneously connected (both local and remote) to a database. It affects the amount of memory that might be allocated for agent private memory and application global memory for that database. Note that this parameter can be set differently for every database. The default is automatic, which has the effect of allowing any number of connected applications. We do not suggest using the default. Set this parameter to the expected number of connections to the database.
- MAXAGENTS and MAX_COORDAGENTS for parallel processing

  These parameters limit the number of database manager agents that can exist simultaneously across all active databases in an instance. (MAX_COORDAGENTS is not shown in the figure.) Together with MAXAPPLS, these parameters limit the amount of memory allocated for agent private memory and application global memory. The value of MAXAGENTS should be at least the sum of the values for maxappls in each database that are allowed to be accessed concurrently.

Database manager shared memory

Memory space is required for the database manager to run. This space can be very large, especially in intra-partition and inter-partition parallelism environments.

Figure 2-6 shows how memory is used to support applications. The configuration parameters shown enable you to control the size of this memory by limiting the number and size of memory segments, which are portions of logical memory.



Database manager shared memory (including FCM)
Monitor heap (MON_HEAP_SZ)
Audit buffer size (AUDIT_BUF_SZ)
Database global memory
Utility heap (UTIL_HEAP_SZ)
Backup buffer
Restore buffer (RESTBUFSZ)
Package cache (PCKCACHESZ)
Buffer pools
Extended memory cache
Lock list (LOCKLIST)
Database heap (DBHEAP)
Log buffer (LOG_BUFSZ)
Catalog cache (CATALOGCACHE_SZ)
Application global memory
(APP_CTL_HEAP_SZ)
Agent private memory
Application heap (APPHEAPSZ)
Agent stack (AGENT_STACK_SZ)
Statistics heap (STAT_HEAP_SZ)
Short heap (SORTHEAP)
DRDA heap
UDF memory
Statement heap (STMTHEAP)
Query heap (QUERY_HEAP_SZ)
Java heap (JAVA_HEAP_SZ)
Client I/Q block (RQRIOBLK) (REMOTE)
Agent/Application shared memory
Application support layer heap (ASLHEAPSZ)
Client I/Q block (QRIOBLK) (LOCAL)
Note: Box size does not indicate relative size of memory.

Figure 2-6 Memory usage

You can predict and control the size of this space by reviewing information about database agents. Agents running on behalf of applications require substantial memory space, especially if the value of MAXAGENTS is not appropriate.

For partitioned database systems, the fast communications manager (FCM) requires substantial memory space, especially if the value of FCM_NUM_BUFFERS is large. In addition, the FCM memory requirements are either allocated from the FCM buffer pool, or from both the database manager shared memory and the FCM buffer pool, depending on whether the partitioned database system uses multiple logical nodes.

2.1.3 Performance-related concepts

The DB2 database management system provides a set of features and concepts that are closely related to the performance of a DB2 database system. Some of them reflect the architectural concepts that we discussed in 2.1.2, "DB2 architecture overview" on page 29. Depending on your specific database application, each of them can cause a performance bottleneck in your database environment if the database is not set up according to the requirements of the application. Thus it is essential to know and understand these concepts for effective DB2 performance tuning. In this section, we give an overview of these concepts and describe how they affect database performance. In Chapter 8, "Tuning DB2 for Content Manager" on page 171, we go through the individual configuration steps to adjust the setup of your database environment. In 11.4, "Monitoring and analysis tools for DB2" on page 295, we provide an overview of the tools and techniques that enable you on the individual platforms to verify that your current database setup meets your requirements.

This section requires you to be familiar with the basic concepts and architecture as outlined in 2.1.1, "Basic DB2 database concepts" on page 20 and 2.1.2, "DB2 architecture overview" on page 29.

Among others, we specifically discuss the following performance-related concepts:

- Query execution and access plan
- Indexes
- Data placement
- Clustering

Query execution and access plan

Any SQL query sent to the database is first passed to the SQL Compiler. As part of this SQL Compiler process, the query optimizer analyzes a query. The Compiler develops alternative strategies, called *access plans*, for processing the query. DB2 UDB evaluates the access plans primarily on the basis of information about the data source capabilities (such as disk I/O capabilities) and about the data itself. In general, many access plans and variations are examined before the most optimum access plan is chosen based on the available information.

Depending on the type of query compilation and access plan, evaluation is done at different points in time:

- For Static SQL, the SQL query is embedded in some application program and explicitly bound to the database during the process of making this application program executable. In this case, compilation of the SQL query and evaluation of access plans is done once during the process of binding the application. For performance reasons, the access plan is then stored so that it can be reused whenever the query is perform. Nevertheless, the user might (and should!) explicitly request to recompile a query whenever any of the facts has changed that led the compiler to its choice of access plan, especially whenever the system catalog statistics have been updated as described in "System catalog statistics" on page 46.
- For Dynamic SQL, including all SQL queries that are entered in some interactive query tool, the SQL query is compiled whenever it is submitted to the database for execution. DB2 might apply some intelligent query caching to avoid recompiling identical queries, but this is not under full control of the user.

The access plan is the key when it comes to the performance of an individual query. Although an access plan is something internal and not automatically exposed to the user, DB2 provides means to visualize and analyze it. (In Chapter 11, "Performance monitoring, analysis, and tracing" on page 281, we show how to use these tools.) An access plan might be poor or good. In a situation when there is just a specific query that shows a poor performance, analysis of the access plan of this query typically provides the clue to the resolution of the performance problem.

Key points:

- An access plan of a query represents the database manager's query execution model.
- The query optimizer selects the optimum access plan based on various information.
- The access plan is the key element to look at when analyzing performance problems of individual queries.

Indexes

An index can be viewed as a means to support efficient access to data in a database table. An index always covers a set of columns of the underlying table, and after it is defined it can be used by the query optimizer to choose an access plan that makes efficient use of an index access instead of a costly table scan. An index implicitly defines an order of the entries in a table based on the values of the index columns. An index does not automatically speed up any query on the underlying table. This is because an index can only speed up access to those columns of a table that are included in the index, but remember that the query optimizer takes into account a variety of information to determine the optimum access plan. An index is just one among many aspects that the optimizer considers.

Technically, an index is an ordered set of pointers to rows in a base table. Each index is based on the values of data in one or more table columns. When an index is created, the database manager builds this object and maintains it automatically.

Although it can speed up query execution, an index can slow down data manipulation such as INSERT, UPDATE, and DELETE on the table due to synchronous maintenance of the index. Furthermore, an index can reduce the degree of parallel access of multiple users and applications because DB2 might place locks on index entries when accessing data. This is similar to locks on table entries. These locks enable the database manager to virtually isolate and protect parallel applications and users from interfering with each other in an undesirable way. Depending on the access pattern, this might even result in deadlocks that would not occur without the index. Thus, an index should be introduced only to speed up a query after carefully evaluating the potential trade-off that might come as negative impact on performance for data manipulation.

In 11.4, “Monitoring and analysis tools for DB2” on page 295, we go into detail about how to find out whether some index may be introduced in order to speed up performance of a query.

In addition to the performance-related aspect of an index that we discussed here, an index can also be used for database design purpose. If an index is characterized as UNIQUE, it does not allow multiple entries in the table with identical values of the index-columns.

Although the database manager takes care of automatically maintaining the index so that its pointers to table entries are kept consistent and correct, an index might need some explicit maintenance. This is, when after many updates, the index gets fragmented. In this situation, we might want to reorganize the index. (In 8.3, “Regular routine maintenance” on page 181, we describe how to do this.)

Key points:

- An index can be created to speed up a query (or a set of queries).
- An index creates an extra load on the database system when updating data in a table.
- An index might need some explicit maintenance for reorganization.

Data placement

Data placement and the types of disks can play a significant role in the performance and availability of a database application. The overall objective is to spread the database effectively across as many disks as possible to try to minimize I/O wait. DB2 supports a variety of data placement strategies including mirroring, striping, and RAID devices. The administrator needs to understand the advantages and disadvantages of disk capabilities to arrive at the most appropriate strategy for an existing environment.

Four aspects of disk-storage management affect performance:

- Division of storage

  How you divide a limited amount of storage between indexes and data and among table spaces determines to a large degree how each will perform in different situations.

- Wasted storage

  Wasted storage in itself might not affect the performance of the system that is using it, but wasted storage is a resource that can be used to improve performance elsewhere.

- Distribution of disk I/O

  How well you balance the demand for disk I/O across several disk storage devices and controllers can affect how fast the database manager can retrieve information from disks.

- Lack of available storage

  Reaching the limit of available storage can degrade overall performance.

You should ensure that the log subdirectory is mapped to different disks than those disks that are used for your data. This can be a substantial performance benefit because the log files and database containers do not compete for the movement of the same disk heads. Also, a disk problem would be restricted to your data or the logs but not both.

Key points:

- The physical placement of data on disks and distribution over multiple disks has major impact on performance.
- A general goal should be to distribute data over multiple disks and to separate the database log from the data.

Clustering

When evaluating a query and executing an access plan, a typical operation of the database manager retrieves records from a table with a given value for one or more of its columns. This directly reflects the “SELECT ... FROM ... WHERE ...” structure of SQL queries. If multiple entries match for the given SELECT criterion, it appears to be desirable to find as many of these entries on as few physical data pages as possible in order to minimize disk I/O. Thus, the concept of *clustering* in general deals with physically arranging data that is supposed to be retrieved in a sequential way such that this logically sequential data resides on sequential physical data pages. However, as data pages fill up over time, clustering cannot be guaranteed and data reorganization might be required.

In a database system such as DB2, the concept of clustering is closely linked to the concept of indexes. That is, the sequential order implicitly defined by an index is used as the clustering order. This implies that there is a maximum of one clustering index for each table.

Physical arrangement of data tables is completely within the responsibility of the database manager, so maintenance of the clustering order is done automatically. In an environment with heavy updates on data, this might result in measurable performance overhead — analogous to automatic index maintenance.

Figure 2-7 on page 44 illustrates how a table with at least the two columns Region and Year may be clustered using an index on Region.

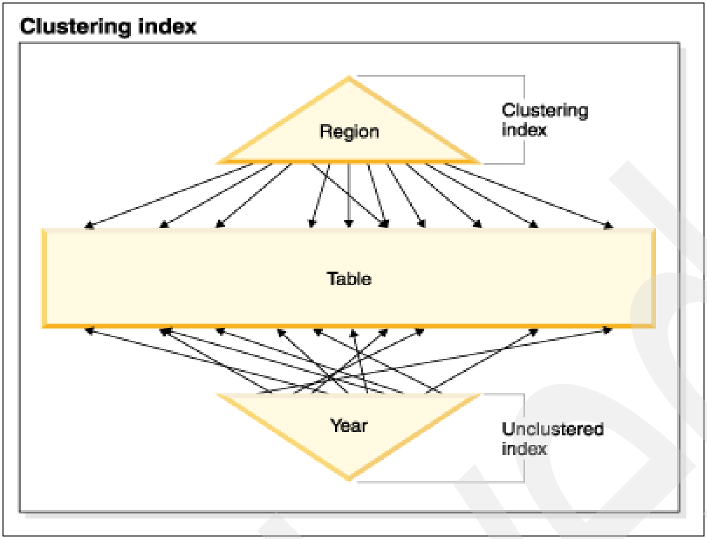


Figure 2-7 Table with clustering index

Key points:

- Clustering can be used to minimize I/O of query processing assuming the query demands sequential access to logically sequential data (range-queries).
- Clustering causes some performance overhead for data update.
- Clustering might require some explicit maintenance for reorganization.

Multi-dimensional clustering

Multi-dimensional clustering extends the idea of physically locating data contiguously when this data is logically coherent. Although the name of this concept might suggest it, this is not just clustering with respect to more than just one clustering dimension. It introduces a new and different organization of pointers to table entries that might improve performance through improved index efficiency and faster data retrieval. Although clustering maintains pointers to table data at record level, multi-dimensional clustering uses pointers to contiguous data blocks. For any of these referenced data blocks, DB2 makes sure that all records in these blocks have the same value with respect to any of the clustering dimensions. Blocks can be extended so that clustering can be maintained as data is updated.

Figure 2-8 illustrates this concept based on the table known from Figure 2-7 on page 44, using columns Region and Year each as clustering dimensions.

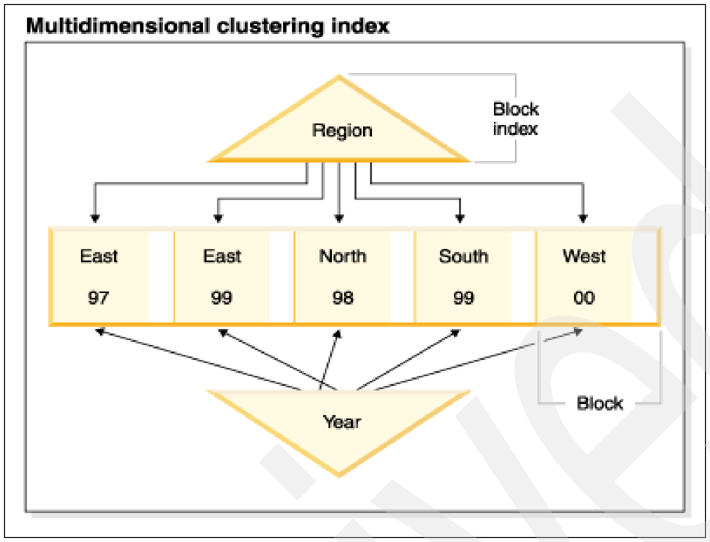


Figure 2-8 Table with multi-dimensional clustering index

In this example both indexes, on Region as well as on Year, use pointers to contiguous blocks of data, making them small and very efficient.

Key points:

- Multi-dimensional clustering can significantly speed up certain types of queries.
- Multi-dimensional clustering does *not* require explicit maintenance because clustering is guaranteed by the database manager.

Materialized query tables

A *materialized query table* (MQT) is a table whose definition is based on the result of a query, and whose data is in the form of pre-computed results that are taken from one or more tables on which the materialized query table definition is based. An MQT can be viewed as a table that caches the result of the query. The MQT thus holds precomputed data redundantly. When you create an MQT, you can select whether you want to be responsible for updating the MQT and keeping it consistent with the query that forms the MQT, or whether you want to leave this up to the database manager. Furthermore, if you leave the database manager responsible for maintaining consistency of the MQT, you can select whether you want to have any update to tables reflected immediately in the MQT or whether you want to explicitly request any update to the MQT. The decision about who is made responsible for maintaining consistency and when updates are propagated is primary guided by functional requirements in your database application environment. From a performance point of view, it might be better to update the MQT in some maintenance period at night, but your application scenario might require that updates are reflected immediately.

When you have created an MQT, this MQT can implicitly be reused when executing further queries. This is *not* restricted to exactly the same query. Instead, an MQT can be reused in execution of any query that subsumes the query that forms the MQT. The decision whether to use the MQT or the query itself is made by the query optimizer as part of the general query execution process. The optimizer bases its decision on the estimation of costs for access plans with and without MQT.

In addition to the performance improvement that comes with MQTs by reusing query results, there is another option for further performance boost: Indexes can be added to MQTs to improve performance for certain queries.

Automatic maintenance of the MQT during updates implies some performance overhead that you should consider when creating an MQT. In a situation where updates are rare and similar expensive queries are repeatedly run against the database, MQTs can be the technique of choice to resolve performance bottlenecks.

Key points:

- MQTs provide for storing precomputed query results redundantly in tables.
- The query optimizer makes use of MQTs whenever this results in a less expensive access plan.
- The database manager supports automatic maintenance of consistency of the MQT with the query that forms the MQT.
- Maintenance of MQT consistency, either user-controlled or automatic, implies some performance overhead when updating data.

System catalog statistics

As discussed in “Query execution and access plan” on page 39, the query optimizer’s decision about the optimum access plan for query execution is guided by its knowledge of the database’s environment (such as disk speeds) and of the data in the database itself. Other than changes in configuration parameters, the database’s environment typically does not change much over time, and most changes are exposed to the query optimizer. But as data is loaded and updated in the database, the characteristics of this data can change significantly over time. This can cause the optimizer to select a completely inappropriate access plan, which can result in severe performance problems. Thus it is very important to keep the optimizer's knowledge about the data in the database up to date.

Important: Maintaining up-to-date statistics in a database is crucial for performance of the database system. It has been shown that many performance bottlenecks, especially in Content Manager environments, are due to outdated statistics.

Knowledge about the data internally is represented as some kind of statistics; for example, about distribution of values in columns of a table, selectivity and cluster ratio of indexes, numbers of rows in tables, and fragmentation of data. Collecting statistics is done by the database manager itself, but because this requires some locks on the data and it consumes performance-related resources, this can interfere with users and applications simultaneously accessing the database. Thus, updating statistics in a database is considered to be administrative task that must be controlled by a database administrator. The database administrator can do this either by manually requesting this or by specifying an administrative time window during which the update of statistics is done automatically. Doing this, the administrator can also control the level of details of the statistics that are collected. In 8.3, "Regular routine maintenance" on page 181, we describe how to do this.

Whenever the nature of the data in the database has changed significantly (for example, after some initial load or after a series of updates), it is important to have the database manager update statistics to ensure that the query optimizer has the most accurate knowledge to make profound decisions about optimum access plans. This is also true if new objects such as indexes and tables are added to the database system, because otherwise the query optimizer will not have any statistics information about these objects.

Key points:

- Statistics about the data in the database is the most important source of knowledge that the query optimizer uses to decide which access plan to select.
- Statistics must be maintained (updated) regularly, especially when data is updated in the database.
- The database manager supports manual as well as automatic update of statistics.

Table space types: SMS versus DMS

As described in 2.1.2, "DB2 architecture overview" on page 29, DB2 supports two different types of table spaces: system managed space (SMS) and database managed space (DMS). The actual storage of a table space is provided by containers that can either be files in the file system or devices (such as a UNIX raw disk) where only SMS table spaces may use only file containers and DMS table spaces can use file containers or device containers. The type of table space is determined when it is created.

Both types come with certain advantages that you should consider before the table space is created. We give a brief overview of the trade-off between these two options. For completeness and because of the importance of this topic, we do not restrict simply to performance-related aspects.

- File container advantage:

  The operating system might cache pages in the file system cache. Thus, even if some pages are not in a buffer pool, it does not require disk I/O to load and access them. That is, the file system caching can eliminate I/O that would otherwise have been required. This can allow for a smaller buffer pool and it might be advised to increase the file system buffer in turn.

  Especially in the case of table space for long data (using LOB or LONG data types), the database manager does not cache the data at all in its buffers. Depending on the access pattern of an application, it might be an advantage for performance if this table space is kept in file containers in order to make use of file system caching.

  Because system catalogs contain some LOB columns, you should keep them in SMS table spaces or in DMS-file table spaces.

- Device container advantage:

  The location of the data on the disk can be controlled, if this is allowed by the operating system.

- SMS table space advantages:
  - If file containers are used, space is not allocated by the system until it is required.
  - Creating a table space requires less initial work, because you do not have to predefine the containers.
- DMS table space advantages:
  - The size of a table space can be increased by adding or extending containers. Existing data can be rebalanced automatically across the new set of containers to retain optimal I/O efficiency.
  - If all table data is in a single table space, a table space can be dropped and redefined with less overhead than dropping and redefining a table.

As a general rule, a well-tuned set of DMS table spaces will outperform SMS table spaces (but it may require hard work to get there).

In 11.4, “Monitoring and analysis tools for DB2” on page 295, we demonstrate how to monitor and analyze whether indicators in your environment show that a different table space type might improve performance.

In 8.4, “SMS to DMS table space conversion” on page 188, we discuss how to convert SMS table spaces into DMS table spaces.

Key points:

- The database manager supports different types of containers that come with different characteristics with respect to performance and administrative overhead.
- The database manager supports different types of table spaces that correspond to container types and come with different characteristics with respect to performance and administrative overhead.
- The type of a table space is determined when it is created; any change of the type afterwards requires a migration.

Prefetching and I/O servers

As described in 2.1.2, “DB2 architecture overview” on page 29, prefetching deals with how the database manager proactively retrieves one or more pages from disk into some buffer pool in the expectation that they will be required by an application. Prefetching index and data pages into the buffer pool can help improve performance by reducing the I/O wait time. In addition, parallel I/O enhances prefetching efficiency, such as when data is spread over multiple disks. The database management system uses dedicated threads (in Windows operating system) response processes (in UNIX-based operating systems) to perform prefetching. While these *prefetchers* are used for asynchronous fetching of pages into buffer pools, the *page cleaners* analogously are used for asynchronous write of pages from buffer pool to disk when they are no longer needed. Prefetchers and page cleaners are also called *I/O servers*. The number of I/O servers that is used by the database management system can be adjusted to reflect the given database environment and the requirements of applications.

Key points:

- Prefetching is a built-in technique to minimize disk wait time for data read.
- I/O between buffer pool and disk in general is done asynchronously.
- I/O can be parallelized using multiple I/O servers.
- Parallelizing I/O might improve performance.

Coordinator agents

As described in 2.1.2, “DB2 architecture overview” on page 29, each connection to database corresponds to a *coordinator agent* at server side that performs all the application requests. These agents can be created as needed or in advance and kept in a pool. Coordinator agents are owned by the database manager and shared among all databases managed by database manager. The maximum number of applications that can be connected to the database simultaneously can be adapted, as can the maximum number of agents that can be created by the database manager. Actually the maximum number of concurrent connections might even be higher than the maximum number of coordinator agents. This is known as *database connection pooling* and can be useful if concurrent connections tend to fall asleep on a regular basis (that is, when the application does not really interact with the database).

There are a certain performance-related aspects regarding the maximum number of connections and the maximum number of coordinator agents:

- Each coordinator agent consumes a certain amount of resources in the database environment (such as shared memory for various heaps) as well as in the underlying operating system. Especially in a UNIX environment where each agent is implemented as process, you should consider this consumption of system resources that comes with each process. There should not be more coordinator agents than are actually needed.
- Dynamic creation of a coordinator agent on demand consumes operating system resources and time. Again, especially in a UNIX environment, you should in general consider the overhead that comes with creation of a process.

On Windows-based platforms, agents are implemented as threads that come with less overhead in terms of operating system resources. With respect to database resources, there is no difference between UNIX and Windows.

In 8.6, “Parameter tuning” on page 202, we show how to control the number of agents and connections.

Key points:

- Agent processes (resp. threads in Windows) are those entities in a database manager environment that perform the application’s requests.
- The number of pooled agents can be controlled as well as the maximum number of agents.
- The configuration of the number of agents in your database environment has an impact on the performance of the database.

Locking

In a database system, typically many applications concurrently work on the same data. If this is done in an uncontrolled way, this might lead to anomalies that are known as:

- Lost update
- Uncommitted read
- Non-repeatable read
- Phantom read

It is the responsibility of the database manager to provide protection against these anomalies to applications. The database manager does this by maintaining locks on individual database objects such as rows in a table, index entries, or even whole tables. If a database lock is held on an object by an application, the database manager takes care that no other application accesses this element in a way such that one of the above anomalies shows up. This is done by suspending one of the applications from further execution, which in turn affects transaction throughput and visible performance. To minimize this impact, applications can specify the degree of protection that they require against interference with concurrent applications. This is done using the concept of *isolation levels*.

Key points:

- Locking is a concept to protect concurrent applications from interfering with each other and to protect consistency of data.
- The number and nature of objects that are locked by concurrent applications generally affects performance in terms of transaction throughput.
- Applications have a certain influence on the number and nature of locks that the database manager holds for them, but otherwise, locking is done implicitly under control of the database manager.
- The database manager uses certain configurable memory pools to store locks.
- Configuration of these memory pools affects the general locking behavior.

2.1.4 Routine maintenance

Routine maintenance of your system is essential in maintaining optimum performance. In this section, we provide an overview of what should be included as part of routine maintenance (ideally performed during a scheduled maintenance window during which there is little to no user access to the database) as well as how often it should be scheduled. Your schedule can be more or less frequent depending on the amount of changes within your system.

See Chapter 8, “Tuning DB2 for Content Manager” on page 171 for details about routine maintenance, including applicable commands.

The following routine maintenance are important:

- Keeping up with the latest software fix packs
- Monitoring system
- Cataloging statistics (runstats and rebind)
- Reorganization tables
- Pruning the log file
- Pruning the diagnostics file

Keeping up with the latest software fix packs

Software bugs are found or reported by customers and new enhancements are continually being developed for software. DB2 is no exception. It is extremely important to keep your software up to the latest fix pack level so that you get the benefit of these software changes. When calls to external support are placed, one of the first things asked is whether your software is at the latest fix pack level. If not, you will be asked to update it before they will continue to troubleshoot your problem so that they do not try to solve a problem that has already been solved.

Monitoring system

Monitoring your system on a regular basis is vital to ensuring that the system is running in a healthy and performance-oriented way. The database monitor switches must be turned on for sufficient time to enable DB2 to collect the data.

We recommend regularly taking a DB2 snapshot of the database manager (instance) and each database, and that you keep the snapshot data in a file, a spreadsheet, or loaded into a DB2 table for historical and trend analysis. Analyze the results for immediate issues such as a high number of package cache overflows or database files closed. Perform a trend analysis to determine whether a problem is building. Is the buffer pool hit ratio decreasing? Are the sort overflows steadily increasing? If so, take appropriate corrective action.

Cataloging statistics (runstats and rebind)

Statistical data stored in the system catalogs helps the optimizer choose the best access plan for queries. Executing the **`runstats`** command to update this statistical data has a tremendous impact on performance and should be one of the first things done (in conjunction with a **`rebind`** command) whenever system performance is perceived to be poor.

We recommend executing **`runstats`** and **`rebind`** commands:

- At frequent regular intervals for tables whose content changes continually. This might be daily or weekly.

- After creating a table or an index.
- After any operation that adds or changes data in a significant number of table rows. Such operations include batch updates and data loading.

These utilities could have a negative impact on performance and should be run during low user activity times, preferably during scheduled routine maintenance.

Reorganization tables

After many changes to table data, logically sequential data might be on nonsequential physical data pages, which can cause the database manager to perform additional read operations to access the data. This increased I/O has a negative impact on performance. Running a table reorganization will re-cluster the data to match the primary index for efficient access and will reclaim space.

Reorganizing a table takes more time than running statistics. A reorganization of a table might be necessary when any of the following are true:

- Table has a high volume of insert, update, and delete activity.
- Executing **`reorgchk`** indicates a need to reorganize the table.

We recommend executing the **`reorgchk`** command at least weekly. If the amount of change to the data of your system is high, you might want to run it daily. Reorganize any table that has been flagged.

Pruning the log file

When using the Log Retain strategy, log files are created continually as needed, filling up disk space of the log directory in the process. When the file system or disk is full, DB2 will be unable to create log records and will start writing error messages continually into the diagnostics log file (db2diag.log) until the problem is resolved. For this reason, old log files must be pruned or removed regularly before the file system gets full. The frequency of this action will depend on how fast it takes to use up most of the space in the log directory.

Log files should not be deleted until it is absolutely sure that they will not be needed for any type of recovery. We recommend moving non-active log files from the log directory to another "logs to be deleted" directory located on a different disk. Depending on your recovery strategy, when you are sure that the logs are no longer necessary, they can be deleted from this directory. TSM can also be used as a destination for the archival of log files.

Pruning the diagnostics file

DB2 continually writes all diagnostic information to the db2diag.log file, which is used by DB2 customer support for troubleshooting purposes. You control the level (and amount) of detailed information that is written to the file by using the DIAGLEVEL configuration parameter. Valid values are from one (minimal information) to four (maximum information), with three being the default. Increasing the amount of diagnostic output can result in performance degradation and insufficient storage conditions in your database instance file system.

The file will continue to grow forever and consume space, so it is necessary to maintain the file. Instead of manually editing the file to crop it down or simply deleting the file, we recommend that you periodically move the file to another disk. (DB2 will automatically create a new db2diag.log file.) Each moved file would overlay the previously moved file, allowing for one version of historical data.

Attention: DB2 is a key underlying software of Content Manager and it plays a critical role in the performance of a Content Manager system. In this section, we merely touched the very basics of the DB2 components and architecture. In 11.4, “Monitoring and analysis tools for DB2” on page 295, we introduce some of the performance tuning monitoring tools for DB2. Chapter 6, “Best practices for Content Manager system performance” on page 143, and especially Chapter 8, “Tuning DB2 for Content Manager” on page 171 discuss DB2-specific tuning for a Content Manager system.

To better understand DB2 and DB2 performance tuning, we recommend reading the publications we mentioned at the beginning of this DB2 Universal Database overview section.

2.2 WebSphere Application Server overview

IBM WebSphere Application Server Version 5.1 is a comprehensive Java 2 Platform, Enterprise Edition 1.4, and Web services technology-based application server. It integrates enterprise data and transactions, and provides the core software to deploy, integrate, and manage Web applications.

Through a rich application deployment environment, you can build, manage and deploy dynamic Web applications, handle high-transaction volumes and extend back-end business data and applications to the Web. WebSphere Application Server Network Deployment offers advanced Web services that can operate across disparate application frameworks and business-to-business (B2B) applications, and provides virtual any-to-any connectivity with transaction management and application adaptability.

With multiple configuration options, WebSphere Application Server supports a wide range of scenarios, from simple administration of a single server to a clustered, highly available, high-volume environment with edge-of-network services. These specialized configuration options give you the flexibility to respond to an ever-changing marketplace without the costs of migrating to a different technology base.

In the following sections, we provide a *general* overview of J2EE, WebSphere Application Server architecture, and architectural components. The material provided are an extraction from the existing WebSphere publications. For more detailed information, refer to:

- IBM WebSphere Application Server, Version 5.1 Information Center

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1/index.jsp`

- *IBM WebSphere Application Server, Version 5.1: Getting started*, SC31-6323

  `http://www.ibm.com/support/docview.wss?uid=pub1sc31632301`

- WebSphere Application Server Information Center:

  `http://www.ibm.com/software/webservers/appserv/was/library`

2.2.1 J2EE overview

Java 2 Platform, Enterprise Edition (J2EE) is the standard for developing, deploying, and running enterprise applications. WebSphere Application Server supports all of the J2EE APIs.

The J2EE standard applies to all aspects of designing, developing, and deploying multi-tier, server-based applications. The J2EE standard architecture consists of:

- Standard application model for developing multi-tier applications
- Standard platform for hosting applications
- Compatibility test suite for verifying that J2EE platform products comply with the J2EE platform standard
- Reference implementation providing an operational definition of the J2EE platform

The J2EE platform specification describes the runtime environment for a J2EE application. This environment includes application components, containers, and Resource Manager drivers. The elements of this environment communicate with a set of standard services that are also specified.

J2EE platform roles

The J2EE platform also defines a number of distinct roles performed during the application development and deployment life cycle:

- Product provider designs and makes available for purchase the J2EE platform, APIs, and other features defined in the J2EE specification.

- Tool provider provides tools used for the development and packaging of application components.
- Application component provider creates Web components, enterprise beans, applets, or application clients for use in J2EE applications.
- Application assembler takes a set of components developed by component providers and assembles them in the form of an enterprise archive (EAR) file.
- Deployer is responsible for deploying an enterprise application into a specific operational environment.
- System administrator is responsible for the operational environment in which the application runs.

Product providers and tool providers have a product focus. Application component providers and application assemblers focus on the application. Deployers and system administrators focus on the runtime.

These roles help identify the tasks to be performed and the parties involved. Understanding this separation of roles is important for explaining the approach that should be taken when developing and deploying J2EE applications.

J2EE benefits

The J2EE standard empowers customers. Customers can compare J2EE offerings from vendors and know that they are comparing parallel products. Comprehensive, independent compatibility test suites ensure vendor compliance with J2EE standards.

Some benefits of deploying to a J2EE-compliant architecture are:

- A simplified architecture based on standard components, services, and clients, that takes advantage of write-once, run-anywhere Java technology.
- Services providing integration with existing systems, including Java Database Connectivity (JDBC); Java Message Service (JMS); Java Interface Definition Language (Java IDL); JavaMail™; and Java Transaction API (JTA and JTS) for reliable business transactions.
- Scalability to meet demand, by distributing containers across multiple systems and using database *connection pooling*, for example.
- A better choice of application development tools, and components from vendors providing off-the-shelf solutions.
- A flexible security model that provides single sign-on support, integration with legacy security systems, and a unified approach to securing application components.

The J2EE specifications are the result of an industry-wide effort that has involved, and still involves, a large number of contributors. IBM alone has contributed to defining more than 80% of the J2EE APIs.

Application components and their containers

The J2EE programming model has four types of application components, which reside in four types of containers in the application server:

- Enterprise JavaBeans™ components, residing in the EJB™ container
- Servlets and JavaServer™ Pages™ files, residing in the Web container
- Application clients, residing in the application client container
- Applets, residing in the applet container

See 2.2.3, “Architectural components” on page 60 for a description of the components and containers.

J2EE containers provide runtime support for the application components. There must be one container for each application component type in a J2EE application. By having a container between the application components and the set of services, J2EE can provide a federated view of the APIs for the application components.

A *container* provides the APIs to the application components used for accessing the services. It also might handle security, resource pooling, state management, naming, and transaction issues.

Standard services

The J2EE platform provides components with a set of standard services that they can use to interact with each other:

- HTTP and HTTPS
- Java Transaction API (JTA)
- Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP)
- Java Interface Definition Language (Java IDL)
- Java Database Connectivity (JDBC)
- Java Message Service (JMS)
- Java Naming and Directory Interface™ (JNDI)
- JavaMail and JavaBeans Activation Framework (JAF)
- Java Transaction API (JTA and JTS)
- XML
- J2EE Connector Architecture
- Resource Managers

For a description of each standard service, see the Sun Web page:

http://java.sun.com/products

J2EE packaging

One of the most significant changes introduced by the J2EE specification is how application components are packaged for deployment. During a process called assembly, J2EE components are packaged into modules. Modules are then packaged into applications. Applications can be deployed on the application server. Each module and application contains a J2EE deployment descriptor. The deployment descriptor is an XML file providing instructions for deploying the application.

2.2.2 Three-tier architecture

WebSphere Application Server provides the application logic layer in a three-tier architecture, enabling client components to interact with data resources and legacy applications. Collectively, three-tier architectures (see Figure 2-5 on page 36) are programming models that enable the distribution of application functionality across three independent systems, typically:

- Client components running on local workstations (tier 1)
- Processes running on remote servers (tier 2)
- A discrete collection of databases, Resource Managers, and mainframe applications (tier 3)

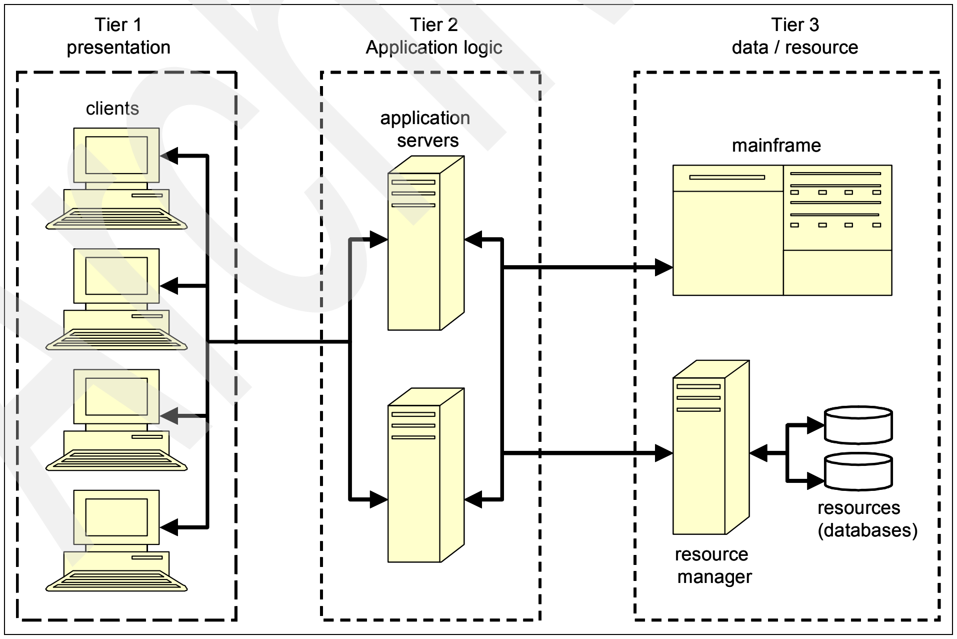


Figure 2-9 Typical WebSphere three-tier architecture

Tier 1 (presentation)

Responsibility for presentation and user interaction resides with the first-tier components. These client components enable the user to interact with the second-tier processes in a secure and intuitive manner. WebSphere Application Server supports several client types.

Clients do not access the third-tier services directly. For example, a client component provides a form on which a customer orders products. The client component submits this order to the second-tier processes, which check the product databases and perform tasks needed for billing and shipping.

Tier 2 (application logic layer)

The second-tier processes are commonly referred to as the application logic layer. These processes manage the business logic of the application, and are permitted access to the third-tier services. The application logic layer is where the bulk of the processing work occurs. Multiple client components are able to access the second-tier processes simultaneously, so this application logic layer must manage its own transactions.

Continuing with the previous example, if several customers attempt to place an order for the same item, of which only one remains, the application logic layer must determine who has the right to that item, update the database to reflect the purchase, and inform the other customers that the item is no longer available. Without an application logic layer, client components access the product database directly. The database is required to manage its own connections, typically locking out a record that is being accessed. A lock can occur simply when an item is placed into a shopping cart, preventing other customers from even considering it for purchase. Separating the second and third tiers reduces the load on the third-tier services, can improve overall network performance, and enables more eloquent connection management.

Tier 3 (data/resource)

The third-tier services are protected from direct access by the client components by residing within a secure network. Interaction must occur through the second-tier processes.

Communication among tiers

All three tiers must be able to communicate with each other. Open, standard protocols, and exposed APIs simplify this communication. Client components can be written in any programming language, such as Java or C++, and can run on any operating system, as long as they can speak with the application logic layer. Likewise, the databases in the third tier can be of any design, as long as the application layer can query and manipulate them. The key to this architecture is the application logic layer.

2.2.3 Architectural components

In this section, we cover major components in IBM WebSphere Application Server, including the application server and its containers, the HTTP server plug-in and embedded HTTP handling, and the virtual host configuration.

HTTP server and HTTP plug-in

IBM WebSphere Application Server works with an HTTP server to handle requests for servlets and other dynamic content from Web applications. The terms HTTP server and Web server are used interchangeably in this section.

The HTTP server and application server communicate using the WebSphere HTTP plug-in for the HTTP server. The HTTP plug-in uses an easy-to-read XML configuration file to determine whether a request should be handled by the Web server or the application server. It uses the standard HTTP protocol to communicate with the application server. It can also be configured to use secure HTTPS, if required. The HTTP plug-in is available for popular Web servers.

Application server

An application server in WebSphere is the process that is used to run servlet and EJB-based applications, providing both Web container and EJB container.

The application server collaborates with the Web server to return customized responses to client requests. Application code including servlets, JavaServer Pages (JSP™) files, Enterprise JavaBeans (EJB) components, and their supporting classes run in an application server. In keeping with the J2EE component architecture, servlets and JSP files run in a Web container, and enterprise beans run in an EJB container. You can define multiple application servers, each running in its own Java virtual machine (JVM).

- EJB container

  The EJB container provides the runtime services needed to deploy and manage EJB components, which are known as enterprise beans. It is a server process that handles requests for both session and entity beans.

  The enterprise beans (inside EJB modules) installed in an application server do not communicate directly with the server; instead, an EJB container provides an interface between the enterprise beans and the server. Together, the container and the server provide the bean runtime environment.

  The container provides many low-level services, including threading and transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained beans. A single container can hold more than one EJB JAR file.

An *EJB module* is used to package one or more enterprise beans into a single deployable unit. An EJB module is represented by a JAR file that contains the enterprise bean classes/interfaces and the bean deployment descriptors. An EJB module can be used as a stand-alone application, or it can be combined with other EJB modules, or with Web modules, to create a J2EE application. An EJB module is installed and run in an enterprise bean container.

- Web container

  Servlets and JavaServer Pages (JSP) files are server-side components that are used to process requests from HTTP clients, such as Web browsers. They handle presentation and control of the user interaction with the underlying application data and business logic. They can also generate formatted data, such as XML, for use by other application components.

  The Web container processes servlets, JSP files, and other types of server-side files. Each Web container automatically contains a single session manager. When handling servlets, the Web container creates a request object and a response object, then invokes the servlet service method. The Web container invokes the servlet destroy() method when appropriate and unloads the servlet, after which the JVM performs garbage collection.

  A Web container handles requests for servlets, JSP files, and other types of files that include server-side code. The Web container creates servlet instances, loads and unloads servlets, creates and manages request and response objects, and performs other servlet management tasks. WebSphere Application Server provides Web server plug-ins for supported Web servers. These plug-ins pass servlet requests to Web containers.

  A *Web module* represents a Web application. It is used to assemble servlets and JSP files, as well as static content such as HTML pages, into a single deployable unit. Web modules are stored in Web archive (WAR) files (.war) which are standard Java archive files.

  A Web module contains one or more servlets, JSP files, and other files. It also contains a deployment descriptor that declares the content of the module, stored in an XML file named web.xml. The deployment descriptor contains information about the structure and external dependencies of Web components in the module, and describes how the components are to be used at runtime.

  A Web module can be used as a stand-alone application, or it can be combined with other modules (other Web modules, EJB modules, or both) to create a J2EE application. A Web module is installed and run in a Web container.

- Application client container

  Application clients are Java programs that typically run on a desktop computer with a graphical user interface (GUI). They have access to the full range of J2EE server-side components and services.

  The application client container handles Java application programs that access enterprise beans, Java Database Connectivity (JDBC), and Java Message Service message queues. The J2EE application client program runs on client machines.

  This program follows the same Java programming model as other Java programs; however, the J2EE application client depends on the application client runtime to configure its execution environment, and uses the Java Naming and Directory Interface (JNDI) name space to access resources.

- Applet container

  An *applet* is a client Java class that typically runs in a Web browser, but can also run in a variety of other client applications or devices.

  Applets are often used in combination with HTML pages to enhance the user experience provided by a Web browser. They can also be used to shift some of the processing workload from the server to the client.

  The applet container handles Java applets embedded in a HTML document that reside on a client machine that is remote from the application server. With this type of client, the user accesses an enterprise bean in the application server through the Java applet in the HTML document.

Embedded HTTP server

A nice product feature is the HTTP handling capability, embedded in the application server, which enables an HTTP client to connect directly to the application server. Or, as described earlier, an HTTP client can connect to a Web server and the HTTP plug-in can forward the request to the application server.

The embedded HTTP transport is a service of the Web container. An HTTP client connects to a Web server and the HTTP plug-in forwards the requests to the embedded HTTP transport. The communication type is either HTTP or HTTPS between the plug-in and the embedded HTTP transport.

Administrative Console

The WebSphere Administrative Console provides an easy-to-use GUI to a WebSphere cell, runs in a browser, and connects directly to a WebSphere server or to a Deployment Manager node that manages all nodes in a cell.

Virtual host

A virtual host is a configuration that enables a single host machine to resemble multiple host machines. Resources that are associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP host name and port number that is used to request the servlet (for example, yourHostName:80). The default ports and aliases are:

- The default alias is *:80, using an external HTTP port that is not secure.
- Aliases of the form *:9080 use the embedded HTTP port that is not secure.
- Aliases of the form *:443 use the secure external HTTPS port.
- Aliases of the form *:9443 use the secure embedded HTTPS port.

When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet. If no match is found, an error (404) is returned to the browser.

WebSphere Application Server provides a default virtual host, aptly named "default_host," with some common aliases, such as the machine's IP address, short host name, and fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet. For example, it is "localhost:80" in the request http://localhost:80/servlet/snoop.

Virtual hosts enable the administrator to isolate, and independently manage, multiple sets of resources on the same physical machine.

2.2.4 Performance-related tools and concepts

WebSphere Application Server provides a set of features and tools that are closely related to the performance of the system. In this section, we give an overview of the performance-related concepts:

- Performance Monitoring Infrastructure
- Performance Advisors
- Dynamic caching
- Java Virtual Machine Profiler Interface
- Parameter hot list

Performance Monitoring Infrastructure

For the WebSphere environment, we provide the Performance Monitoring Infrastructure (PMI) to capture performance data with minimal performance impact to incorporate that data into an overall monitoring solution.

PMI uses a client-server architecture. The server collects performance data in memory within the WebSphere Application Server. This data consists of counters such as servlet response time and data connection pool usage. A client can then retrieve that data using a Web client, a Java client, or a Java Management Extension (JMX™) client. A client is an application that retrieves performance data from one or more servers and processes the data. Clients can include:

- Graphical user interfaces (GUIs) that display performance data in real time
- Applications that monitor performance data and trigger different events according to the current values of the data
- Any other application that needs to receive and process performance data

The PMI components and infrastructure have been extended and updated in WebSphere Application Server V5.1 to support the new management structure and to comply with the Performance Data Framework of the J2EE Management Specification. WebSphere Application Server contains *Tivoli Performance Viewer*, a Java client that displays and monitors performance data.

PMI is composed of components for collecting performance data on the application server side and components for communicating between runtime components and between the clients and servers.

Figure 2-10 on page 65 shows the overall PMI architecture. On the right side, the server updates and keeps PMI data in memory. The left side displays a Web client, Java client, and JMX client retrieving the performance data.

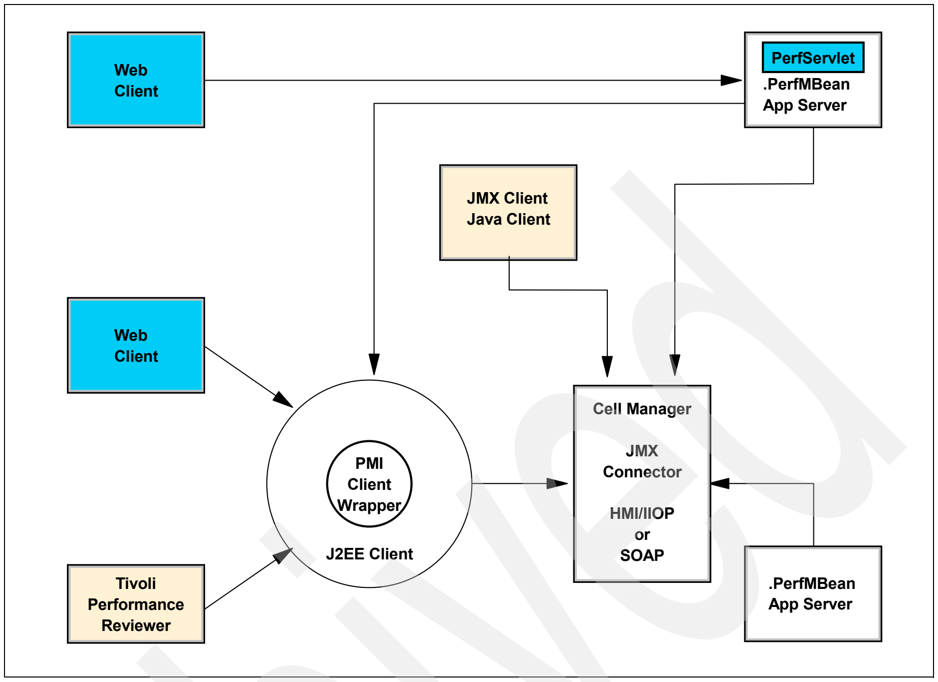


Figure 2-10   General PMI architecture

Performance Advisors

The Performance Advisors use the PMI data to suggest configuration changes to ORB service thread pools, Web container thread pools, connection pool size, persisted session size and time, prepared statement cache size, and session cache size. The Runtime Performance Advisor runs in the application server process, and the other advisor runs in the Tivoli Performance Viewer (TPV).

Dynamic caching

Server-side caching techniques have long been used to improve performance of Web applications. In general, caching improves response time and reduces system load. Until recently, caching has been limited to *static content*, which is the content that rarely changes. Performance improvements are greatest when *dynamic content* is also cached. Dynamic content is the content that changes frequently, or data that is personalized. Caching dynamic content requires proactive and effective invalidation mechanisms, such as event-based invalidation, to ensure the freshness of the content. Implementing a cost-effective caching technology for dynamic content is essential for the scalability of today's dynamic, data-intensive, Web infrastructures.

There are two basic types of content from the caching point of view:

- *Static content* does not change over long periods of time. The content type can be HTML, JSP rendering less dynamic data, GIF, and JPG.
- *Dynamic content* includes frequently updated content (such as stock exchange rates), as well as personalized and customized content. Although it is changing content, it also must be stable over a long enough time for meaningful reuse to occur. However, if some content is very frequently accessed, such as requests for pricing information of a popular stock, then even a short time of stability might be long enough to benefit from caching.

Java Virtual Machine Profiler Interface

The Java Virtual Machine Profiler Interface (JVMPI) enables the collection of information, such as data about garbage collection, and the Java virtual machine (JVM) API that runs the application server. The Tivoli Performance Viewer leverages a JVMPI to enable more comprehensive performance analysis.

JVMPI is a two-way function call interface between the JVM API and an in-process profiler agent. The JVM API notifies the profiler agent of various events, such as heap allocations and thread starts. The profiler agent can activate or deactivate specific event notifications based on the needs of the profiler.

JVMPI supports partial profiling by enabling the user to choose which types of profiling information to collect and to select certain subsets of the time during which the JVM API is active. JVMPI moderately increases the performance impact.

Parameter hot list

Each WebSphere Application Server process has several parameters that influence application performance. You can use the WebSphere Application Server Administrative Console to configure and tune applications, Web containers, EJB containers, application servers, and nodes in the administrative domain.

Always review the tuning parameter hot list, which is a subset of the tuning parameter index. These parameters have an important impact on performance. Because these parameters are application dependent, the parameter settings for specific applications and environments can vary.

2.2.5 Routine maintenance

WebSphere Application Server collects data about runtime and applications through the Performance Monitoring Infrastructure (PMI). Performance data can then be monitored and analyzed with a variety of tools.

Routine maintenance of WebSphere can be performed with the following tools:

- *Tivoli Performance Viewer* (TPV) is a GUI performance monitor for WebSphere Application Server. TPV can connect to a local or a remote host. Connecting to a remote host minimizes performance impact to the application server environment.

  The Performance Advisor in TPV provides advice to help tune systems for optimal performance and gives recommendations about inefficient settings by using collected PMI data. The Performance Advisor in TPV provides more extensive advice than the Runtime Performance Advisor. For example, TPV provides advice about setting the dynamic cache size, setting the JVM heap size, and using the DB2 Performance Configuration Wizard.
- IBM Tivoli Monitoring for Web Infrastructure: Provides best-practice monitoring of the key elements of WebSphere Application Server. This is the inside-out view, which enables administrators to quickly address problems before they affect users.
- IBM Tivoli Monitoring for Transaction Performance: Provides a monitoring perspective unique from that of the users. This approach is the outside-in view, which verifies that end-to-end components provide a positive user experience. This tool monitors performance of actual and synthetic transactions, and verifies that the delivered content meets predefined guidelines.

Attention: WebSphere Application Server is a key underlying software of Content Manager, and it plays a critical role in the performance of a Content Manager system. In this section, we merely touched the very basics of the J2EE, WebSphere architecture, and architectural components. In 11.5, “Monitoring tools for WebSphere Application Server” on page 339, we introduce the performance tuning monitoring tools for WebSphere. Chapter 6, “Best practices for Content Manager system performance” on page 143, and especially Chapter 9, “Tuning WebSphere for Content Manager” on page 255, discusses WebSphere-specific tuning for a Content Manager system.

To better understand WebSphere and WebSphere performance tuning, we recommend visiting the WebSphere Application Server Information Center and the publication we mentioned earlier in this section.

2.3 Tivoli Storage Manager overview

IBM Tivoli Storage Manager (TSM), now a part of the IBM TotalStorage® Open Software Family, ensures availability of business applications by providing data protection and resource utilization that scales with business needs. TSM is an enterprise-class recovery solution, protecting business critical data from laptop to systems, regardless of where it resides.

TSM supports business continuity by helping to automate disaster recovery planning and recovery execution based on business priorities. TSM integrates the power of applications-aware technology in the recovery of leading database, content management, and workflow applications to ensure that the entire business process is protected.

In the following sections, we provide a *very general* overview of TSM capabilities, advantages, and architecture. Our materials are extracted from existing TSM publications. For more detailed information, see:

- *Tivoli Storage Management Concepts*, SG24-4877
- *IBM Tivoli Storage Manager: A Technical Introduction*, REDP0044
- *Tivoli Storage Manager for AIX - Administrator's Guide*, GC32-0768
- Tivoli Storage Manager home page:

  http://www.ibm.com/software/tivoli/products/storage-mgr/

2.3.1 TSM capabilities overview

TSM provides an interface to a range of storage media, including disks, tapes, and optical disks. It enables Content Manager systems to support different storage devices, drives, and libraries. TSM provides the following capabilities:

- Backup and restore
- Disaster preparation and recovery
- Archive and retrieve
- Hierarchical Storage Management
- Application protection, 24 hours a day, 365 days a year

Backup and restore

Complete data protection starts with data backups. Backups are a copy of the active online data stored on offline storage. If an online storage device fails, a data error occurs, or someone accidentally delete a file, the offline copy of that data can be copied back to online storage restored. TSM provides backup and restore functionality, using multiple techniques to reduce data transfer sizes to the minimum possible. These techniques reduce the total time required for both data backups and, more important, data restores.

Disaster preparation and recovery

Complete data protection also involves disaster preparation and recovery. Local copies of data protect against discrete failures or errors in equipment, storage, or people. But disasters tend to happen to entire facilities, not just a portion of the equipment inside those facilities.

Using Tivoli Storage Manager, you can prepare an additional copy of the active data for safekeeping at an off-site location to provide extra insurance against disasters. If a disaster strikes and destroys online storage and computers, the off-site copy of the active data can be restored to new computers to get business up and running quickly.

Archive and retrieve

Tivoli Storage Manager goes beyond data backups to include data archiving. Archiving inactive data is an effective way to reduce your online storage costs.

The archive process creates a copy of a file or a set of files representing an end point of a process for long-term storage. Files can remain on the local storage media or can be deleted. The customer determines the retention period (how long an archive copy is to be retained).

TSM also provides retrieval functionality to fetch the archived data for usage. The retrieval process locates the copies in the archival storage and places them back into a customer-designated system or workstation.

Hierarchical Storage Management

Tivoli Storage Manager includes an automated version of archive called *Hierarchical Storage Management* (HSM), which provides the automatic and transparent movement of operational data from the user system disk space to a central storage repository. If the user accesses this data, it is dynamically and transparently restored to the client storage.

As with archiving, HSM removes data from online storage and puts it on less expensive offline storage. Unlike archiving, HSM leaves a data-stub on online storage that shows the name of the removed file. With this stub, you can easily access the offline data, albeit much more slowly than if it were online. HSM capabilities are automated. It watches online data files to see how often they are used. If the data are not opened for an administrator-specified length of time, they will be removed to offline storage, leaving only the data -stub behind. For a business with large amounts of data that do not need to be online at all time, HSM is the best way to save on storage costs.

Application protection - 24x365

By directly controlling the time and method by which a data transfer to offline storage occurs, the application can continue operating with no interruption, 24 hours a day, 365 days a year.

2.3.2 TSM architecture

TSM is implemented as a client/server software application, consisting of a server software component, backup/archive client, storage agent, and other complementary Tivoli and vendor software products. Figure 2-11 shows the main components of Tivoli Storage Manager.

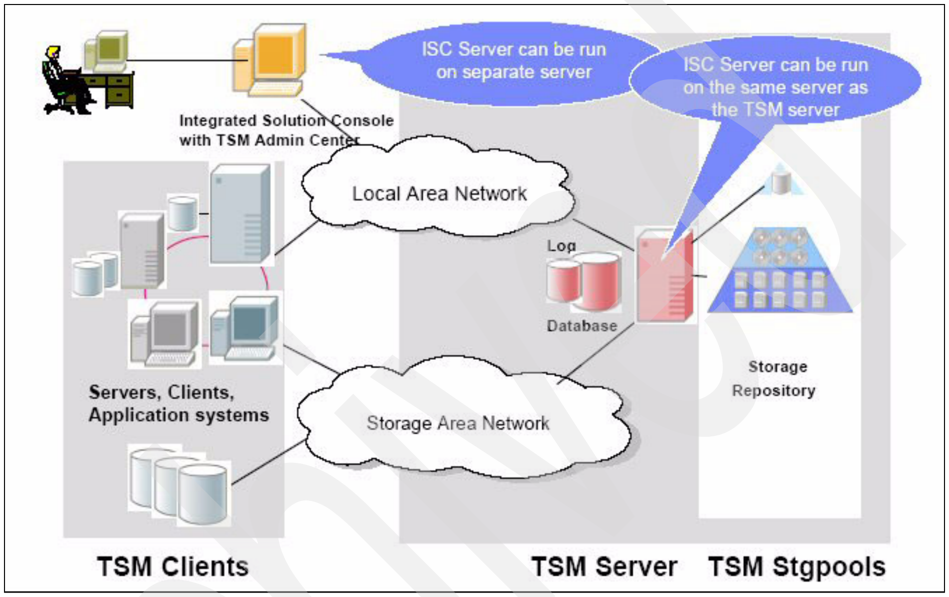

ISC Server can be run on separate server
ISC Server can be run on the same server as the TSM server
Integrated Solution Console with TSM Admin Center
Local Area Network
Log
Database
Storage Repository
Servers, Clients, Application systems
Storage Area Network
TSM Clients
TSM Server
TSM Stgpools

Figure 2-11 Tivoli Storage Manager architecture

The *TSM server* provides a secure environment, including automation, reporting, and monitoring functions, for the storage of client data. It also provides storage management policies and maintains all object inventory information.

The TSM client software and complementary products implement data management functions such as data backup and recovery, archival and hierarchical storage management, and disaster recovery.

The client software can run on different systems, including laptop computers, PCs, workstations, and server systems. The client and server software can also be installed on the same system for a local backup solution. The storage agent software in conjunction with the server software enables the implementation of LAN-free backup solutions exploiting the SAN infrastructure. It is also possible to define server hierarchies or multiple peer-to-peer servers in order to provide a multi-layer storage management solution or an electronic vaulting solution.

TSM server

One of the principal architectural components of the TSM server is its built-in relational database. The TSM database was designed especially for the task of managing data, and it implements zero-touch administration. All policy information, logging, authentication and security, media management, and object inventory is managed through this database. Most of the fields are externalized through the TSM high-level administration commands, SQL SELECT statements, or for reporting purposes, by using an ODBC driver. This database is fully protected with software mirroring, roll-forward capability, and with its own management and online backup and restore functions.

For storing the managed data, the TSM server uses a storage repository. The storage repository can be implemented using any combination of supported media including magnetic and optical disk, tape, and robotic storage devices, which are locally connected to the server system or are accessible through a SAN. To exploit SAN technology, the TSM server has features implemented to dynamically share SAN-connected automated tape library systems among multiple TSM servers.

The TSM server provides built-in device drivers for more than 300 different device types from every major manufacturer. It can utilize operating system device drivers and external library manager software.

Within the storage repository, the devices can operate stand-alone or can be linked to form one or more storage hierarchies. The storage hierarchy is not limited in the number of levels and can span over multiple servers using so-called virtual volumes.

Backup and archive client

Data is sent to the TSM server using the TSM backup/archive client and complementary Tivoli and non-Tivoli products. These products work with the TSM server base product to ensure that the data you need to store is managed as defined.The TSM backup and archive client, included with the server, provides the operational backup and archival function. The client implements the patented progressive backup methodology, adaptive sub-file backup technology, and unique record-retention methods.

The backup and archive clients are implemented as multi-session clients, which means that they can exploit the multi-threading capabilities of modern operating systems. This enables the running of backup and archive operations in parallel to maximize the throughput to the server system.

Depending on the client platform, the backup and archive client may provide a graphical, command line, or Web user interface. Many platforms provide all three interfaces. The command line interface is useful for experienced users and enables generation of backup or restore scripts for scheduled execution. The graphical interface is designed for ease of use for the end user for ad hoc backups and restores. The Web client is especially useful for those clients, such as NetWare, where no native GUI is available, or for performing remote backup and restore operations (for example, in a help desk environment).

Some clients (including some UNIX variants and Microsoft platforms) use a new plug-in architecture to implement an image backup feature for raw device backup. This allows you to back up and recover data stored in raw (that is, not a file system) volumes. It also provides an additional method to make point-in-time backups of entire file systems as single objects (image backup) and recover them in conjunction with data backed up by using the progressive backup methodology.

Storage agent

TSM storage agent supports LAN-free backup solutions using a SAN infrastructure. The storage agent dynamically shares SAN connected tape libraries and disks with the TSM server, and it has the ability to write and read client data directly to and from server-owned storage media.

The storage agent receives data objects via the TSM API and communicates with the server over the LAN using TCP/IP to exchange control information and metadata about the objects being backed up. The data movement itself utilizes the LAN-free path over the SAN to write directly to the storage media. Thus, the data movement is removed from both the LAN and the server processor for potentially greater scalability.

The storage agent is available for selected backup and archive clients as well as for backing up popular databases and applications.

Administration

For the central administration of one or more TSM server instances, TSM provides command line or Java-based administration interfaces, also called administration clients.

Using the unique enterprise administration feature, it is possible to configure, monitor and manage all server and client instances from one administrative interface, known as the enterprise console. It includes:

- Enterprise configuration
- Administrative command routing
- Central event-logging functions

The enterprise configuration allows TSM server configurations to be defined centrally by an administrator and then propagated to other servers.

Administrative command routing enables administrators to issue commands from one TSM server and route them to other target servers. The commands are run on the target servers, and the command output is returned and formatted on the server where the command was issued.

In an enterprise environment with multiple TSM servers, client and server events can be logged to a central management server through server-to-server communications, thereby enabling centralized event management and automation.

Externalized interfaces

TSM provides a data management API, which can be used to implement *application clients* to integrate popular business applications, such as databases or groupware applications. The API also adheres to an open standard (XBSA) and is published to enable customers and vendors to implement specialized or custom clients for particular data management needs or non-standard computing environments.

Tivoli Storage Manager is a key underlying software of Content Manager and it plays an important role in the performance of a Content Manager system. In this section, we merely touched on the very basics of TSM. In Chapter 10, “Tuning TSM for Content Manager” on page 275, we discuss TSM-specific tuning for a Content Manager system.

Part 2

Performance tuning overview and planning

In this part, we introduce performance tuning basics. We define performance, how it is measured, why we need to tune it, and when to plan and tune your system for performance. We cover performance methodology, and describe the performance improvement process along with a set of general guidelines that you should use for planning a new system, or maintaining and improving an existing system.

Attention: We highly recommend that you read this part before you continue to the rest of the book.

3

Performance tuning basics

In this chapter we introduce performance tuning basics. We define what performance is, how it is measured, some of the reasons for performance tuning, and when to plan and tune your system for performance. In addition, we describe the *performance methodology* that you should follow for the entire life cycle of a Content Manager system, and the *performance improvement process* you should use to maintain and improve an existing system. Finally, we include general performance tuning guidelines that you should keep in mind while doing performance tuning.

We recommend that you complete this chapter before you read the remaining chapters of this book.

3.1 Introduction

To perform means to begin a task and to carry the task through to completion. *Performance* refers to the way in which a system performs or functions given a particular workload. In the Content Manager world, a Content Manager system must run efficiently and effectively, and it must be able to perform the expected workload within a business-required time frame. Designing, configuring, and tuning a Content Manager system to achieve the desired results is essential for your business to succeed.

Performance measurement

Performance is typically measured in terms of response time, throughput, and availability.

Response time is the elapsed time between when a request is submitted and when the response from that request is returned. Examples include how long a database query takes and how long it takes to display a retrieved document.

Throughput is a measure of the amount of work over a period of time. In other words, it is the number of workload operations that can be accomplished per unit of time. Examples include database transactions per minute, kilobytes of a file transferred per second, and total number of legacy documents imported per hour.

Major performance bottlenecks

Four major performance bottlenecks are:

- CPU
- Memory
- Disks
- Network

Source of performance problems

System performance is affected by many factors, some of which include:

- The available hardware and software in the system
- The configuration of hardware and software in the system
- The number of users, the types of users, and the number of concurrent users in the system
- The number of applications running in the system, the types of running applications, and the application workload

Performance problems might be caused by poor application and system design, inadequate system resources such as CPU, memory, disks, or non-optimal tuning of system resources.

Performance tuning goals

Some of the goals we want to accomplish through performance tuning include:

- Remove a bottleneck.
- Increase existing throughput: Processing a larger or more demanding workload without additional processing cost such as adding new hardware or upgrade existing software.
- Increase response time: Obtaining faster system response times without additional processing cost.
- Increase system availability: Extending the availability of a system, thereby increasing productivity of the system.
- Reduce processing cost: Reducing the number of users or the length of their working hours needed to process existing workload.

Performance tuning is a complex and iterative process. It involves establishing quantitative objectives, constant system monitoring, and selective and careful tuning to ensure that the objectives are met over time.

Publication references

Many publications have been written about performance tuning for Content Manager, DB2, WebSphere, and AIX. These guides are well written and contain very useful information. Most of the materials presented in this chapter are extractions from and summaries of these publications. The publications are:

- *Database Performance Tuning on AIX*, SG24-5511
- *IBM DB2 Universal Database Version 8.2 Administration Guide: Performance*, SC09-4821
- *DB2 UDB/WebSphere Performance Tuning Guide*, SG24-6417
- *IBM Content Manager v8.3 Performance Tuning Guide*
  `http://www.ibm.com/support/docview.wss?uid=swg27006452`
- CMv8.3 Performance Monitoring and Monitoring Guide
  `http://www.ibm.com/support/docview.wss?uid=swg27006451`
- CMv8.3 Performance Troubleshooting Guide
  `http://www.ibm.com/support/docview.wss?uid=swg27006450`

We include what we think is important for a quick overall view of the performance tuning basics for the Content Manager system. If time allows, we recommend that you read these publications because we found them extremely helpful.

3.2 When to plan and tune your system

Everyone wants value for their money and tangible benefit for the cost of performance tuning. When should you plan and tune your system for performance?

You need to plan and tune your system in any of the following situations:

- Planning and designing a new system

  Keep the system performance goal in mind during the initial planning and designing phase of a system. Many performance problems arise from poor planning and poor design. It is hard to tune a system to correct design problems.

- Installing a new system

  A newly built system requires initial tuning for maximum performance before going into production. In most of the cases, however, this system might already be in production because of the difficulty of generating user workload artificially and not being able to predict real user workload and work patterns.

  Ideally, the number of users and workload is still growing and has not reached maximum expected capacity. This allows tuning before system response times become unacceptable.

- Regular maintenance

  Regular periodic monitoring and tuning should be standard practice. It may be a good idea to perform a system performance review on a quarterly, bi-annual, or yearly interval. Be proactive in identifying problem machines, applications, and systems. This will help avoid potential crises that halt business activities.

- User complaint or a crisis just occurred

  Users are complaining or a crisis in performance or response time just occurred. Based on the severity of the performance problem, you might need to identify the bottleneck or issue immediately, recommend and run a solution that might include tuning, then work out procedures for avoiding these types of issues in the future.

- Upgrading an existing system or changes in workload

  New resources might be added to the existing system or to replace some existing resources; additional workload might be demanded of the existing system. For example, you need to add a new Resource Manager to accommodate new and additional digital content. The current system must go through tuning activities for the new system configuration.

If you are among the fortunate few, you are currently in the planning and designing phase of your future system. We strongly recommend you read through 3.3, "Performance methodology" on page 81 to get you started the right way. It is extremely important for you to understand how to design a system that maximizes potential system performance.

If you already have an existing Content Manager system that is up and running in a production environment, your main task is to maintain or improve performance, and resolve any crisis or problems along the way. In this situation, we advise you to read the performance methodology, and understand the principles behind it. Most important, we recommend that you read through 3.4, "Performance improvement process" on page 83, which provides a structured way to maintain and improve your existing system performance.

3.3 Performance methodology

The Content Manager performance team has recommended a list of "best practices in the *IBM DB2 Content Manager V8.3 Enterprise Edition Performance Tuning Guide* at:

http://www.ibm.com/support/docview.wss?uid=swg27006452

This guide is useful during the entire life cycle of a Content Manager system. The best practices tuning guide is incorporated into Chapter 6, "Best practices for Content Manager system performance" on page 143.

Note: If you are planning and designing a new Content Manager system, read it through to understand and follow the methodology presented. If you are working with an existing system, we strongly recommend reading it anyway.

In this section, we extract the performance methodology from the Performance Tuning Guide. The primary goal is to avoid surprises, start on the right path from day one, and routinely maintain your system to ensure optimal performance for your system.

Performance and scalability cannot be achieved by performance tuning alone, but tuning is one key element of a Content Management application's lifecycle. To build and maintain high performance and scalability requires a full lifecycle performance methodology, from pre-sales through installation to production, with the primary goal of avoiding surprises.

The Content Manager performance methodology has a five-stage life cycle. By setting objectives early, performance tuning before going into production, and establishing processes that monitor key performance metrics, the methodology helps ensure achievement of initial scalability objectives and provide information for capacity planning as workloads change and grow.

The five stages of the performance methodology are:

1. Planning for performance and scalability.

   Document the projected system topology, the projected workload, and define measurable performance and scalability objectives. Perform initial capacity planning and sizing estimates.

2. Making solution design choices and performance trade-offs.

   Understand topology choices (types of client, eClient, Document Manager, Client for Windows, custom client), feature choices (such as replication, high availability, text search, and versioning) and their performance implications.

3. Performance tuning.

   Plan for an initial tuning period to maximize confidence and reduce risk, using the tools and techniques from the monitoring and troubleshooting phases to meet the objectives set in the planning phase. Focus on effective tuning of Content Manager server calls, memory, disk I/O, and network bandwidth. The sections that follow provide key parameters in various Content Manager components that give you the greatest performance impact.

4. Monitoring and maintaining performance.

   Maintain a performance profile of the workload metrics and resource utilizations on the Content Manager servers. Monitor over time to observe trends before they become a problem.

5. Performance troubleshooting.

   When performance issues arise, use a disciplined and organized approach to solve the problem using the performance profile, data, and performance tuning guidelines.

Attention: If you are dealing with a new system not yet in production, plan for an initial tuning period to maximize performance and reduce risk before going into production. If possible, use automated test tools to drive a multi-user test load based on your projected workload. During this tuning period, focus on one area at a time and change only a small number of tuning parameters. Run the test workload to evaluate the effect of each set of changes before making additional tuning changes.

3.4 Performance improvement process

If you are working with a running Content Manager system, your primary goal is to improve system performance through performance tuning. This requires planning, strategy, methodology, resources, time, and money. Be sure to read through the performance methodology in the previous section before you continue.

Performance improvement process is an iterative and long-term approach to monitoring and tuning your system for performance. Depending on the results of monitoring, you and your performance team should adjust the configuration of the system resources and applications to maximize system performance, reduce down time, and avoid potential problems or a crisis.

We recommend the following performance improvement process:

1. Gather business requirements on performance and user input.
2. Understand system topology and workload.
3. Define performance objectives.
4. Establish performance indicators for the major constraints in the system.
5. Develop a performance monitoring plan.
6. Run the plan.
7. Evaluate system performance, analyze the measured outcome, and determine whether objectives have been met.

   If objectives are met, consider simplifying the performance monitoring plan by reducing the number of measurements, because performance tuning can be resource intensive. Iterate the entire process from step 6 when necessary.
8. If objectives are not met, identify the area and major causes of the problems.
9. Decide where and how changes should be make in the hardware or software configuration that can address the problems and make a major impact. Understand, consider, and make trade-off decisions. When appropriate, re-assess performance objectives; modify if necessary so the objectives are realistic, achievable, yet still acceptable.
10. Adjust the system configuration according to the solution from the previous step. Make one change at a time. Work on one level at a time. If the system has already reached its optimal performance or no other configuration can make any significant impact, consider upgrading your hardware, software, or application re-design.

    Remember to follow the performance tuning methodology covered in later sections.
11. Iterate the entire process from step 6.

Periodically, and especially after significant changes to your system or workload:

- Re-examine your objectives and indicators.
- Refine your monitoring plan and tuning methodology.

3.5 General performance tuning guideline

When you are working on improving system performance, we recommend that you follow this set of general guidelines for performance tuning. This is a collection of wisdom from the performance tuning experts:

- Establish quantitative, measurable, and realistic objectives.
- Understand and consider the entire system.
- Change one parameter at a time.
- Measure and reconfigure by levels.
- Consider design and redesign.
- Remember the law of diminishing returns.
- Recognize performance tuning limitationss.
- Understand the configuration choices and trade-offs.
- Do not tune just for the sake of tuning.
- Check for hardware as well as software problems.
- Put fall-back procedures in place before you start tuning.

Establish quantitative, measurable, and realistic objectives

Performance is measured in response time and throughput. Performance tuning objectives must be quantitative, measurable, and realistic. Units of measurement include one or more of these: response time for a given workload, transactions per second, I/O operations, CPU use. A system performance can also be measured in terms of throughput, or how much specific workload such as loading and archiving can be completed over a specified time.

Understand and consider the entire system

Never tune one parameter or one portion of a system in isolation. Before you make any adjustments, understand how the entire system works and consider how your changes could affect the entire system.

Change one parameter at a time

Change one performance tuning parameter at a time to help you assess whether the changes have been beneficial. If you change multiple parameters at once, you might have difficulty in evaluating what changes contribute to what results. In addition, it might be difficult for you to assess the trade-off you have made. Every time you adjust a parameter to improve one area, there is a great chance that it affects at least one other area that you might not have considered. By changing only one at a time, this gives you a benchmark to evaluate whether the result is what you want to accomplish.

Measure and reconfigure by levels

Tune one level of your system at a time, for the same reasons that you would change one parameter at a time. Use the following list of levels within a Content Manager system as a guide:

- Hardware
- Hardware configuration
- Operating system
- Library Server configuration
- Resource Manager configuration
- Database manager
- Database configuration
- WebSphere Application Server
- Application configuration

Consider design and redesign

Design plays an important role in system performance. If the design of a system is poor, it might be impossible to improve performance and achieve business requirements. When considering improving a system's performance, the impact of the design on a system should be considered. Sometimes, you need to go back to the drawing board, review the entire system design, and work with architects and developers to redesign the application or system if necessary to achieve desired performance. There might uncertainty whether redesign is considered part of performance tuning; nevertheless, design should be considered and if necessary redesign should be followed.

Remember the law of diminishing returns

Greatest performance benefits usually come from a few important changes in the initial effort. Minor changes in the later stage usually produce smaller benefits and require more effort. Always remember the law of diminishing returns: Stop when the performance effort no longer justifies the outcome and when the outcome is already acceptable.

Recognize performance tuning limitations

There are times when tuning improves performance dramatically if a system is encountering a performance bottleneck, which can be corrected. Other times, tuning will not make any additional difference because the system is already performing at its optimal limit. There is a point beyond which tuning can no longer help. You must recognize this limit. When this situation arises, the best way to improve performance might be to add additional disks, CPU, or memory or replace the existing ones with faster and higher performance hardware.

Understand the configuration choices and trade-off

Nearly all tuning involves trade-offs among system resources and their performances. Understanding the configuration choices and trade-offs and deciding where you can afford to make a trade-off and which resources can bear additional load is important.

Do not tune just for the sake of tuning

Make sure you have clear objectives and concentrate on the source of the performance problems. Make changes that address and relieve problems. Do not tune just for the sake of tuning. If you make changes to the area that is not the primary cause of the performance problems, it has little or no effect on the resolution of the problem. Furthermore, it may complicate any subsequent performance tuning. Again, make sure you have clear objectives.

Check for hardware and software problems

Some performance problems may be corrected by applying service to your hardware, your software, or both. Do not spend excessive time monitoring and tuning your system when simply applying service might make it unnecessary.

Put fall-back procedures in place before you start tuning

Some tuning can cause unexpected performance results. If this leads to poorer performance, it should be reversed and alternative tuning should be tried. If the former setup is saved in such a manner that it can be simply recalled, backing out of the incorrect information becomes much simpler.

4

Planning for performance

To deliver good performance, a Content Manager system requires sufficient system resources to run the necessary workload. One of the most important tasks from the first day is to plan for performance. To determine how much system resources will be required, we need to estimate the workload the business application is likely to create, then translate that to workload for the Content Manager server (or servers). When we have an estimate of the load, we can compare it to the measured workloads tested on particular hardware configurations.

In this chapter we run through the principles of sizing a system and look at the tools and techniques available to estimate the workload generated by a Content Management solution.

4.1 Introduction

Eventually, after all of the tuning and system design has been optimized, the system performance comes down to the ability of the hardware resources to process commands, access data, and communicate across a network. Unfortunately, the hardware resources are the first components acquired when building a system. Although Content Manager provides the flexibility to expand and take advantage of more hardware that is added into the system, it is usually more cost effective to get the correct hardware resources in the first place.

For new systems, we need to be able to determine the required resources even before the detailed system design has been done. If you already have an existing system, you might want to perform the sizing exercise to determine whether the existing hardware configuration can support new workload or projected growth.

How can we determine what system resources are required? To do this, we need to accurately estimate the current workload and its rate of growth, then translate that into hardware and software configurations by comparing this workload to measured systems and workload. This is a critical exercise when implementing new systems, but should also be performed periodically for existing systems as part of a capacity management review. To create a capacity plan, we must:

- Understand the current process and data.
- Translate the current process to the new data model.
- Analyze the impact of the new workload.
- Estimate the storage requirements.
- Perform benchmark measurements.
- Forecast future workload and system performance.

An organization considering a new Content Manager application should examine the current methods of processing their documents and data. They need to have some business goals defined for the system. The task then is to use attributes, resource objects, documents, folders, links, authorization schemes, and work processes to develop a model that best represents the organization's data flow. They will need to understand their business process and user needs well enough to translate the existing process into a Content Manager data model and document routing and work flow process. See 5.5.3, "Data model" on page 115" for details. Having mapped the existing process you might be able to enhance it with new capabilities provided by an electronic Content Manager system.

The data model you have developed and the data volumes measured from the current processes enable you to calculate projected activities in the Content Manager system. To automate some of the calculations, sizing support is available to IBM representatives through TechLine. Your IBM representative can access it from this Web site:

http://w3.ibm.com/support/americas/techline/sizewise.html

Sizing should not be done in a vacuum. You need to analyze the environment before and after any sizing exercise to ensure that the results match the real-world environment for which you are planning.

Goal of the sizing exercise

With a data model that properly represents business data and processes, and an understanding of the volume of documents in these processes, you can effectively size the system. The ultimate goal of the sizing exercise is to estimate:

- Storage requirements for the objects on disk, optical, or other media
- Disk requirements for the databases, prerequisite software, and other requirements on the servers
- The processing power of the servers, along with memory recommendations rewired to support the workload and deliver the response time required

4.2 Understanding current processes and data

To start the capacity planning exercise, we need to understand how the existing system is dealing with this content. This includes:

- Document types and counts

  Documents should be grouped and categorized by type (for example, invoices or claims). These groupings will be based on the way the documents are to be treated for security, and indexing data and storage requirements. The groups will probably translate to Content Manager item types.

- Document arrival rate

  When, where, and how many documents are entering the system? Although annual document volumes are often used as a benchmark, it is dangerous to apply these statistics without knowing document arrival peaks and valleys. It is critical to identify the peak arrival rate and size the system accordingly. It is also useful to understand the locations and formats of the documents that can be captured. Are all application forms sent to a central mail room and processed there, or are they sent to individuals in branch offices?

- Document usage

  When, where, and how is each kind of document used? For example, one type of form might go through a business process, following a specific workflow. The workflow process comprises seven different activities or steps by different people. At each workflow step, this document is viewed by the person performing the activity. This example means that each form is retrieved at least seven times. That does not seem like much, but if there are 10,000 of these forms arriving each day, then there are 70,000 retrievals happening each day.

4.3 Translating current processes to the new data model

When sizing the system, first you must determine what item types will make up the system. This can be the most difficult task in sizing the system. You need to understand how the customer currently uses these documents, and how the customer wants to conduct business with the Content Manager system.

Develop the data model that will define the documents as item types and describe what attributes and objects the documents have. If folders or links are to be used, determine what they represent to the business and how they fit into the process.

When choosing attributes for your item types, your primary concerns are identifying data that will be searched, and fields that will distinguish one document from another when a list of search results are displayed. When a user looks for an item in the system, what information about that item does the user already have? In a customer service application, when a customer initiates a query, does the customer already have a customer number, a reference number, or just the customer name? Is there a need to search by the date sent, received, processed, published, or a combination of the values?

Remember that the Library Server tables (specifically the item type root component or child component tables) will have a column for every key field defined, and there will be one row in the root component table for every item in the item type, and possibly many rows in the child component table or tables. Avoid having an attribute for every thing you can think of attached to each item. Although you might want everything about the item to be used for searching, it could lead to masses of unnecessary data on the Library Server and, as a consequence, increased search times and more data to back up and maintain.

4.4 Analyzing the impact of a new workload

Every action that is performed by a client talking to a Library Server uses some small amount of the resources of the Library Server. The key to estimating the total load on the server is to estimate the total number of actions the client community must perform to support the customer's business requirements. For instance, if you create 10,000 documents per day, that is 10,000 store requests on the Library Server. If the user retrieves each document five times, that is 50,000 retrieve requests. You can determine the number of searches, folder opens, and document route actions per day by asking appropriate questions of the customer in their terms and in the context of their existing process.

The following questions deal with numbers and sizes of documents, folders, and item types, and the amount of stress placed on the servers and workstations.

Questions for new workload analysis

You could have documents that vary greatly in size and other characteristics. You might want to answer each of the following questions by item type:

- How many documents and folders of each type will be stored in the system?

  How many documents per day will users create in the system? How many folders per day will be created? The use of folders will be dependent on the data model developed for this application. Documents can be grouped in folders, and folders placed inside other folders. Also, note that one document can exist in multiple folders.

  Do users have old documents that will be scanned into the system? Is there a back-capture exercise to account for? Do we need to migrate an existing solution? Are we bulk loading old documents and images from a file system?

  Note: This is the most important question to answer, because it affects all of the estimates.

- What is the size of the documents?

  How many bytes of data are required to hold the average document? If we are scanning data, this is the size of the document image file scanned at the required resolution. If this is an electronic document, then what is the average file size? Sample the existing data to determine this. If there are different types of documents, establish an average size and count for each type of document. Understand the size of each part that makes up a document if there is more than one. For example, there might be a high-resolution image part of 500 KB, a low-resolution image part of 40 KB, and another part that has a plain ASCII text description of 5 KB. The size of the documents and parts has the greatest effect on the amount of final media storage needed, and the network requirements needed to transmit the objects when they are stored or retrieved.

- How many days will documents reside on disk?

  Along with the number of objects and the size of the objects, this question relates to the amount of disk required for storage before migration to other media (TSM). Typically, Content Manager implementations assume that new documents will reside on disk for a set number of days while they are actively worked on and frequently retrieved. Afterwards, they are migrated off to slower media, such as optical, for archiving purposes, where they are less likely to be retrieved but are still needed for auditing purposes.

- Will any documents be replicated to other Resource Managers? If so, how many copies will be held?

  If the documents are duplicated, they will take up twice as much storage space. These have to be counted as well.

- Are you using the out-of-the-box Windows client, the eClient, or are you writing your own custom application?

  The Windows client and the eClient make assumptions about the document model. The eClient requires a mid-tier server. Will you convert document formats on the mid-tier server?

- How many users will access the system?

  This includes both LAN and Web-based users. This affects the per-user interactions with the system.

- Is there a target platform for the servers?

  Many customers have hardware platform preferences for the servers, and this can affect the size and the number of required machines.

- What information will be used to index the documents and folders (the number and the size of attributes)?

  This translates to the data that is held in the Library Server database; hence its size. Each item type creates at least one Library Server database table (ICMUT01nnnnnn) that will have a column for each of these attributes, and a row for every item. In the case of child components, there might be multiple rows for each item. These tables are searched when retrieving items by attribute values. They also hold system defined and maintained attributes.

  In Content Manager there is indexing. The term “indexing data” can mean the parametric data that describes the image, document, or file, and is stored as rows and columns in the database. The term can also apply to database indexes created to help the database operate effectively. The database indexes enable you to search the data in the index much faster than searching across all data in the table. These database indexes take up additional space on the database and must be included in our calculations. Typically, database indexes are set up on a subset of the parametric data (indexing data) that describes these items. You should select item attributes that are used for the majority of user-initiated searches. Refer to “Indexes” on page 110 for more information about database indexes.

- How many times will documents and folders be retrieved and updated?

  Understanding the business process in which these documents exist helps us understand how often retrieve requests are expected per day.

  Workflow-driven processing: As a document passes through a fixed process it may be retrieved, on average, a certain number of times for it to be fully processed. For instance, if 10,000 claim forms are received each day and each is retrieved five times during its lifetime, the daily retrieves done by the system will be 50,000. This is to keep a steady state in the system; there must be as many items completing the process each day as entering, and on average, the same number passing through every step of the process. For example, 10,000 items are received, 10,000 are retrieved at step 1, 10,000 are retrieved at step 2, another 10,000 at step 2, and 4, and 5, giving a total of 50,000 retrievals.

  Note: If one of these retrieval steps can be eliminated from the process, then there can be a significant reduction in the Resource Manager and network load.

- If you are using the Content Manager document routing, you also need to determine:
  - Through how many steps or work nodes will the document be routed?

    Each route places a certain amount of stress on the Library Server.
  - How many days will items reside in the document routing process?

    This affects the size of the Library Server database. The information concerning items in work lists and processes is kept in the database tables. There is at least one row in the document routing tables for every item that is currently in a work process.
- How many ad hoc searches for documents and folders will be done per day?

  Will the customer service officers be searching the system to handle enquiries? If so, how many enquiries do they handle per day, and of those how many would involve searching the Content Manager system? Are there business analysts who might crawl the system to find out more about the organization's customers? Will external customers or business partners be accessing data through a Web interface? Are there Web site statistics that identify the number of requests that are likely to retrieve pages or items from the Content Manager system?

  Searches for items are done on the user tables for the item type. Each search places a stress on the Library Server itself, and may lead to the user actually retrieving and viewing a document.

- How many hours per day will the system be available for input, processing, and object migration?

  Assuming a fixed workload, if the workday is long (for example, 10, 16, or 24 hours), a smaller processor can be used on the Library Server than for a system with standard 8-hour workday to process the workload. You might want to size for a single hour of the peak workload, if one can be determined.

  One of the bottlenecks in the system might be migrating objects to optical or slower media off-shift. Increasing the number of hours allowed for off-shift could allow for a reduction in the number of Resource Manager servers otherwise recommended.

Sizing a Content Manager system

When these questions have been answered, we can begin to calculate storage and load generated by this Content Manager application. Almost every client action interacts with the server and causes some load generated on that server. From logon through search, retrieve, update, and delete, they all generate work for the server.

Stored Procedure Calls (SPC) and HTTP requests can be used as the workload metrics. SPC represents a generic unit of work for the Library Server and the HTTP request represents a unit of work for the Resource Manager. SPC replaces the Library Server Functional Request (LSFR) as the unit of measure for the Library Server. Readers with experience of earlier versions of Content Manager may recall LSFRs.

To determine your workload, you can either trace a sample workload or have your IBM representative use the TechLine Sizing Support. To measure a workload, use the Library Server performance-level trace to monitor a sample workload on a test system client, count the number of SPCs generated, and multiply that by the total number of requests determined by the questions above. This will give an estimate of the number of SPCs likely to be generated by the system. We can then compare this to benchmark systems.

The TechLine Sizing Support uses a Content Manager Sizer to generate a report that calculates the number of SPCs that are likely to be generated by the Windows client or eClient to perform the activity defined by the answers to the questions above. It also compares this calculated workload to measurements made on specific hardware configuration. TechLine will use the Sizer to help suggest the hardware that is required for the customer load.

Equipped with this information, we can identify a current model server of similar relative performance. We then need to find the best value hardware combination for a server of that performance range. Always be sure to allow additional capacity for growth. Similarly, the performance metrics were taken under laboratory conditions and a well-optimized system; these conditions might not be the same for your application so make allowance for that.

The report also provides relative performance measures for the *Resource Manager* capacity. The Resource Manager is an application that runs and uses processor and memory capacity as well as I/O capacity.

4.5 Estimating storage requirements

Apart from processing capacity, the system must also allow sufficient data storage capacity. Part of the planning must be to calculate the projected storage requirements now and into the future. Storage is required for the Library Server as well as the Resource Managers. The Library Server holds the data in the Library Server database and its size must be estimated and accounted for. The Resource Manager, as the primary data storage component, must have sufficient space to hold the images, documents, and files that make up the system, in combination with the TSM system, if used. It also has a database and we need to include this in our storage calculations.

Library Server database

The size of the Library Server database grows as the number of documents and folders grows. To estimate its size, you need to know the number of documents or folders you expect to have in the system, the number that will be in workflow at a given time, the number of folders you expect each item to be contained in, the number of resource objects you expect to have, the total length of the attributes of each item type, and the number of items that will be in each of the item types. To estimate the size of the database accurately, we have to understand all of the database tables and how they are used. The database tables are documented in the API online reference, so we can calculate the table sizes exactly. This is quite tedious. Alternatively, we can measure sample systems and extrapolate, approximate by sizing only the largest tables, usually the item type tables, and then allowing additional space for the remaining control tables.

Each folder or item will have:

- A row in the item type table and, where applicable, items in the child component tables
- A row in the links table for every link, or it “contains” the relationship it has
- A row in the document routing tables for every process in which it participates

Each document will have:

- A row in the item type table and, where applicable, items in the child component tables

- A row in the parts table for that item type
- A row in the media object class table for that part type
- A row in the document routing tables for every process in which it participates

Resource items will have:

- A row in the item type table and, where applicable, items in the child component tables
- A row in the media object class table for that part type
- A row in the document routing tables for every process in which it participates

In each of these rows, there is a certain amount of system data. See Table 4-1 for root component system data, and Table 4-2 on page 97 for child component system data. There is also user-defined attribute data. It is possible to calculate the row lengths for each of the major tables; multiply them by the projected numbers of items; add in factors for the database row, page, and table overheads, and the other system tables; then add in extra space for the user and system indexes. For more about calculating database table and index sizes, see *IBM DB2 Universal Database Version 8.2 Administration Guide: Planning, SC09-4822*.

Table 4-1 System entries for root component tables

Column name	Type	Length	Scale	Nulls
COMPCLUSTERID	INTEGER	4	0	No
COMPONENTID	CHARACTER	18	0	No
ITEMID	CHARACTER	26	0	No
VERSIONID	SMALLINT	2	0	No
ACLCODE	INTEGER	4	0	No
SEMANTICTYPE	INTEGER	4	0	No
EXPIRATIONDATE	DATE	4	0	Yes
COMPKEY	VARCHAR	23	0	No
CREATETS	TIMESTAMP	10	0	No
CREATEUSERID	CHARACTER	32	0	No
LASTCHANGEDTS	TIMESTAMP	10	0	No
LASTCHANGEDUSERID	CHARACTER	32	0	No

Table 4-2   System entries for child component tables

Column name	Type	Length	Scale	Nulls
COMPCLUSTERID	INTEGER	4	0	No
COMPONENTID	CHARACTER	18	0	No
ITEMID	CHARACTER	26	0	No
VERSIONID	SMALLINT	2	0	No
PARENTCOMPID	CHARACTER	18	0	No
COMPKEY	VARCHAR	23	0	No
PAERENTCOMPKEY	VARCHAR	23	0	No

Alternatively, the sizing support output provides an estimated size of the Library Server database after conversion of existing items, after a year of production at the projected volumes, and at the end of the estimate period (as specified by the user). These estimates use the numbers of documents, the attribute value lengths, and the index sizes to perform the calculations suggested above.

From our experience, we used the sizing support to calculate the database size based on the numbers of documents in the system. We compared the estimated database size against the actual size of the database as measured on the file system. The estimates provided were very generous and increasingly so as the number of items increased; however, we still recommend that you use the Techline Sizing Support through your IBM representative as it is far better in these situations to overestimate than to underestimate. Any excess capacity realized can allow additional room for growth or change in the future.

Text indexes for Library Server

If you plan to implement text indexes for documents in your system, you should allocate storage space on the Library Server for those indexes. The disk space you need for an index depends on the size and the type of data to be indexed. As a guideline for an index, reserve disk space for about 0.7 times the size of the documents being indexed for single-byte documents. For double-byte documents, reserve the same size as the documents being indexed.

The amount of space needed for the temporary files in the work directory is 1.0 to 4.0 times the amount of space needed for the final index file in the index directory. If you have several large indexes, you should store them on separate disk devices, especially if you have concurrent access to the indexes during index update or search.

Resource Manager database

The Resource Manager database holds storage management information for the stored content, so it must be accounted for in storage estimation. The Resource Manager database is significantly smaller than the Library Server database (by a factor of about 10 to 1 in our test systems). There will be a Resource Manager database for each and every Resource Manager in the system. The Resource Manager database tables are fully documented, so you could calculate the exact size of each row held. In general, however, they hold a row in the database for each image, document, or file stored on that Resource Manager and additional rows for each replica stored there.

You can estimate it by adding up the size of the columns of the tables and multiplying by the number of resource objects stored, or you can use the results from the TechLine Sizing Support. The sizer output provides a database size estimate for conversion items, the size after one year of processing, and the size after a period you specify. The Resource Manager database size estimate is also quite generous compared to the size actually measured in our test systems.

File storage for Resource Manager

Usually, the largest storage usage in the system is the Resource Manager file system or LBOSDATA directory. To estimate the required amount of storage, simply multiply the average file size by the number of files expected to be stored here. The amount stored here will continue to grow while new data is added and old data is maintained. Old data may be removed from the system by deliberate deletions after the data is no longer required, or because it is migrated to another storage system, such as TSM. If files need only be stored on disk for 120 days, then the amount of storage required for the LBOSDATA directory is 120 times the arrival rate per day, multiplied by the average size of the documents.

The Techline Sizing Support provides estimates for this.

TSM database and storage requirements

If TSM is to be used in the system, we need to account for its requirements as well. If TSM is the long-term repository, then we need sufficient storage capacity to hold the data. TSM provides the tools to efficiently maintain files on other storage media, specifically disk, tape and optical disk. The storage capacity required for the TSM system can be calculated the same as Resource Manager: the number of items held for a period we specify, multiplied by their average size.

The TechLine Sizing Support also provides estimates for this.

The TSM database, though even smaller per item than that of the Resource Manager, can over time grow to be a considerable size and must be accounted for. TSM provides tools to estimate the database size, as does the Techline Sizing Support spreadsheet.

5

Designing and configuring for performance

Good performance for a Content Manager system requires good design. In this chapter we look at many of the choices that have to be made during design and configuration phases and how they can affect performance.

Content Manager systems are flexible and can be reconfigured based on your system requirements. This discussion might also be of interest to anyone who is maintaining an existing Content Manager system.

5.1 Introduction

Design and configuration includes the overall architecture of a system, its layout, and what features and functions are used to meet the business requirements. It also means understanding these features and functions sufficiently well to predict their performance behavior. This sounds like a major challenge, but if we can understand the principles, then we can begin to make general predictions. If we need more detail information, then we can model and test.

Content Manager is designed and tuned for performance. Out-of-the-box Content Manager system uses sophisticated techniques to optimize its own performance. On installation, Content Manager adjusts a range of DB2 parameters from default settings to production-ready values ready for at least medium-scale use. Content Manager also uses unique technology such as writing its own programs during run time, specifically for you and your implementation. It uses pre-compiled SQL wherever possible to reduce the load on the Library Server database and reduce response time at the server. If we understand some of these techniques, we can tune for them and optimize performance.

5.2 General design considerations

For a Content Manager system, consider three main areas for performance tuning: the Library Server, the Resource Manager, and the client application. Two additional important areas to keep in mind are the hardware resources and the network that connects them. Always keep in mind during design:

- Library Server as a database application
- Resource Manager as a Web application
- Client application design impact on performance

Library Server as a database application

The Library Server is a database application, and the tuning is database specific. You should focus on database design when designing a Content Manager system. In addition, you should tune the database for maximum Content Manager system performance.

Resource Manager as a Web application

The Resource Manager is a Web application that delivers files on request. It is primarily a Java application running as a WebSphere Web application that holds control data in a database, and objects in a file system. We need to tune WebSphere Application Server, the database, and the file system.

Client application design impact on performance

The design of the client application can have a significant impact on performance. When building and testing the client application, be sensitive to the design and capabilities of the overall system. This includes the network that carries the data between distributed system components.

Content Manager supplies two general-purpose clients (Windows client and eClient) and a list of APIs and application components that can be assembled to customize client applications. There can be many variations of client applications, which we will discuss in general. Usually, there is not much to tune on a client application after it is built. Most client performance issues are related to the design and the APIs that are used to implement the application. As with all application tuning, the biggest gains in performance come from well-designed client applications.

Design the client applications to take best advantage of the servers' capabilities and to minimize their limitations. For example, it is possible to store image files as binary large objects in the Library Server database and store nothing on the Resource Manager. This is possible, but it does not take advantage of the Resource Manager's capabilities, but increases the load on the Library Server. There might be circumstances where this is the most appropriate design, although we cannot think of any.

Most of our discussion and scenario performance tuning covers a two-tier application. The same considerations should apply for *n*-tier applications, at least from the Content Manager Connector, and down to the server. Other tiers can have an impact and should be tuned accordingly.

5.3 Configuration choices and trade-off

An important aspect of planning a new system is to understand the configuration choices and trade-off. In this section, we include some of the important configuration and application design choices, and their implication to system performance. This material is an extraction from *IBM Content Manager V8.2 Performance Tuning Guide*, which can be found at:

`http://www.ibm.com/software/data/cm/cmgr/mp/support.html`

Search for "performance tuning guide" under white papers.

Content Manager configuration choices and the performance trade-off:

- Web clients or desktop clients?
  - Web clients are typically easier to deploy and maintain.

- Desktop clients typically have faster response time and more functionality.

- For Web clients: direct retrieve or mid-tier conversion?
  - Direct retrieve is faster and more scalable.
  - Direct retrieve might require browser plug-ins or viewer applets.
- For Web clients: direct connect or federated access?

  Federated access is slower than direct connection to Library Server, but more flexible supporting search across heterogeneous back-end servers.
- For Web clients: connection pooling?
  - Connection pooling makes more efficient use of Library Server resources.
  - Scales to larger number of Web clients.
  - Allows more control of mid-tier impact on Library Server resources.
- Out-of-the-box client program or custom client program?
  - A custom client program can be tuned to your exact business requirements.
  - Out-of-the-box client already includes the latest general-purpose tuning.
- For custom client: beans or Java/C++ API?
  - Beans implement only the document models.
  - Beans support rapid application development with a federated “reach.”
  - Java/C++ APIs have the best performance.
- For Java/C++ API custom clients: document data model or custom data model?
  - A custom data model can be tuned to your exact business requirements.
  - Data model already includes the latest general-purpose tuning.
- Document routing or advanced workflow?
  - Document routing has better performance and higher scalability.
  - Advanced workflow has more functions unavailable with document routing.
- Versioning?
  - Versioning increases the Library Server database size.
  - Accessing current version is faster than accessing previous versions.
- Text search?
  - Text indexes increase the Library Server database size.
  - Indexing text content reduces scalability as content must be retrieved from the Resource Manager.
- Unicode?
  - There is a substantial performance penalty for Unicode-enabling the Library Server.
  - Avoid using this feature unless you need the functionality.

- Attribute indexes?
  - Appropriate indexes improve performance of searches and reduce the Library Server resource usage.
  - Indexes increase the Library Server database size and affect store and update time.
- Library Server and Resource Manager on the same machine or separate machines?
  - There is higher scalability when they are on separate machines.
  - If the system is small, putting them together is easier to maintain.
- Single or multiple Resource Managers? LAN cache? Replication?
  - Multiple Resource Managers give higher total bandwidth for larger objects.
  - Multiple Resource Managers give higher migrator parallelism.
  - Distributed Resource Managers close to users provide better performance.
  - If the system is small, single Resource Manager is easier to maintain.
- Number of Resource Manager collections?

  Multiple collections give higher migrator parallelism (one thread per collection).
- Resource Manager asynchronous and third-party ingest/delivery?

  Asynchronous and third-party require custom clients. This is appropriate for very large objects, such as IBM DB2 Content Manager VideoCharger™.
- Server platform?

  For mid-tier server, Content Manager Java API is supported on AIX, Sun, Linux, and Windows platforms. Some other connectors are Windows only. The Java conversion engine is cross-platform.

  For Library Server and Resource Manager, AIX, Sun, and Linux offer higher scalability than Windows.

  There will be more in-depth discussion on some of the design configuration choices and trade-offs in the subsequent chapter.

5.4 Hardware

As with all systems, we need sufficient system resources to begin with. This means that we need to invest in hardware. At some point, all of the tuning in the world would not make the system deliver the throughput or response time required; there are just not enough processing resources available. So, scope the requirements, do an initial design, size the system, estimate the cost of the system, and compare that to the projected return on investment.

Your IBM representative can help with sizing estimations using information that you can provide about the nature of your application. These estimates compare projected workload against measured results on specific hardware configurations.

5.4.1 Scalability

Content Manager is a scalable solution that provides a number of ways to increase capacity using additional hardware resources. Increased capacity can be achieved by adding more memory, disk, CPU, or servers.

Content Manager is multi-threaded and multi-processor enabled. Additional processors at the servers can be used by the server components. DB2 and WebSphere both take advantage of lots of memory. Additional memory can also increase system performance. Refer to upcoming chapters for more information about determining whether memory contention exists on your system. In general, the more the better.

Again, additional server resources in the form of more disk drives can be used by the system. There are a number of I/O operations that occur for most Content Manager server activities, such as logging database changes for both the Library Server and the Resource Manager (even more if dual logging is enabled), updates to the database tables themselves, and storing the resource objects to the Resource Manager file system. The more the I/O activities are spread across multiple disk drive spindles or arrays and device adapters, the less the contention is likely, and the more likely faster performance.

Resource Managers

The system can utilize additional Resource Managers. As a first step, the Resource Manager can be split off from the Library Server to provide more capacity to the Library Server and possibly the Resource Manager. More Resource Managers can then be added to the configuration to spread the Resource Manager workload even more. If a single system is limited by its network connection speed or the rate at which data can be moved to or from its disk system, then a second server can double the throughput. If used, the TSM server function can be separated from the Resource Manager to further reduce workload and possible contention on any of the Resource Managers.

As a WebSphere application, the Resource Manager can be deployed as the same application on multiple WebSphere Application Servers. The Web server plug-in can redirect requests to multiple WebSphere Application Server machines, each running a copy of the Resource Manager application. The processing load can be spread across multiple machines. Each instance of the Resource Manager code requires identical access to the Resource Manager database, item storage file system, and TSM system.

Alternatively, we can add independent Resource Manager servers. At some point, the I/O capacity of the system and its storage capacity or performance capacity might be exceeded. Additional Resource Managers provide the facility to add more storage capacity to the system. This provides distributed storage with a central point of control without the complications of a distributed database and multi-site updates.

Using multiple instances of a Resource Manager running on multiple WebSphere Application Servers enables the work to be distributed dynamically. Using independent Resource Managers requires the workload to be distributed by Content Manager. WebSphere Application Server distributes the workload to the least busy application server while Content Manager distributes the load according to the Resource Manager, specified either by the application or as the configured default Resource Manager. You can set the default Resource Manager system to be derived from the Resource Manager associated with the item type or with the user. If you specify item type as the pointer to the Resource Manager, then all items of that type will go, by default, to that specific Resource Manager. If you set the user profile to specify the default Resource Manager, then all items regardless of their item type go to the users associated Resource Manager.

Workload distribution by the default Resource Manager (either item type specified or user profile specified) works only for storing documents. The Resource Manager to service retrieval requests is, by default, the Resource Manager that holds the primary copy of the requested item. If there is a high number of requests for an item, they will all go to the Resource Manager where the primary copy of that item is stored while the other Resource Managers are not used. LAN cache can be used to spread this load. See 5.7.2, “Physical locations” on page 132” for more information about LAN cache.

Mid-tier servers

Mid-tier workload can also be separated from the Library Server and Resource Manager servers. Where appropriate, the eClient Web application server or custom Web application server can be separated from the other Content Manager servers to remove the workload to a separate server. The mid-tier server can also be duplicated for additional capacity. Like the Resource Managers, the mid-tier servers can be multiple instances of a single Web application running on multiple WebSphere Application Servers or as multiple independent Web applications.

Vertical scalability

What we have discussed so far is termed *horizontal scalability,* spreading the workload over more servers; Content Manager also offers *vertical scalability*. It enables customers to move to bigger, higher-capacity servers and operating environments. For example, customers can move from Windows systems to AIX, SUN, or Linux systems.

Clients

The client machines. including the machines supporting browser clients, also need sufficient hardware resources to give the users the response time they require. For these clients, memory and CPU speed are the main requirements to manipulate and display the images. Refer to your favorites browser's hardware requirements, and allow extra resources to support the client side of the Web application and viewer applets if you plan to use them.

Content Manager provides the client APIs on a range of platforms. Client APIs are available in Java on Windows 32-bit operating systems, AIX, Linux, and Sun Solaris, and the C++ APIs are supported on Windows 32-bit OSs and AIX. This increases your options for where and how to deploy client applications. Having the OO APIs available on Sun and AIX enables you to deploy thick clients or mid-tier servers on these platforms for performance, scalability, or server consolidation.

5.5 Library Server

The Library Server is the heart of the Content Manager system. The Library Server controls the system configuration, indexing data, and user access. The Library Server in Content Manager Version 8 is now an extension of the database. It is primarily a set of database tables and stored procedures that hold the data and respond to client requests.

Because it is the single point of control, almost all activities on the system interact with the Library Server. Optimizing the performance of the Library Server can improve response time for almost every user and process in the system.

Stored procedures

The Client API calls are translated by the OO API layer and the DB2 client into SQL and sent to the Library Server database. The generated SQL makes calls to the Library Server stored procedures. The stored procedures run on the server, perform a number of actions on the Library Server database, then return a response to the DB2 client, the OO APIs, and the client application. This model has the distinct advantage of being able to make a single call across the network to perform a range of database searches, updates, and retrieves.

The stored procedures are dynamically linked libraries (DLLs) or programs that run on the server and are called by the database engine in response to the client calls. The stored procedures perform system functions such as checking for access control privileges and run user functions such as searching for an item. The stored procedures call other DLLs as required.

When you define an item type to Content Manager, it uses Data Definition Language (DDL) calls to the database to create tables that match the data definitions that you specified for that item type. It creates a table for each root component and a table for each child component as well as tables to link to parts, and hold links and references. The SQL in these DLLs is pre-compiled, and bound to the database automatically during this process.

These generated DLLs, along with other DLLs installed with the system, use pre-compiled SQL. This can reduce the database server load when these DLLs are run. Using pre-compiled SQL enables the database to pre-calculate the best way to access the data that is required by the SQL. The database stores the access path information in the application package created for each particular version of each specific DLL. At run time, the database optimizer does not need to re-calculate the access path for the data; it merely reads it from the package. In general, this saves only a fraction of a second each time the SQL in the DLL is run, but it might be run many thousands of times each minute and the savings therefore can be significant.

The potential penalty you pay here is that the method the database chooses to get to the data is fixed at the time. The best method to get to the data might change over time. If the table has only one row in it, then getting the whole table at once is the best method to get to the data with or without any indexes. If you then add 10,000,000 rows to that table, the best method to get to the data is likely to be via an index.

To get the database to re-evaluate the access path, rebind the application to the database. To complicate things just a little, the database will only check to see how many rows exist in a table when it is explicitly asked — by you. So to have the database choose the best access path, we must maintain the database by periodically updating the database statistics and rebinding the packages to the database, commonly referred as **`runstats`** and **`rebind`**.

See 8.3.4, “Reorganizing tables through reorg” on page 184” for detail information about how to run **`runstats`** and **`rebind`**.

Table space

Content Manager Version 8 uses database table space by default. Table space allows the system a number of configuration options of which even the default installation takes advantage. Table space enables the database user, in our case the Content Manager Library Server, to use multiple page sizes and multiple different buffer pools. We can also spread the database tables over multiple disk drives.

Table 5-1 lists the default table spaces that are created during Library Server installation, their associated buffer pool, and a description of how these groups of tables are used. This information can be useful when deciding how to lay out table space containers across physical disks.

Table 5-1 Library Server default table spaces

Buffer pool	Table space	Table space description
ICMLSMAINBP32	ICMLFQ32	Holds all the large, persistent, "frequently used" tables – most notably, all item-type tables. When a new item type is defined, the system administration client defines the new tables in this table space. If you customize the DDL, this table space should be defined across multiple containers for performance and scalability.
	ICMLNF32	Holds the large, less frequently used tables, such as event logs, versions, and replicas. If you do not use any of these features, this table space will not need much space.
ICMLSVOLATILE BP4	ICMVFQ04	Holds the "volatile" tables, whose size is usually small but which can vary widely, and for which both updates and deletes are very common (for example, the tables associated with document routing "in progress" items, checked-out items, and so on). Putting this on a separate physical disk should help performance.
ICMLSFREQBP4	ICMSFQ04	Holds all the small, frequently used but seldom updated tables, such as system information, user definitions, ACLs, and so on, that do not take up much space but that are always needed in memory for good performance.
CMBMAIN4	CMBINV04	Holds the tables used for Information Integrator for Content federated administration.

The table space groups together tables that are used in similar ways, and enables us to tune each table space I/O set up individually. Note the numeric suffix on the table space and buffer pool names: This reflects the page size in kilobytes used by the tables in this table space. The database table is made up of pages, which in turn are made up of rows. Indexes are also broken up into pages for storage and retrieval. The database fetches and stores pages of data, not individual rows. This is done for performance particularly when queries access rows sequentially. The larger the page size, the larger the chunk of data that is moved into and out of memory at once.

The size of a row must be slightly smaller than the size of the page. If any of your item-type index data components have a combined attribute field length of greater than 32 KB excluding LOBs and CLOBs, then you should consider using LOBs or CLOBs. The database handles these differently from normal columns, and they are processed more efficiently inside the database.

If the row size is just slightly more than half the size of a page, then each page will hold only one row. This creates a lot of unusable space in the database tables and can slow system operation. If this is the case, you might wish to consider using the DB2 control tools to create a new table space with a more appropriate page size and move your table there.

Buffer pools

Separate buffer pools means that we can tune the memory usage differently for each table space. DB2 uses buffer pools to hold large amounts of database table data in memory, in an attempt to avoid every possible I/O operation. I/O operations take thousands of times as long as accessing data in memory. Any table data that is read frequently should be in memory, in a buffer pool. The larger the buffer pool, the more likely the data will be in memory and the faster the database operations can proceed. However, if you oversubscribe real memory, the operating system will start paging this data out to disk and getting to the data will be even slower than going directly from disk.

DB2 allocates memory from the operating system because it requires up to the maximum specified in the buffer pool size. You can set the size of each buffer pool individually.

Separating the data into different buffer pools enables certain sets of table data to be churned rapidly while preserving other data in memory. Some data is rarely ever reused; it was read in from the disk to answer a query perhaps, but there is not likely to be another query that needs that data again, at least not for some considerable time. This type of data is eventually moved out of memory to make way for other data that is required. In a congested system, other data that was more likely to be reused might have been pushed out of memory to make way for that data. All data has to be paged into the buffer pool before it can be used. All the time that the page of data was in memory after it was used, this page was occupying memory possibly better used by other data.

Multiple buffer pools means that we can keep the data that is frequently re-referenced in one buffer pool and volatile data in another. The volatile data now cannot swamp the other frequently accessed data, because it is in a separate buffer pool.

Indexes

Indexes are vital to the performance of the databases as they grow. When a table grows to more than a few thousand rows, it is much faster to access the data by an index than by looking at each row in the table. The index is maintained as an ordered list of column entries with pointers to the corresponding database table row. Ideally, all access to the indexing data in Content Manager is via an index.

Indexes come at a cost. There is a minor performance cost associated with updating the indexes every time a change is made to table data. Additional storage is required to hold each index, which can, in the extreme, be as large as the table itself.

Indexes are also accessed in the buffer pools and can be placed into separate table spaces and therefore different buffer pools. By separating the index data into a separate table space and buffer pool, it can be protected from being swamped by the table data. This is not done by default and you would need to modify the Content Manager database manually.

The Content Manager administration tool provides a point-and-click interface to create database indexes for your application. Refer to 5.5.3, "Data model" on page 115 for more information.

For a great deal more information about designing and configuring your databases refer to the *DB2 Administration Guide: Planning*.

5.5.1 Library Server configuration

Configuration and design options in the Library Server database have performance ramifications. You can use the administration interface or the administration APIs to set and change configuration parameters, turn functions on or off, and define entities to the system.

To get to the Library Server Configuration, open the system administration client. From the left-side pane, click the **Library Server database** → **Library Server Parameters** → **Configurations**. From the right-side pane, double-click **Library Server Configuration**.

We discuss several configuration options in the Library Server Configuration that might influence system performance.

Definition tab

You can set the following configuration in the Library Server Configuration Definition window.

Max users

Content Manager enables administrators to enforce client licensing restrictions, either rigidly or more loosely. The Max user action parameter allows three options:

Reject logon	Rigidly deny access to users in excess of the license.
Allow logon with warning	Allow the additional users (in excess of the license) in with a warning message.
Allow logon without warning	Allow the additional users in without any warning message.

The second and third options assume that sufficient additional client licenses will be acquired as soon as possible to accommodate these additional users.

An alternative configuration option is to set the maximum number of users to zero. This bypasses license checking although it does not bypass the requirement to abide by the license agreement. License compliance checking requires processing at the Library Server. Bypassing the check removes this load.

For a client to be considered concurrent in the Content Manager Version 8 client licensing scheme, the user must have had an interaction with the Library Server within the past 30 minutes. Users may be logged on, which will make them active for the first 30 minutes. If they have no further interaction with the Library Server, for example, they did not store or retrieve any information for the next 30 minutes, then they are no longer considered a concurrent user for licensing purposes. Tracking this level of user activity places a load on the system and is proportional to the number of users and their activity. Bypassing this checking reduces the load on the server.

If you choose to bypass checking, you should periodically check your license compliance. This can be turned back on by setting the max users to the number of client licenses held.

Features tab

You can set the following options in the Feature tab.

Text Information Extender (now Net Search Extender)

New in Content Manager Version 8 is the internal text search capability. Text search works across both index fields and document text content, and is provided by the Net Search Extender (NSE) in Content Manager Version 8.2. The new text search capabilities are tightly integrated with the Library Server database that enables automatic synchronization of the Library Server data and the text index data. It also simplifies and controls user access to the text index.

The text search engine resides on the Library Server machine, creating additional processing, storage, and network load for this machine.

Text indexing requires the text, from a document or attribute field, to be disassembled into text components and then have them indexed and ordered. The index is built and stored on the NSE server in the Library Server. Similarly, search requests involving traversing these indexes to identify the qualifying items are also done in the Library Server.

Being centrally located has implications for large systems with multiple Resource Managers. During indexing operations, a copy of every document that is to be fully text-indexed is transferred from the Resource Manager to the Library Server, converted where necessary to a suitable format, then indexed by the NSE server. Typically, the number of documents in a Document Management style Content Manager system is *relatively* few; however, the number and the size must be assessed for network impact and index size calculations.

To enhance performance during indexing, consider the following issues:

- Using a VARCHAR data type to store the text attributes instead of LONG VARCHAR or CLOB.
- Using different hard disks to store the text index and the database files. You can specify the directory in which an index is created when defining the text index as part of the item-type definition.
- Ensuring that your system has enough real memory available for all this data. Net Search Extender can optimize text index searches by retaining as much of the index as possible in memory. If there is insufficient memory, the operating system uses paging space instead. This decreases the search performance.
- Leaving the update commit count parameter as blank during text index definition of the item type definition. This is used during the automatic or manual updating of the index. It slows down the indexing performance during incremental indexing. Setting it to blank will disable the updating function.

ACL User Exit

The ACL user exit enables you to have external systems to decide whether access to system resources (such as items and document routing objects) should be granted to users or groups. If this is enabled, every time access is requested to an object, this function will be invoked. It is vital for good system performance that this external function returns its result quickly. Any delays here can be accumulated many times for any one system activity. If a search results in a hit list of many items, each one must be tested through this user exit for this user before they can be presented to the user.

Defaults tab

The defaults tab enables you to define where the default access control list, the default Collection ID, and the default Resource Manager are taken from when a new item is added to the Content Manager system. Unless you build your own client and use it to specify these values explicitly when creating an item, it is assigned the values that are calculated from the information specified by these defaults. The Resource Manager to which an item is assigned can have a major impact on response time to the end user. Refer to 5.7.2, “Physical locations” on page 132 for a discussion about how the location of users and their Resource Managers affects performance.

Setting the default storage option “Get default Resource Manager from” to a value of “item type” ensures that all items of the same item type reside on a particular Resource Manager. This can be used to locate items close to the offices that use them if specific item types are used in specific locations. Alternatively, if the default Resource Manager is taken from the user profile, and the user generally works in the same location as the Resource Manager, then benefits accrue in storing and retrieving items from the closest Resource Manager. These options can also affect the load balancing across Resource Managers.

Default Replication options

Content Manager provides a fail-over service that verifies that Resource Managers are available. If you are trying to store objects into a Resource Manager that is unavailable, then Content Manager tries to store into the next available Resource Manager. Without this fail-over service, you would get an error if you tried to store objects in a Resource Manager that was unavailable.

The fail-over service monitors the availability of the Resource Managers based on the interval that you set on the “Interval to check server availability” field in the Library Server Configuration window. For example, if you set 60 seconds as the interval, it checks availability every 60 seconds. This service should remain running. The Library Server monitor service is named ICMPLSAP (Portable Library Server Asynchronous Process). Although this check does not take a significant amount of processing, it represents some work for the Library Server, the Resource Manager, and the network. Unless there are significant risks of servers becoming unavailable, we recommend that you set this value to 600.

The server time-out threshold specifies how long the server should wait for a response from the Resource Manager before considering it to be unavailable. Set this value high enough to avoid “false positives” — that is, allow the Resource Manager sufficient time to respond when it and the network connection are under load. However, do not set it too high, such that users have to wait a long time before the system switches to an alternative Resource Manager.

Logs and Trace

All logging will increase the load on the Library Server, so only turn on the necessary logging or tracing to meet the business requirements (for example, audit trails). When you turn on system administration event logging, entries are written to the ICMSTSYSADMEVENTS table for every system change made by the system administration client or administration APIs.

The events table grows with each logged event. To reduce the size of the events table, you can remove the expired and unused events from the table. The EventCode column in the events table indicates the classification of events as the values shown in Table 5-2.

Table 5-2 Administration event codes

Value	Definition
1 - 200	System administration function event codes
200 - 900	Item, document routing, and resource management function event codes
1000+	Application event codes

You can delete events from the events table by issuing the following SQL command:

```
delete from ICMSTSYSADMEVENTS where eventcode <=xxx and Created <
yyyy-mm-dd-hh.mm.ss.uuuuuu
```

- `xxx` is the highest event code you wish to delete (for example. 200).
- `yyyy-mm-dd-hh.mm.ss.uuuuuu` is the timestamp of the oldest event you wish to keep in the tables (for example, 2002-01-01-12.00.00.000000).

When tracing, wherever possible, set the log file and path to a different physical disk from other system activity, such as the database logs, database table spaces, and the Resource Manager file systems.

5.5.2 Authentication

New in Content Manager Version 8 are compiled ACLs and administrative domains. Compiled ACLs enable Content Manager to pre-calculate a user's access to every defined control list when they or their components are modified. Administrative domains enable a Content Manager system to be subdivided into domains; this includes the users and the managed objects.

Pre-calculated access control lists speed up the control list processing during run time. The system does the general privilege check (is this user allowed to perform this function at all?) against the ICMSTCompiledPerm table, and then it looks for the user, the tested ACL, and the specific privilege in the ICMSTCompiledACL table. This table holds a row for every privilege in every ACL that every user holds. Note that a large number of users and a large number of ACLs could make this a very large table.

5.5.3 Data model

The primary issue for the data model is to capture the data required by the business problem. When that is established it can be massaged for performance. As mentioned earlier, the Library Server holds the metadata and system control data and is primarily a database application. That means that the same principles that are applied to good database design can also be applied to the Content Manager data model. Normalize the data to reduce the table size and de-normalize for performance. Add indexes to fields that are queried. Cluster the table into a sequence that reflects the way data is accessed.

Normalizing the data means to identify data that might be repeated for one particular entry and extract that data to a separate database table. Imagine that we wish to record information about a customer for whom we have an application form image. That customer might have one name but two addresses, a business address and a residential address. To represent this in our system, we could create two rows, one for each of this customer's addresses. When we search on his or her name we find both addresses. We also see that the database table has the customer's name and possibly other details repeated in our database, which might not be the most efficient use of our resources, especially as the number of our customers grows. One solution is to change our database model and store the data that gets repeated (in this case, the customer's name and other customer details in one table and the address data in another table), and maintain a database relationship between the two tables.

In Content Manager, this means implementing folders, links, references, or child components as needed. Refer to the System Administration Guide for more details about these components and additional information about how to model your data.

In the example above, this could be implemented as a root and child component relationship. The customer's name would be part of the root data, and their addresses would become entries in a child component attached to the root. It is also possible to model this using links. The customer name information may be modelled as a root component that is linked to other root components, which each hold the address information.

Using our previous example, we might have several applications or other types of documents that relate to this same customer. We could have his or her name recorded many times. This could be another opportunity to reduce the amount of duplication. We could create a folder that contains all customer-related documents and attach to the folder the name and addresses.

Normalizing the data in this way has performance advantages and disadvantages. Removing the duplication of the data makes the database smaller. Two (or more) tables are created whose combined size is smaller than the original. This means that the database has less data to scan when searching, the data rows are shorter so we get more rows in each page, so this reduces the number of I/O operations and increases the number of entries in memory, thereby potentially avoiding I/O operations altogether.

On the other hand, we now have to go to two tables for the complete set of information about any one item. We need to go to the customer table and the address table and perform a database *table join.* In this operation, rows from one table must be matched against the corresponding rows in the other table. We also need to create a build and maintain the database relationship between the two sets of related data. All of these actions cost processing time on the server.

In general, normalize the database design for size and simplicity, and de-normalize only when query performance becomes a major issue.

Versioning

Versioning creates a new copy of index data and the associated document when an update is made. The system preserves the original as well as the new updated item. This can be extremely useful where users wish to regress to an earlier version of a document or undo erroneous changes, or to create a new derivative work that might take a different development path. It can also be required by legislation or management practice to retain the history of development of an article or policy contained in a document. This can have an impact on performance.

When the previous versions of a resource object are maintained in the system, they occupy space in both the Library Server and the Resource Managers. Similarly, if previous versions of only the index data is kept, then the impact is limited to storage on the Library Server.

Additional rows in the database can have an impact on its performance. Versioning typically requires the system to restrict the query results to one version of the item — usually the latest. This means additional criteria for the query (predicates) must be resolved, so the indexes must be maintained to match the criteria (filter) for the required version. This adds complexity to the database and to the query processing. Filtering for the correct version has to be done for almost every action on an item in the system.

Logging

Just like system administration logging, item-level activity logging will increase the load on the Library Server. Turn on only to the level of logging necessary to meet the business requirements. If you enable Read logging, the system will warn you that this can have an effect on performance because of the amount of data that is likely to be recorded. When you turn on item event logging, entries are written to the ICMSTITEMEVENTS table for every selected action on all items in this item type made by all of the clients and APIs in your system.

The events table grows with each logged event. To reduce the size of the events table, you can remove the expired and unused events from the table. The EventCode column in the events table indicates the classification of events as the values shown in Table 5-3.

Table 5-3 Item event codes

Value	Definition
1 - 200	System administration function event codes
200 - 900	Item, document routing, and resource management function event codes
1000+	Application event codes

You can delete events from the events table by issuing this SQL command:

```
delete from ICMSTITEMEVENTS where eventcode <=xxx and Created <
yyyy-mm-dd-hh.mm.ss.uuuuuu
```

- `xxx` is the highest event code you wish to delete (for example, 600).
- `yyyy-mm-dd-hh.mm.ss.uuuuuu` is the timestamp of the oldest event you wish to keep in the tables (for example, 2002-01-01-12.00.00.000000).

Document model - item type classification

When defining the data model, you choose the item-type classification, which defines the relationship between the resource object and the customer's indexing data. Item-type classifications are items, resources, documents, and document parts.

In the simplest case (the items), there is only indexing data, and no resource object. This is somewhat like a folder or a database table entry, in that there are data and relationships to other items in the system but no image or document file associated with this item. Unlike the folder, however, there are absolutely no resource objects associated with it, and that includes no note log.

The resources item type is indexing data that *is* associated with a resource object, such as an image file, a word-processing document, or an XML file. It is important to note that there is a one-to-one relationship here: only one set of indexing data for one resource object. For many situations, this could be sufficient. For other applications, we might need to keep multiple resource items for one set of indexing data. For example, in some implementations of Content Manager system, an image might be composed of multiple parts:

- The image itself
- The file that holds annotations and mark-ups
- A note log
- The text extracted by OCR from the image

The documents item type allows multiple resource objects to be linked to a single set of indexing data, enabling us to accommodate resources that might be made up of different parts, as in the example above. Specifically, the system comes with predefined parts for image, text, note log, annotation, and streaming data.

If these predefined parts are not sufficient for a particular application, you can define customized document parts using the document parts item type. This enables you to define an item type that carries index data to support specific properties of the part you wish to create. Content Manager then makes this a part you can include in an item type you define with an item type classification of documents.

What then are the performance implications of choosing one item-type classification over the another? Several trade-offs should be weighed here. Only the document model item type classification is supported by the out-of-the-box clients. Unless you are designing and building customized clients, you should stick with the document classification. If you are building a custom client, the next trade-off depends on what the particular application demands. If you are integrating scanning systems that assume the document classification, you should stick with that. If the particular files and documents to be used are best modelled as multi-part documents, then the document model provides the simplest implementation.

If your application utilizes the items and resources item model, in general, this is more efficient. The documents model requires intermediate tables to hold the links between the document and the parts of which it is comprised. This means that there are additional tables that must be updated when documents are added or changed and must be traversed when fetching data, as compared to items and resource items. All of these cost processing time on the Library Server and there are consequent performance advantages if they can be avoided.

Database indexes

As discussed earlier, indexes are vital to Content Manager performance, particularly as the number of items stored in the system increases. When you define a new item type, the appropriate indexes on the new tables are created automatically. Indexes on user-defined item attributes are not generated automatically. You must define them manually. For the attributes that are frequently used for regular searches, define indexes on these attributes to improve response time and reduce Library Server resource usage for queries over these attributes. The system administration client provides a point-and-click interface to define indexes. You can choose one or a combination of attributes to create each index and you can specify the order of the index, either ascending or descending. Use combinations of attributes to create the index where users typically use that combination of attribute values to find items in the item type.

Content Manager cannot anticipate the way your application will choose to search for the data. Application queries are embedded in the compiled programs as Dynamic SQL, which enables the system to calculate the access path at run time. This means that the database optimizer calculates the best access path to the data for your queries when they are run. The database then immediately starts using your indexes for queries after the indexes are created, assuming that the optimizer chooses to use them. The optimizer might decide that the indexes do not help resolving the queries because they are on the wrong columns or because the optimizer is working with outdated statistical information about the table.

It is always a good idea to run **`runstats`** soon after you create an index so that the optimizer has the best information. If you do run **`runstats`**, it is also a good idea to **`rebind`** the database packages as well. The optimizer might be able to use the new index to improve access speed for other SQL and might improve other access paths based on the new statistical information.

Unless there is a major performance problem affecting most users at the time, it is best to create indexes, run **`runstats`**, and **`rebind`** at times of low user activity. These utilities can have a negative impact on server performance while they are running.

See 8.2.5, “Create attribute indexes” on page 177 for details about creating indexes for attributes. See 8.3.3, “Keeping database statistics and execution plans up to date through runstats/rebind” on page 183 for more about **`runstats`** and **`rebind`**.

Multiple item types

Many Content Manager applications tend to work actively on an item while it is relatively new to the system. At some point, that work completes and the item is archived. For these types of applications, it can be a performance advantage to separate the active documents into a separate item type from those that have already been dealt with (archived). Sooner or later the archived documents will out-number the active documents, and all being well, the number of active documents should plateau or at least have relatively slow growth. The number of archived documents will continue to grow at about the rate new documents enter the system.

This can provide performance benefits in several ways. The active documents item type can be tuned from maximum performance. It might be feasible to create additional indexes on this item type that were too costly on all the data. This new, active data tends to be updated more frequently and that can lead to fragmentation of the database tables. If this is a smaller database table, it takes less time to re-organize. Because these tables are smaller and more frequently accessed, they can be retained in the system memory. The item type can be separated into a new table space and have a dedicated buffer pool. Poor queries that cause table scans will not have as dramatic an effect on performance because these tables will be of limited sizes.

Note that information from previous versions of an item is not transferred across when the item type is changed.

5.5.4 Document routing

Document routing is a fast and efficient method of managing document-centric work flows. Because it is built into the Content Manager system, no additional program layers are required to connect to the work flow and synchronize operation. If document routing offers enough function to meet your business requirements, then it is the simplest and most efficient work flow tool to use.

Note that the work flow APIs and user exists enable you to extend the functionality of document routing if required. For example, parallel routing can be emulated by sending the same item through two different processes at the same time.

5.6 Resource Managers

How we use Resource Managers can play an important part in overall system performance. The Resource Manager provides the resource object storage management functions. It receives, stores, and sends images and document files from and to the client applications. New in Version 8 is the use of standard Web ports and protocols to send and receive requests. The Resource Manager acts like a Web server, serving resource objects on request. However, the request must have a valid token supplied by the Library Server.

Behind the scene, the Resource Manager runs in a Web application host: WebSphere Application Server, which is a Java application that holds control data in the Resource Manager database and stores the resource objects on the file systems connected to the server, and in an optional TSM backup archive.

Note: The Resource Manager uses its own database connection pooling methods, not the standard WebSphere database connections.

Upcoming chapters cover tuning the configuration parameters for the database, WebSphere, and TSM. First, we should consider configuration and design options in the Resource Manager that can affect performance. As in the case of the Library Server, you use the system administration client or the administration APIs to set and change configuration parameters, turn functions on or off, and define storage to the system.

5.6.1 Resource Manager properties

To reach the Resource Manager properties, open the system administration client. From the left-side pane, expand the Library Server (ICMNLSDB) Æ **Resource Managers** → the Resource Manager database (RMDB). Right-click the Resource Manager database to select **Properties** to open the Resource Manager Properties window. Right-click the Resource Manager database to select the Staging area to open the Staging Area panel.

Properties

In the Resource Manager Properties panel, you can set the details stored in the Library Server about the Resource Manager, including communications details, logon details, token duration, the LAN cache switch, and a disable switch.

The Library Server and the system administration client use this connection information to make connections to the Resource Manager. The connection information includes host name, user ID, password, protocols, port numbers, and access data used to log on.

Token duration

Token duration specifies how long the security token, provided to a client to access a document on this Resource Manager, will remain valid. The default is 48 hours. A client application may continue to access the resource object without reference to the Library Server for the period that the token remains valid. If your application frequently re-reads the same item on the Resource Manager, you can store the item's URL and go directly to the Resource Manager for the item for as long as the token remains valid.

LAN cache

LAN cache enables objects to be retrieved from the user's default Resource Manager if a copy is cached there. The user's default Resource Manager should be the physically closest Resource Manager or the Resource Manager with the highest bandwidth network connection to the user. If the requested resource object is not cached on the Resource Manager, then the original is sent to the requestor from the Resource Manager on which it is stored and a second copy is sent to the user's default Resource Manager. If the user or a colleague in the same location requests that resource object again, it can now be sent from the user's local (default) Resource Manager.

LAN cache can provide a big benefit where there are relatively slow or busy network links between locations, and users need the same item multiple times. If the item is accessed only once, then there is unnecessary network load generated sending a second copy to the Resource Manager that will never be used.

In earlier versions of Content Manager, there existed a function to "pre-stage" or copy a list of items from one Resource Manager to the staging area or LAN cache of another Resource Manager. This meant that objects could be copied to the cache prior to users requesting them. This function is no longer supported. The same effect can be achieved using replication and a specialized client that finds the nearest copy.

Staging area

The staging area is the cached disk space used to hold copies of objects stored permanently elsewhere in the system. The staging area holds files recalled from TSM and, when LAN cache is enabled, files from other Resource Managers. Unlike previous versions of Content Manager, new resource objects are written directly to the first storage location for that item's storage management hierarchy. That is, they now bypass the staging area. This significantly reduces the internal movement of data inside the Resource Manager and leaves the staging area as a pure cache.

Also new in Content Manager Version 8.2 is the ability to divide the cache into multiple subdirectories and set a limit on the largest file cached there. The ability to create subdirectories reduces the number of files per directory, which helps avoid the file system limitations when the directory holds a large number of files (10,000 to 40,000+).

The ability to control the maximum size of cached resource objects means that we can avoid flooding the cache with very large objects. A few very large files can force the deletion of a large number of smaller files to make space.

The staging area is managed by the system to a fixed size. Items are stored in other locations according to a time period specified in the management class assigned to the item. Items in the staging area are there only while there is sufficient room. When the stagging area grows too large, the system deletes items from it in order, starting with the least recently used. This function is carried out by the purger process.

In the Staging Area panel, you can specify the size of the staging area and the thresholds by which the size of the storage is controlled. The thresholds are checked by the purger process to determine when to start and stop the purging. At its scheduled start time, the purger determines whether the current size of the staging area exceeds the “Start purge when size equals” value (as a percentage of the maximum staging area size). If it is, then the purger removes the least recently referenced resource objects until the size is reduced to the “Stop purge when size equals” value (as a percentage of the maximum size). When setting these values, the aim is to keep the staging area as full as possible, while always allowing room for new items to be added.

Files coming from TSM, placed in the staging area, indicate that they might have been migrated before users had finished using them. Files from LAN cache could indicate that usage patterns do not match the item type, user profile, or Resource Manager assignment expected by the system design. Finding the item in the cache usually provides significant performance improvement for the end user over getting it from near-line media through TSM or a remote Resource Manager.

The larger the staging area, the more likely resource objects are to be found there; thus, better performance. However, this sacrifices storage for other activities. The staging area will overrun the maximum size if required, as long as there is space on that file system. Unless the file system on which the staging area resides is close to 100% utilized, set the “Start purge when size equals” threshold close to 100% of the staging area size and the staging area as large as you can without interfering with other server storage requirements, and leaving some spare.

The “Stop purge when size equals” threshold should be tuned as best as you can to the processing of these exceptions. For example, the majority of items in the staging area might have been recalled from TSM because of customer inquiries about prior transactions and these are handled on the spot. They are unlikely to be re-referenced, so there is no point in retaining them in the cache. Conversely, the files could be in the stagging area because of periodic reviews as part of some business control process that might take several days or weeks to complete. In this case, it is worth keeping as much of the old data in the cache as possible. This is done by setting the “Stop purge when size equals” threshold closer to the maximum value. If access is unpredictable, set the stop threshold to 80%.

5.6.2 Configuration

Configuration within the Resource Manager has performance ramifications. You can use the system administration client or the administration APIs to set and change configuration parameters, turn functions on or off, and define entities to the system.

To reach the Resource Manager Configuration, open the system administration client. From the left-side pane, click the Library Server (**ICMNLSDB**) → **Resource Managers** → the Resource Manager database (**RMDB**). From the right-side pane, double-click **Resource Manager Configuration**.

Definition tab

The Resource Manager Configuration's Definition panel sets the timeout values for Library, client, and purger.

Time out - Library, client, and purger

These values specify how long the Resource Manager waits for a response. If no response is received within that time frame, the Resource Manager assumes that the component is no longer able to communicate and it should start the appropriate recovery processing.

Set these values high enough to avoid "false positives"; that is, allow the Library Server, client, and purger sufficient time to respond when they and the network connections are under load. However, do not set them so high that users have to wait too long before the system recognizes an error condition.

Cycles tab

The Cycles panel sets the time period between executions of the background Resource Manager tasks. In general, because no user is waiting on the completion of these tasks, the performance considerations relate to minimizing the impact of these tasks on the system.

On the Cycles panel, set the frequency at which these tasks run. The Schedule panel for the migrator and replication specify when these functions run. To have them run at all, the functions must be enabled elsewhere.

For Resource Managers running on Windows 2000, each of these processes is installed as a System service. They can be started and stopped from the standard Windows 2000 services control window.

To start these tasks on AIX selectively, enter the following command:

```
etc/rc.cmrmproc start dbname rmappname application
```

In this example:

- `dbname` is the database name on which these processes are running.
- `rmappname` is the name of the Resource Manager Web application.
- `application` is the Resource Manager stand-alone process that you want to start (RMMigrator, RMPurger, RMReplicator or RMStager).

For example, **`/etc/rc.cmrmproc start rmdb icmrm RMMigrator`** starts the Resource Manager migrator on the database `rmdb` with `icmrm` as the name of the Resource Manager Web application.

Start only the processes you need. For example:

- Unless you are running VideoCharger on this server, it is not necessary to run the RMStager.
- If you are not running TSM or LAN cache is not turned on for this Resource Manager, it is not necessary to run the RMPurger.
- If you are not running replication of resource objects from this Resource Manager, it is not necessary to run the RMReplicator.

Do start the migrator even if you are not running TSM. The migrator is responsible for deleting items and tidying up after failed transactions. See the migrator discussion below.

Purger

The purger clears free space in the staging area by deleting old copies of items. In the purger field, type or select the amount of time in hours and minutes that must elapse before the purger begins purging (if necessary). Run the purger often enough to avoid running out of space in the staging area file systems.

Threshold

In the Threshold field, type or select the amount of time in hours and minutes that must elapse before the migrator checks the capacity of the Resource Manager disk volumes to ensure that they are below the capacity threshold that was set when defining the volumes.

New in Content Manager Version 8.2 is the ability to define migration based on available capacity on Content Manager managed volumes, in addition to time. When defining a new volume to a Resource Manager, you can specify the capacity threshold at which migration will begin. If that threshold is set below 100%, the migrator will check the capacity of this volume. If the used space on the volume is greater than the threshold value, the migrator will begin to move resource objects to their next storage location. The next storage location is defined by the workstation collection definition for the resource object. It is the next available volume in the storage group for the collection that is of the storage class defined by the next step in the migration policy for that collection. In this case, migration occurs earlier than is defined by the migration policy.

Threshold migration works like normal migration, except that it occurs when the volume space used exceeds the threshold. The objects are migrated to the same place as during normal migration. First it performs normal migration (action date expired), then it migrates according to least recently used and nearest action date until the percentage of volume used is under the threshold.

Stager

In the Stager field, type or select the amount of time in hours and minutes that must elapse before the stager checks whether the capacity threshold of the VideoCharger volume has been reached.

Migrator

The migrator is a function of the Resource Manager that moves objects to the next storage class according to their assigned migration policy. The migrator is also responsible for deleting from the file system the items that have been marked for deletion by users' explicit action.

The movement of items to other storage media is an important task that can add significant workload to the Resource Manager, both for the database and its I/O subsystem. Items become eligible for migration to the next storage location based on a date field (OBJ_ACTION_DATE column of the RMOBJECTS table in the Resource Manager database). This date is calculated when the item was stored based on its assigned management class. It is recalculated every time it is eligible for migration.

The Content Manager Version 8 administration tool does not allow you to create a migration policy whose last assigned storage class is anything other than "forever." This reduces the server load that migration previously caused when continuously updating the next action date of resource objects with no next storage class but a due date to be migrated. Previously, it was possible to create a migration policy that stored items to disk for only 30 days. After 30 days, the migrator would set the next action date for all items stored on the first day to 30 days into the future. So every day the migrator updated 1/30th of the items in the system, but with no change in the actual storage of the items. In the current version of Content Manager, this does not happen. If necessary you can change the next action date using SQL to make items store forever on a particular storage class eligible to be migrated.

The next action date field is not a timestamp but a date field. All objects due for migration on any particular day become eligible to migrate at 12:00 AM that day. As such, it is usually best practice to migrate all items outside of prime business hours at one go instead of letting the migrator run until the following day.

The query to determine items due for migration can take a long time. This query is run for each *batch* of files until there are no more qualifying items. By setting the batch size larger you can reduce the number of times the query is run each day and speed up the migration process. The trade-off here is that your

Resource Manager JVM needs to be large enough to hold and process the whole batch.

In Content Manager Version 8, the migrator is multi-threaded. This means that multiple jobs can be initiated by the migrator to move data. The migrator creates a thread for each volume from which data is being moved. To get concurrent processing, you need to have more than one volume defined in the storage class from which data is migrating, which might improve overall migration time.

The migrator is also responsible for delete processing. Deletion of the Resource Manager resource objects is an asynchronous activity. Resource deletion consists of three steps:

1. A Content Manager application deletes an item from the Library Server.
2. The Asynchronous Recovery Deletion Reconciliation utility marks the resource for deletion on the Resource Manager.
3. The Resource Manager migrator physically deletes the resource during scheduled running periods.

When the migrator is started but not scheduled to run, it periodically runs the Asynchronous Recovery utility.

The *Asynchronous Recovery* utility's purpose is to periodically restore data consistency between the Library Server and its Resource Manager. This process is necessary for the following reasons:

- To provide a rollback function for failed transactions. The Library Server and the Resource Manager can become inconsistent in the event that the Resource Manager crashes or communications between the Information Integrator for Content (formerly known as EIP) toolkit and Resource Manager fails. The inconsistent state can be reconciled with the Asychronous Transaction Reconciliation utility.
- To complete scheduled deletion of items that were designated for deletion. Note this only occurs during times when the Asynchronous Recovery Utility runs and the migrator is scheduled to run.
- To delete tracking table records (for both the Library Server and the Resource Manager) for transactions that are determined to have completed successfully. As each create or update transaction for resource items completes, a record is placed into the Library Server database. The number of these records and the database table that holds them becomes larger over time. The records are removed by the Transaction Reconciliation utility. It is important to run the utility on all of the Content Manager Version 8 Resource Managers.

The migrator should be started even when it is not scheduled to run because it helps to maintain the transaction tables.

Items are checked out from the Library Server during the migration process to control the movement transaction. This means that the Library Server will be involved in the migration as well.

Replication

Replication copies resource objects to a workstation collection on the same or another Resource Manager. If the primary Resource Manager becomes unavailable, a replica of a required resource can be retrieved from the other Resource Manager. This provides increased availability, but should not take the place of normal backups.

In the Replicator field, type or select the amount of time in hours and minutes that must elapse before the system checks to see whether replication is necessary.

Similar to migration, replication involves moving a potentially large amount of data at a scheduled time. Unlike migration, items can become eligible for replication throughout the day. Typically, customers want the replicas copied as soon as possible after the original is stored. Another complicating factor for replication is that it frequently involves moving data over relatively slow network links.

Replication is also multi-threaded. A number of threads are pooled for each target Resource Manager. Each replication request is assigned to one of these worker threads. Again, the item is checked out for the duration of the replication process. You specify the maximum number of replicator worker threads as ICM_MAX_REP_THREADS (value between 1 and 50, default 10) in the icmrm.properties file.

Replication uses significant processing power on the Resource Manager. It is recommended that you run replication almost continuously. If it is run only after prime shift, the system might not be able to keep up with the load. If necessary, schedule replication to pause during a few of the peak hours of the day.

In Windows, the process runs as a service: ICM Replicator (RMDB). For AIX, **`rc.cmrmproc`** can be used to manually or automatically start the service.

The replication process *must* run as the same user that runs the application server. The scripts start a Java virtual machine using the WebSphere Application Server JVM and the deployed Resource Manager class files. You might need to modify the JVM memory settings to allow for a larger virtual machine size on very active systems. If the Resource Manager servlet is cloned, only one clone should run the Replication process.

5.6.3 System-managed storage

Device Managers, Storage Classes, Storage Systems, Storage Groups, Migration Policies, and Workstation Collections are the components used to build the storage management policies that are applied to each item or part stored in a Content Manager system. For convenience, we deal with the entire process as a group.

Much of the design and terminology of system-managed storage (SMS) goes back a very long time, at least in computing terms. The principle behind SMS is that storage is assumed to be both more expensive and faster, the closer it is to the CPU. The purpose of the system-managed storage components is to give you the flexibility to configure the system to keep active data on faster media and move inactive data to slower, cheaper media.

The system's ability to achieve this depends very much on the designer's ability to predict the way in which data will be accessed. In general, time is a very good indicator of the way in which files will be accessed. For forms-processing systems, the very nature of the application is that there is a certain target time for handling these forms, and almost every form will be actively used during that period. At completion of the forms process, the document is rarely accessed.

For document-management systems, the period of high activity is less well defined but typically occurs at the early part of the document life cycle. During creation and editing, there are frequent accesses by the author or the editor as the document is prepared for release. Most documents have high read activity soon after release, then the read activity dies down.

Rich media and Web application information typically have a similar pattern: lots of activity during preparation, and high read access tapering off over time. In this environment, there can also be archiving activity to remove items from the pubic access area when they become outdated.

Archive systems usually do not have this early life period of activity. This activity has already occurred in the archiving application.

Time-based migration

In setting the SMS configuration, you need to match the access patterns with the speed of the storage media. Keep the items that are likely to be accessed on fast media and move the inactive data to slower or near-line media.

Wherever possible, spread the data on your disk based on storage classes over multiple volumes and use the maximum number of subdirectories to reduce the number of files per directory.

SMS fills each volume assigned to a storage class in a storage group, then moves to the next volume. This means that if you have multiple disk volumes dedicated to a storage class in a workstation collection, each volume will be filled in turn. SMS enables you to share disk volumes between storage groups and workstation collections, so that items from multiple workstation collections can exist on the same disk volumes. To spread the workload over the disk drives, you should dedicate the drives to a particular workstation collection. If you need to share them, create multiple storage groups and change the order in which the volumes are defined.

Space-based migration

Content Manager Version 8.2 enables you to define migration according to capacity as well. Setting a volume threshold below 100% will cause the migrator to monitor the volume's free space and potentially move data before the migration policy predicts. If you are using migration and you have a good understanding of the usage patterns, set this value to 100% and use overflow volumes to avoid running out of storage.

Alternatively, if you wish to keep as much data on the Content Manager managed disk as possible, set the migration period long (but not forever) and the volume capacity threshold below 100%. Define a next storage class in the migration policy as the destination for resource items when they are eventually migrated. The migration period for the Content Manager disk storage class (or classes) can be set long enough that any access after this time is very unlikely. Resource objects will be on disk for this period and get fast access. Set the migration thresholds on the volumes for the Content Manager managed disk under 100 % to avoid running out of space. The resource items will reside on disk for up to the migration period, as long as you might possibly need them, unless you begin to run out of disk capacity before the migration period. The volume threshold will become the trigger for migration; old data will move to the next storage location, maintaining some free space on disk for your new data.

5.6.4 Logging

There is a Resource Manager activity log file as well, but it is not controlled by the administration panels. The method and detail recorded in this log is specified in the icmrm_logging.xml file, which is located in the following directory:

```
.../Websphere/AppServer/installedApps/<node name>/<Resource Manager
name>.ear/icmrm.war/
```

In this syntax:

- <node name> is the WebSphere Application Server node name. This is usually the server host name.

- <Resource Manager> name is the name of the Resource Manager Web application, which is icmrm by default. It is followed by .ear as the directory name.

It might be necessary to increase the level of detail recorded in the log to resolve problems with the Resource Manager. As mentioned earlier, additional logging involves additional workload on the server. The system uses the Log4J standard logging utility and by default it logs asynchronously. In the logging configuration file, you have the option to specify that logging is done directly to a file. This too can have an impact on system performance. In one of our scenarios, with pure load test run with debug level tracing directly to a file, the system throughput dropped by more than half as compared to the default logging settings.

5.6.5 Move Resource Manager storage area

Move the Resource Manager storage areas to a different disk to achieve better I/O and increase performance on a data loaded system.

The instructions to perform this action are the same as replacing or repartitioning a hard drive. This instructions can be found in the *IBM Content Manager for Multiplatforms System Administration Guide*, SC27-1335.

Plan ahead. You can avoid this kind of procedure if at installation time you plan for growth of your system. This task includes many steps that have to be performed by an experienced DBA because of the data involved. This procedure really affects general performance; thus it distributes the charges and I/O capacity. If you are having I/O contention, this is a good procedure to improve system performance and avoid I/O contention. You can check the I/O contention problem on AIX using **`topas`**. Look for the disk utilization where the Resource Manager storage area is located. If it is constantly near 100% utilization, you have a I/O contention problem.

5.7 Network

Network plays an important part in a system's response time and performance.

5.7.1 The triangle or "inverted V"

The data flows between the client, Library Server, and Resource Manager changed dramatically in Content Manager Version 8. The client became the central cohesive element in fetching resource objects from the Resource Manager. In general, there is very little communication between the Library Server and the Resource Manager. The client communicates with the Library

Server to find and request access to data. When the Library Server authorizes the access, it sends a key (a token) related to the asset directly back to the client, unlike previous Content Manager implementations. The client then passes the key (token) to the Resource Manager to request the resource object (Figure 5-1).

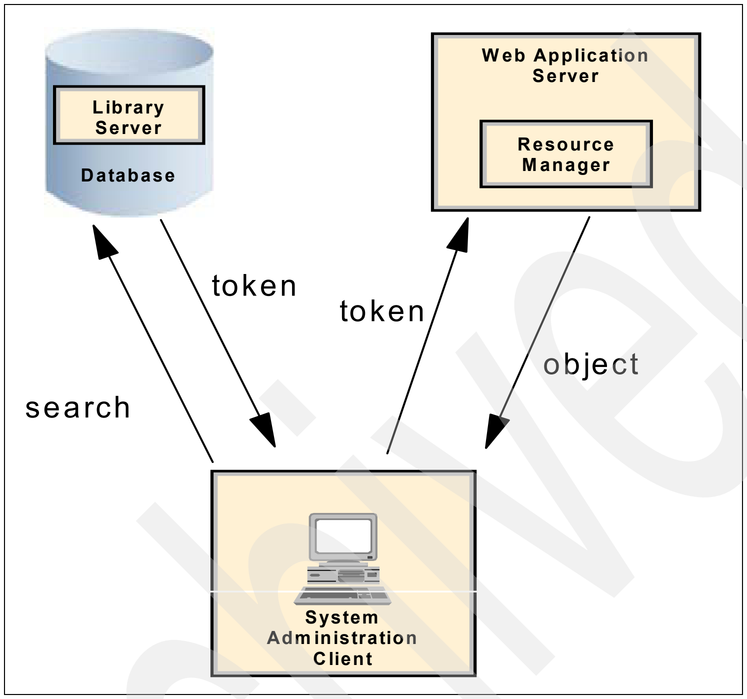


Figure 5-1 Data flow triangle

5.7.2 Physical locations

One aspect of scalability and performance that has a major impact on design is the support for multiple Resource Managers in the same system. Additional Resource Managers can spread the I/O and network load involved in storage and retrievals. If installed at physically remote sites, this can mask wide area network delays.

The Resource Managers can be located next to users to potentially reduce the network traffic and delays associated with moving a relatively large amount of data across a wide area network. A customer with multiple sites can run a single Content Manager system with a central Library Server for a single point of control and have distributed Resource Managers at each location. Users can store and retrieve their data to and from their local server. All data is available to all authorized users throughout the system, but the frequently accessed data is located on a nearby server.

For example, a global organization has offices in Chicago, Munich, and Singapore. Almost all work done in Munich revolves around Munich's customers and their interactions with the organization. All the information about the customer, their interactions, and the local staff can be held in Munich. The Chicago office holds similar information about its own customers and staff, as does Singapore. For this global organization, it is rare that Munich-based customers would have dealings with the Singapore office. If they do, the system still can support them. The Singapore staff can access the Munich data, but response time might be slower because of the requirement to move large images or document files over the relatively slow network links to the Munich office.

There are three ways to configure Content Manager to store data in a particular location:

- You can set the storage location for each item as it is stored using APIs in your customized client application.
- Create specific item types for each location and have the item types set the default Resource Manager.
- Set the system to get the default Resource Manager based on the user storing the item, and for each user, set the default Resource Manager to be the one in their physical location.

In Version 8.2, Content Manager reintroduces the LAN Cache facility. LAN Cache enhances this distribute operation by taking another copy of remote objects and storing it in the requestor's local Resource Manager staging area. Subsequent requests for that resource object at that location would be served by the local Resource Manager. In the global organization example above, after the first time Singapore requested the Munich data that resided only in Munich's Resource Manager, any subsequent request to access the Munich data will be served by the Singapore Resource Manager, and the data will be from the Singapore Resource Manager staging area.

5.7.3 Network and resource objects

Moving resource objects within a network affect network performance. On the other hand, network performance can have a big impact on the Content Manager system performance. Compared to typical transactional data, documents or resource objects are, in general, relatively large: a database row update might be a matter of a dozen bytes and an insert might be up to 1000 bytes; whereas an image file is typically 50,000 bytes. The time taken to move this amount of data affects user response time.

Every store and retrieve operation moves the document across the network (WAN or LAN). Consider the forms processing application referred to earlier in this chapter. If each item is stored once and retrieved and viewed in five workflow steps, then the document must traverse the network between the client application and the server six times. If the resource image is 50 KB and we have to process 10,000 images per 8-hour day, then we can calculate the network traffic approximately as follows:

$$\frac{50\ \text{KB} * 10{,}000\ \text{images/day} * 6\ \text{times on the network}}{(8\ \text{hours} * 3600\ \text{seconds/hour})}$$

= 100 KB / second = 800 Kb per second (average)

Although the work process might take weeks to complete, in order to keep a steady workload (in this example, no build-up of work in the system), then the system needs to process the number of items equivalent to the number entering the system each day. This means, in our example where 10,000 items entering the system everyday, the system needs to process 10,000 items every day to avoid a backlog.

The calculation above is only approximate, because additional header and trailer information is appended by the HTTP response, and neither the request nor calls to the Library Server are considered.

This does not seem to be much load when minimum LAN speed is 10 Mb per second; however, the network is not installed specifically for this system. All estimates for the network traffic have to take account of existing network traffic and that the effective throughput of a 10 Mb LAN is really only about 6 Mb.

Another factor that affects the calculation above is the difference between average and peak utilizations. Often, the end users' work habits can mean that the majority of the work is done in certain highly productive time slots. Some users might get a late start each morning, by doing things other than "real" productive work, such as reading e-mail, getting coffee, conducting meetings, and doing other administrative tasks. Users might delay serious production work until mid-morning. This could present a peak system utilization demand during mid-morning. Lunch time can have an effect as some users go early and some go late. The workload declines around the lunch break. There might be another peak period in the afternoon after the late lunch. Finally, closing time also has an effect because some users go home early, and others stop using the system in preparation for closing up for the night or finishing other activities.

This all means that peak load can be two or more times the average load.

When considering network performance, batch processing and overnight processing must be included. Object replication and migration move large amount of data over the network. Typically these background tasks can be done at any time; they are best deferred to times when user activities are low. Prefetching items during the off hours in preparation for the next day's processing is another effective way of reducing the network traffic during the day and improving system response time. Other network operations outside of Content Manager also can have network impact. Things such as file system backups shift large amounts of data across the network that potentially interfere with system response time or the elapsed time of the background operations. If network contention is severe, it could delay background activities into times of high user activity and cause user response time problems. Planning and scheduling background tasks are important to improve system performance and response time.

5.7.4 Network and Web clients

So far, we discussed moving the resource objects directly to the user requesting them. This happens by default for client/server or two-tier configurations. In the default configuration of eClient, resource objects are sent by the Resource Manager to the Web Application Server, which then sends them to the browser client. On a single-server system this might work, but when the Resource Manager and the eClient are on separate servers this can load the network unnecessarily.

Consider our previous example, of the global organization again. Suppose the Munich office establishes a Web interface to their system and encourages the whole organization to use it. If they use the default configuration, users in Chicago requesting their own resource objects via the Web will cause the files to go from the Chicago Resource Manager across the Atlantic Ocean to the Munich eClient application, only to be turned around and sent back to Chicago.

The solution is to configure the eClient to send the resource object directly from the Resource Manager to the user's browser. In our example, the resource request from the browser client in Chicago receives a URL pointing to the resource on the Chicago Resource Manager, so the resource object goes directly from the Chicago Resource Manager to the Chicago browser.

When the resource object is sent directly to the browser, there is an additional complication. The browser becomes responsible for handling the particular MIME type of the resource object. In the default configuration, the eClient application on the mid-tier server converted a range of MIME types into browser-friendly format. TIFF is not supported directly by Netscape or Internet Explorer, so the eClient converts TIFF to GIF or JPEG before sending the objects to the browser. When the objects are sent to it directly, the browser needs to handle them. In this case, the eClient can be configured to send an applet to view a number of MIME types that are not supported by most browsers.

Sending the files directly to the browser can enhance system performance because the workload of converting the file from one MIME type to another is removed from the eClient server. The machine on which browser is running now does the work of rendering the image in the applet and therefore requires sufficient resources to meet the response time expectations.

There is a cost for this configuration. The applet must be downloaded from the eClient Web application server to the browser every time the browser is shut down and restarted to access a resource object that requires the applet. The applet is nearly 400 KB. This could add more network traffic than it saves for a large number of infrequent users.

5.7.5 Network and Library Server

Most API calls interact with the Library Server database. Every time there is an interaction, there are network delays and additional workload on the database. We should avoid these as much as possible. To avoid calls to the Library Server database, assemble all parts to an item before adding it and conversely only get from the Library Server (or the Resource Manager) the data your application needs. If you are building a collection of child components, links, or parts, assemble the collection in the client application and add them all at once wherever possible.

5.8 Client applications

The Content Manager clients vary between particular Content Manager applications styles and even between functions within any one system. Content Manager provides sample applications that demonstrate many functions that are available in a Content Manager system. The sample applications can also help developers in the use of the APIs and creating custom Content Manager clients.

The Windows client and the eClient, which provide production-ready application clients for Content Manager, are generalized applications that have a wide range of capabilities and applications. They are designed for maximum flexibility and assume very little about the setup of the Library Server to which they will connect. They do not take any short cuts or assume the existence of the components that might be required for any one particular implementation. As a consequence, they might not be a fast or efficient as a custom-built application client. A custom client application can take advantage of a predefined data model and make assumptions about the data that have been enforced elsewhere in the application.

For example, the system designer knows that folders are created only when they contain a document and documents cannot be deleted, so the application can assume that there will be a document contained by that folder and jump straight to it. The generic, out-of-box clients cannot make this assumption, because it might not be true in other customer situations. These clients must determine whether there is a contained document, and if so, assume that there might be multiple documents and have to deal with them all. This can take a little additional time and create additional load for the server.

To develop clients specific to a business requirement, we provide some general development rules for better performance:

- Call the right APIs. This might seem intuitive but is an important point. A wide range of function calls and designs is available to the developer. Refer to the code samples and test your application against a database of a size that will be representative of the real system. In our testing scenario, we were able to improve performance of one particular function by choosing a different set of APIs and ways to deliver that function. The response time was reduced by more than half — generating the biggest performance gain in the performance tuning process.
- Minimize the number of calls to the server. Refer to 5.7.5, "Network and Library Server" on page 136. This reduces the network delays and the workload on the servers. In addition, wherever feasible, cache data and definitions on the client.
- Request only the data required. Get only the indexing data if you do not need to view the document; get only the PID if you do not need the indexing data. When retrieving data from the Content Manager servers, get only the data you need by specifying the retrieval option most appropriate for the task at hand. See Table 5-4 for a list of options. The table is taken from the sample code supplied with the Content Manager connector for Enterprise Information Integrator.
- In three-tier designs, offload as much work from the mid-tier servers to the end client as possible. Where possible, allow the viewer applet to perform the image rendering on the client.
- Deliver the items from the Resource Manager to the client directly wherever possible.

Table 5-4 Retrieve options

Retrieval	Option constant	Explanation
PID information only	`DK_CM_CONTENT_IDONLY`	Useful in Query/Search. Query will return collection with empty DDOs but complete PID information.
Attributes only	`DK_CM_CONTENT_ATTRONLY`	Retrieves only attributes.
Children	`DK_CM_CONTENT_CHILDREN`	Retrieve the child components.

Retrieval	Option constant	Explanation
One level	`DK_CM_CONTENT_ONELEVEL`	Retrieve one level of information, children, and attributes.
Outbound links	`DK_CM_CONTENT_LINKS_OUTBOUND`	Retrieve outbound link information.
Inbound links	`DK_CM_CONTENT_LINKS_INBOUND`	Retrieve inbound link information.
Item tree	`DK_CM_CONTENT_ITEMTREE`	Retrieve entire tree of information, including all metadata / attributes, children and sub-children, links.
Item tree without links	`DK_CM_CONTENT_ITEMTREE_NO_LINKS`	Retrieve the entire tree of information, but save time and increase performance by not checking for links.
No resource content	`DK_CM_CONTENT_NO`	If it is a resource item, do not retrieve the Resource object content.
Resource content	`DK_CM_CONTENT_YES`	If it is a resource item, retrieve the resource object content.
Only resource content	`DK_CM_CONTENT_ONLY`	If it is a resource item, retrieve only the resource object content.
Check out item	`DK_CM_CHECKOUT`	In addition, check out / lock the item.
Latest version	`DK_CM_VERSION_LATEST`	If versioning is enabled, retrieve the latest version, despite what the specific object the PID information specifies.

5.8.1 The OO API layer component of the client

New in Content Manager Version 8 is the use of the DB2 client as the connection from the client or mid-tier server to the Library Server. In many earlier implementations of the Content Manager, the DB2 client was installed alongside the Content Manager client to support other functions required by the client. Now the Content Manager client can take advantage of the DB2 communications capability. The Content Manager client OO APIs convert the API calls to the database calls. This removes the multiple conversions that previously occurred in the earlier Information Integrator for Content (EIP) client APIs.

The major redesign of Content Manager Version 8 takes advantage of database stored procedure calls, which reduce the amount of back-and-forth communication between the client application or mid-tier application and the database. In a single stored procedure request, the client can initiate a long series of database queries and changes that might otherwise have taken many client-to-database requests. This provides a major performance boost, especially over low-bandwidth connections. Although this reduces the workload, it is still good to avoid any unnecessary client requests and data transfer to or from the servers.

To monitor the interactions with the database, you can run the DB2 CLI or JDBC traces from the client or mid-tier server and see the calls and response times.

By default, these facilities are disabled and use no additional computing resources. When enabled, the trace facilities generate one or more text log files whenever an application accesses the appropriate driver (DB2 CLI or DB2 JDBC). These log files provide detailed information about:

- The order in which CLI or JDBC functions were called by the application
- The contents of input and output parameters passed to and received from CLI or JDBC functions
- The return codes and any error or warning messages generated by CLI or JDBC functions

DB2 CLI and DB2 JDBC trace file analysis can benefit application developers in a number of ways. First, subtle program logic and parameter initialization errors are often evident in the traces. Second, DB2 CLI and DB2 JDBC traces might suggest ways of better tuning an application or the databases it accesses.

The configuration parameters for both DB2 CLI and DB2 JDBC traces facilities are read from the DB2 CLI configuration file db2cli.ini. By default, this file is located in the \sqllib path on the Windows platform and the /sqllib/cfg path on UNIX platforms. You can override the default path by setting the DB2CLIINIPATH environment variable. On the Windows platform, an additional db2cli.ini file might be found in the user’s profile (or home) director if any user-defined data sources are defined using the ODBC Driver Manager. This db2cli.ini file will override the default file.

To view the current db2cli.ini trace configuration parameters from the command line processor, issue the following command:

```
db2 GET CLI CFG FOR SECTION COMMON
```

There are three ways to modify the db2cli.ini file to configure the DB2 CLI and DB2 JDBC trace facilities:

- Use the DB2 Configuration Assistant if it is available.
- Manually edit the db2cli.ini file using a text editor.
- Issue UPDATE CLI CFG command from the command line processor.

For example, the following command issued from the command line processor updates the db2cli.ini file and enables the JDBC tracing facility:

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING jdbctrace 1
```

Important: Disable the DB2 CLI and DB2 JDBC trace facilities when they are not needed. Unnecessary tracing can reduce application performance and might generate unwanted trace log files. DB2 does not delete any generated trace files and will append new trace information to any existing trace files.

It is best to perform these traces on a client or mid-tier server that is physically separated from the Library Server and Resource Manager servers. These trace files are common to all client activities, so they can include server-side SQL and JDBC calls from the Resource Manager Web application to the Resource Manager database.

For more about JDBC and CLI tracing, refer to the DB2 Information Center at:

http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp

The trace information enables you to see the stored procedures called by the application and the time it takes to service those requests. From this data, you can get a clear picture of the interactions with the Library Server and identify areas of poor performance.

Part 3

Performance tuning details

In this part, we summarize the best practices for tuning a Content Manager system for performance using check lists and a flowchart. For details, we cover the tuning of the operating system, DB2, WebSphere Application Server, and Tivoli Storage Manager for Content Manager. In most cases, for each parameter tuning, we describe what the parameter is used for, the default value if any, how to view, set it, or update it, and our recommendation based on our experience.

6

Best practices for Content Manager system performance

In this chapter we cover a variety of performance tuning best practice tips for Content Manager. We look at common performance tuning considerations; operating system tuning for AIX, Linux, and Windows; network tuning; Library Server and Resource Manager tuning; and application tuning.

These best practice steps are assembled into a convenient checklist for you to follow.

6.1 Introduction

Tuning a Content Manager system involves tuning the operating system, DB2, WebSphere Application Server, and Tivoli Storage Manager. Some of the tuning and configuration can be done during or immediately after the system installation. It is very important to include operating system tuning considerations before the system goes into production. Some other tasks should be performed during regular maintenance tasks.

Proper performance tuning is more than a solution to a performance problem. With proper monitoring, performance tuning can be addressed during regular maintenance schedules, minimizing performance impact on the system users.

Attention: Most of the material covered in this chapter is extracted from the following publications, written by the Content Manager Performance Team:

- *IBM Content Manager v8.3 Performance Tuning Guide*

  http://www.ibm.com/support/docview.wss?uid=swg27006452
- *CMv8.3 Performance Monitoring Guide*

  http://www.ibm.com/support/docview.wss?uid=swg27006451
- *CMv8.3 Performance Troubleshooting Guide*

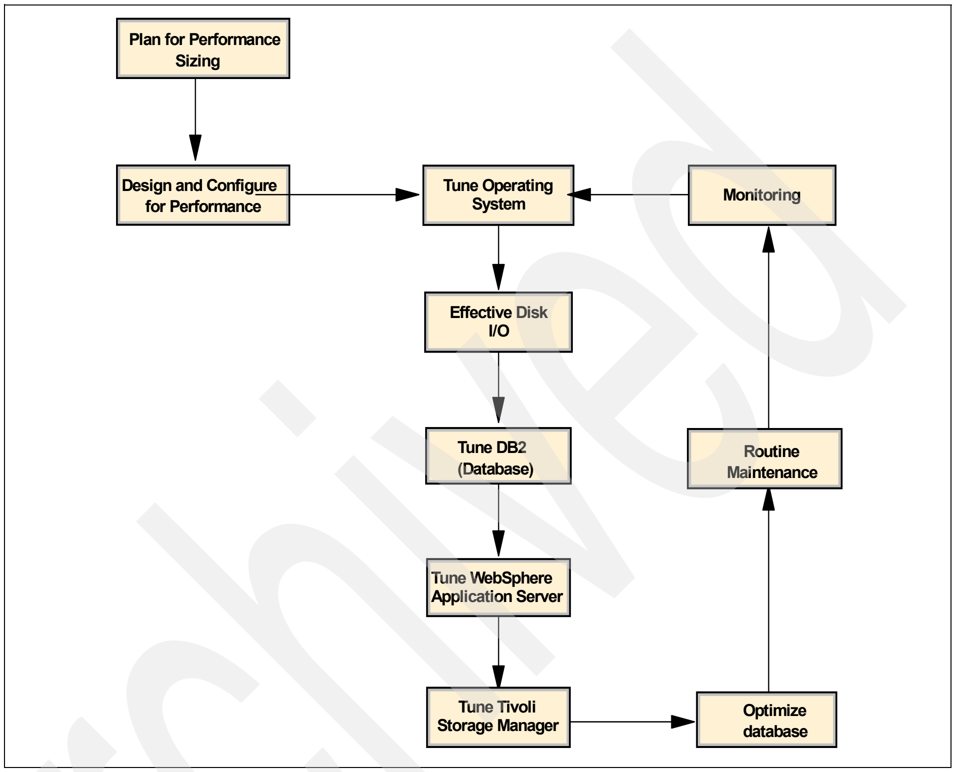
  http://www.ibm.com/support/docview.wss?uid=swg27006450

6.2 Best practices for system performance

This checklist includes some of the most common applicable performance tuning recommendations from the Content Manager Performance Team at the Silicon Valley Lab and services practitioners in the field. The tasks presented here are the most common suggestions for an initial install. These tasks are not all of the performance tuning tips available, but they are the most common and considered the best practice steps. The tasks are covered in detail in their respective chapters in this book. We do not cover all operating systems and databases specifically. Keep in mind that many of the principals and methodologies are still applicable.

Figure 6-1 on page 145 shows the performance methodology over the life cycle of a Content Manager system we followed when designing the best practices checklist. Even though the assumption is that the best practices are followed during an initial install and configuration, these best practices can be implemented at any time.

After initial installation, implementing routine maintenance and monitoring can start the performance tuning process again. This practice can help to ensure that your Content Manager system is always being tuned for performance to ensure that the user community is not affected by performance problems.



Plan for Performance Sizing
Design and Configure for Performance
Tune Operating System
Monitoring
Effective Disk I/O
Tune DB2 (Database)
Routine Maintenance
Tune WebSphere Application Server
Tune Tivoli Storage Manager
Optimize database

Figure 6-1 Content Manager best practices performance methodology

6.2.1 AIX

Table 6-1 provides a line-item overview of steps to be followed, with links to the detailed descriptions in their respective chapters.

Table 6-1 Best practice checklist in an AIX environment

	AIX best practice checklist
	Plan for performance - Chapter 4, “Planning for performance” on page 87
	Design and configure for performance - Chapter 5, “Designing and configuring for performance” on page 99

	AIX best practice checklist
	Adjust maximum number of processes - 7.2.1, “Adjust maximum number of PROCESSES allowed per user” on page 152
	Use JFS2 and EJFS for logical volumes and file systems - 7.2.2, “Use JFS2 and EJFS for logical volumes and file systems” on page 153
	Configure DB2 instance owners ulimit values - 7.2.3, “Check the default values in ulimit” on page 154
	Set all machines to full duplex - 7.2.5, “Set all machines to full duplex” on page 155
	Set MTU size to maximum supported by LAN - 7.2.6, “Set MTU size to the maximum supported by LAN” on page 155
	Configure effective disk I/O usage - 7.2.4, “Configure disk I/O for effective usage” on page 155
	Spread DB components over multiple disks - 8.2.1, “Spread the database components over multiple disks” on page 173
	Create separate instances during installation - 8.2.3, “Create separate instances during installation” on page 176
	Set default database location - “Change the database default location” on page 173
	Set default database log location - “Change log file location” on page 174
	Initial parameter tuning for Library Server database - 8.2.4, “Initial parameter tuning for Library Server database” on page 176
	Increase default buffer pool if more than 1 GB RAM - 8.5, “Buffer pool tuning” on page 196
	WebSphere Application Server: Tune Java heap size - 9.6.1, “Initial and maximum heap size for the JVM” on page 273
	Tune TSM buffer pool size - 10.2.1, “Self-tune buffer pool size (BUFPOOLSIZE)” on page 277
	Initial index configuration: Use runstats/rebind - “Database indexes” on page 118
	Regular maintenance: Use snapshots - 8.3.2, “Monitoring system performance using DB2 snapshot” on page 181

	AIX best practice checklist
	Regular maintenance: Use runstats/rebind - 8.3.3, “Keeping database statistics and execution plans up to date through runstats/rebind” on page 183
	Regular maintenance: Reorgch/reorg - 8.3.4, “Reorganizing tables through reorg” on page 184

6.2.2 Linux

Table 6-2 provides a line-item overview of steps to be followed, with links to the detailed descriptions in their respective chapters.

Table 6-2 Best practice checklist in a Linux environment

	Linux best practice checklist
	Plan for performance - Chapter 4, “Planning for performance” on page 87
	Design and configure for performance - Chapter 5, “Designing and configuring for performance” on page 99
	Set limits - 7.3.1, “Configure appropriate ulimit” on page 157
	Configure shared Memory - 7.3.2, “Configure shared memory” on page 157
	Configure semaphores - 7.3.3, “Configure semaphores settings” on page 158
	Configure message queues - 7.3.4, “Configure message queues” on page 158
	Tune network - 7.3.5, “Tune network” on page 159
	Tune file systems for external storage- 7.3.6, “Tune file systems for external storage” on page 160
	Configure effective disk I/O usage - 7.2.4, “Configure disk I/O for effective usage” on page 155
	Spread DB components over multiple disks - 8.2.1, “Spread the database components over multiple disks” on page 173
	Create separate instances during installation - 8.2.3, “Create separate instances during installation” on page 176

	Linux best practice checklist
	Configure default database location - “Change the database default location” on page 173
	Change database log location - “Change log file location” on page 174
	WebSphere Application Server: Configure Java heap size - 9.6.1, “Initial and maximum heap size for the JVM” on page 273
	Configure TSM buffer pool size - 10.2.1, “Self-tune buffer pool size (BUFPOOLSIZE)” on page 277
	Initial index configuration: Use runstats/rebind - “Database indexes” on page 118
	Regular maintenance: Use snapshots - 8.3.2, “Monitoring system performance using DB2 snapshot” on page 181
	Regular maintenance: Use runstats/rebind - 8.3.3, “Keeping database statistics and execution plans up to date through runstats/rebind” on page 183
	Regular maintenance: Use reorgch/reorg - 8.3.4, “Reorganizing tables through reorg” on page 184

6.2.3 Windows

Table 6-3 provides a line-item overview of steps to be followed, with links to the detailed descriptions in their respective chapters.

Table 6-3 Best practice checklist in a Windows environment

	Windows best practice checklist
	Plan for performance - Chapter 4, “Planning for performance” on page 87
	Design and configure for performance - Chapter 5, “Designing and configuring for performance” on page 99
	Disable Indexing service - 7.4.1, “Disable Indexing Service” on page 162
	Disable computer browser service - 7.4.2, “Disable Computer Browser Service” on page 162
	Disable Internet information service - 7.4.3, “Disable Internet Information Service” on page 163

	Windows best practice checklist
	Disable 8.3 file name creation service - 7.4.4, “Disable 8.3 file name creation (NtfsDisable8dot3NameCreation)” on page 163
	Check memory - 7.4.5, “Check memory” on page 164
	Tune network - 7.4.6, “Tune Windows network” on page 164
	Set boot.ini parameters - 7.4.8, “Set the /3GB parameter option in boot.ini” on page 166
	Configure effective disk I/O usage - 7.4.7, “Configure disk I/O for effective usage” on page 166
	Spread DB components over multiple disks - 8.2.1, “Spread the database components over multiple disks” on page 173
	Create separate instances during installation - 8.2.3, “Create separate instances during installation” on page 176
	Change default database location - “Change the database default location” on page 173
	Change default database log location - “Change log file location” on page 174
	WebSphere Application Server: Configure Java heap size - 9.6.1, “Initial and maximum heap size for the JVM” on page 273
	Tune TSM buffer pool size - 10.2.1, “Self-tune buffer pool size (BUFPOOLSIZE)” on page 277
	Initial index configuration: Use runstats/rebind - “Database indexes” on page 118
	Regular maintenance: Use snapshots - 8.3.2, “Monitoring system performance using DB2 snapshot” on page 181
	Regular maintenance: Use runstats/rebind - 8.3.3, “Keeping database statistics and execution plans up to date through runstats/rebind” on page 183
	Regular maintenance: Use reorgch/reorg - 8.3.4, “Reorganizing tables through reorg” on page 184

6.2.4 Maintenance

After a Content Manager system has been designed, installed, and configured, it is important to perform maintenance tasks regularly.

By maintaining a Content Manager system properly and in a timely manner, you get the best possible performance from it over time, as well as potentially avoid problems that can manifest themselves as system-endangering situations.

The checklist shown in Table 6-4 provides a line-item overview of the maintenance steps to be followed, with links to the detailed descriptions in their respective chapters.

Table 6-4 Best practice for performance maintenance checklist

	Maintenance checklist
	Optimize server databases using runstats/rebind - “runstats and rebind database tables” on page 398
	Optimize server databases using reorgchk - “reorgchk” on page 400
	Monitor LBOSDATA directory size - 14.3, “Monitoring LBOSDATA directory size” on page 403
	Manage staging directory space - 14.4, “Managing staging directory space” on page 406
	Remove entries from the events table - 14.5, “Removing entries from the events table” on page 409
	Remove log files - 14.6, “Removing log files” on page 410
	Manage Resource Manager utilities and services - 14.7, “Managing Resource Manager utilities and services” on page 411

7

Tuning operating systems for Content Manager

This chapter covers common performance tuning considerations for the operating system that Content Manager runs on. The operating systems that we cover in this chapter are AIX, Linux, and Windows, with additional information for Windows 2003 Enterprise Edition. Also included in the operating system are the network tuning options.

7.1 Introduction

Tuning a Content Manager system involves tuning the operating system. Most of the tasks performed by Content Manager depend directly on the operating system functionalities and capabilities. It is very important to include operating system tuning, preferably before the system goes into production.

Attention: Most of the material covered in this chapter is extracted from the following publications, written by the Content Manager Performance Team:

- *IBM DB2 Content Manager V8.3 Performance Tuning Guide*

  http://www.ibm.com/support/docview.wss?uid=swg27006452
- *Performance Tuning Guide for Content Manager V8.3 on Linux (x86)*

  http://www.ibm.com/support/docview.wss?uid=swg27007130

7.2 Tuning AIX operating system

Certain configuration changes can be made to increase performance of a Content Manager system running on an AIX platform. This section covers the changes you can make to the OS to maximize your system performance:

- Adjust maximum number of PROCESSES allowed per user.
- Use JFS2 and EJFS for logical volumes and file systems.
- Check the default values in ulimit.
- Configure disk I/O for effective usage.

Tuning the network improves Content Manager system performance. In this section, we also cover the changes you can make to the network to maximize your system performance:

- Set all machines to full duplex.
- Set MTU size to the maximum supported by LAN.

Attention: For more information about AIX performance tools and tuning, refer to *AIX 5L Practical Performance Tools and Tuning Guide*, SG24-6476.

7.2.1 Adjust maximum number of PROCESSES allowed per user

Increase the allowable number of processes by user from 1024 to a higher value depending on how many users will be connected to your system. It is important to update this value due to the large number of processes created on the database instance owners.

Default value

1024

How to do it

1. Log in as root.
2. Type `smitty`.
3. Click **System Environments** → **Change/Show Characteristics of Operating System**.
4. Change the value for **Maximum number of PROCESSES allowed per user** to a higher value.

Our recommendation

For a system with a large number of concurrent users, you might need to increase it up to as high as 8192. Check the number of processes that are running in your system and the number of the concurrent users in your system, and increase this number accordingly.

7.2.2 Use JFS2 and EJFS for logical volumes and file systems

For AIX 5L™, define your logical volumes and file systems using Journaled File System 2 (JFS2) and Enhanced Journaled File System (EJFS). This is necessary to manage the high volume of documents that Content Manager creates and manages on the same file system.

Enhanced Journaled File System (EJFS) uses database journal techniques to keep its structure consistent. This prevents damage to the file system if the system is halted abnormally. EJFS supports both 32-bit and 64-bit file system semantics. It supports 4-petabyte file systems and dynamically allocates inodes so that disk space is not wasted on unused inodes.

Journaled File System 2 (JFS2) is an enhanced and updated version of the Journaled File System (JFS) on AIX V4.3, and previous releases. JFS2 provides a robust, quickly restartable, transaction-oriented, log-based, scalable byte-level file system implementation for AIX environments. JFS2 features also include extent-based allocation, sorted directories, and dynamic space allocation for file system objects. Although tailored primarily for the high throughput and reliability requirements of servers, JFS2 is also applicable to client configurations where performance and reliability are desired.

Our recommendation

Verify all operating system issues before starting the installation process. Plan ahead. Make sure that the use of EJFs or JFS2 is needed in your business environment before proceeding.

7.2.3 Check the default values in ulimit

The **`ulimit`** command sets and reports user process resource limits, as defined in the /etc/security/limits file. This file contains these default limits:

```
fsize = 2097151
core = 2097151
cpu = -1
data = 262144
rss = 65536
stack = 65536
nofiles = 2000
```

During installation, DB2 creates an entry in the limits file and changes the defaults for the instance owner (user db2inst1) with the following values:

```
db2inst1:
fsize = -1
core = -1
data = 491519
rss = -1
stack = 32767
nofiles = 2000
```

How to view or set

To view the actual values, you can either edit the file (`/etc/security/limits`) or use the **`ulimit -a`** command.

To change the values for the number of files and the size of the files, you can either edit the file or use commands such as:

```
ulimit -n 2000
ulimt -f -1
```

For large SMP machines with clones, issue the **`ulimit -unlimited`** command:

Our recommendation

Be sure that all values for the db2inst1 user are in the limits file. If not, we recommend you to add it.

7.2.4 Configure disk I/O for effective usage

For an I/O bound system, you can reduce I/O contention by distributing I/O to multiple disks. We recommend installing the following items to separate physical disks if the resource is available:

- Library Server database
- Library Server database log
- Resource Manager database
- Resource Manager database log
- Resource Manager LBOSDATA
- Resource Manager staging area
- JFS logs
- Content Manager log file

Consider how your disks are attached to the system. Do not move the bottleneck from the disk to the disk controller card. Distribute the load over the disk controller cards.

7.2.5 Set all machines to full duplex

As part of network tuning, always set your machines on the network to the same value (100 Mbps - full duplex) and be sure that your hub/switch is set to the same value.

NIC driver settings for auto detection of line speed and direction can sometimes be misleading. Trouble can sometimes be avoided by explicitly setting the line speed and direction and not trusting auto-detection settings.

7.2.6 Set MTU size to the maximum supported by LAN

If your traffic is predominantly local, set MTU size to the maximum amount that is supported by your LAN. This minimizes the fragmentation of packets that are exchanged by local systems. The offsetting cost is fragmentation in gateways that connect your LAN to other LANs with smaller MTUs.

On AIX, use the `no` command to configure network attributes to get the best result for data transfers. The `no` command sets or displays current network attributes in the kernel. This command operates only on the currently running kernel. The command must be run again after each startup, or after the network has been configured. Whether the command sets or displays an attribute is determined by the accompanying flag.

Attention: Use this command with care. The **no** command does not perform range checking. It accepts whatever values you use for variables. If used incorrectly, the **no** command can cause your system to become inoperable.

This is a sample setting (provided by the Content Manager Performance Team) of a large AIX performance test server:

```
no –o sb_max=6192000
no –o tcp_sendspace=4096000
no –o tcp_recvspace=4096000
no –o udp_sendspace=65536
no –o udp_recvspace=655360
no –o rfc1323=1
no –o ipqmaxlen=150
no –o clean_partial_conns=true
```

7.3 Tuning Linux operating system

There are configuration changes you can make to increase performance of a Content Manager system running on the Linux platforms. On the Linux platforms, this is particularly important because many default settings do not take into account relatively large SMP systems, usually with more memory.

This section covers the following changes you can make to maximize your system performance:

- Configure appropriate ulimit.
- Configure shared memory.
- Configure semaphores settings.
- Configure message queues.
- Tune network.
- Tune file systems for external storage.
- Configure disk I/O for effective usage.

Most of the parameters mentioned apply to both Red Hat Enterprise Linux and SUSE Linux Enterprise Server 8.

7.3.1 Configure appropriate ulimit

This specifies several maximums allowed per user, but changes affect all users on the system. Of particular importance are open files, stack size, and number of processes. If any of these numbers is set too low, errors might appear in the DB2 databases, Resource Manager, Web server, or operating system error logs. These values are all maximums, so they can be set arbitrarily large or to unlimited. However, this might remove some safeguards against runaway processes or system attacks.

To display the current value and list the available flags, use the **ulimit –a** command.

To change to a different value, use the command, with -1 for unlimited.

For a database application, we recommend setting the ulimit to at least 8000.

```
ulimit –n <new_value>
ulimit –n -1
```

You may also change the settings in the /etc/security/limits file.

7.3.2 Configure shared memory

Under a large workload, the DB2 database tuning for ASLHEAPSZ might require a large value for the DB2 registry variable DB2_FMP_COMM_HEAPSZ. This in turn might require a greater amount of shared memory allowed by the operating system. To accomplish this, change the value of /proc/sys/kernel/shmmax.

```
echo 268435456 > /proc/sys/kernel/shmmax
```

This value takes effect immediately, but the command must be added to a boot script such as rc.local to make it permanent. You will know if any of these values are too small because errors will appear in db2diag.log, the database instance will not start, or database connections after a certain point will be dropped.

Fortunately, DB2 UDB Version 8.2 has a new feature that checks the values of the kernel.semmni, kernel.msgmni, and kernel.shmmax parameters when you enter the **db2start** command, and changes them for you if the current values are not optimal. This new feature makes these changes:

- The semmni kernel parameter is changed to 1024.
- The msgmni kernel parameter is changed to 1024.
- The shmmax kernel parameter is changed to 268435456 (32-bit) or 1073741824 (64-bit).

For example, after you issue the **db2start** command for the first time, you should receive output similar to the following messages in your db2diag.log file:

```
ADM0506I DB2 has automatically updated the "semmni" kernel
parameter from "128" to the recommended value "1024".
2004-07-31-16.38.59.074791 Instance:db2inst1
Node:000
PID:15996(db2sysc) TID:8192 Appid:none
base sys utilities sqlesysc_main Probe:9
ADM0506I DB2 has automatically updated the "msgmni" kernel
parameter from "16" to the recommended value "1024".
2004-07-31-16.38.59.076916 Instance:db2inst1
Node:000
PID:15996(db2sysc) TID:8192 Appid:none
base sys utilities sqlesysc_main Probe:9
ADM0506I DB2 has automatically updated the "shmmax" kernel
parameter from "33554432" to the recommended value "268435456".
2004-07-31-16.39.01.262594 Instance:db2inst1
Node:000
PID:15994(db2star2) TID:8192 Appid:none
base sys utilities startdbm Probe:911
ADM7513W Database manager has started.
```

7.3.3 Configure semaphores settings

Linux default kernel tuning for semaphores is not meant for enterprise applications such as DB2. On RHEL3 for example, it is:

```
# more /proc/sys/kernel/sem
250 32000 32 1024
```

Using the default values might cause the application to run out of semaphores. To increase semaphore settings, use the following command:

```
echo 500 512000 64 2048 > /proc/sys/kernel/sem
```

The parameters are, in order:

SEMMSL	Semaphores per ID
SEMMNS	(SEMMNI*SEMMSL) Max semaphores in system
SEMOPM	Max operations per semaphore call
SEMMNI	Max semaphore identifiers

7.3.4 Configure message queues

Message queues are used by the Linux operating system and by databases to facilitate communications among processes. Databases might run out of message queues for large numbers of concurrent connections under the default Linux message queue settings.

To increase the number of queues systemwide:

```
echo 2048 > /proc/sys/kernel/msgmni
```

To set the maximum size of a queue, use:

```
echo 64000 > /proc/sys/kernel/msgmax
```

As with other changes under /proc, which apply only for the current session, you must add the command to rc.local or the equivalent settings in sysctl.conf to make them permanent.

7.3.5 Tune network

Based on the tested, privately switched 1 GB Ethernet, the default parameters from Linux sufficed. Because tuning varies based on network load and topology, we offer some important parameter details. Network tuning is especially critical for applications that store or retrieve large amounts of data from the Resource Manager, especially when FTP tests for large files show low throughput, or when netstat reports high numbers dropped or error packets.

To get information about the current network adapter settings, run:

```
mii-tool –v or ethtool <adapter name>
```

Depending on the network topology (switches, adapters, or both), the adapter should probably be running in full duplex mode. In this case, full duplex can be set by typing:

```
mii-tool -F 100baseTx-FD
```

The following kernel network settings might be helpful. Each can be set with the command:

```
sysctl –w parameter=value
```

Table 7-1 lists the TCP tuning to support a large number of multiple connections.

Table 7-1 Network tuning for both RHEL and SUSE

Tuning parameter	Recommended value	Description of impact
net.ipv4.tcp_tw_reuse	1	Reuse sockets in the time-wait state.
net.ipv4.tcp_tw_recycle	1	Fast recycling of TIME-WAIT sockets.

Tuning parameter	Recommended value	Description of impact
net.core.wmem_max	8388608	Increase the maximum write buffer queue size.
net.core.rmem_max	8388608	Increase the maximum read buffer queue size.
net.ipv4.tcp_rmem	“4096 87380 8388608”	Set the minimum, initial, and maximum sizes for the read buffer. This maximum should be less than or equal to the value set in net.core.rmem_max.
net.ipv4.tcp_wmem	“4096 87380 8388608”	Set the minimum, initial, and maximum sizes for the write buffer. This maximum should be less than or equal to the value set in net.core.wmem_max.

7.3.6 Tune file systems for external storage

Several UNIX file system parameters should be modified according to your expected workload and the dynamics of your environment. These settings enable you to take advantage of caching, throughput optimization, and performance of UNIX attached external storage devices.

One of the most important choices is which file system to use. In the test environment, *ext3* is used for Red Hat and *reiser* for SUSE. These file systems offer excellent throughput and represent the file systems in use by much of the Linux community. However, there are many other good options such as JFS and XFS, especially if there is a special need such as allowing for huge files or sharing data for a high-availability environment.

After choosing the file system, several kernel and mounting options could affect it. One such kernel setting is the elevator algorithm. Tuning the elevator algorithm helps the system balance the need for low latency with the need to collect enough data to efficiently organize batches of read and write requests to the disk. The elevator algorithm can be adjusted with the following command:

```
elvtune -r 1024 -w 2048 /dev/sda
```

The parameters are: read latency (-r), write latency (-w), and the device affected. Red Hat recommends using a read latency half the size of the write latency (as shown). As usual, to make this setting permanent, add the **elvtune** command to the /etc/rc.d/rc.local script.

For tuning mount options, the *noatime* mount option can improve I/O performance in some cases. For example, directories with many files might benefit from avoiding updates for the access time of the file. When noatime is not specified, every time a file is accessed, its inode information is updated to reflect the last access time that incurs a write to the file system metadata. To change the setting, edit the file system entry in /etc/fstab, appending noatime to the existing entry or replacing .defaults. With all I/O stopped, the change can be applied by typing:

```
umount <mount point>
mount <mount point>
```

Another mounting consideration is how to journal on journal file systems. Journalling helps guarantee the consistency of data written to disk and reduces recovery time after a crash. The default option, data=ordered, offers the best access time while ensuring consistency. In storage devices with backup power or in situations where data consistency is not critical, data=writeback offers faster access time at the cost of consistency after a crash.

7.3.7 Configure disk I/O for effective usage

For an I/O bound system, you can reduce I/O contention by distributing I/O to multiple disks. We recommend installing the following items to separate physical disks if the resource is available:

- Library Server database
- Library Server database log
- Resource Manager database
- Resource Manager database log
- Resource Manager LBOSDATA
- Resource Manager staging area
- Content Manager log file

Consider how your disks are attached to the system. Do not move the bottleneck from the disk to the disk controller card. Distribute the load over the disk controller cards.

7.4 Tuning the Windows operating system

You can make certain configuration changes to increase performance of a Content Manager system running on the Windows platform. This section covers the changes you can make to maximize your system performance, and the areas you need to check to ensure optimal system performance:

- Disable Indexing Service.
- Disable Computer Browser Service.

- Disable Internet Information Service.
- Disable 8.3 file name creation (NtfsDisable8dot3NameCreation).
- Check memory.
- Tune network.
- Configure disk I/O for effective usage.
- Windows 2003 Enterprise Edition /3GB boot.ini parameter.
- Windows 2003 Enterprise Edition /PAE boot.ini parameter.
- Windows 2003 Enterprise Edition unnecessary services.

7.4.1 Disable Indexing Service

Microsoft Windows 2000 Indexing Services is a search engine that enables users to perform full-text search of online sites using their browsers. Search results include Word, Excel®, Power Point, and HTML documents. To avoid overhead on your server from indexing all Content Manager files, turn off the Windows Indexing Service:

1. Click **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**.
2. Right-click **Indexing Service** → **Properties**.
3. Set Startup Type to **Disable**.
4. Reboot your machine for the setting to take effect.

7.4.2 Disable Computer Browser Service

The primary function of a Windows Browser Service is to provide a list of computers that share resources in a client's domain, along with a list of other domains and workgroup names across a wide area network (WAN). This list is provided to clients who view network resources with Network Neighborhood or the NET VIEW command. This is an additional workload for your server. Because Content Manager uses TCP/IP and not NetBIOs, it is better to turn off the Computer Browser Service:

1. Click **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**.
2. Right-click **Computer Browser** → **Properties**.
3. Set Startup Type to **Disable**.
4. Reboot your machine for the setting to take effect.

7.4.3 Disable Internet Information Service

If you do not use Windows Internet Information Server (IIS) as your HTTP server, it is better to turn off IIS to avoid problems with the IBM HTTP server that is installed with WebSphere:

1. Click **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**.
2. Right-click **IIS** → **Properties**.
3. Set Startup Type to **Disable**.
4. Reboot your machine for the setting to take effect.

7.4.4 Disable 8.3 file name creation (NtfsDisable8dot3NameCreation)

Every time you create a file with a long file name, NTFS creates a second file entry that has a similar 8.3 short file name.

If you have a large number of files (300,000 or more) in a folder, and the files have long file names with the same initial characters, the time required to create the files increases. The increase occurs because NTFS bases the short file name on the first six characters of the long file name. In folders with more than 300,000 files, the short file names start to conflict after NTFS uses all of the 8.3 names that are similar to the long file names. Repeated conflicts between a generated short file name and existing short file names cause NTFS to re-generate the short file name from six to eight times.

Default value

The default value for this parameter is zero (enable short name file creation).

How to view or set

To reduce the time required to create short name files, you can change the parameter value to 1 (Disable short name file creation). To do this:

1. Click **Start** → **Run**.
2. Type `regedit`. Press Enter.
3. Click **HKEY_LOCAL_MACHINE** → **SYSTEM** → **ControlSet001** → **Control** → **FileSystem**.
4. Change the value of NtfsDisable8dot3NameCreation from 0 to `1`.
5. Reboot your machine for the setting to take effect.

Our recommendation

Content Manager usually creates files with the first six characters that are the same. By disabling the 8.3 file name creation, you avoid the NTFS overhead of creating the short name files for Content Manager long-name (30 CHARS) files that it stored in the LBOSDATA area. From our experience, this improves system performance.

7.4.5 Check memory

Verify that you have enough physical memory for your Content Manager system. For a Library Server machine, you should have about 1 GB RAM plus 10 MB RAM for each Windows desktop client plus about 10 MB RAM for every 20 Web clients without DB2 Connection concentrator turned on. With DB2 Version 8.1 Connection concentrator turned on, allocate about 10 MB RAM for every 10 desktop clients.

7.4.6 Tune Windows network

For Windows servers, use both the network adapter properties and Windows registry to update the TCP layer parameters. Table 7-2 lists the registry entries, parameters, and recommended values. If your environment is uniformly Windows platform, make sure all of these parameters are set to the same values across all of your servers (that is, the Library Server, Resource Manager server, and eClient Server) to optimal network throughput. We recommend that you restart your server for these changes to take effect.

Table 7-2 Windows 2003 TCP tuning for CM

Registry entry	Parameter and value

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters`	`TcpTimedWaitDelay= 30 (decimal)`
	`MaxUserPort= 32768 (dec)`
	`KeepAliveInterval= 1 (dec)`

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\`	`TcpAckFrequency= 1`
`Tcpip\Parameters\Interfaces\ID`	`TcpMaxDataretranmissions= 5 (dec)`

Registry entry	Parameter and value
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AFD\Parameters	"EnableDynamicBacklog"=dword:00000001 "MinimumDynamicBacklog"=dword:00000020 "MaximumDynamicBacklog"=dword:00001000 "DynamicBacklogGrowthDelta"=dword:00000010

The descriptions of the tuned TCP parameters are as follows:

- TcpTimedWaitDelay: Determines the time (in seconds) that must elapse before TCP/IP can release a closed connection and reuse its resources. This interval between closure and release is known as the TIME_WAIT state or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP/IP can release closed connections faster and provide more resources for new connections. Adjust this parameter if your Content Manager application requires rapid release, the creation of the new connections, or an adjustment because of a low throughput caused by multiple connections in the TIME_WAIT state.
- MaxUserPort: Determines the highest port number that TCP/IP can assign when an application requests an available user port from the system.
- KeepAliveInterval: Determines how often TCP repeats keep-alive transmissions when no response is received.
- TcpAckFrequency: TCP/IP can be the source of some significant remote method delays. You can increase TCP performance by immediately acknowledging incoming TCP segments, in all situations.
- TcpMaxDataRetransmissions: Determines how many times TCP retransmits an unacknowledged data segment on an existing connection.
- MaxConnect Backlog: If many connection attempts are received simultaneously, increase the default number of pending connections that are supported by the operating system. Set all four parameters according to the table above, Enable DynamicBacklog, MinimumDynamicBacklog, MaximumDynamicBacklog, and DynamicBacklogGrowthDelta.

For more information about the network driver, refer to:

http://www.microsoft.com/whdc/device/network/taskoffload.mspx#EEAA

7.4.7 Configure disk I/O for effective usage

For an I/O bound system, you can reduce I/O contention by distributing I/O to multiple disks. We recommend installing the following items to separate physical disks if the resource is available:

- Library Server database
- Library Server database log
- Resource Manager database
- Resource Manager database log
- Resource Manager LBOSDATA
- Resource Manager staging area
- Content Manager log file

Consider how your disks are attached to the system. Do not move the bottleneck from the disk to the disk controller card. Distribute the load over the disk controller cards.

7.4.8 Set the /3GB parameter option in boot.ini

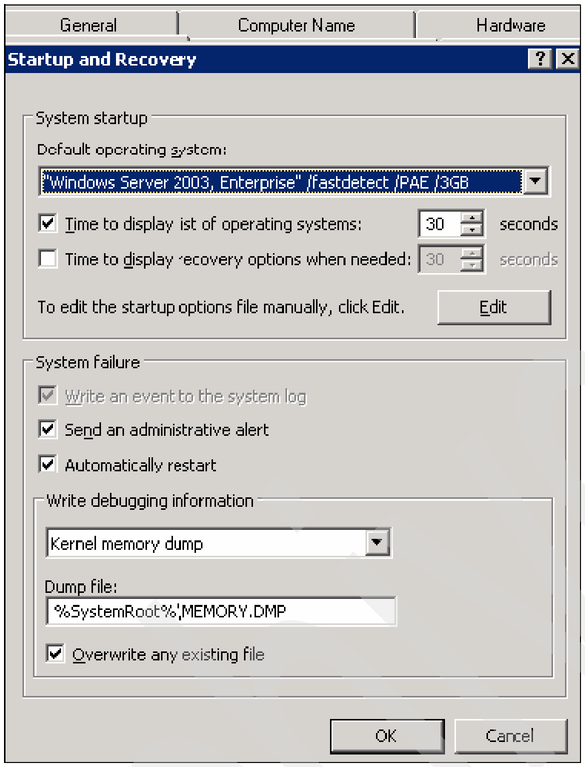
By default, the 32-bit editions of Windows can address a total of 4 GB of virtual address space. This is the limitation of the 32-bit architecture. Normally, 2 GB of this is reserved for the operating system kernel requirements (privileged-mode) and the other 2 GB is reserved for application (user-mode) requirements. Under normal circumstances, this creates a 2 GB per-process address limitation. For Content Manager, the user-mode process affected is db2syscs.exe, and for Oracle, the affected process is oracle.exe.

Windows provides a /3GB parameter to be added to the boot.ini file that reallocates 3 GB of memory to be available for user-mode applications and reduces the amount of memory for the system kernel to 1 GB. Both DB2 and Oracle can derive performance benefits from having large amounts of addressable memory available for creating agent processes that handle a large number of concurrent database connections.

To edit the boot.ini file to make this change, complete the following steps:

1. Open the System Control Panel.
2. Select the **Advanced** tab.
3. In the Startup and Recovery frame, click **Settings**.
4. Click **Edit**. Notepad opens to edit the current boot.ini file.
5. Edit the current ARC path to include the /3GB switch in the same line that has `/fastdetect`.
6. Restart the server for the change to take effect.

Figure 7-1 shows the default Startup and Recovery frame settings.



System startup

Default operating system:
"Windows Server 2003, Enterprise" /fastdetect /PAE /3GB

[✓] Time to display list of operating systems: 30 seconds

[ ] Time to display recovery options when needed: 30 seconds

To edit the startup options file manually, click Edit. Edit

System failure

[✓] Write an event to the system log

[✓] Send an administrative alert

[✓] Automatically restart

Write debugging information

Kernel memory dump

Dump file:
%SystemRoot%\MEMORY.DMP

[✓] Overwrite any existing file

Figure 7-1 Windows 2003 boot.ini settings

7.4.9 Set the /PAE parameter option in boot.ini

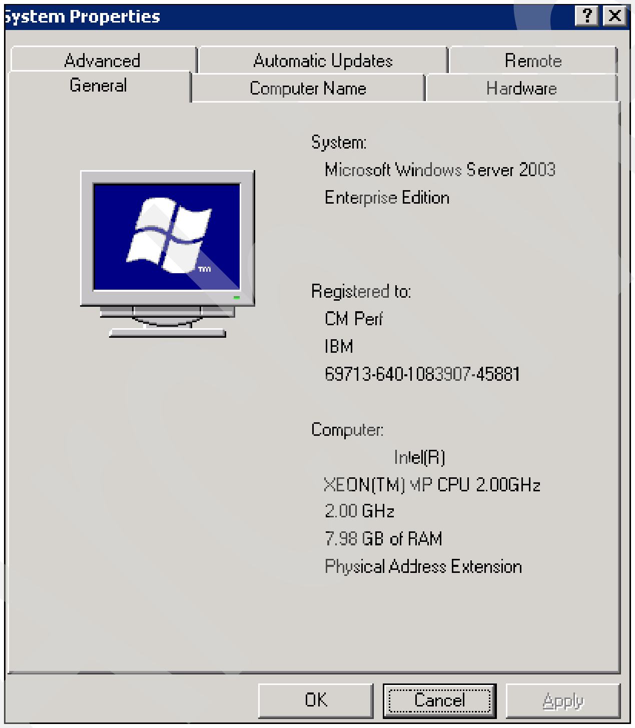
Many SMP servers come with more than 4 GB RAM, but the Windows OS might not see the total RAM if the /PAE switch is not set in the boot.ini file.

The native 32-bit architecture of the x86 processor allows a maximum addressable memory space of 4 GB. The Intel® Physical Address Extension (PAE) is a 36-bit memory addressing mode that allows 32-bit systems to address memory above 4 GB. PAE requires the appropriate hardware and operating system support to be implemented. Intel introduced PAE 36-bit physical addressing with the Intel Pentium® Pro processor. Windows has supported PAE since Windows NT® Server 4.0 Enterprise Edition, and it is supported with the Advanced and Datacenter Editions of Windows. Specify `./PAE.` in boot.ini to allow full use of 8 GB physical RAM.

To edit the boot.ini file to make this change, complete the following steps:

1. Open the System Control Panel.
2. Select the **Advanced** tab.
3. In the Startup and Recovery frame, click **Settings**.
4. Click **Edit**. Notepad opens to edit the current boot.ini file.
5. Edit the current ARC path to include the /PAE switch in the same line that has /fastdetect.
6. Restart the server for the change to take effect.

The amount of RAM should now be 8 GB in the System Properties settings, as in Figure 7-2.



System:
Microsoft Windows Server 2003
Enterprise Edition

Registered to:
CM Perf
IBM
69713-640-1083907-45881

Computer:
Intel(R)
XEON(TM) MP CPU 2.00GHz
2.00 GHz
7.98 GB of RAM
Physical Address Extension

Figure 7-2 Windows 2003 system summary

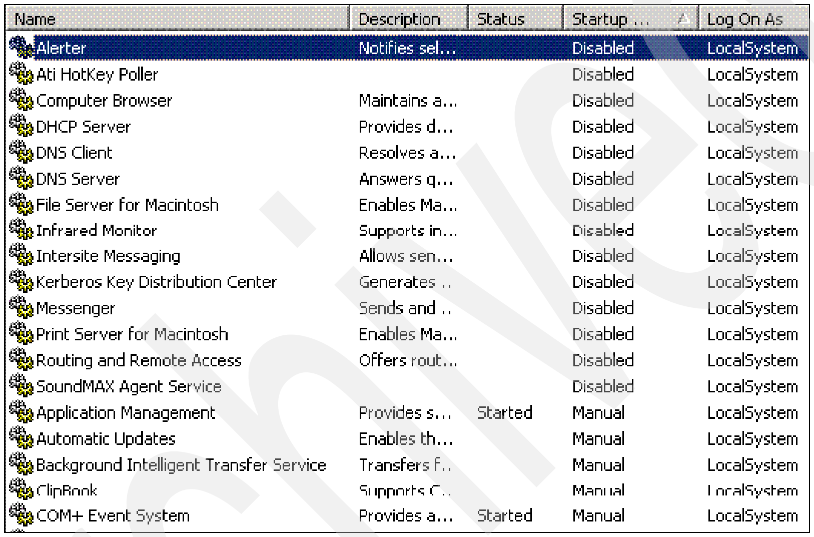
7.4.10 Disable unnecessary Windows services

When Windows is first installed, many services are enabled that might not be necessary for a particular server. Inexperienced users could also inadvertently add additional services that are not required for a given system. Each service requires system resources. It is best to disable non-required services to improve performance.

To view the services running in Windows, complete the following steps:

1. Right-click **My Computer** and select **Manage**.
2. Expand the **Services and Applications** icon.
3. Select the **Services** icon.

Figure 7-3 shows a sample display of disabled Windows services.



Name	Description	Status	Startup ...	Log On As
Alerter	Notifies sel...		Disabled	LocalSystem
Ati HotKey Poller			Disabled	LocalSystem
Computer Browser	Maintains a...		Disabled	LocalSystem
DHCP Server	Provides d...		Disabled	LocalSystem
DNS Client	Resolves a...		Disabled	LocalSystem
DNS Server	Answers q...		Disabled	LocalSystem
File Server for Macintosh	Enables Ma...		Disabled	LocalSystem
Infrared Monitor	Supports in...		Disabled	LocalSystem
Intersite Messaging	Allows sen...		Disabled	LocalSystem
Kerberos Key Distribution Center	Generates ..		Disabled	LocalSystem
Messenger	Sends and ..		Disabled	LocalSystem
Print Server for Macintosh	Enables Ma...		Disabled	LocalSystem
Routing and Remote Access	Offers rout...		Disabled	LocalSystem
SoundMAX Agent Service			Disabled	LocalSystem
Application Management	Provides s...	Started	Manual	LocalSystem
Automatic Updates	Enables th...		Manual	LocalSystem
Background Intelligent Transfer Service	Transfers f..		Manual	LocalSystem
ClipBook	Supports C..		Manual	LocalSystem
COM+ Event System	Provides a...	Started	Manual	LocalSystem

Figure 7-3 Sample display of disabled Windows services

You can sort the Services by clicking the **Startup** column. Table 7-3 shows a list of services that we recommend to disable or to set them to manual startup.

Table 7-3 Recommended services to disable to set to manual startup

Service	Default startup setting	Recommended setting
Alerter	Automatic	Disabled
Application Management	Manual	Disabled
Automatic Updates	Automatic	Manual
Background Intelligent	Automatic	Disabled
Clipbook	Disabled	Disabled
Computer Browser	Automatic	Disabled

Service	Default startup setting	Recommended setting
DHCP Server	Manual	Disabled
Distributed file system	Automatic	Disabled
Distributed link tracking client	Automatic	Disabled
Distributed transaction coordinator	Automatic	Manual
DNS Client	Automatic	Disabled
Error Reporting Service	Automatic	Disabled
Fax Service	Manual	Disabled
File Replication	Manual	Disabled
File Server for Macintosh	Manual	Disabled
Help and Support	Automatic	Disabled
HTTP SSL	Manual	Disabled
Infrared Monitor	Automatic	Disabled
Intersite messaging	Manual	Disabled
License Logging	Manual	Disabled
Logical Disk Manager	Automatic	Manual
Message Queuing	Automatic	Disabled
Messenger	Automatic	Disabled
Portable Media Serial Number Service	Manual	Disabled
Shell Hardware Detection	Automatic	Disabled
SNMP Service	Manual	Disabled
Windows Audio	Automatic	Disabled
Wireless Configuration	Automatic	Disabled

For more Windows Server 2003 tuning tips, refer to the IBM Redpaper *Tuning Windows Server 2003 on IBM eServer xSeries Servers*, REDP3943, at:

http://www.redbooks.ibm.com/redpieces/abstracts/redp3943.html

8

Tuning DB2 for Content Manager

This chapter covers the tuning of DB2 for Content Manager. It includes design and configuration setup, regular routine maintenance, SMS-to-DMS table space conversion instructions, buffer pool tuning, and parameter tuning.

8.1 Introduction

DB2 plays an extremely important role in Content Manager system performance. Library Server, the central focal point for the entire Content Manager system, is largely an extension of DB2. Performance tuning is affected by many factors, including the workload scenario and the system environment. For this reason, proper DB2 configuration setup, regular routine maintenance, and careful tuning of the buffer pools and the configuration parameters are the keys to a solidly performing Content Manager system. In the following sections, we cover these key areas in detail. Some of the recommendations might have been presented in previous chapters, but we decided to include all DB2-related performance tuning in one section for easy reference.

We have provided general examples of the related DB2 commands. Always check the product publications for the correct syntax as it applies to your situation:

- *IBM DB2 Universal Database Version 8.2 SQL Reference Volume 2*, SC09-4845
- *IBM DB2 Universal Database Version 8.2 Command Reference*, SC09-4828
- *IBM DB2 Universal Database Version 8.2 Administration Guide: Performance*, SC09-4821

8.2 Design and configuration setup

Design and configuration setup influences system performance. Without proper design and configuration setup, the system might never be able to perform at the expected level. In this section, we recommend a list of actions that you should consider taking to improve overall system performance. They include:

- Spread the database components over multiple disks.
- Customize CM databases deployment at install through DDL.
- Create separate instances during installation.
- Initial parameter tuning for Library Server database.
- Create attribute indexes.
- Create indexes for document routing.
- Additional configuration and tuning suggestions.

Attention: You must be an experienced DBA to perform the suggested tasks. If you are not familiar with what they are and how to do them, we do not recommend you experimenting with them.

8.2.1 Spread the database components over multiple disks

If possible, it is extremely useful to distribute major DB components such as table spaces and log files over all available physical disks. Properly configuring the database components over multiple disks avoids I/O contention.

Default values

By default, the Content Manager installation program creates the database, all database table spaces, and the log files on the same disk and file system. You should change the defaults in the Data Definition Language (DDL) during installation time or by using the administrative tools to change the parameters for the database components.

Change the database default location

During installation, the database is created on the default database path specified in the database manager configuration file (DFTDBPATH parameter). This is the C:\ drive for Windows or the /home/ file system for AIX. It is better if the database is separated from the drive or file system that contains the operating system. It is important to note that you can set the database destination path only *before* the database is created. After it is created, it cannot be moved without dropping and re-creating it.

Change the “create database” DDL to specify a drive (if you have more than just C:\) or path other than that used by the base operating system:

```
db2 create db <database name> on <new path> ....
```

An alternative to changing the “create database” DDL is to change the default database path specified in the database manager configuration file (DFTDBPATH parameter). Changing this parameter must be done after the instance is created but before any database is created:

```
db2 update dbm cfg using DFTDBPATH <new path>
```

Spread table spaces over multiple containers

In order to promote the distribution of data across multiple disks, thereby increasing I/O parallelism, the table spaces should be altered to add a container. After the new container is added, DB2 will rebalance the contents of the table space across all the containers belonging to the table space.

Use the **alter** command to add a file on another disk path. In the following examples, a 5120-page file on the d:\ drive is being added to a table space on Windows and a 2000-page file on file system cmdb1 is being added to a table space on AIX:

```
db2 alter tablespace <tsname> add (file 'd:\tsname1\<filename>' 5120)
db2 alter tablespace <tsname> add (file '/cmdb1/<filename>' 2000)
```

In these examples, `<tsname>` is the table space name.

Spreading the table spaces across multiple containers can also be achieved by converting the SMS table spaces to DMS. This procedure is discussed in 8.4, “SMS to DMS table space conversion” on page 188.

Change log file location

Even if the rest of the database components are not going to be spread across multiple disks, the database log files should be put on a different disk. The log files should be on a physical disk that does not have high I/O. This allows for efficient logging activity with a minimum of overhead such as waiting for I/O. To change the log file location, use the following command:

```
db2 update db cfg for <database name> using NEWLOGPATH = <new log path>
```

In this example, `<new log path>` is an existing path on another disk.

Our recommendation

Installing the databases, table spaces, and log files on different disks improves both performance and scalability by maximizing I/O concurrency. As soon as possible, try to adjust these components according to your system availability and capacity. Use as many disks as possible. Invest in multiple disks — it pays performance dividends. Divide your disks using the following rules:

- Separate and spread table spaces across multiple physical disks.
- Separate log files from the databases and put them on separate disks.

Separating the instances makes it possible to tune the Content Manager databases independently. This provides more flexible tuning by allowing, as an example, the Resource Manager database instance to be stopped and started without affecting the Library Server. This topic is discussed in 8.2.3, “Create separate instances during installation” on page 176.

8.2.2 Customize CM databases deployment at install through DDL

In the previous section, we showed how to spread database components to different physical disks using DB2 commands. An alternative way is to change the database installation scripts (the DDL files) at installation time. Content Manager gives you the flexibility to change the DDL files to customize your Library Server or Resource Manager database designs at install time. An example of the changes you can make include assigning multiple physical disks to a table space container.

Default values

The default installation script installs all databases onto the same disk and file system.

How to change

1. After Content Manager installation, drop the Library Server database (icmnlsdb) or the Resource Manager database (rmdb), or both, that were created during the installation process:

   ```
   db2 drop db icmnlsdb
   db2 drop db rmdb
   ```

2. Back up the DDL files that are located in the <icmroot>\config directory. The primary file names are:

icmlsdb2def.ddl	Library Server database, buffer pool, and table space definitions
icmplsct.ddl	Library Server table definitions
icmplsci.ddl	Library Server index definitions
icmrmcrdb.ddl	Resource manager server database, buffer pool, and table space definitions
icmrmcrtab.ddl	Resource manager server table and index definitions
cmbadm81.db2	Library Server federated administration table definitions
icmpwfcr.ddl	Library Server federated advanced workflow table definitions
icmjcr.ddl	Library Server JCR table definitions (reserved for future use)
icmjcrup.ddl	Library Server JCR table definitions (reserved for future use)

3. Edit the DDL files and reorganize the physical design as you planned.
4. After all DDL files are changed, re-create the databases by running the **Create Library Server Database** or **Create Resource Manager Database** utilities (or both). These utilities use the updated DDL files to create the new databases.

Our recommendation

This is an advanced DB2 task. You need to be very familiar with the DDL, table space, and container concepts in a DB2 environment. This task should be performed by an experienced DB2 database administrator (DBA) and Content Manager system administrator.

Remember, lay out the plan first. Think through before you start changing the content of the DDL. For your physical system layout, we recommend:

- Separate the operational system paging area from the other components.
- Separate the physical location of databases from each other.
- Separate the staging area from LBOSDATA.
- Separate the paging area from LBOSDATA.
- Separate the table space containers onto different disks.

For more information about DDLs, databases, table spaces, and containers, refer to the publications listed at the beginning of this chapter.

8.2.3 Create separate instances during installation

The default Content Manager installation uses one database instance for all databases. Therefore, if you install the Library Server and the Resource Manager database on the same AIX system, you will have two databases managed by one database instance (db2inst1):

- Library Server database (icmnlsdb)
- Resource Manager database (rmdb)

The advantage of this approach is that you avoid the administration work of managing multiple instances. The potential disadvantages of the single-instance approach are:

- Shared memory: A single DB2 instance on a 32-bit AIX can use only up to 1.7 GB of memory for database buffer pools, which means that all of the databases would have to share this memory.
- Resource contention.
- Stoppages to the instance affect all resident databases instead of just the intended database.

Because the database manager (instance) is a complete environment, the parameter settings apply to every database in that instance. We recommend creating one instance per database. For details about how to do this, refer to 6.7, "Creating and configuring DB2 instances" on page 177 in *Content Manager Version 8.1 Migration Guide for Multiplatforms*, SG24-6877.

8.2.4 Initial parameter tuning for Library Server database

You can improve performance when the database manager is started (cold starts) by having a pool of worker agents created during startup. These initial idle agents will be available to handle initial queries instead of being created at query time. This improves the performance of those initial queries.

The following commands (which are discussed in detail later in this chapter) will configure DB2 to initialize the Library Server's database processes:

```
db2 update dbm cfg using NUM_POOLAGENTS <n>
db2 update dbm cfg using NUM_INITAGENTS <n>
db2 update dbm cfg using MAXAGENTS <m>
```

In these examples:

- `<n>` is the expected number of concurrent users, and
- `<m>` is at least twice <n>.

If other databases share this instance, you can increase the `MAXAGENTS` parameter to reflect all the total number of agents that are working on all applications.

8.2.5 Create attribute indexes

The Library Server and the Resource Manager databases are installed with the appropriate indexes to optimize create, retrieve, update, and delete operations. When you define a new item type, the appropriate indexes are created automatically. Indexes on user-defined item attributes are not generated automatically. For the attributes that are frequently used for regular searches, creating indexes on these attributes often improves response time and reduces resource usage for queries using these attributes.

Indexes can negatively affect response times for create, update, and delete operations for that item type as well as increase the database size and buffer pool usage. You should monitor overall system performance and response time before and after adding an index to evaluate its overall value. A part of this process should be running your SQL through the DB2 Explain function to verify that the new indexes are being used.

How to create an index

You can use the Content Manager system administration client or the DB2 Control Center to add an index for a single attribute. If you want to create an index over multiple attributes, then you must create it manually. In order to be effective, manually created indexes must include certain key system-defined attributes.

For example, for a simple (no child components) item type myitemtype, with user-defined attributes myattr1 and myattr2, perform the following steps to create indexes on the attributes:

1. Query to get numeric values AAAA for myattr1 and BBBB for myattr2:

   ```
   db2 select keywordcode, keywordname from icmstnlskeywords where
   keywordclass=1 and keywordname in ('myattr1','myattr2')
   ```

2. Query to get numeric value CCCC for myitemtype:

   ```
   db2 select c.componenttypeid from
   icmstnlskeywords k, icmstcompdefs c where
   k.keywordname='myitemtype' and k.keywordclass=2 and
   k.keywordcode=c.itemtypeid and c.comptypeclass=0
   ```

3. Create the index using the obtained numeric values:

   ```
   db2 create unique index mynewindex1 on icmut0<CCCC> (
      ATTR000000<AAAA> asc,
      ATTR000000<BBBB> asc,
      versioned asc,
      componentid asc
   ) include (itemid, compkey)
   ```

   (This index is unique, even if myattr1 and myattr2 are not.)

4. If you want the myattr1 and myattr2 combination to be unique within myitemtype, create this index as well:

   ```
   db2 create unique index mynewindex2 on icmut0<CCCC>001 (
      ATTR000000<AAAA> asc,
      ATTR000000<BBBB> asc)
   ```

After creating a new index, either manually or through the DB2 Control Center, run the **runstats** and **rebind** commands to make sure that the DB2 UDB optimizer can effectively use the new index.

See 8.3.3, “Keeping database statistics and execution plans up to date through runstats/rebind” on page 183 for details about how to run the processes.

8.2.6 Create indexes for document routing

If you use document routing with “filter on owner,” adding this index to the Library Server database might improve performance:

```
db2 create index icmux002040017x on icmut00204001(
   itemid asc,
   assigneduserid asc);
```

Other document routing indexes that can improve performance if appropriate are:

```
db2 create index icmux002040018x on icmut00204001(
   notifytime asc);

db2 create index icmux002040019x on icmut00204001(
   suspendflag asc,
   resumetime asc);
```

8.2.7 Additional configuration and tuning suggestions

You might want to consider other additional configuration and tuning suggestions. Some of them (using a striped file system and separating indexes from base tables) are especially helpful if you need to improve performance on a high-volume production system. We recommend the following actions:

- Avoid Unicode.
- Use same code pages.
- Use a striped file system for database table spaces.
- Assign indexes to separate table spaces from the base tables.
- Use DB2 multi-page file allocation.

Avoid Unicode

There is a substantial performance penalty to using Unicode for the Library Server database. Avoid using this feature unless you need the functionality.

Use same code pages

Use the same code page for both Library Server and Resource Manager databases if possible. Data conversion might be required to map data between application and database code pages when your application and database do not use the same code page. This overhead negatively affects performance.

The code page can be specified in the DDL files described earlier.

Use a striped file system for database table spaces

If your data is placed on Redundant Array of Independent Disks (RAID) devices, use the striped file systems for the database table spaces. If possible, use four to six or more physical disks. For each table space that uses a RAID device, you should:

- Define a single container for the table space (using the RAID device).
- Make the EXTENTSIZE of the table space equal to, or a multiple of, the RAID stripe size.

- Ensure that the PREFETCHSIZE of the table space is:
  - The RAID stripe size multiplied by the number of RAID parallel devices
  - A multiple of EXTENTSIZE
- Use the DB2_PARALLEL_IO registry variable to enable parallel I/O for the table space (DB2 must be stopped and started afterwards):

```
db2set DB2_PARALLEL_IO=*
```

Assign indexes to separate table spaces from the base tables

Assign indexes to separate table spaces from the base tables. This can allow for more efficient use of disk storage by reducing the movement of read/write heads during index access. You can also create index table spaces on faster physical devices. Having separate index table spaces gives you the option of assigning them to a different buffer pool than that being used by the data table spaces. This might keep the index pages in the buffer longer because they do not compete with the table data pages.

Use DB2 multi-page file allocation

With the SMS table spaces, when you are inserting or updating a table in the table space, the file associated with the SMS container is extended one page at a time, which can lead to a performance degradation if you are inserting large numbers of rows.

For SMS databases, consider the use of DB2 multi-page file allocation for improved performance. This is done by running the **`db2empfa`** tool. With multi-page file allocation enabled, disk space is allocated one extent at a time rather than one page at a time.

This parameter has no effect on DMS table spaces because space is pre-allocated. Nor does it have any effect on temporary table because all of the required pages are allocated when the temporary table space is created.

Attention: When you set this parameter to YES for a particular database, you cannot set it back to NO.

Prior to Version 8.2 of DB2, the default for the MULTIPAGE_ALLOC database configuration parameter was NO, which caused one page to be allocated at a time. In Version 8.2, the default setting is YES, which means that a full extent is allocated at a time by default.

8.3 Regular routine maintenance

In this section, we describe what you need to do as routine maintenance to ensure that your system is performing at the optimal level. The maintenance tasks include:

- Keeping up with the latest software fix packs
- Monitoring system performance using the DB2 snapshot
- Keeping database statistics and execution plans up to date by running **`runstats`** and **`rebind`** commands
- Reorganizing tables as necessary
- Pruning log file regularly
- Pruning diagnostics file
- Cleaning up the event log

8.3.1 Keeping up with the latest software fix packs

Software bugs are found and software enhancements are continually being developed. DB2 is no exception. It is extremely important to keep your software up to the latest fix pack level so that you get the benefit of these software changes. When calls to external support are placed, one of the first things they ask is whether your software is at the latest fix pack level. If not, they typically ask you to update to the latest version of the software before they continue to troubleshoot your problem to avoid trying to solve a problem that has already been solved.

Check for new fix packs at least quarterly and whenever you see a recurring problem that other tuning suggestions cannot fix.

Fix packs for DB2 can be found at:

`http://www.ibm.com/software/data/db2/udb/support/`

Always read the README.TXT file before installing the fix pack to be sure that all prerequisites are known.

8.3.2 Monitoring system performance using DB2 snapshot

System monitoring helps you understand what is happening to the system and what kind of action you might need to take. Monitor your system on a regular basis to ensure that it is running healthy, and to profile system usage and performance. To do this, we recommend using the DB2 monitoring tool snapshot. Chapter 11, “Performance monitoring, analysis, and tracing” on page 281 goes into great detail about the use of snapshot and other monitoring tools, and the content of this section is derived from the way a particular client monitors their system.

How to take a snapshot

1. DB2 monitors must be set on before they can collect data. It is not necessary to turn on all of the monitors for regular maintenance. At a minimum, have the DFT_MON_BUFPOOL, DFT_MON_LOCK, and DFT_MON_SORT turned on. The other switches provide a great deal of data and could be turned on to investigate a specific problem. The switches can be turned on as follows:
   a. Click **Programs** → **IBM DB2** → **General Administration Tools** → **Control Center**.
   b. Change the following database instance parameters state to on:
      ```
      DFT_MON_BUFPOOL
      DFT_MON_LOCK
      DFT_MON_SORT
      DFT_MON_STMT
      DFT_MON_TABLE
      DFT_MON_TIMESTAMP
      DFT_MON_UOW
      HEALTH_MON
      ```
   You can also turn on all monitoring switches through a DB2 command line:
   ```
   update monitor switches using BUFFERPOOL on, LOCK on, SORT on,
   STATEMENT on, TIMESTAMP on, TABLE on, UOW on
   ```
   The database monitor switches must be turned on for sufficient time to allow DB2 to collect meaningful data.
2. When you start database monitors, you also have to increase the database manager monitor heap size (MON_HEAP_SZ) to allow enough memory allocation to get the monitoring results:
   a. Click **Programs** → **IBM DB2** → **General Administration Tools** → **Control Center**.
   b. Change the database instance parameter MON_HEAP_SZ to at least 2048.
3. Run snapshot from DB2 command window:
   a. Click **Start** → **Run**.
   b. Type **`db2cmd`**.
   c. Type the following command:
      ```
      db2 get snapshot for all databases > fname.txt
      ```
      - *fname* is the name of the file the monitoring results will be saved in.

4. Alternative to running the ALL snapshot, you can run individual snapshots that enable the output to be saved in a more specific manner. Example 8-1 shows a list of individual snapshots you can run.

Example 8-1 Running individual snapshots

```
db2 get snapshot for database manager > snapdbm.txt
db2 get snapshot for database on rmdb > snapdbr.txt
db2 get snapshot for database on icmnlsdb > snapdbi.txt
db2 get snapshot for bufferpools on rmdb > snapbpr.txt
db2 get snapshot for bufferpools on icmnlsdb > snapbpi.txt
db2 get snapshot for tablespaces on rmdb > snaptspr.txt
db2 get snapshot for tablespaces on icmnlsdb > snaptspi.txt
db2 get snapshot for tables on rmdb > snaptblr.txt
db2 get snapshot for tables on icmnlsdb > snaptbli.txt
```

Our recommendation

We recommend regularly taking a DB2 snapshot of the database manager (instance) and each database, and that you keep the snapshot data in either a file, a spreadsheet, or loaded into a DB2 table for historical and trend analysis. Analyze the results for immediate issues such as a high number of package cache overflows or database files closed. Perform a trend analysis to determine whether a problem is building. Is your buffer pool hit ratio going down? Are your sort overflows steadily increasing? Take appropriate corrective action and continue to monitor.

8.3.3 Keeping database statistics and execution plans up to date through runstats/rebind

Routinely keep the Library Server and the Resource Manager database statistics and execution plans up to date using the **runstats** and **rebind** commands to maintain good performance. For **runstats** to be effective, a **rebind** of the application is necessary after the **runstats** execution.

Script generation

Example 8-2 contains a set of commands you can run from a DB2 command line window that creates a script you need to run.

Example 8-2 Commands to generate a script for runstats and rebind

```
db2 connect to db user userid using password
echo db2 connect to db user userid using password > fname.bat
db2 –x "select 'db2 runstats on table 'concat tabschema concat
```

```
'.' concat tabname concat 'with distribution and detailed indexes all'
from syscat.tables where tabschema='schema' and type='T'">> fname.bat
echo db2 connect reset >> fname.bat
db2 connect reset
echo db2rbind db –l bind.log all /u userid /p password >> fname.bat
```

Change *db* to the name of your database and change *userid* and *password* for your system values. Change the schema name based on your system and be sure to use capital letters.

Our recommendation

Run the generated file fname.bat script daily for the first few weeks and during initial database loading. After that, run it weekly or at least monthly as routine performance maintenance, and whenever your database has changed significantly in size. This should be part of your database administration.

Attention: Library Server relies heavily on DB2 stored procedures and precompiled access modules to perform its functions. This is why **runstats** is so important for maintaining the performance of a Content Manager. To learn more about how Library Server works, see 5.5, "Library Server" on page 106 for more details.

8.3.4 Reorganizing tables through reorg

Over a period of time, after many insertion, deletion, and updating to database table data, logically sequential data may be on non-sequential physical data pages. This causes the database manager to perform additional read operations to access data. Additional read operations are also required if a significant number of rows have been deleted. In such a case, you need to reorganize the table to re-cluster the data to match the primary index and to reclaim space. You can reorganize the system catalog tables as well as database tables.

Reorg required?

The DB2 command **reorgchk** can be used to suggest if a reorganization of the tables and indexes is warranted. You can run the command from a script and schedule to run the script when the system usage is low. You can use **reorgchk** to recalculate the table statistics using the "update statistics" option but it does not give the level of control over recalculating the index statistics that the **runstats** command does. Run **runstats** before running a **reorgchk**.

To see whether you need to reorganize a table, use the following command from a DB2 command line window (after connecting to the database):

```
db2 reorgchk current statistics on table all > reorgchk.txt
```

Example 8-3 displays a partial sample of the output results that would be stored in reorgchk.txt. In the complete output listing, statistics for all of the tables followed by statistics for all of the indexes are included. The last column in the output (REORG) is the column that indicates by the presence of one or more asterisks whether a reorganization might be necessary (for example, -*- or --*).

Example 8-3 Sample reorgchk output

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA    NAME                  CARD    OV    NP    FP ACTBLK    TSIZE  F1  F2  F3 REORG
----------------------------------------------------------------------------------------
RMADMIN   RMACCESS                 2     0     1     1      -      386   0   - 100 ---
RMADMIN   RMBLOBS                  -     -     -     -      -        -   -   -   - ---
RMADMIN   RMCNTL                   1     0     1     1      -      225   0   - 100 ---
RMADMIN   RMCOLLECTIONS           16     0     1     1      -      992   0   - 100 ---
RMADMIN   RMDEVMGR                 6     0     1     1      -      540   0   - 100 ---
RMADMIN   RMMGTCLASS               1     0     1     1      -       44   0   - 100 ---
RMADMIN   RMMGTTRANSITION          1     0     1     1      -       20   0   - 100 ---
RMADMIN   RMOBJECTS           611242     0  3748  3748      - 1.22e+08   0  99 100 ---
RMADMIN   RMPARTS                  -     -     -     -      -        -   -   -   - ---
RMADMIN   RMREPLICATION            -     -     -     -      -        -   -   -   - ---
RMADMIN   RMSERVER                 2     0     1     1      -      750   0   - 100 ---
RMADMIN   RMSTAGING                1     0     1     1      -       57   0   - 100 ---
RMADMIN   RMSTGGRPCLASS           16     0     1     1      -      224   0   - 100 ---
RMADMIN   RMSTGGRPVOLUME          16     0     1     1      -      256   0   - 100 ---
RMADMIN   RMSTORAGECLASS           1     0     1     1      -       57   0   - 100 ---
RMADMIN   RMSTORAGEGROUP          16     0     1     1      -      704   0   - 100 ---
RMADMIN   RMSTRINGS                1     0     1     1      -       30   0   - 100 ---
RMADMIN   RMSYNCPARM               1     0     1     1      -       36   0   - 100 ---
RMADMIN   RMSYNCREPORT             -     -     -     -      -        -   -   -   - ---
RMADMIN   RMSYNCSCHED              1     0     1     1      -       31   0   - 100 ---
RMADMIN   RMSYNCSTATUS             1     0     1     1      -       53   0   - 100 ---
RMADMIN   RMSYNCVOL                -     -     -     -      -        -   -   -   - ---
RMADMIN   RMTABLES                18     0     1     1      -      882   0   - 100 ---
RMADMIN   RMTRACKING               -     -     -     -      -        -   -   -   - ---
RMADMIN   RMVERSION                1     0     1     1      -       59   0   - 100 ---
RMADMIN   RMVOLUMES               15     0     2     2      -     4920   0 100 100 ---
SYSIBM    SYSATTRIBUTES            -     -     -     -      -        -   -   -   - ---
SYSIBM    SYSBUFFERPOOLNODES       -     -     -     -      -        -   -   -   - ---
SYSIBM    SYSBUFFERPOOLS           8     0     1     1      -      400   0   - 100 ---
```

To reorganize a specific table, use the following command from a DB2 command line window:

```
db2 reorg table <table name>
```

In this example, `<table name>` is the specific table you want to reorganize, such as RMADMIN.RMPARTS.

To reorganize the indexes for a specific table, use the following command from a DB2 command line window:

```
db2 reorg indexes all for table <table name>
```

Again, `<table name>` is the specific table you want to reorganize.

Our recommendation

When you reorganize tables, you remove empty spaces and arrange table data for efficient access. Reorganizing tables takes a lot more time than simply checking (**`reorgchk`**) what tables might require reorganization. It is always a good idea to perform reorgchk first. If you have already identified the pattern of what tables most likely need reorganization, you can schedule the **`reorg`** task on these tables without first running **`reorgchk`**. Do not reorganize tables when there are a lot of server activities because **`reorg`** has an impact on performance. DB2 locks any data in a table that is currently being reorganized.

Refer to *IBM Content Manager for Multiplatforms System Administration Guide*, SC27-1335, for more information about how to perform this optimization task.

8.3.5 Pruning log file

When using the Log Retain strategy, log files are continually created as needed, filling up disk space of the log directory in the process. When the file system or disk is full, DB2 will be unable to create log records and will start writing error messages continually into the diagnostics log file (db2diag.log) until the problem is resolved. For this reason, regular pruning or moving of old log files must be done before the file system gets full. The frequency of this action depends on how fast it can use up most of the space in the log directory.

Example 8-4 displays a sample script that would handle moving log files.

Example 8-4 Log file rotation script example

```
#!/bin/sh

# Here is the cron entry:
# 59 23 * * * /home/script/DB2LogRotation.sh >> /home/script/rotation.log 2>&1
# Rotates server log files, without affecting users who may be
# connected to the server.

DATE=`date +%Y.%m.%d`

LOGDIR=$1
ENDDIR=$2
LOGS=$3

(cd $LOGDIR;

 for i in $LOGS; do
   if [ -f $i ]; then
     cp -p $i $i.$DATE
     cp /dev/null $i
     /usr/bin/gzip -v $i.$DATE
   fi
 done
)

if [ -f $LOGDIR/*.gz ]; then
        # chown wasuser:wasgroup $LOGDIR/*.gz
        chmod 644 $LOGDIR/*.gz
        mv $LOGDIR/*.gz $ENDDIR
fi
```

Our recommendation

Log files should not be deleted until you are absolutely sure that they will not be needed for any type of recovery. We recommend moving non-active log files from the log directory to another "logs to be deleted" directory located on a different disk. Depending on your recovery strategy, when you are sure that you no longer need the logs, you can delete them from this directory.

You can also use Tivoli Storage Manager as a destination for log-file archival.

8.3.6 Pruning diagnostics file

DB2 continually writes all diagnostic information to the db2diag.log file, which is intended to be used by DB2 customer support for troubleshooting purposes. You control the level (and amount) of detailed information that is written to the file by using the DIAGLEVEL configuration parameter.

Valid values are:

0	No diagnostic data captured
1	Severe errors only
2	All errors
3	(default) All errors and warnings
4	All errors, warnings, and informational messages

Increasing the amount of diagnostic output can result in both performance degradation and insufficient storage conditions in your database instance file system if that is where your diagnostics file is stored.

Our recommendation

Because the file will continue to grow forever and consume space, it is necessary to maintain the file. Instead of manually editing the file to crop it down or simply deleting it, we recommend that you periodically move (manually or via a scheduled script) the file to another disk. (DB2 will automatically create a new db2diag.log file.) Each moved file would overlay the previously moved file, allowing for one version of historical data.

8.3.7 Cleaning up the event log

An event monitor captures system monitor information after particular events have occurred. These include the end of a transaction, the end of a statement, or the detection of a deadlock. This information can be written to files or to a named pipe. If you have Events enabled, monitor the size of the Events table and periodically prune it.

8.4 SMS to DMS table space conversion

As discussed in 2.2.4, "Performance-related tools and concepts" on page 63, there are distinct advantages related to whether to use system-managed storage (SMS) table spaces or database-managed space (DMS) table spaces. As a general rule, a well-tuned set of DMS table spaces outperforms SMS table spaces for reasons such as:

- Assigning indexes to separate table spaces from the base table.
- The ability to add or extend containers.
- Does not use the file system of the operating system.

It is not necessary or advisable for all of your table spaces to be converted to DMS. DMS is recommended for tables that have a high volume of activity or have a high cardinality. A Content Manager system that has a lot of users concurrently accessing the system and retrieving data would benefit from DMS table spaces.

Using DMS table spaces enables DB2 to bypass querying the operating system for file containers' locations and sizes because DB2 already has that information. How much of a performance gain still depends greatly on the application and database setup. These areas still must be monitored and tuned.

If you decide that converting any of your table spaces from SMS to DMS is right for your system, this section highlights the necessary steps. Although these steps reference AIX file names, the process is the same for Windows.

The SMS-to-DMS conversion tasks include:

1. Running **`cdb2look`**
2. Exporting table data
3. Creating new logical volumes
4. Creating the new DMS DDL
5. Loading the data

Attention: The commands, object names, and file names in this section are used to illustrate the examples. They are provided *for reference only*. Do not simply use them as they are presented. Always consult the platform-specific product manual for the proper syntax.

8.4.1 Running db2look

The command **`db2look`** generates data definition language statements by object type. Run this command on your current system to see how the table spaces and containers currently exist:

```
db2look -d ICMNLSDB -u icmadmin -z ICMADMIN -e -p -l -o db2look.sql
```

In this syntax:

- `-d` specifies the database name.
- `-u` specifies the creator.
- `-z` specifies the schema.
- `-e` specifies the extract DDL statements for database objects.
- `-p` specifies plain text.
- `-l` specifies to generate DDL for user-defined table spaces, database partition groups, and buffer pools.
- `-o` specifies a meaningful output file name.

Example 8-5 on page 190 shows an example of the output from **`db2look`**.

Example 8-5 Output from db2look

```
------------------------------------
-- DDL Statements for TABLESPACES --
------------------------------------

CREATE REGULAR TABLESPACE ICMLFQ32 IN DATABASE PARTITION GROUP IBMDEFAULTGROUP
PAGESIZE 32768 MANAGED BY SYSTEM
         USING ('/lsdb/db2inst1/db2inst1/NODE0000/SQL00001/ICMSXITEMS0010013U')
         EXTENTSIZE 32
         PREFETCHSIZE AUTOMATIC
         BUFFERPOOL ICMLSMAINBP32
         OVERHEAD 12.670000
         TRANSFERRATE 0.180000
         DROPPED TABLE RECOVERY ON;
```

8.4.2 Exporting table data

The conversion from SMS to DMS involves dropping the SMS table spaces. Before this is done, you must save your existing table data using the **export** command. Example 8-6 shows an example of the **export** command.

Example 8-6 Export table data command

```
EXPORT TO /lsdb/sms/ exportICMUT01016001.ixf of ixf messages
/lsdb/sms/exp_msgICMUT01016001.msg
SELECT * FROM ICMADMIN.ICMUT01016001;
```

8.4.3 Creating new logical volumes

Create new logical volumes and file systems on separate hard disks if possible for each table space. This allows more parallelism in disk I/O and enables the **filemon** and **iostat** commands to clearly display which file systems have the highest disk I/O.

Example 8-7 shows an example of the commands that could be necessary. These commands can be in a script that loops through an input file that defines the new file systems for the new DMS tables.

Example 8-7 AIX logical volume commands

```
#
# Library Server
#
#  ls db 400GB
```

```
while read sz fs lv vg pv logv
do
#
# Setup VGroups (using a partition size of 32MB -s )
#
#    /usr/sbin/mkvg -y$vg -s'32' $pv
#    /usr/sbin/lsvg -L -p $vg

#
# Setup LVs
#
# for jfs2, use:     /usr/sbin/mklv -v'n' -w'n' -y$lv -t'jfs2' -e'x' $vg $sz $pv
     /usr/sbin/mklv -v'n' -w'n' -y$lv -t'jfs' -e'x' $vg $sz $pv

#
# Setup filesystems
#
#    /usr/sbin/crfs -v jfs2 -d${lv} -m'/'${fs} -A yes -p'rw' -t no -a size=$sz -a
agblksize='4096' -a logname='INLINE'
     /usr/sbin/crfs -v jfs -d${lv} -m'/'${fs} -A yes -p'rw' -t no -a bf=true -a
size=$sz -a frag='4096' -a nbpi='32768' -a ag='16' -a logname=$logv

done<ls.input

/usr/sbin/mount -a
/usr/bin/df -k
```

Example 8-8 shows a sample input file that could be used as input to the script.

Example 8-8   Sample input file

```
## more ls.input
5G lsdbfast1 lsdblv1 sysfvg hdisk5 /dev/loglsdbfast1
5G lsdbfast2 lsdblv2 sysfvg hdisk4 /dev/loglsdbfast2
5G lsdbfast3 lsdblv3 sysfvg hdisk3  /dev/loglsdbfast3
5G lsdbfast4 lsdblv4 sysfvg hdisk2 /dev/loglsdbfast4
5G lsdbfast5 lsdblv5 sysfvg hdisk1  /dev/loglsdbfast5
5G lsdbfast6 lsdblv6 sysfvg hdisk5 /dev/loglsdbfast6
5G lsdbfast7 lsdblv7 sysfvg hdisk4 /dev/loglsdbfast7
5G lsdbfast8 lsdblv8 sysfvg hdisk3 /dev/loglsdbfast8
5G lsdbfast9 lsdblv9 sysfvg hdisk2  /dev/loglsdbfast9
5G lsdbfast10 lsdblv10 sysfvg hdisk1  /dev/loglsdbfast10
5G lsdbfast11 lsdblv11 sysfvg hdisk5 /dev/loglsdbfast11
5G lsdbfast1i lsdblv1i sysfvg hdisk4  /dev/loglsdbfast1i
5G lsdbfast2i lsdblv2i sysfvg hdisk3 /dev/loglsdbfast2i
```

```
5G lsdbfast3i lsdblv3i sysfvg hdisk2 /dev/loglsdbfast3i
5G lsdbfast4i lsdblv4i sysfvg hdisk1  /dev/loglsdbfast4i
5G lsdbfast5i lsdblv5i sysfvg hdisk5 /dev/loglsdbfast5i
5G lsdbfast6i lsdblv6i sysfvg hdisk4 /dev/loglsdbfast6i
5G lsdbfast7i lsdblv7i sysfvg hdisk3 /dev/loglsdbfast7i
5G lsdbfast8i lsdblv8i sysfvg hdisk2 /dev/loglsdbfast8i
5G lsdbfast9i lsdblv9i sysfvg hdisk1  /dev/loglsdbfast9i
5G lsdbfast10i lsdblv10i sysfvg hdisk5  /dev/loglsdbfast10i
5G lsdbfast11i lsdblv11i sysfvg hdisk4  /dev/loglsdbfast11i
```

In Example 8-8 on page 191, one logical volume manager will be created for each file system. This reduces the bottleneck that would have occurred if all file systems were under one logical volume manager.

Example 8-9 shows results similar to the look of the newly created file system. In our example, the “i” is used to indicate table spaces for indexes of the corresponding tables.

Example 8-9 File system output

```
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4         63176704  47458360   25%    50016     1% /
/dev/hd2         32505856  29354860   10%    63111     1% /usr
/dev/hd9var       4456448   4423500    1%      456     1% /var
/dev/hd3         26214400  18412260   30%      180     1% /tmp
/dev/hd1          7340032   7267456    1%      493     1% /home
/proc                   -         -    -         -     -  /proc
/dev/hd10opt     27262976  25581188    7%    54239     1% /opt
/dev/fslv01        393216    390992    1%       45     1% /root
/dev/usaflv      98304000  84807100   14%    24707     1% /lsdb
/dev/lsdblv1      5242880   5221944    1%       17     1% /lsdbfast1
/dev/lsdblv2      5242880   5221944    1%       17     1% /lsdbfast2
/dev/lsdblv3      5242880   5221944    1%       17     1% /lsdbfast3
/dev/lsdblv4      5242880   5221944    1%       17     1% /lsdbfast4
/dev/lsdblv5      5242880   5221944    1%       17     1% /lsdbfast5
/dev/lsdblv6      5242880   5221944    1%       17     1% /lsdbfast7
/dev/lsdblv7      5242880   5221944    1%       17     1% /lsdbfast8
/dev/lsdblv8      5242880   5221944    1%       17     1% /lsdbfast9
/dev/lsdblv10     5242880   5221944    1%       17     1% /lsdbfast10
/dev/lsdblv11     5242880   5221944    1%       17     1% /lsdbfast11
/dev/lsdblv1i     5242880   5221944    1%       17     1% /lsdbfast1i
/dev/lsdblv2i     5242880   5221944    1%       17     1% /lsdbfast2i
/dev/lsdblv3i     5242880   5221944    1%       17     1% /lsdbfast3i
/dev/lsdblv4i     5242880   5221944    1%       17     1% /lsdbfast4i
/dev/lsdblv5i     5242880   5221944    1%       17     1% /lsdbfast5i
```

```
/dev/lsdblv6i     5242880   5221944    1%       17     1% /lsdbfast7i
/dev/lsdblv7i     5242880   5221944    1%       17     1% /lsdbfast8i
/dev/lsdblv8i     5242880   5221944    1%       17     1% /lsdbfast9i
/dev/lsdblv10i     5242880   5221944    1%       17     1% /lsdbfast10i
/dev/lsdblv11i     5242880   5221944    1%       17     1% /lsdbfast11i
```

8.4.4 Creating the DMS DDL

Using the DDL that was produced (or a copy of it) from the running of **db2look**, create the DMS DDL that will be used to rebuild the table space, the table, the indexes, the views, and any other DB2 objects contained in the table space that is being converted. You can run all of the create DDLs together in one script or you can run the **create** commands separately. This section makes the assumption that all of the DDLs will be run in one script by the **db2 -tvf** command.

Start by creating the DMS DDL for the new table spaces. Example 8-10 shows an example in which two DMS table spaces are to be created, the first for table data and the second for the table indexes. Note the `MANAGED BY DATABASE`, and that more than one container is created for each table space to achieve parallelism.

Example 8-10 DMS table space DDL

```
------------------------------------
-- DDL Statements for TABLESPACES --
------------------------------------

CREATE REGULAR TABLESPACE tsicmut01001a IN DATABASE PARTITION GROUP
ICMLSONENODE PAGESIZE 32768 MANAGED BY DATABASE
         USING (FILE '/lsdbfast3/icmut01001a-1' 3000, FILE
'/lsdbfast3/icmut01001a-2' 3000)
         EXTENTSIZE 32
         PREFETCHSIZE 32
         BUFFERPOOL ICMLSMAINBP32
         OVERHEAD 12.670000
         TRANSFERRATE 0.180000
         DROPPED TABLE RECOVERY ON;


CREATE REGULAR TABLESPACE tsicmut01001ai IN DATABASE PARTITION GROUP
ICMLSONENODE PAGESIZE 32768 MANAGED BY DATABASE
         USING (FILE '/lsdbfast3i/icmut01001a-1i' 3000, FILE
'/lsdbfast3i/icmut01001a-2i' 3000)
         EXTENTSIZE 32
```

```
          PREFETCHSIZE 2
          BUFFERPOOL ICMLSMAINBP32
          DROPPED TABLE RECOVERY ON;
```

Part of modifying the DDL to add containers to a table space involves calculating the size that the containers should be. In our example, each container has been assigned 3000 pages. To determine what size to make the containers:

1. Use **`reorgchk`** to determine the table size (tsize) in bytes.
2. Convert the table size into pages by dividing it by the page size.

   Assume table ICMUT01016001 is 62,900,000 bytes divided by 32768 (32 K page size), which is almost 2000 pages.
3. Adjust the number of container pages based on workload and growth expectations.

When the table spaces have been created, change the table DDL to reference the new table spaces. This is done in the `in` and `index in` options of the **`create table`** command, which is shown in Example 8-11.

Example 8-11   Create table command

```
CREATE TABLE "ICMADMIN"."ICMUT01016001"  (
        "COMPCLUSTERID" INTEGER NOT NULL ,
        "COMPONENTID" CHAR(18) NOT NULL ,
        "ITEMID" CHAR(26) NOT NULL ,
        "VERSIONID" SMALLINT NOT NULL ,
        "ACLCODE" INTEGER NOT NULL ,
        "SEMANTICTYPE" INTEGER NOT NULL ,
        "EXPIRATIONDATE" DATE ,
        "COMPKEY" VARCHAR(23) FOR BIT DATA NOT NULL ,
        "CREATETS" TIMESTAMP NOT NULL ,
        "CREATEUSERID" CHAR(32) NOT NULL ,
        "LASTCHANGEDTS" TIMESTAMP NOT NULL ,
        "LASTCHANGEDUSERID" CHAR(32) NOT NULL ,
        "ATTR0000001036" VARCHAR(254) NOT NULL ,
        "ATTR0000001037" VARCHAR(254) ,
        "ATTR0000001038" VARCHAR(254) ,
        "ATTR0000001039" VARCHAR(254) ,
        "ATTR0000001040" VARCHAR(254) ,
        "ATTR0000001041" VARCHAR(254) ,
        "ATTR0000001042" VARCHAR(254) ,
        "ATTR0000001043" VARCHAR(254) ,
        "ATTR0000001044" VARCHAR(254) ,
        "ATTR0000001045" VARCHAR(254) ,
        "ATTR0000001046" VARCHAR(254) ,
        "ATTR0000001047" VARCHAR(254) ,
        "ATTR0000001048" INTEGER ,
        "ATTR0000001049" INTEGER ,
        "ATTR0000001050" INTEGER ,
        "ATTR0000001051" INTEGER ,
        "ATTR0000001052" INTEGER ,
        "ATTR0000001053" INTEGER ,
        "ATTR0000001054" INTEGER ,
        "ATTR0000001055" INTEGER ,
        "ATTR0000001056" INTEGER ,
        "ATTR0000001057" INTEGER ,
        "ATTR0000001058" VARCHAR(3975) )
       IN "tsicmut16001" INDEX IN "tsicmut16001i";
```

The DDL for the other DB2 objects that relate to the table space (indexes, views) do not change as part of the conversion to DMS.

If you want to use the same name for the new DMS table space as is being used by the SMS table space, drop the old SMS table space before creating the new one:

```
db2 drop tablespace tscmut16001
```

If you want to use a different name for the new DMS table space, drop any old SMS tables that belong to the table spaces being converted to DMS before creating the new tables because the table names are not changing:

```
db2 drop table ICMADMIN.ICMUT01016001
```

As discussed at the start of this section, we have assumed that all of the DDL related to the conversion of a table space from SMS to DMS has been put into a single file. For the purpose of this example, the dms-icmut16001.sql file contains all of the DDL related to our table space. To run the DDLs for DMS table space creation, issue the following command:

```
db2 -tvf dms-icmut16001.sql
```

8.4.5 Loading the data

After you have created the table space and all of the objects that relate to it, load the data back into the tables. Example 8-12 on page 196 shows an example of the **load** command.

Example 8-12 Load command

```
LOAD FROM /lsdb/sms/exportICMUT01016001.ixf of ixf messages
/lsdb/sms/load_msgICMUT01016001.msg REPLACE INTO ICMADMIN.ICMUT01016001
CHECK PENDING CASCADE DEFERRED ;
```

After loading the data, run the **`runstats`** command to update the catalog statistics, then run the **`rebind`** command.

8.5 Buffer pool tuning

A buffer pool is memory that is used to cache table and index data pages as they are being read from disk or being modified. The buffer pool improves database system performance by enabling data to be accessed from memory instead of from disk. Because memory access is much faster than disk access, the less often the database manager needs to read from or write to a disk, the better the performance. Because most data manipulation takes place in buffer pools, configuring buffer pools is the single most important tuning area. Only large objects and long field data are not manipulated in a buffer pool.

Attention: The buffer pool is probably the single most important database tuning area for DB2.

All buffer pools reside in the Database Global Memory, which is available to all applications using the database. All buffer pools are allocated when the first application connects to the database, or when the database is explicitly activated using the **`activate database`** command. Use the **`activate database`** command to keep the buffer pools primed even if all of the connections terminate. This will be useful when the connection load is highly dynamic (for example, Web servers).

As an application requests data from the database, pages containing the data are transferred to one of the buffer pools from disk. The next time an application requests data, the buffer pool is checked first to see whether the data is in the memory area; if it is found, there is no need to read data from the disk. Avoiding data retrieval from disk storage results in faster performance. If the buffer pools are not large enough to keep the required data in memory, data must be read from disk. Pages are not written back to the disk until the page is changed, or one of the following occurs:

- All applications disconnect from the database.
- The database is explicitly deactivated.
- The database quiesces (that is, all connected applications have committed).

- Space is required for another incoming page.
- A page cleaner is available (NUM_IOCLEANERS database configuration parameter is not zero) and is activated by the database manager.

After changed pages are written out to disk, they are not removed from the buffer pool unless the space they occupy is needed for other pages.

The size and tuning of a buffer pool is handled in one of two ways:

- The BUFFPAGE database configuration parameter controls the size of a buffer pool when the **`create bufferpool`** or **`alter bufferpool`** statement was run with `npages -1`.
- If the **`create bufferpool`** or **`alter bufferpool`** statement was run with a value in NPAGES, then the BUFFPAGE database parameter is ignored and the buffer pool will be created with the number of pages specified in the NPAGES parameter.

Note: npages in syscat.bufferpools overrides buffpage.

To determine whether the BUFFPAGE parameter is active for a buffer pool, issue this command:

```
SELECT * FROM SYSCAT.BUFFERPOOLS
```

The AVG_APPLS parameter provides the optimizer with information regarding the average number of active applications. This parameter is used by the optimizer to determine how much of each buffer pool can be used for an application. A value of 1 for this parameter will cause the optimizer to treat the entire buffer pool to be available to one application.

Note: Buffer pool tuning involves monitoring the buffer pool hit ratio and adjusting the value of the BUFFPAGE parameter according to the hit ratio.

8.5.1 Buffer pool hit ratios

You must be careful how you allocate space for each buffer pool. Allocating a large buffer pool to a table space that contains a large number of small, rarely used tables or a small buffer pool to a table space containing a large, frequently accessed table will lead to performance problems. The size of the buffer pools should reflect the size of the tables in the table space, and how frequently the tables are updated or queried.

The DB2 optimizer utilizes different buffer pools to achieve the best query performance. When selecting the access plan, the optimizer considers the I/O cost of fetching pages from disk to the buffer pool. In its calculation, the optimizer estimates the number of I/Os required to satisfy a query. This estimate includes a prediction of buffer pool usage, because additional physical I/Os are not required to read rows in a page that is already in the buffer pool.

The optimizer considers the value of the `npages` column in the `buffer pools` system catalog table in estimating whether a page will be found in the buffer pool.

The I/O costs of reading the tables can have an impact on:

- How two tables are joined: for example, outer versus inner
- Whether an index will be used to read the data

Snapshot monitoring of the buffer pools, discussed in 8.3.2, “Monitoring system performance using DB2 snapshot” on page 181, can be used to capture information about the number of reads and writes and the amount of time taken. The buffer pool snapshot commands are:

```
db2 get snapshot for bufferpools on rmdb > snapbpr.txt
db2 get snapshot for bufferpools on icmnlsdb > snapbpi.txt
```

Example 8-13 shows a sample snapshot for the main Library Server buffer pool, `ICMLSMAINBP32`.

Example 8-13 Sample snapshot for ICMLSMAINBP32 buffer pool

```
Bufferpool Snapshot

Bufferpool name                           = ICMLSMAINBP32
Database name                             = ICMNLSDB
Database path                             =
/cmdb/db2inst1/NODE0000/SQL00002/
Input database alias                      = ICMNLSDB
Snapshot timestamp                        = 08-03-2005 13:34:53.639150

Buffer pool data logical reads            = 562228265
Buffer pool data physical reads           = 135122990
Buffer pool temporary data logical reads  = 48991
Buffer pool temporary data physical reads = 1196706
Buffer pool data writes                   = 1648067
Buffer pool index logical reads           = 9917310051
Buffer pool index physical reads          = 11930407
Buffer pool temporary index logical reads  = 0
Buffer pool temporary index physical reads = 0
Total buffer pool read time (ms)          = 282369176
Total buffer pool write time (ms)         = 16890198
Asynchronous pool data page reads         = 134496074
Asynchronous pool data page writes        = 1643270
Buffer pool index writes                  = 28368
Asynchronous pool index page reads        = 11534788
Asynchronous pool index page writes       = 27399
Total elapsed asynchronous read time      = 307886999
Total elapsed asynchronous write time     = 16851553
Asynchronous data read requests           = 10905817
Asynchronous index read requests          = 1501310
No victim buffers available               = 149548444
Direct reads                              = 9186176
Direct writes                             = 0
Direct read requests                      = 1344
Direct write requests                     = 0
Direct reads elapsed time (ms)            = 404940
Direct write elapsed time (ms)            = 0
Database files closed                     = 24
Data pages copied to extended storage     = 0
Index pages copied to extended storage    = 0
Data pages copied from extended storage   = 0
Index pages copied from extended storage  = 0
Unread prefetch pages                     = 1510752
Vectored IOs                              = 12407051
Pages from vectored IOs                   = 147227466
Block IOs                                 = 0
Pages from block IOs                      = 0
Physical page maps                        = 0

Node number                               = 0
Tablespaces using bufferpool              = 3
Alter bufferpool information:
 Pages left to remove                     = 0
 Current size                             = 8000
 Post-alter size                          = 8000
```

If the server needs to read a page of data, and that page is already in the buffer pool, then data will be accessed faster than if the page had to be read from the disk. It is then desirable to access as many pages as possible in the buffer pool. Reducing disk I/O is the main action when trying to improve the performance of your server.

The following snapshot data elements can be measured to evaluate how the buffer pool is being used:

- Buffer pool data logical reads (pool_data_l_reads): The number of logical read requests for data pages that have gone through the buffer pool.

- Buffer pool data physical reads (pool_data_p_reads): The number of physical read requests that required I/O to get data pages into the buffer pool.
- Buffer pool index logical reads (pool_index_l_reads): The number of logical read requests for index pages that have gone through the buffer pool.
- Buffer pool index physical reads (pool_index_p_reads): The number of read requests for index pages that required I/O activity to place the index pages into the buffer pool.

Using these elements, you can measure the overall buffer pool hit ratio, the data page hit ratio, and the index buffer pool hit ratio.

Buffer pool hit ratio

The buffer pool hit ratio indicates the percentage of time that the database manager did not need to load a page from disk in order to service a page request. That is, the page was already in the buffer pool. The greater the buffer pool hit ratio, the lower the frequency of disk I/O. An ideal hit ratio is over 90%.

Use the following formula to calculate the overall buffer pool hit ratio:

```
(1 - ((pool_data_p_reads + pool_index_p_reads)/
(pool_data_l_reads + pool_index_l_reads))) * 100
```

Data page hit ratio

The data page hit ratio for a buffer gives you an idea of the number of logical reads that are occurring for every physical read. It is calculated as the number of data physical reads divided by the number of data logical read requests as shown in the following example. A ratio over 75% is desirable, although it is more important to have a high index buffer pool hit ratio.

Use the following formula to calculate the data page hit ratio:

```
(1 - (pool_data_p_reads / pool_data_l_reads)) * 100
```

Index buffer pool hit ratio

Similarly, the index buffer pool hit ratio gives you an idea of the number of logical index reads that are occurring for every physical read. It is calculated as the number of index physical reads divided by the number of index logical read requests as shown in the following example. Ideally, this ratio should be over 90%.

Use the following formula to calculate the buffer pool index hit ratio:

```
(1 - (pool_index_p_reads / pool_index_l_reads)) * 100
```

Tuning tips

Increasing the buffer pool size generally improves the hit ratio, but you will reach a point of diminishing returns. Ideally, if you can allocate a buffer pool large enough to store your entire database, then once the system is up and running, you would get a hit ratio of 100%. However, this is unrealistic in most cases. The significance of the hit ratio depends on the size of your data, and the way it is accessed.

For a large database, increasing the buffer pool size might have minimal effect on the buffer pool hit ratio. Its number of data pages can be so large that the statistical chances of a hit are not improved by increasing its size. In this case, tuning the index pool hit ratio could achieve the desired result. This can be done using two methods:

- Split the data and indexes into two different buffer pools and tune them separately.
- Use one buffer pool, but increase its size until the index hit ratio stops increasing.

The first method is often more effective, but because it requires indexes and data to reside in different table spaces, it might not be an option for existing databases. It also requires you to tune two buffer pools instead of one, which can be more complicated, particularly when amount of memory is limited.

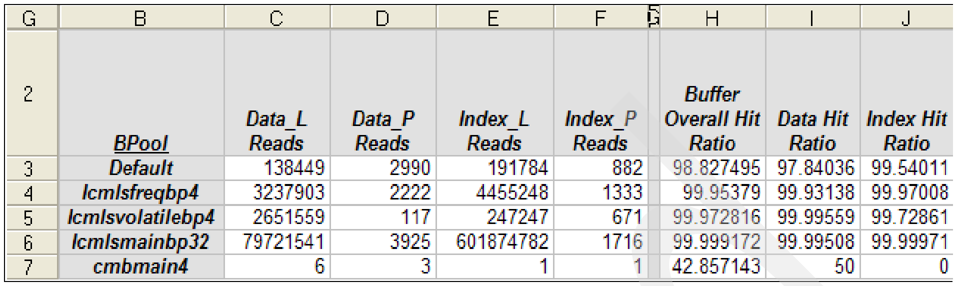
A very large database where data is accessed evenly would have a poor hit ratio. There is little you can do with very large tables. In such a case, you should focus on smaller, frequently accessed tables, and on the indexes — perhaps assigning them to individual buffer pools, in which you can aim for higher hit ratios.

Our recommendation

We recommend that buffer pools be defined to keep all system tables that contain definitions (for example: ItemTypes, ComponentTypes, attributes) in memory to avoid disk I/O. Defining very large buffer pools on user data is a high-cost way to eliminate I/O and there will be a limit eventually. It is more important to design applications and indexes so that a smaller number of pages have to be accessed, which reduces the need for large buffer pools.

Take snapshots and keep a history of the results. Automate the buffer pool calculations (as well as those to follow in this chapter) by placing them into a spreadsheet program such as Excel. As you enter the snapshot data into this spreadsheet, you will see trending that will help you determine whether you need to adjust your buffer pool size or move indexes and tables to other buffer pools.

Figure 8-1 shows an example of a simple IBM Lotus 1-2-3® spreadsheet using the snapshot elements to compute the buffer pool ratios.



G	B	C	D	E	F	G	H	I	J
2	BPool	Data_L Reads	Data_P Reads	Index_L Reads	Index_P Reads		Buffer Overall Hit Ratio	Data Hit Ratio	Index Hit Ratio
3	Default	138449	2990	191784	882		98.827495	97.84036	99.54011
4	Icmlsfreqbp4	3237903	2222	4455248	1333		99.95379	99.93138	99.97008
5	Icmlsvolatilebp4	2651559	117	247247	671		99.972816	99.99559	99.72861
6	Icmlsmainbp32	79721541	3925	601874782	1716		99.999172	99.99508	99.99971
7	cmbmain4	6	3	1	1		42.857143	50	0

Figure 8-1 Lotus 1-2-3 spreadsheet

If your server exceeds 1 GB RAM, consider increasing the default database buffer pool sizes. This can improve performance for heavy loads. To see whether increasing the buffer pool sizes could help, monitor buffer pool logical and physical reads for both data and indexes as mentioned earlier; also monitor committed memory and buffer pool hit ratios. In general, keep the total of all buffer pools below 50% of total physical memory, and even less if a Resource Manager or middle-tier server shares the same machine.

8.6 Parameter tuning

There are two types of DB2 configuration parameters that are typically the subject of tuning: database manager (instance) and database. This section provides a discussion of some of the more important of these configuration parameters that you can tune to improve Content Manager performance. Each parameter is discussed in detail, along with its default value and our recommendation as to what the value should be set to based on our experiences. Keep in mind that these are only recommendations and that your specific system environment might require different values based on your tuning.

For each parameter, the type of parameter (dbm or db) as well as the relative performance impact is provided:

High	Can have a significant impact on performance.
Medium	Can have some impact on performance.
Low	Has a less general or less significant impact on performance.

Table 8-1 on page 203 lists in order the configuration parameters discussed in this section.

Table 8-1 Configuration parameter table

Long name	Short name	Impact
Intra-partition parallelism	INTRA_PARALLEL	DBM - High
Maximum number of concurrently active databases	NUMDB	DBM - Low
Maximum number of active applications	MAXAPPLS	DB - Medium
Average number of active applications	AVG_APPLS	DB - Low
Maximum number of database manager agents	MAXAGENTS	DBM - Medium
Maximum number of client connections	MAX_CONNECTIONS	DBM - Medium
Maximum number of coordinating agents	MAX_COORDAGENTS	DBM - Medium
Initial number of agents in pool	NUM_INITAGENTS	DBM - Medium
Agent pool size	NUM_POOLAGENTS	DBM - High
Maximum number of fenced processes	FENCED_POOL	DBM - Medium
Keep fenced process	KEEPFENCED	DBM - Medium
Initial number of fenced processes	NUM_INITFENCED	DBM - Medium
Maximum database files open per application	MAXFILOP	DB - Medium
Maximum total files open	MAXTOTFILOP	DBM - Medium
Database heap	DBHEAP	DB - Medium
Catalog cache size	CATALOGCACHE_SZ	DB - High
Package cache size	PCKCACHESZ	DB - High
Log buffer size	LOGBUFSZ	DB - High
Maximum storage for lock list	LOCKLIST	DB - High
Application control heap size	APP_CTL_HEAP_SZ	DB - Medium
Application heap size	APPLHEAPSZ	DB - Medium
Sort heap size	SORTHEAP	DB - High
Sort heap threshold	SHEAPTHRES	DBM - High
Statement heap size	STMTHEAP	DB - Medium

Long name	Short name	Impact
Query heap size	QUERY_HEAP_SZ	DBM - Medium
Java interpreter heap size	JAVA_HEAP_SZ	DBM - High
Private memory threshold	PRIV_MEM_THRESH	DBM - Medium
Client I/O block size	RQRIOBLK	DBM - Medium
Application support layer heap size	ASLHEAPSZ	DBM - High
Changed pages threshold	CHNGPGS_THRESH	DB - High
Number of asynchronous page cleaners	NUM_IOCLEANERS	DB - High
Number of I/O servers	NUM_IOSERVERS	DB - High
Maximum percent of lock list before escalation	MAXLOCKS	DB - High
Lock timeout	LOCKTIMEOUT	DB - Medium
Size of the log files	LOGFILSIZ	DB - Medium
Number of primary logs	LOGPRIMARY	DB - Medium
Number of secondary log files	LOGSECOND	DB - Medium
Minimum commits before write logs to disk	MINCOMMIT	DB - High
Connection concentrator	MAX_CONNECTIONS MAXAGENTS NUM_INITAGENTS NUM_INITFENCED MAX_COORDAGENTS NUM_POOLAGENTS	DBM - Medium/High

You can tune some or all of the following parameters during the initial installation of your Content Manager system. We also recommend maintenance tuning on a regular basis to ensure that performance does not degrade over time. Many of the parameters can be tuned without having to turn off your system. Remember the general performance tuning guideline: change one parameter at a time.

Changing database manager configuration parameters

Use this generic procedure to retrieve and update database manager (DBM) configuration parameters. The specific update command appears in the respective parameters subsection.

To retrieve:

1. Open a DB2 command line window.

   On Windows: Click **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**.

   On AIX: Log in as db2inst1.

2. Run the following DB2 command:

   On Windows: **`db2 get dbm cfg`**

   On AIX: **`db2 "get dbm cfg"`**

To update:

1. Open a DB2 command line window.

   On Windows: Click **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**.

   On AIX: Log in as db2inst1.

2. Run the DB2 command. (Refer to subsections for the specific parameter-related command.)

3. Stop and start the database.

Changing database (db) configuration parameters

Use this generic procedure to connect, retrieve, and update database configuration parameters. The specific update command will appear in the respective parameters subsection.

To retrieve:

1. Open a DB2 command line window.

   On Windows: Click **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**.

   On AIX: Log in as db2inst1

2. Run the following DB2 command:

   On Windows: **`db2 get db cfg for <database name>`**

   On AIX: **`db2 "get db cfg for <database name>"`**

To update:

1. Open a DB2 command line window.

   On Windows: Click **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**.

   On AIX: Log in as db2inst1

2. Run the DB2 command. (Refer to subsections for the specific parameter-related command.)
3. Stop and start the database if it is not an online configurable parameter.

Note: Much of the material in this section is extracted from the following publications and augmented with our personal experiences:

- *IBM DB2 Universal Database Version 8.2 Administration Guide: Planning*, SC09-4822
- *IBM DB2 Universal Database Version 8.2 Administration Guide: Performance*, SC09-4821
- *IBM DB2 Universal Database Version 8.2 System Monitor Guide and Reference*, SC09-4847

If you are interested in more in-depth understanding of DB2 and DB2 tuning, we recommend reading these publications. They are excellent guides to DB2, especially with regard to performance tuning.

8.6.1 Intra-partition parallelism (INTRA_PARALLEL)

Impact

DBM - High

Description

This parameter specifies whether the database manager can use intra-partition parallelism. If intra-partition parallelism is disabled (this is the default), then the coordinator agent performs all the application's requests. If intra-partition parallelism is enabled, then DB2 assigns a set of subagents to the application to work on processing the application's requests. Some of the operations that can take advantage of intra-partition parallelism include database queries and index creation.

Default value

No

To update

Run the following DB2 command:

```
db2 update dbm cfg using INTRA_PARALLEL <parameter value>
```

Our recommendation

Keep this parameter as No. If INTRA_PARALLEL is set to Yes, for each connection, DB2 splits SQL statements to be processed in parallel, causing the use of more agents and resources. If your system is transactional (several connections), the recommendation is not to use INTRA_PARALLEL.

If you change this value, you must rebind all packages defined in that database.

8.6.2 Maximum number of concurrently active databases (NUMDB)

Impact

DBM - Low

Description

This parameter specifies the number of local databases that can be concurrently active (that is, have applications connected to them), or the maximum number of different database aliases that can be cataloged on a DB2 Connect™ server. In a partitioned database environment, it limits the number of active database partitions on a database partition server, regardless of whether that server is the coordinator node for the application.

Default values (range)

8 (1 - 256) for UNIX and Windows database server with local and remote clients

3 (1 - 256) for Windows database server with local clients

To update

Run the following DB2 command:

```
db2 update dbm cfg using NUMDB <parameter value>
```

Our recommendation

DB2 allocates memory for each database during the database manager startup. It is generally best to set this value to the actual number of databases that are already defined to the database manager. In a general Content Manager system, we set the parameter to 3:

- Library Server database
- Resource Manager database
- toolsdb database (this database is used to monitor DB2)

8.6.3 Maximum number of active applications (MAXAPPLS)

Impact

DB - Medium

Description

This parameter specifies the maximum number of concurrent applications that can be connected (both local and remote) to a database. Because each application that attaches to a database causes some private memory to be allocated, allowing a larger number of concurrent applications will potentially use more memory.

Setting MAXAPPLS to Automatic has the effect of allowing any number of connected applications. DB2 will dynamically allocate the resources it needs to support new applications.

If you do not want to set this parameter to automatic, the value of this parameter must be equal to or greater than the sum of the following:

- Total connected applications
- The number of these same applications that might be concurrently in the process of completing a two-phase commit or rollback
- The anticipated number of in-doubt transactions that might exist at any one time

When an application attempts to connect to a database, but MAXAPPLS has already been reached, an error is returned to the application indicating that the maximum number of applications have been connected to the database.

As more applications use the Data Links Manager, the value of MAXAPPLS should be increased based on the following formula (to a maximum of 2000):

```
<MAXAPPLS> = 5 * (number of nodes) + (peak number of active
applications using the Data Links Manager)
```

Default values (range)

Automatic (Automatic; 1 - 60,000)

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using MAXAPPLS <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 200 for the Library Server database, and to 512 for the Resource Manager database.

Change these values depending on your environment, based on how many concurrent users you have.

Increasing the value of this parameter without lowering the MAXLOCKS parameter or increasing the LOCKLIST parameter could cause you to reach the database limit on locks (LOCKLIST) rather than the application limit and as a result cause pervasive lock escalation problems.

Keep in mind that an application can connect to the database only if there is an available connection (MAXAPPLS) as well as an available agent (MAXAGENTS). In addition, the maximum number of applications is also controlled by the MAX_COORDAGENTS configuration parameter, because no new applications (that is, coordinator agents) can be started if MAX_COORDAGENTS has been reached.

8.6.4 Average number of active applications (AVG_APPLS)

Impact

DB - Low

Description

This parameter specifies the average number of active applications. It is used by the SQL optimizer to help estimate how much buffer pool will be available at run time for each application. With a value of 1, for example, the optimizer assumes that the entire buffer pool will be available to one application.

Default values (range)

1 (1 - MAXAPPLS)

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using AVG_APPLS <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 5 for the Library Server database, which should be sufficient.

When running DB2 in a multi-user environment, particularly with complex queries and a large buffer pool, you might want the SQL optimizer to know that multiple query users are using your system; therefore, the optimizer would be more conservative in assumptions about buffer pool availability.

When setting this parameter, you should estimate the number of complex query applications that typically use the database. This estimate should exclude all light OLTP applications.

If you have trouble estimating this number, you can multiply the following two numbers::

- An average number of all applications running against your database. The database system monitor can provide information about the number of applications at any given time, and using a sampling technique, you can calculate an average over a period of time. The information from the database system monitor includes both OLTP and non-OLTP applications.
- Your estimate of the percentage of complex query applications.

As with adjusting other configuration parameters that affect the optimizer, you should adjust this parameter in small increments. This enables you to minimize path selection differences.

You should consider rebinding applications (using the **`rebind package`** command) after changing this parameter.

8.6.5 Maximum number of database manager agents (MAXAGENTS)

Impact

DBM - Medium

Description

This parameter indicates the maximum number of database manager agents, whether coordinator agents or subagents, that are available at any given time to accept application requests. If you want to limit the number of coordinating agents, use the MAX_COORDAGENTS parameter.

This parameter can be useful in memory-constrained environments to limit the total memory usage of the database manager, because each additional agent requires additional memory.

Default values (range)

200 (1 – 64,000)

400 (1– 64,000) on partitioned database servers with local and remote clients

To update

Run the following DB2 command:

```
db2 update dbm cfg using MAXAGENTS <parameter value>
```

Our recommendation

Content Manager changes this value to 500 during Library Server installation.

Use this value initially. If you have more clients connecting concurrently, change this value. The value of MAXAGENTS should be at least the sum of the values for MAXAPPLS in each database that is allowed to be accessed concurrently.

An agent is created for each Library Server connection. For each retrieve, one or more agents are created on Resource Manager. Each agent allocates memory. If you have memory constraints, this parameter can help you limit the memory usage by limiting the number of concurrent agents.

8.6.6 Maximum number of client connections (MAX_CONNECTIONS)

Impact

DBM - Medium

Description

This parameter controls the maximum number of applications that can be connected to the instance. Typically, each application is assigned a coordinator agent. An agent facilitates the operations between the application and the database.

When the default value (-1) is used, the connection concentrator feature (see 8.6.39, “Connection concentrator” on page 251) is not activated. The value of MAX_CONNECTIONS will equal the value of MAX_COORDAGENTS. As a result, each agent operates with its own private memory and shares database manager and database global resources such as the buffer pool with other agents.

When the parameter is set to a value greater than the default, then the connection concentrator is enabled. The intent of the concentrator is to reduce the server resources per client application to the point where a DB2 Connect gateway can handle more than 10,000 client connections.

Default values (range)

-1 (-1, MAX_COORDAGENTS – 64,000)

To update

Run the following DB2 command:

```
db2 update dbm cfg using MAX_CONNECTIONS <parameter value>
```

Our recommendation

Use the default.

8.6.7 Maximum coordinating agents (MAX_COORDAGENTS)

Impact

DBM - Medium

Description

When the connection concentrator is off, (when MAX_CONNECTIONS is equal to MAX_COORDAGENTS), this parameter determines the maximum number of coordinating agents that can exist at one time on a server in a partitioned or non-partitioned database environment.

One coordinating agent is acquired for each local or remote application that connects to a database or attaches to an instance. Requests that require an instance attachment include **`create database`**, **`drop database`**, and database system monitor commands.

When the connection concentrator is on (that is, when MAX_CONNECTIONS is greater than MAX_COORDAGENTS), there might be more connections than coordinator agents to service them. An application is in an active state only if there is a coordinator agent servicing it. Otherwise, the application is in an inactive state. Requests from an active application will be serviced by the database coordinator agent (and subagents in SMP or MPP configurations). Requests from an inactive application will be queued until a database coordinator agent is assigned to service the application. This application then becomes active. As a result, this parameter can be used to control the load on the system.

Default values (range)

-1 (-1, 0 - MAXAGENTS)

For non-partitioned database environments and environments in which INTRA_PARALLEL is set to no, MAX_COORDAGENTS must equal MAXAGENTS.

To update

Run the following DB2 command:

```
db2 update dbm cfg using MAX_COORDAGENTS <parameter value>
```

Our recommendation

Use the default.

8.6.8 Initial number of agents in pool (NUM_INITAGENTS)

Impact

DBM - Medium

Description

This parameter determines the initial number of idle agents that are created in the agent pool at **db2start** time.

Default values (range)

0 (0 - NUM_POOLAGENTS)

To update

Run the following DB2 command:

```
db2 update dbm cfg using NUM_INITAGENTS <parameter value>
```

Our recommendation

Define this value to be at least 50% of the MAXAGENTS value or to the value of NUM_POOLAGENTS. In general, when Content Manager connects to DB2 (client connection), an agent is already created and is available, thus avoiding the overhead of creating new agents.

8.6.9 Agent pool size (NUM_POOLAGENTS)

Impact

DBM - High

Description

When the connection concentrator is off (when MAX_CONNECTIONS is equal to MAX_COORDAGENTS), this parameter determines the maximum size of the idle agent pool. Idle agents can be used as parallel subagents or as coordinator agents. If more agents are created than is indicated by the value of this parameter, they will be terminated when they finish executing their current request, rather than be returned to the pool.

When the connection concentrator is enabled (when MAX_CONNECTIONS is greater than MAX_COORDAGENTS), agents will always be returned to the pool, no matter what the value of this parameter is. The number of agents specified here is only advisory. More agents might be in the agent pool at any given time.

Based on the system load and the time agents remain idle in the pool, agents may terminate themselves, as necessary, to reduce the size of the idle pool to the configured parameter value.

Default values (range)

-1 (-1, 0 - MAXAGENTS)

To update

Run the following DB2 command:

```
db2 update dbm cfg using NUM_POOLAGENTS <parameter value>
```

Our recommendation

Use the default -1 (MAXAGENTS), which will enable DB2 to calculate the value.

If you have memory constraints, consider reducing this value gradually, down to 50% of MAXAGENTS. Otherwise, the general rule is to reduce this parameter value if the system has few applications connected concurrently, and to increase the value if the system has many applications connected concurrently.

Use the database manager snapshot to monitor this element. If the ratio of “Agents created from empty pool” divided by “Agents assigned from pool” is high (5:1 or more), increase NUM_POOLAGENTS. If the ratio is low, then NUM_POOLAGENTS is probably set too high and there are some agents wasting system resources.

8.6.10 Maximum number of fenced processes (FENCED_POOL)

Impact

DBM - Medium

Description

For threaded **db2fmp** processes (processes serving thread-safe stored procedures and user-defined functions), this parameter represents the number of threads cached in each **db2fmp** process. For non-threaded **db2fmp** processes, this parameter represents the number of processes that are cached.

If your environment uses fenced stored procedures or user-defined functions, then this parameter can be used to ensure that an appropriate number of **db2fmp** processes are available to process the maximum number of concurrent stored procedures and user-defined functions, which run on the instance.

If the database manager configuration parameter KEEPFENCED is set to yes (which is the default), then each **db2fmp** process that is created in the cache pool will continue to exist and use system resources even after the fenced routine call has been processed and returned to the agent.

If KEEPFENCED is set to no, then non-threaded **db2fmp** processes will terminate when they complete execution, and there is no cache pool. Multi-threaded **db2fmp** processes will continue to exist, but no threads will be pooled in these processes. This means that even when KEEPFENCED is set no, you can have one threaded C **db2fmp** process, and one threaded Java **db2fmp** process on your system.

Default values (range)

-1 (MAX_COORDAGENTS)

To update

Run the following DB2 command:

```
db2 update dbm cfg using FENCED_POOL <parameter value>
```

Our recommendation

Library Server relies heavily on stored procedures. We recommend that you maintain enough **db2fmp** processes for them. The best way is by using the value -1, which sets the value equal to that of MAX_COORDAGENTS.

If you have memory constraints and it is affecting database manager performance, think about reducing this value; but keep in mind the overhead of creating new processes when there are no more processes available to process the concurrent stored procedures.

8.6.11 Keep fenced process (KEEPFENCED)

Impact

DBM - Medium

Description

This parameter indicates whether a fenced mode process is kept after a fenced mode routine call is complete. Fenced mode processes are created as separate system entities in order to isolate user-written fenced mode code from the database manager agent process.

Default values (range)

Yes (Yes; No)

To update

Run the following DB2 command:

```
db2 update dbm cfg using KEEPFENCED <parameter value>
```

Our recommendation

Library Server relies heavily on stored procedures. We recommend that you use the default value -1.

8.6.12 Initial number of fenced processes (NUM_INITFENCED)

Impact

DBM - Medium

Description

This parameter indicates the initial number of non-threaded, idle **db2fmp** processes that are created in the **db2fmp** pool at **db2start** time. Setting this parameter will reduce the initial startup time for running non-threadsafe C and COBOL routines.

This parameter is ignored if KEEPFENCED is not specified.

It is much more important to set FENCED_POOL to an appropriate size for your system than to start up a number of **db2fmp** processes at **db2start** time.

Default values (range)

0 (0 to max_connections + (maxagents - max_coordagents))

To update

Run the following DB2 command:

```
db2 update dbm cfg using NUM_INITFENCED <parameter value>
```

Our recommendation

We recommend that you set this parameter to equal the value of parameter NUM_INITAGENTS.

8.6.13 Maximum database files open per application (MAXFILOP)

Impact

DB - Medium

Description

This parameter specifies the maximum number of file handles that can be open for each database agent. If opening a file causes this value to be exceeded, some files in use by this agent will be closed. If MAXFILOP is too small, the overhead of opening and closing files so as not to exceed this limit will become excessive and might degrade performance.

Both SMS table spaces and DMS table space file containers are treated as files in the database manager's interaction with the operating system, and file handles are required. More files are generally used by SMS table spaces compared to the number of containers used for a DMS file table space. If you are using SMS table spaces, you will need a larger value for this parameter compared to what you would require for DMS file table spaces.

You can also use this parameter to ensure that the overall total of file handles used by the database manager does not exceed the operating system limit by limiting the number of handles per agent to a specific number; the actual number will vary depending on the number of agents running concurrently.

Default values (range)

64 (2 - 1,950) for UNIX platforms

64 (2 - 32,768) for Windows platforms

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using MAXFILOP <parameter value>
```

Our recommendation

Use the defaults to start with. Monitor the database snapshot value for "Database files closed." Ideally, the value should be zero.

If you detect excessive opening and closing of files, increase MAXFILOP in small amounts until the "Database files closed" snapshot value becomes zero; be sure that the overall total of file handles does not exceed the operating system limit.

8.6.14 Maximum total files open (MAXTOTFILOP)

Impact

DBM - Medium

Description

This parameter defines the maximum number of files that can be opened by all agents and other threads executing in a single database manager instance. If opening a file causes this value to be exceeded, an error is returned to your application.

Note: This parameter does not apply to UNIX-based platforms.

Default values (range)

16,000 (100 – 32,768)

To update

Run the following DB2 command:

```
db2 update dbm cfg using MAXTOTFILOP <parameter value>
```

Our recommendation

Keep the default value to start with. Monitor the total number of files open in general and increase this value when you need to.

8.6.15 Database heap (DBHEAP)

Impact

DB - Medium

Description

There is one database heap per database, and the database manager uses it on behalf of all applications connected to the database. It contains control block information for tables, indexes, table spaces, and buffer pools. It also contains space for the log buffer (LOGBUFSZ), catalog cache (CATALOGCACHE_SZ), and temporary memory used by utilities. Therefore, the size of the heap will be dependent on a large number of variables. The control block information is kept in the heap until all applications disconnect from the database.

The minimum amount the database manager needs to get started is allocated at the first connection. The data area is expanded as needed up to the maximum specified by DBHEAP.

Default values (range)

Automatic (32 - 524,288) 4-KB pages

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using DBHEAP <parameter value>
```

Our recommendation

Use the default values to start with. As you increase related variables such as CATALOGCACHE_SZ and LOGBUFSZ, you will have to increase DBHEAP. Use the database system monitor to track the highest amount of memory that was used for the database heap, using the DB_HEAP_TOP (maximum database heap allocated) element.

8.6.16 Catalog cache size (CATALOGCACHE_SZ)

Impact

DB - High

Description

This parameter is allocated out of the database shared memory, and is used to cache system catalog information. In a partitioned database system, there is one catalog cache for each database partition.

Caching catalog information at individual partitions enables the database manager to reduce its internal overhead by eliminating the need to access the system catalogs (or the catalog node in an partitioned database environment) to obtain information that has previously been retrieved.

The catalog cache is used to store:

- SYSTABLES information (including packed descriptors)
- Authorization information, including SYSDBAUTH information and execute privileges for routines
- SYSROUTINES information

The use of the catalog cache can help improve the overall performance of:

- Binding packages and compiling SQL statements
- Operations that involve checking database-level privileges
- Operations that involve checking execute privileges for routines

- Applications that are connected to non-catalog nodes in a partitioned database environment

By taking the default (-1) in a server or partitioned database environment, the value that is used to calculate the page allocation is four times the MAXAPPLS configuration parameter. The exception to this occurs if four times MAXAPPLS is less than 8. In this situation, the default value of -1 will set CATALOGCACHE_SZ to 8.

Default values (range)

-1 (8 – 524,288) 4-KB pages

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using CATALOGCACHE_SZ <parameter
value>
```

Our recommendation

Start with the default value (-1).

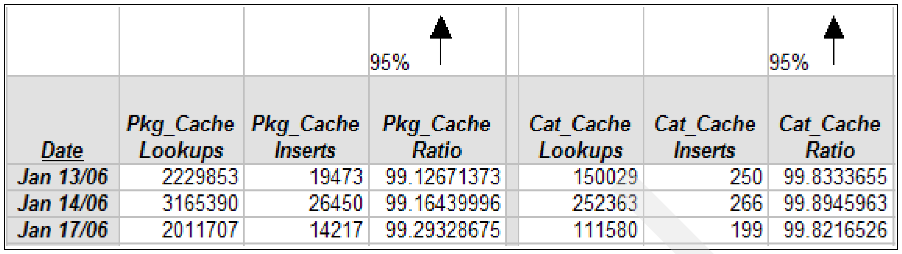
Use the database system monitor (snapshots) to determine whether this number needs to be increased. The snapshot monitor elements of importance are:

- Catalog cache lookups (`cat_cache_lookups`): the number of times that the catalog cache was referenced to obtain table descriptor or authorization information.
- Catalog cache inserts (`cat_cache_inserts`): the number of times that the system tried to insert table descriptor or authorization information into the catalog cache.
- Catalog cache overflows (`cat_cache_overflows`): the number of times that the catalog cache overflowed the bounds of its allocated memory. If the number of overflows is large, the catalog cache might be too small for the workload and should be increased.

The catalog cache hit ratio indicates how well the catalog cache is avoiding catalog accesses. A ratio of above 80% indicates that the cache is performing well. The formula to calculate the catalog cache hit ratio is:

```
(1 - (cat_cache_inserts / cat_cache_lookups)) * 100
```

Figure 8-2 on page 221 shows an example of a Lotus 1-2-3 spreadsheet using the snapshot elements to compute catalog cache and package size hit ratio.



			95%			95%
Date	Pkg_Cache Lookups	Pkg_Cache Inserts	Pkg_Cache Ratio	Cat_Cache Lookups	Cat_Cache Inserts	Cat_Cache Ratio
Jan 13/06	2229853	19473	99.12671373	150029	250	99.8333655
Jan 14/06	3165390	26450	99.16439996	252363	266	99.8945963
Jan 17/06	2011707	14217	99.29328675	111580	199	99.8216526

Figure 8-2 Hit ratios in a spreadsheet

The Health Monitor also gives you information about this parameter. When tuning this parameter, you should consider whether the extra memory being reserved for the catalog cache might be more effective if it were allocated for another purpose, such as the buffer pool or package cache.

Tuning this parameter is particularly important if a workload involves many SQL compilations for a brief period of time, with few or no SQL compilations thereafter. If the cache is too large, memory might be wasted holding copies of information that will no longer be used.

In general, more cache space is required if a unit of work contains several dynamic SQL statements, or if you are binding packages that contain a large number of static SQL statements.

8.6.17 Package cache size (PCKCACHESZ)

Impact

DB - High

Description

This parameter is allocated out of the database shared memory, and is used for caching of sections for static and dynamic SQL statements on a database. In a partitioned database system, there is one package cache for each database partition.

Caching packages enables the database manager to reduce its internal overhead by eliminating the need to access the system catalogs when reloading a package; or, in the case of dynamic SQL, eliminating the need for compilation.

Sections are kept in the package cache until one of the following actions occurs:

- The database is shut down.
- The package or dynamic SQL statement is invalidated.
- The cache runs out of space.

This caching of the section for a static or dynamic SQL statement can improve performance, especially when the same statement is used multiple times by applications connected to a database. This is particularly important in a transaction-processing application.

By taking the default (-1), the value used to calculate the page allocation is eight times the value specified for the MAXAPPLS configuration parameter. The exception to this occurs if eight times MAXAPPLS is less than 32. In this situation, the default value of -1 will set PCKCACHESZ to 32.

Default values (range)

-1 (-1, 32 - 128,000) 4-KB pages for 32-bit platforms

-1 (-1, 32 - 524,288) 4-KB pages for 64-bit platforms

To update

Run the following command:

```
db2 update db cfg for <database name> using PCKCACHESZ <parameter value>
```

Our recommendation

Start with using the default.

Use the database system monitor (snapshots) to determine whether this number should be increased. The snapshot monitor elements of importance are:

- Package cache lookups (`pkg_cache_lookups`): the number of times that an application looked for a section or package in the package cache.
- Package cache inserts (`pkg_cache_inserts`): the total number of times that a requested section was not available for use and had to be loaded into the package cache.
- Package cache overflows (`pkg_cache_num_overflows`): the number of times that the package cache overflowed the bounds of its allocated memory. If the number of overflows is large, the package cache might be too small for the workload and should be increased.

The package cache hit ratio tells you whether the package cache is being used effectively. A ratio of more than 80% indicates that the cache is performing well. The formula to calculate the package cache hit ratio is:

```
(1 - (pkg_cache_inserts / pkg_cache_lookups)) * 100
```

If you do consider tuning this parameter, you should think whether the extra memory being reserved for the package cache might be more effective if it is allocated for another purpose, such as the buffer pool or catalog cache. For this reason, you should use benchmarking techniques when tuning this parameter.

Tuning this parameter is particularly important when several sections are used initially, and then only a few are run repeatedly. If the cache is too large, memory is wasted holding copies of the initial sections.

The limit specified by the PCKCACHESZ parameter is a soft limit. This limit may be exceeded, if required and if memory is still available in the database shared set. You can use the PKG_CACHE_SIZE_TOP monitor element to determine the largest that the package cache has grown.

Note: The package cache is a working cache, so you cannot set this parameter to zero. There must be sufficient memory allocated in this cache to hold all sections of the SQL statements that are currently being run. If there is more space allocated than currently needed, then sections are cached. These sections can simply be run the next time they are needed without having to load or compile them.

8.6.18 Log buffer size (LOGBUFSZ)

Impact

DB - High

Description

This parameter enables you to specify the amount of the database heap (defined by the DBHEAP parameter) to use as a buffer for log records before writing these records to disk.

The log records are written to disk when one of the following actions occurs:

- A transaction commits or a group of transactions commit, as defined by the MINCOMMIT configuration parameter.
- The log buffer is full.
- As a result of some other internal database manager event.

This parameter must be less than or equal to the DBHEAP parameter. Buffering the log records will result in more efficient logging file I/O because the log records will be written to disk less frequently, and more log records will be written at each time.

Default values (range)

8 (4 - 4,096) 4-KB pages for 32-bit platforms

8 (4 - 65,535) 4-KB pages for 64-bit platforms

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using LOGBUFSZ <parameter value>
```

Our recommendation

Start with the default. Increase the size of this buffer area if there is considerable read activity on a dedicated log disk, or there is high disk utilization.

When increasing the value of this parameter, you should also consider increasing the DBHEAP parameter because the log buffer area uses space controlled by DBHEAP.

You can use the database system monitor to determine:

- Maximum total log space used (TOT_LOG_USED_TOP): The maximum amount of total log space used (in bytes). This value includes space used in both the primary and secondary log files.
- Unit of work log space used (UOW_LOG_SPACE_USED): Log buffer space used for a particular transaction (or unit of work).

8.6.19 Maximum storage for lock list (LOCKLIST)

Impact

DB - High

Description

This parameter indicates the maximum amount of storage that is allocated to the lock list. There is one lock list per database and it contains the locks held by all applications concurrently connected to the database. Locking is the mechanism that the database manager uses to control concurrent access to data in the database by multiple applications. Both rows and tables can be locked. The database manager may also acquire locks for internal use.

On 32-bit platforms, each lock requires 36 or 72 bytes of the lock list, depending on whether other locks are held on the object:

- 72 bytes are required to hold a lock on an object that has no other locks held on it.
- 36 bytes are required to record a lock on an object that has an existing lock held on it.

On 64-bit platforms, each lock requires 56 or 112 bytes of the lock list, depending on whether other locks are held on the object:

- 112 bytes are required to hold a lock on an object that has no other locks held on it.
- 56 bytes are required to record a lock on an object that has an existing lock held on it.

When the percentage of the lock list used by one application reaches MAXLOCKS, the database manager will perform lock escalation, from row to table, for the locks held by the application (described below). Although the escalation process itself does not take much time, locking entire tables (versus individual rows) decreases concurrency, and overall database performance might decrease for subsequent accesses against the affected tables.

When the lock list is full, performance can degrade because lock escalation will generate more table locks and fewer row locks, thus reducing concurrency on shared objects in the database. Additionally, there could be more deadlocks between applications (because they are all waiting on a limited number of table locks), which results in transactions being rolled back. An application gets -912 when the maximum number of lock requests has been reached for the database.

Default values (range)

100 (4 - 524,288) 4-KB pages for UNIX

50 (4 - 524,288) 4-KB pages for Windows database with local and remote clients

50 (4 - 60,000) 4-KB pages for Windows 64-bit database server with local clients

25 (4 - 60,000) 4-KB pages for Windows 32-bit database server with local clients

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using LOCKLIST <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 1000 for both the Library Server and the Resource Manager databases.

Initially, use these default values. Use the snapshot command to monitor your databases. Keep an eye on the LOCK_ESCALS (lock escalations) monitor element. If you are having problems with lock escalation, you might need to increase the value of this parameter or the MAXLOCKS parameter.

Also watch LOCK_LIST_IN_USE (total lock list memory in use). If this element is high (exceeds 50% of the defined LOCKLIST size), you might want to consider increasing the size of the parameter. Note that the monitor element is reported in bytes, but the LOCKLIST parameter is in 4-kilobyte pages.

To determine the maximum number of locks held by a given transaction, check the LOCKS_HELD_TOP (maximum number of locks held) monitor element. The monitor information is provided at a transaction level, not an application level.

The following steps help you determine the number of pages required for your lock list:

1. Calculate a lower bound for the size of your lock list using one of the following formulas, depending on your environment:
   - `(512 * x * MAXAPPLS) / 4096`
   - With connection concentrator enabled:
     `(512 * x * MAX_COORDAGENTS) / 4096`

   512 is an estimate of the average number of locks per application and *x* is the number of bytes required for each lock against an object that has an existing lock (36 bytes on 32-bit platforms, 56 bytes on 64-bit platforms).
2. Calculate an upper bound for the size of your lock list using the formula:

   `(512 * y * MAXAPPLS) / 4096`

   y is the number of bytes required for the first lock against an object: 72 bytes on 32-bit platforms, 112 bytes on 64-bit platforms.
3. Estimate the amount of concurrency you will have against your data and based on your expectations; choose an initial value for locklist that falls between the upper and lower bounds, which you have calculated.
4. Using the database system monitor, tune the value of this parameter.

You might also want to increase LOCKLIST if MAXAPPLS is increased, or if the applications being run perform infrequent commits.

You should consider performing a **`rebind`** after changing this parameter.

8.6.20 Application control heap size (APP_CTL_HEAP_SZ)

Impact

DB - Medium

Description

For partitioned databases, and for non-partitioned databases with intra-parallelism enabled, this parameter specifies the average size of the shared memory area allocated for an application. For non-partitioned databases where intra-parallelism is disabled, this is the maximum private memory that will be allocated for the heap. There is one application control heap per connection per partition.

The application control heap is required primarily for sharing information between agents working on behalf of the same request. Use of this heap is minimal for non-partitioned databases when running queries with a degree of parallelism equal to 1.

This heap is also used to store descriptor information for declared temporary tables. The descriptor information for all declared temporary tables that have not been explicitly dropped is kept in this heap's memory and cannot be dropped until the declared temporary table is dropped.

Default values (range)

128 (1 - 64,000) pages for database server with local and remote clients

64 (1 - 64,000) pages for database server with local clients, non-UNIX platforms

128 (1 - 64,000) pages for database server with local clients, UNIX platforms

256 (1 - 64,000) pages for partitioned database server with local and remote clients

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using APP_CTL_HEAP_SZ <parameter
value>
```

Our recommendation

The Content Manager installation program changes this to 1000 for the Library Server and 128 for the Resource Manager.

Use these default values. If you have memory constraints, reduce the value on the Library Server database.

8.6.21 Application heap size (APPLHEAPSZ)

Impact

DB - Medium

Description

This parameter defines the number of private memory pages that are available for use by the database manager on behalf of a specific agent or subagent.

The heap is allocated when an agent or subagent is initialized for an application. The amount that is allocated will be the minimum amount needed to process the request given to the agent or subagent. As the agent or subagent requires more heap space to process larger SQL statements, the database manager will allocate more memory as needed, up to the maximum specified by this parameter.

In a partitioned database environment, APP_CTL_HEAP_SZ is used to store copies of the executing sections of SQL statements for agents and subagents. SMP subagents use APPLHEAPSZ as do agents in all other environments.

Default values (range)

256 (16 - 60,000) 4-KB pages

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using APPLHEAPSZ <parameter
value>
```

Our recommendation

The Content Manager installation program sets this value to 4,096 for the Library Server and the Resource Manager. Use this value initially.

Increase this if your applications receive an error indicating insufficient storage in the application heap. APPLHEAPSZ is allocated out of agent private memory. Be sure that you have enough private memory before increasing this number.

8.6.22 Sort heap size (SORTHEAP)

Impact

DB - High

Description

This parameter defines the maximum number of private memory pages to be used for private sorts, or the maximum number of shared memory pages to be used for shared sorts. If the sort is a private sort, then this parameter affects agent private memory. If the sort is a shared sort, then this parameter affects the database shared memory. Each sort has a separate sort heap that is allocated as needed by the database manager. This sort heap is the area where data is sorted. If directed by the optimizer, a smaller sort heap than the one specified by this parameter is allocated using information provided by the optimizer.

Default values (range)

256 (16 - 524,288) 4-KB pages for 32-bit platforms

256 (16 - 1,048,575) 4-KB pages for 64-bit platforms

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using SORTHEAP <parameter value>
```

Our recommendation

The default values for both the Library Server and the Resource Manager will not be sufficient and will have to be increased in conjunction with database monitoring. Increasing this value unnecessarily can cause memory problems.

The database snapshot monitor elements of importance are:

- Total sorts (`total_sorts`): the total number of sorts that have been performed.
- Sort overflows (`sort_overflows`): the total number of sorts that ran out of sort heap and might have required disk space for temporary storage.
- Number of hash join overflows (`hash_join_overflows`): the number of times that hash join data exceeded the available sort heap space.

Sort overflows divided by total sorts will provide the percentage of sorts that had to overflow to disk. If this percentage is high, you should increase the value of SORTHEAP.

If the number of hash join overflows is not zero, increase the value of SORTHEAP by increments of 256 until it is.

When tuning this parameter, you should consider:

- Appropriate indexes can minimize the use of the sort heap.
- Hash join buffers and dynamic bitmaps use sort heap memory. Increase the size of this parameter when these techniques are used.
- You might increase the size of this parameter when frequent large sorts are required.
- When increasing the value of this parameter, examine whether the SHEAPTHRES parameter in the database manager configuration file also should be adjusted.
- The sort heap size is used by the optimizer in determining access paths.

Consider performing a **`rebind`** of the applications after changing this parameter.

8.6.23 Sort heap threshold (SHEAPTHRES)

Impact

DBM - High

Description

The SHEAPTHRES parameter is used differently for private and shared sorts:

- For private sorts, this parameter is an instance-wide *soft* limit on the total amount of memory that can be consumed by private sorts at any given time. The size of the private sort memory is unrestricted. When the total private-sort memory consumption for an instance reaches this limit, the memory allocated for additional incoming private-sort requests will be reduced considerably.
- For shared sorts, this parameter is a database-wide *hard* limit on the total amount of memory consumed by shared sorts at any given time. The size of the shared sort memory area is statically predetermined at the time of the first connection to a database based on SHEAPTHRES. When this limit is reached, no further shared-sort memory requests will be allowed (until the total shared-sort memory consumption falls below the limit specified by SHEAPTHRES).

Examples of those operations that use the sort heap include: sorts, hash joins, dynamic bitmaps, and operations where the table is in memory.

Explicit definition of the threshold prevents the database manager from using excessive amounts of memory for large numbers of sorts.

Generally, you do not need to change the value of this parameter when moving from a non-partitioned to a partitioned database environment. After you have tuned the database and database manager configuration parameters on a single database partition environment, in most cases the same values will work well in a partitioned database environment.

The SHEAPTHRES value applies across the entire DB2 instance. To update this parameter to different values on different nodes or partitions, create more than one DB2 instance.

Default values (range)

20,000 (250 - 2,097,152) 4-KB pages for UNIX 32-bit platforms

10,000 (250 - 2,097,152) 4-KB pages for Windows platforms

20,000 (250 - 2,147,483,647) 4-KB pages for 64-bit platforms

To update

Run the following DB2 command:

```
db2 update dbm cfg using SHEAPTHRES <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 10,000 during Library Server installation.

Use this value initially. Be sure that it is a multiple of the largest sortheap parameters in your database instance. This parameter should be at least two times the largest SORTHEAP defined for any database within the instance.

If you are doing private sorts and your system is not memory constrained, you may consider setting the parameter value as follows:

1. Calculate the typical sort heap usage for each database:

   ```
   TypicalConcurrentAgents * SORTHEAP
   ```

2. Add the typical sort heap for all the databases to get the total sort heap.

Monitor your system using snapshot to verify that your system is healthy. Try to find the proper balance between sort performance and memory usage. You can use the database manager system monitor to track the sort activity, using the post threshold sorts (POST_THRESHOLD_SORTS) monitor element, which indicates the number of sorts that have requested heaps after the sort heap threshold has been exceeded. If this element value is a double digit, increase the sort heap threshold.

8.6.24 Statement heap size (STMTHEAP)

Impact

DB - Medium

Description

The statement heap is used as a work space for the SQL compiler during compilation of an SQL statement. This parameter specifies the size of this work space.

This area does not stay permanently allocated, but is allocated and released for every SQL statement handled. Note that for dynamic SQL statements, this work area will be used during execution of your program. For static SQL statements, it is used during the bind process, but not during program execution.

Default values (range)

2,048 pages (128 - 65,535) 4-KB pages

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using STMTHEAP <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 16,384 for the Library Server database.

We recommend using this default value. If your application uses very large SQL statements and an error occurs when optimizing a statement (stating that the statement is too complex), increase this value in increments (such as 256 or 1024) until the error situation is resolved.

8.6.25 Query heap size (QUERY_HEAP_SZ)

Impact

DBM - Medium

Description

This parameter specifies the maximum amount of memory that can be allocated for the query heap. A query heap is used to store each query in the agent's private memory. The information for each query consists of the input and output SQLDA, the statement text, the SQLCA, the package name, creator, section number, and consistency token. When an application (either local or remote) connects to the database, query heap is allocated. When the application disconnects from the database or detaches from the instance, it is freed. This parameter ensures that an application does not consume unnecessarily large amounts of virtual memory within an agent.

The query heap is also used for the memory allocated for blocking cursors. This memory consists of a cursor control block and a fully resolved output SQLDA.

The initial allocated query heap will be the same size as the application support layer heap, as specified by the ASLHEAPSZ parameter. The query heap size must be greater than or equal to 2, and must be greater than or equal to the ASLHEAPSZ parameter. If this query heap is not large enough to handle a given request, it will be reallocated to the size required by the request (not exceeding QUERY_HEAP_SZ). If this new query heap is more than 1.5 times larger than ASLHEAPSZ, the query heap will be reallocated to the size of ASLHEAPSZ when the query ends.

Default values (range)

1,000 (2 – 524,288) 4-KB pages

To update

Run the following DB2 command:

```
db2 update dbm cfg using QUERY_HEAP_SZ <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 32,768 during the Library Server installation, and the program changes the value to 16,384 during the Resource Manager installation. If you install both servers on the same machine and on the same instance, the value set by the Resource Manager installation will be used for both servers. You should change the value back to 32,768.

Use this value initially. If you have memory constraints, reduce the value — but not lower than five times the value of the ASLHEAPSZ. This will allow for queries larger than ASLHEAPSZ and provides additional memory for three or four blocking cursors to be open at a given time.

If you have very large LOBs, you might need to increase the value of this parameter for the query heap to be large enough to accommodate those LOBs.

8.6.26 Java interpreter heap size (JAVA_HEAP_SZ)

Impact

DBM - High

Description

This parameter determines the maximum size of the heap that is used by the Java interpreter that was started to service Java DB2 stored procedures and UDFs.

There is one heap for each DB2 process (one for each agent or subagent on UNIX-based platforms, and one for each instance on other platforms). There is one heap for each fenced UDF and fenced stored procedure process. There is one heap per agent (not including sub-agents) for trusted routines. There is one heap per **db2fmp** process running a Java stored procedure. For multi-threaded **db2fmp** processes, multiple applications using threadsafe fenced routines are serviced from a single heap.

In all situations, only the agents or processes that run Java UDFs or stored procedures ever allocate this memory. On partitioned database systems, the same value is used at each partition. The memory is freed when the **db2fmp** process (fenced) or **db2agent** process (trusted) terminates.

Default value (range)

512 (0 - 524,288) 4-KB pages

To update

Run the following DB2 command:

```
db2 update dbm cfg using JAVA_HEAP_SZ <parameter value>
```

Our recommendation

We suggest increasing this parameter to 2048. If your are having a memory bottleneck, consider reducing the value of this parameter. Library Server heavily relies on stored procedures, which depend on how this parameter is set. Reducing this value will make more memory available.

8.6.27 Private memory threshold (PRIV_MEM_THRESH)

Impact

DBM - Medium

Description

This parameter is used to determine the amount of an unused agent's private memory that will be kept allocated. When an agent is terminated, instead of automatically deallocating all of the memory that was used by that agent, the database manager will deallocate only excess memory allocations as determined by the following formula:

```
PrivateMemoryAllocated - (PrivateMemoryUsed + PRIV_MEM_THRESH)
```

If this formula produces a negative result, no action will be taken.

If this parameter value is -1, then it uses the MIN_PRIV_MEM value.

Note: This parameter does not apply to UNIX-based platforms.

Default values (range)

1,296 (-1; 32 – 112,000) 4-KB pages

To update

Run the following DB2 command:

```
db2 update dbm cfg using PRIV_MEM_THRESH <parameter value>
```

Our recommendation

When setting this parameter, you should consider the client connection and disconnection patterns as well as the memory requirements of other applications on the same server.

If you have memory constraints, reduce this value. If you are running other applications on the same server that Content Manager is running, and the other applications need more memory, reduce this value to make more memory available for other applications.

If memory is not a problem, and you have a constant number of users connecting and disconnecting to your system, use the default or increase this value to avoid the overhead of allocating and deallocating memory.

8.6.28 Client I/O block size (RQRIOBLK)

Impact

DBM - Medium

Description

This parameter specifies the size of the communication buffer between remote applications and their database agents on the database server. When a database client requests a connection to a remote database, this communication buffer is allocated on the client. On the database server, a communication buffer of 32,767 bytes is initially allocated, until a connection is established and the server can determine the value of RQRIOBLK at the client. As soon as the server knows this value, it will reallocate its communication buffer if the client's buffer is not 32,767 bytes. When the remote application disconnects from the server database, the communication buffer is freed.

In addition to this communication buffer, this parameter is also used to determine the I/O block size at the database client when a blocking cursor is opened. This memory for blocked cursors is allocated out of the application's private address space, so you should determine the optimal amount of private memory to allocate for each application program. It is freed when the blocking cursor is closed. If the database client cannot allocate space for a blocking cursor out of an application's private memory, a non-blocking cursor will be opened.

Default values (range)

32,767 (4,096 – 65,535) bytes

To update

Run the following DB2 command:

```
db2 update dbm cfg using RQRIOBLK <parameter value>
```

Our recommendation

Use the default.

For non-blocking cursors, consider increasing the value if the data (for example, large object data) to be transmitted by a single SQL statement is so large that the default value is insufficient.

You should also consider the effect of this parameter on the number and potential size of blocking cursors. Large row blocks might yield better performance if the number or size of rows being transferred is large (for example, if the amount of data is greater than 4 KB). The trade-off is that larger record blocks increase the size of the working set memory for each connection.

Larger record blocks might also cause more fetch requests than are actually required by the application. You can control the number of fetch requests using the `optimize for` clause on the `select` statement in your application.

8.6.29 Application support layer heap size (ASLHEAPSZ)

Impact

DBM - High

Description

The application support layer heap represents a communication buffer between the local application and its associated agent. This buffer is allocated as shared memory by each database manager agent that is started. It is freed when the database agent process is terminated.

If the request to the database manager or its associated reply do not fit into the buffer, they will be split into two or more send-and-receive pairs. The size of this buffer should be set to handle the majority of requests using a single send-and-receive pair. The size of the request is based on the storage required to hold:

- The input SQLDA
- All of the associated data in the SQLVARs
- The output SQLDA
- Other fields that do not generally exceed 250 bytes

In addition to this communication buffer, this parameter is used for two other purposes:

- It is used to determine the I/O block size when a blocking cursor is opened. This memory for blocked cursors is allocated out of the application's private address space, so you should determine the optimal amount of private memory to allocate for each application program. If the database client cannot allocate space for a blocking cursor out of an application's private memory, a non-blocking cursor will be opened.
- It is used to determine the communication size between agents and **`db2fmp`** processes. (A **`db2fmp`** process can be a user-defined function or a fenced stored procedure.) The number of bytes is allocated from shared memory for each **`db2fmp`** process or thread that is active on the system.

The data sent from the local application is received by the database manager into a set of contiguous memory allocated from the query heap. The ASLHEAPSZ parameter is used to determine the initial size of the query heap (for both local and remote clients). The maximum size of the query heap is defined by the QUERY_HEAP_SZ parameter.

Default values (range)

15 (1 - 524,288) 4-KB pages

To update

Run the following DB2 command:

```
db2 update dbm cfg using ASLHEAPSZ <parameter value>
```

Our recommendation

Use the default value.

If your application's requests are generally small and the application is running on a memory-constrained system, you could reduce the value of this parameter. If your queries are generally very large, requiring more than one send and receive request, and your system is not constrained by memory, you could increase the value of this parameter.

Use the following formula to calculate a minimum number of pages for ASLHEAPSZ:

```
ASLHEAPSZ >=(sizeof(input SQLDA)
   + sizeof(each input SQLVAR)
   + sizeof(output SQLDA)
   + 250 )/4096
```

where `sizeof(`*x*`)` is the size of *x* in bytes that calculates the number of pages of a given input or output value.

You should also consider the effect of this parameter on the number and potential size of blocking cursors. Large row blocks might yield better performance if the number or size of rows being transferred is large (for example, if the amount of data is greater than 4 KB). However, there is a trade-off in that larger record blocks increase the size of the working set memory for each connection.

Larger record blocks can also cause more fetch requests than are actually required by the application. You can control the number of fetch requests using the `optimize for` clause on the `select` statement in your application.

8.6.30 Changed pages threshold (CHNGPGS_THRESH)

Impact

DB - High

Description

Asynchronous page cleaners write changed pages from buffer pools to disk before the space in the buffer pool is required by a database agent. As a result, database agents can use the space in the buffer pool without having to wait for changed pages to be written out. This improves overall performance of the database applications.

This parameter specifies the percentage of changed pages at which the asynchronous page cleaners will start, if they are not currently active. When the page cleaners are started, they build a list of the pages to write to disk. After they have completed writing those pages to disk, they become inactive again and wait for the next trigger to start.

Default values (range)

60 (5 – 99) percent

To update

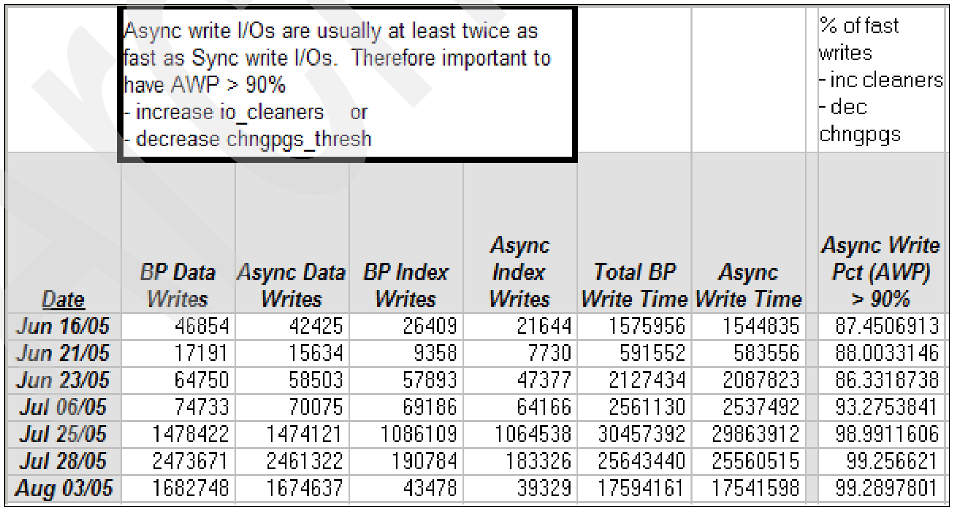
Run the following DB2 command:

```
db2 update db cfg for <database name> using CHNGPGS_THRESH <parameter
value>
```

Our recommendation

Content Manager has some very large tables. Decrease this value slightly each time and check the results using **`db2 get snapshot`** command, until you find an ideal value for your environment. Do not go lower than 30.

Figure 8-3 shows an example of a Lotus 1-2-3 spreadsheet using the snapshot elements to compute the asynchronous write percentage, which helps determine whether the changed pages threshold is set correctly.



Async write I/Os are usually at least twice as fast as Sync write I/Os. Therefore important to have AWP > 90%
- increase io_cleaners or
- decrease chngpgs_thresh

% of fast writes
- inc cleaners
- dec chngpgs

Date	BP Data Writes	Async Data Writes	BP Index Writes	Async Index Writes	Total BP Write Time	Async Write Time	Async Write Pct (AWP) > 90%
Jun 16/05	46854	42425	26409	21644	1575956	1544835	87.4506913
Jun 21/05	17191	15634	9358	7730	591552	583556	88.0033146
Jun 23/05	64750	58503	57893	47377	2127434	2087823	86.3318738
Jul 06/05	74733	70075	69186	64166	2561130	2537492	93.2753841
Jul 25/05	1478422	1474121	1086109	1064538	30457392	29863912	98.9911606
Jul 28/05	2473671	2461322	190784	183326	25643440	25560515	99.256621
Aug 03/05	1682748	1674637	43478	39329	17594161	17541598	99.2897801

Figure 8-3 Async write spreadsheet

The async write percentage gives you an indication of how the cleaners and changed pages threshold are performing. A percentage over 90% is desirable. This is the formula to calculate the percentage:

```
((Async pool data writes + Async pool index writes) / (Buffer pool data
writes + Buffer pool index writes)) * 100
```

For databases with a heavy update transaction workload, you can generally ensure that there are enough clean pages in the buffer pool by setting the parameter value to be equal to or less than the default value. A percentage larger than the default can help performance if your database has a small number of very large tables.

Starting with DB2 version 8.1.4, there is an alternative method of configuring page cleaning. This alternative method differs from the default behavior in that page cleaners behave more proactively in choosing which dirty pages get written out at any given time. This new method differs from the default page cleaning method in two major ways:

- Page cleaners do not consider the CHNGPGS_THRESH configuration parameter.
- Page cleaners no longer respond to LSN gap triggers issued by the logger.

To use the new method of page cleaning, set the DB2_USE_ALTERNATIVE_PAGE_CLEANING registry variable to ON.

8.6.31 Number of asynchronous page cleaners (NUM_IOCLEANERS)

Impact

DB - High

Description

This parameter enables you to specify the number of asynchronous page cleaners for a database. These page cleaners write changed pages from the buffer pool to disk before the space in the buffer pool is required by a database agent. As a result, database agents should not have to wait for changed pages to be written out so that they might use the space in the buffer pool. This improves overall performance of the database applications.

If you set the parameter to zero, no page cleaners are started, and as a result, the database agents will perform all of the page writes from the buffer pool to disk. This parameter can have a significant performance impact on a database stored across many physical storage devices, because in this case there is a greater chance that one of the devices will be idle. If no page cleaners are configured, your applications could encounter periodic log-full conditions.

If the applications for a database primarily consist of transactions that update data, an increase in the number of cleaners will speed up performance. Increasing the page cleaners will also decrease recovery time from soft failures, such as power outages, because the contents of the database on disk will be more up-to-date at any given time.

Default values (range)

1 (0 - 255)

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using NUM_IOCLEANERS <parameter
value>
```

Our recommendation

Use the default initially. As discussed in 8.6.30, "Changed pages threshold (CHNGPGS_THRESH)" on page 238, increase the number of IOCLEANERS if the AWP is less than 90%, unless you are using proactive page cleaning. Monitor your system during normal workload to verify the necessity of changes.

Consider the following factors when changing the value for this parameter:

- Application type
  - If it is a query-only database that will not have updates, set this parameter to zero. The exception is if the query workload results in many TEMP tables being created. (You can determine this by using the explain utility.)
  - If transactions are run against the database, set this parameter to between one and the number of physical storage devices used for the database.
- Workload

  Environments with high update transaction rates might require more page cleaners to be configured.
- Buffer pool size

  Environments with large buffer pools also might require more page cleaners to be configured.

You may use the database system monitor to help you tune this configuration parameter using information from the event monitor about write activity from a buffer pool:

- The parameter can be reduced if both of the following conditions are true:
  - pool_data_writes is approximately equal to pool_async_data_writes.
  - pool_index_writes is approximately equal to pool_async_index_writes.

- The parameter should be increased if either of the following conditions is true:
  - pool_data_writes is much greater than pool_async_data_writes.
  - pool_index_writes is much greater than pool_async_index_writes.

8.6.32 Number of I/O servers (NUM_IOSERVERS)

Impact

DB - High

Description

I/O servers are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities such as backup and restore. This parameter specifies the number of I/O servers for a database. A database cannot have more than this number of I/Os for prefetching and utilities in progress at any time. An I/O server waits while an I/O operation that it initiated is in progress. Non-prefetch I/Os are scheduled directly from the database agents and as a result are not constrained by num_ioservers.

Default values (ranges)

3 (1 - 255)

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using NUM_IOSERVERS <parameter
value>
```

Our recommendation

Customize these values for your environment. To fully exploit all of the I/O devices in the system, a good value to use is generally one or two more than the number of physical devices on which the database resides. It is better to configure additional I/O servers, because there is minimal overhead associated with each I/O server and any unused I/O servers will remain idle.

8.6.33 Maximum percent of lock list before escalation (MAXLOCKS)

Impact

DB - High

Description

Lock escalation is the process of replacing row locks with table locks, reducing the number of locks in the list. This parameter defines the maximum percent of the lock list held by an application that must be filled before the database manager performs escalation. When the number of locks held by any one application reaches this percentage of the total lock list size, lock escalation will occur for the locks held by that application. Lock escalation also occurs if the lock list runs out of space.

The database manager determines which locks to escalate by looking through the lock list for the application and finding the table with the most row locks. If after replacing these with a single table lock, the MAXLOCKS value is no longer exceeded, lock escalation will stop. If not, it will continue until the percentage of the held lock list is below the value of MAXLOCKS.

Important: The MAXLOCKS parameter multiplied by the MAXAPPLS parameter cannot be less than 100.

Default values (range)

10 (1 - 100) percent for UNIX platforms

22 (1 - 100) percent for Windows platforms

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using MAXLOCKS <parameter value>
```

Our recommendation

Use the default as is. If you need to increase this value because your application must use a large number of locks, use the following guidelines.

- To allow an application to hold twice the average number of locks, set:

  ```
  MAXLOCKS = 2 * 100 / MAXAPPLS
  ```

  `2` is used to achieve twice the average and `100` represents the largest percentage value allowed.

- If you have only a few applications that run concurrently, you could use the following formula as an alternative to the first formula:

  ```
  MAXLOCKS = 2 * 100 / (average # of applications running
  concurrently)
  ```

- One of the considerations when setting MAXLOCKS is to use it in conjunction with the size of the lock list (`locklist`). The actual limit of the number of locks held by an application before lock escalation occurs is:

  `MAXLOCKS * LOCKLIST * 4,096 / (100 * 36)` on a 32-bit system

  `MAXLOCKS * LOCKLIST * 4,096 / (100 * 56)` on a 64-bit system

In these equations, `4,096` is the number of bytes per page, `100` is the largest percentage value allowed for MAXLOCKS, `36` is the number of bytes per lock on a 32-bit system, and `56` is the number of bytes per lock on a 64-bit system.

- If you know that one of your applications requires 1,000 locks, and you do not want lock escalation to occur, then you should choose values for MAXLOCKS and LOCKLIST in this formula so that the result is greater than 1,000. (For example, use 10 for MAXLOCKS and 100 for LOCKLIST.)
- If MAXLOCKS is set too low, lock escalation happens when there is still enough lock space for other concurrent applications. If MAXLOCKS is set too high, a few applications can consume most of the lock space, and other applications will have to perform lock escalation. The need for lock escalation in this case results in poor concurrency.

Use the database system monitor to help you track and tune this configuration parameter.

8.6.34 Lock timeout (LOCKTIMEOUT)

Impact

DB - Medium

Description

This parameter specifies the number of seconds that an application will wait to obtain a lock. This helps avoid global deadlocks for applications.

If you set this parameter to zero, locks are not waited for. In this situation, if no lock is available at the time of the request, the application immediately receives a -911.

If you set this parameter to -1, lock timeout detection will be turned off. In this situation, if no lock is available at the time of the request, the application will wait for a lock until:

- The lock is granted.
- A deadlock occurs.

Default values (range)

-1 (-1; 0 - 30,000) seconds

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using LOCKTIMEOUT <parameter
value>
```

Our recommendation

The Content Manager installation program changes this value to 30.

Initially, use this value. In our test, we did not run into any lock timeout issues and thus did not need to change this value.

You can use the database system monitor to help you track the number of times an application (connection) experienced a lock timeout, or that a database detected a timeout situation for all applications that were connected.

High values of the `lock_timeout` (number of lock time-outs) monitor element can be caused by:

- Too low a value for this configuration parameter.
- An application (transaction) that holds locks for an extended period. You can use the database system monitor to further investigate these applications.
- A concurrency problem, which could be caused by lock escalations (from row-level to a table-level lock).

8.6.35 Size of the log files (LOGFILSIZ)

Impact

DB - Medium

Description

This parameter defines the size of primary and secondary log files. The primary log file is preallocated at this specified value. The size of these log files limits the number of log entries that can be written to them before they become full and new log files are required.

The use of primary and secondary log files, as well as the action taken when a log file becomes full, are dependent on the type of logging that is being performed:

- Circular logging

  A primary log file can be reused when the changes recorded in it have been committed. If the log file size is small and applications have processed a large number of changes to the database without committing the changes, a primary log file can quickly become full. If all primary log files become full, the database manager will allocate secondary log files to hold the new log records.

- Log retention logging

  When a primary log file is full, it is archived and a new primary log file is allocated.

Default values (range)

1,000 (4 - 262,144) 4-KB pages for UNIX platforms

250 (4 - 262,144) 4-KB pages for Windows platforms

To update

Run the following command:

```
db2 update db cfg for <database name> using LOGFILSIZ <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 1,000 for Library Server database in both Windows and AIX platforms.

You must balance the size of the log files with the number of primary log files. Larger log files provide better performance but potentially increase the degree of lost transactions in case of system crash or a lost log file.

If your system has a large number of write activities such as update, insert, and delete transactions, you might increase this parameter value because the log files can become full too quickly in this environment.

Note: The upper limit of log file size, combined with the upper limit of the number of log files (LOGPRIMARY + LOGSECOND), gives an upper limit of 256 GB of active log space.

If the log file size is set too small, it could create too much overhead for the system to constantly archive old log files and create new ones.

If the size is set too large, it could take a long time for the system to create new log files. Working with large log files can also be inconvenient because some text editors might not be able to display files that exceed certain size limits. Large log files also take up a lot of disk space. You might need to reduce the parameter value if disk space is scarce, because primary logs are preallocated at this size.

Note: If you are using log retention, the current active log file is closed and truncated when the last application disconnects from a database. Try to use a log file size, which will not allocate excessive amounts of wasted space.

8.6.36 Number of primary log files (LOGPRIMARY)

Impact

DB - Medium

Description

The primary log files establish a fixed amount of storage allocated to the recovery log files. This parameter enables you to specify the number of primary log files to be preallocated.

Under circular logging, the primary logs are used repeatedly in sequence. That is, when a log is full, the next primary log in the sequence is used if it is available. A log is considered available if all units of work with log records in it have been committed or rolled-back. If the next primary log in sequence is not available, then a secondary log is allocated and used. Additional secondary logs are allocated and used until the next primary log in the sequence becomes available or the limit imposed by the logsecond parameter is reached. These secondary log files are dynamically deallocated when the database manager no longer needs them.

The number of primary and secondary log files must be less than or equal to 256.

Default values (range)

3 (2 - 256)

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using LOGPRIMARY <parameter 
value>
```

Our recommendation

The Content Manager installation program changes this value to 10 for the Library Server database, and 25 for the Resource Manager database.

The value chosen for this parameter depends on several factors, including the type of logging being used, the size of the log files, and the type of processing environment (for example, length of transactions and frequency of commits).

Increasing this value increases the disk requirements for the logs because the primary log files are preallocated during the very first connection to the database.

If you find that secondary log files are being allocated frequently, you might be able to improve system performance by increasing the log file size (LOGFILSIZ) or by increasing the number of primary log files.

You can save disk storage for databases that are not accessed frequently by setting the parameter to 2. For databases enabled for roll-forward recovery, set the parameter larger to avoid the overhead of allocating new logs almost immediately.

Use the database system monitor to help size the primary log files. Observing the following monitor values over a period of time will aid in better tuning decisions, because average values might be more representative of your ongoing requirements:

- `sec_log_used_top` (maximum secondary log space used)
- `tot_log_used_top` (maximum total log space used)
- `sec_logs_allocated` (secondary logs allocated currently)

8.6.37 Number of secondary log files (LOGSECOND)

Impact

DB - Medium

Description

This parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed). When the primary log files become full, the secondary log files (of size LOGFILSIZ) are allocated one at a time as needed, up to a maximum number as controlled by this parameter. An error code will be returned to the application, and the database will be shut down, if more secondary log files are required than are allowed by this parameter.

If you set LOGSECOND to -1, the database is configured with infinite active log space. There is no limit on the size or the number of in-flight transactions running on the database. You still use the LOGPRIMARY and LOGFILSIZ configuration parameters to specify how many log files DB2 should keep in the active log path. If DB2 needs to read log data from a log file, but the file is not in the active log path, DB2 will invoke the user exit program to retrieve the log file from the archive to the active log path. (DB2 retrieves the files to the overflow log path, if you have configured one.)

When the log file is retrieved, DB2 caches this file in the active log path so that other reads of log data from the same file will be fast. DB2 manages the retrieval, caching, and removal of these log files as required.

If your log path is a raw device, you must configure the OVERFLOWLOGPATH configuration parameter and set the userexit configuration parameter to Yes in order to set LOGSECOND to -1.

By setting LOGSECOND to -1, you will have no limit on the size of the unit of work or the number of concurrent units of work. However, rollback (both at the savepoint level and the unit of work level) could be very slow due to the need to retrieve log files from the archive. Crash recovery could also be very slow for the same reason. DB2 will write a message to the administration notification log to warn you that the current set of active units of work has exceeded the primary log files. This is an indication that rollback or crash recovery could be extremely slow.

Default values (range)

2 (-1; 0 - 254)

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using LOGSECOND <parameter value>
```

Our recommendation

The Content Manager installation program changes this value to 20 for Library Server database and to 50 for Resource Manager database

In general, keep the default values.

Secondary log files do not require permanent file space, so there is not need to change these values. If secondary log files begin to be used more frequently, think about changing the values of LOGFILSIZ or LOGPRIMARY parameters.

8.6.38 Minimum commits before writing logs to disk (MINCOMMIT)

Impact

DB - High

Description

This parameter enables you to delay the writing of log records to disk until a minimum number of commits has been performed. This delay helps to reduce the database manager overhead associated with writing log records. As a result, this might improve performance when you have multiple applications running against a database, and many commits are requested by the applications within a very short time frame.

This grouping of commits occurs when the value of MINCOMMIT is greater than 1 and when the number of applications connected to the database is greater than or equal to the value of this parameter. When commit grouping is being performed, application commit requests could be held until either one second has elapsed or the number of commit requests equals the value of this parameter. If there is not enough transaction work to do, the database manager will commit transactions every second. Changes to the parameter value take effect immediately. There is no need to wait until all applications are disconnected from the database.

Default values (range)

1 (1 - 25)

To update

Run the following DB2 command:

```
db2 update db cfg for <database name> using MINCOMMIT <parameter value>
```

Our recommendation

Use the default.

If you need to perform a large initial load operation prior to launching a new Content Manager system, increasing this value temporarily will help reduce logging file I/O.

In general, if your application requires large concurrent write activities, simple SQL statements to process, and single row updates (such as daily load operation), you might want to increase the default value. This will result in more efficient logging file I/O from occurring less frequently, and will write more log records each time it does occur.

Grouping the commits might compromise response time of small transactions, because they have to wait until the minimum number of transactions is ready to be committed.

You could also sample the number of transactions per second and adjust this parameter to accommodate the peak number of transactions per second (or some large percentage of it). Accommodating peak activity would minimize the overhead of writing log records during transaction-intensive periods.

If you increase MINCOMMIT, you might also have to increase the LOGBUFSZ parameter to avoid having a full log buffer force a write during these transaction-intensive periods. In this case, the LOGBUFSZ should be equal to:

```
MINCOMMIT * (log space used, on average, by a transaction)
```

Use the database system monitor to help tune this parameter in these ways:

- Calculating the peak number of transactions per second

  Taking monitor samples throughout a typical day, you can determine your transaction-intensive periods. Calculate the total transactions by adding the following monitor elements:

  – `commit_sql_stmts` (commit statements attempted)
  – `rollback_sql_stmts` (rollback statements attempted)

  Using this information and the available timestamps, you can calculate the number of transactions per second:

  ```
  MINCOMMIT = (commits + rollbacks) / 10
  ```

- Calculating the log space used per transaction

  Using sampling techniques over a period of time and a number of transactions, you can calculate an average of the log space used with the following monitor element:

  – `log_space_used` (unit of work log space used)

Changes to the value specified for this parameter take effect immediately; you do not have to wait until all applications disconnect from the database.

8.6.39 Connection concentrator

Impact

DBM - Medium/High

Description

With connection concentrator, there is a shared pool of agent and stored procedure processes for the Library Server database. Each connection uses processes from the pool only during transactions. This enables many more client connections to be processed efficiently. It also reduces memory use for each connection and decreases the number of context switches. Connection concentrator started with DB2 UDB V8.1.

Note: The connection concentrator is enabled when the MAX_CONNECTIONS database configuration parameter is set larger than the MAX_COORDAGENTS configuration parameter.

All of the following parameters are database manager configuration parameters of medium performance impact, with the exception of NUM_POOLAGENTS, which has a high performance impact. Each of these parameters has been discussed in detail in earlier sections of this chapter.

How to view or set

You can enable and tune connection concentrator with the following parameters:

- MAX_CONNECTIONS

  This parameter sets the largest number of allowed database connections. This parameter should be larger than the total number of connections for the mid-tier server plus the number of Windows client users. If you are using mid-tier server connection pooling, then you can roughly calculate this parameter as follows:

  ```
  1/10 (# of Web users) + (# of Windows client users)
  ```

- MAXAGENTS

  This parameter sets the largest number of database agents. At minimum, it should be as high as the expected maximum number of concurrent transactions in your system. If NUM_INITAGENTS (see next parameter) is set very high, you might have to set this value even higher, in order to make room for other kinds of agents. As a rough estimate, you can set it to:

  ```
  1/10 (MAX_CONNECTIONS)
  ```

- NUM_INITAGENTS

  This parameter sets the initial number of database agents. It should be set somewhere around the expected or typical number of concurrent transactions in your system.

- NUM_INITFENCED

  This parameter sets the initial number of **`db2fmp`** “fenced” processes. Set this to the same value as NUM_INITAGENTS.

- MAX_COORDAGENTS

  This parameter sets the largest number of concurrent coordinating agents. This number is usually set to the same as the number as MAXAGENTS.

- NUM_POOLAGENTS

  This parameter sets the size of the agent pool. This number is usually set to the same as the number of MAXAGENTS. When connection concentrator is enabled, the number of agents specified here is only advisory. More agents might be in the agent pool at any given time.

For example, consider a single database partition system in which, on average, 2,000 users are connected to the database. At times, the number of concurrent transactions is as high as 100, but never higher than 200. Transactions are short.

For this workload example, the database manager configuration parameters involved in connection concentrator would be:

- MAX_CONNECTIONS is set to 2,000 to ensure support for the average number of connections.
- MAX_COORDAGENTS is set to 200 to support the maximum number of concurrent transactions.
- MAXAGENTS is set high enough to support all of the coordinator agents and subagents (where applicable) that are required to run transactions on the node.

  For a Content Manager system, we recommend setting INTRA_PARALLEL to OFF (see 8.6.1, "Intra-partition parallelism (INTRA_PARALLEL)" on page 206). If this is the case, MAXAGENTS is set to 200 because in such an environment, there are no subagents.

  If INTRA_PARALLEL is ON, MAXAGENTS should be set large enough to accommodate the coordinator agent and the subagents required for each transaction that accesses data on the node. For example, if each transaction requires four subagents, MAXAGENTS should be set to `(4+1) * 200`, which is 1,000. To tune MAXAGENTS further, take monitor snapshots for the database manager. The high-water mark of the agents will indicate the appropriate setting for MAXAGENTS.
- NUM_POOLAGENTS is set to at least 200, or as high as 1000, depending on the value of MAXAGENTS to ensure that enough database agents are available to service incoming client requests without the overhead of creating new ones.

  However, this number could be lowered to reduce resource usage during low-usage periods. Setting this value too low causes agents to be deallocated instead of going into the agent pool, which requires new agents to be created before the server can handle an average workload.
- NUM_INIT_AGENTS is set to be the same as NUM_POOLAGENTS because you know the number of agents that should be active. This causes the database to create the appropriate number of agents when it starts instead of creating them before a given request can be handled.
- NUM_INITFENCED is set to be the same as NUM_INIT_AGENTS.

This db2 command would be necessary to update the database manager configuration parameters:

```
db2 update dbm cfg using MAXAGENTS 200 MAX_COORDAGENTS 200
NUM_POOLAGENTS 200 MAX_CONNECTIONS 2000 NUM_INITAGENTS 200
NUM_INITFENCED 200
```

Our recommendation

In addition to the recommendations provided in the earlier sections of this chapter for setting the parameters, consider as well the ability of the underlying hardware that handles the given workload. If the hardware cannot handle the ideal number of agents, reduce this number to the maximum that the hardware can support. For example, if the maximum is only 100 agents, then this limits the maximum number of concurrent transactions that can be handled. You need to monitor these kinds of performance-related settings because it is not always possible to determine exact requests sent to other nodes at a given point in time.

9

Tuning WebSphere for Content Manager

This chapter covers tuning WebSphere for Content Manager. It includes hardware selection and configuration, operating system tuning, Web server tuning, WebSphere Application Server tuning, and Java virtual machines tuning.

9.1 Introduction

Tuning WebSphere for Content Manager is about processing as many requests as possible by utilizing the resources to their fullest potential. Many components in WebSphere Application Server have direct performance impact on Resource Manager. This chapter covers the tuning area including our recommendations that might have implications on Resource Manager performance.

As a shortcut into WebSphere tuning, we offer Table 9-1. This table is designed for easy access to symptoms and related tuning area for the symptoms.

Table 9-1 WebSphere performance tuning short cut

Symptoms	Related tuning area
Throughput and response time are undesirable.	Processor speed
AIX: Memory allocation error.	AIX file descriptors (ulimit)
Windows NT or 2000: Netstat shows too many sockets are in TIME_WAIT.	Windows NT or 2000 TcpTimedWaitDelay
Throughput is undesirable and the application server priority has not been adjusted.	Adjusting the operating system priority of the WebSphere Application Server process
Under load, client requests do not arrive at the Web server because they time out or are rejected.	For IBM HTTP Server on Windows NT, see ListenBackLog
Tivoli Performance Viewer Percent Maxed Metric indicates that the Web container thread pool is too small.	Thread pool
Netstat shows too many TIME_WAIT state sockets for port 9080.	MaxKeepAliveConnections, MaxKeepAliveRequests
Too much disk input/output occurs due to paging.	Heap size settings

Much of the material provided here is extracted from *IBM WebSphere Application Server, Version 5.1: Monitoring and Troubleshooting*, found in:

http://www.ibm.com/software/webservers/appserv/was/library

For more in-depth tuning information, you can also check out the WebSphere Application Server Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1/index.jsp

9.2 Hardware tuning

This section discusses considerations for selecting and configuring the hardware on which the application servers will run.

9.2.1 Processor speed

In the absence of other bottlenecks, increasing the processor speed often helps throughput and response times. A processor with a larger L2 or L3 cache can yield higher throughput even if the processor speed is the same as a CPU with a smaller L2 or L3 cache.

9.2.2 Disk speed

Disk speed and configuration can have a dramatic effect on the performance of application servers that run applications that are heavily dependent on database support, that use extensive messaging, or are processing workflow. Disk I/O subsystems that are optimized for performance, such as RAID array, are essential components for optimum application server performance in these environments.

9.2.3 System memory

Increasing memory to prevent the system from paging memory to disk is likely to improve performance.

Try adjusting the parameter when the system is paging (and processor utilization is low because of the paging).

Allow at least 256 MB memory for each processor. We recommend a minimum value of 512 MB.

9.2.4 Networks

Run network cards and network switches at full duplex. Running at half-duplex decreases performance.

Verify that the network speed can accommodate the required throughput. Make sure that 100 MB is in use on 10/100 Ethernet networks.

9.3 Operating system tuning

This section discusses parameters you can tune on the operating system side to potentially enhance system performance. We describe each parameter in detail, with its default value and our recommendation for what value you should set.

9.3.1 TCP timed wait delay (TcpTimedWaitDelay)

This is a Windows NT/2000 parameter. This parameter determines the time that must elapse before TCP can release a closed connection and reuse its resources. This interval between closure and release is known as the TIME_WAIT state or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server cost less than establishing a new connection. Reducing the value of this entry enables TCP to release closed connections faster, providing more resources for new connections. Adjust this parameter if the running application requires rapid release, creation of new connections, and there is a low throughput due to many connections sitting in TIME_WAIT.

Default values

0xF0 (240 seconds = 4 minutes)

How to view or set

1. Using the **`regedit`** command, access HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters and create a new REG_DWORD named `TcpTimedWaitDelay`.
2. Set the value to decimal 30, which is Hex 0x0000001e, or another value.
3. Restart the system.

By using the **`netstat`** command, you will be able to see that there are fewer connections in TIME_WAIT.

Our recommendation

Use the minimum value of 0x1E (30 seconds) for Content Manager OLTP applications.

Adjust this parameter if the application requires rapid release and creation of new connections, and there is a low throughput due to many connections sitting in TIME_WAIT.

9.3.2 Maximum user ports (MaxUserPort)

This is a Windows NT/2000 parameter. It determines the highest port number TCP can assign when an application requests an available user port from the system.

Default values

1024 - 5000 are available

How to view or set

1. Using the **`regedit`** command, access HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters and create a new REG_DWORD named `MaxUserPort`.
2. Restart the system.

Our recommendation

The server might run out of ports if too many clients try to connect and this value is not changed. Use at least decimal 32768 until 65534 (the maximum). With this, you increase the number of possible ports to be used to connect to the Content Manager system.

> **Important:** These MaxUserPort and TcpTimedWaitDelay should be used together when tuning WebSphere Application Server on a Windows NT or Windows 2000 operating system.

9.3.3 Number of open files permitted (ulimit)

This AIX parameter specifies the permitted number of open files. The default setting is typically sufficient for most applications. If the value set for this parameter is too low, a memory-allocation error is displayed.

Default value

2000

How to view or set

To set ulimit, use the following command:

```
ulimit -n <parameter value>
```

In this syntax, <parameter value> stands for the number of open files permitted.

For SMP machines, to unlimit the ulimit value, use the following command:

```
ulimit -unlimited
```

To display the current values for all limitations on system resources, use the following command:

```
ulimit -a
```

Our recommendation

The default setting is typically sufficient for most applications. If the value set for this parameter is too low, a memory-allocation error is displayed. This error occurs when a cluster of four clones is run on an S80 24-way and the ulimit value should be changed to unlimited.

See 7.3.1, "Configure appropriate ulimit" on page 157 for additional discussion about default values for ulimit.

9.3.4 Duration of an active connection (TCP_KEEPIDLE)

The keepAlive packet ensures that a connection stays in an active/ESTABLISHED state.

Default value

14400 milliseconds

How to view or set

Use the **no** command to determine the current value or to set the value. The change takes effect immediately. The change is effective until the next time you restart the machine. To permanently change the value, add the **no** command to the /etc/rc.net directory. For example:

```
no -o tcp_keepidle=600
```

Our recommendation

600 milliseconds

9.4 Web server tuning

Tuning IBM HTTP Server increases the throughput while balancing the system resources. WebSphere Application Server provides plug-ins for several Web server brands and versions. Each Web server operating system combination has specific tuning parameters that affect application performance.

IBM HTTP Server settings are set in the configuration file called httpd.conf, which is normally located on UNIX under /usr/IBMHttpServer/conf directory, and on

Windows under <installed drive>:\IBM\IBMHttpServer\conf directory. The definition of each parameter is described in the httpd.conf file itself.

In this section, we discuss the performance tuning settings that are associated with the Web servers.

9.4.1 Access logs

Access logs collect all incoming HTTP requests. Logging degrades performance because the I/O operation overhead causes logs to grow significantly in a short time.

Default value

Logging of every incoming HTTP request is enabled.

How to view or set

1. Open the **IBM HTTP Server httpd.conf** file, located in the directory IBM_HTTP_Server_root_directory/conf.
2. Search for a line with the text `CustomLog`.
3. Comment out this line by placing `#` in front of the line.
4. Save and close the httpd.conf file.
5. Stop and restart the IBM HTTP Server.

Our recommendation

Disable the access logs.

9.4.2 Web server configuration refresh interval (RefreshInterval)

WebSphere Application Server administration tracks a variety of configuration information about WebSphere Application Server resources. Some of this information, such as URIs pointing to WebSphere Application Server resources, must be understood by the Web server. This configuration data is pushed to the Web server through the WebSphere Application Server plug-in at intervals specified by this parameter. Periodic updates enable new servlet definitions to be added without having to restart any of the WebSphere Application Server servers. However, the dynamic regeneration of this configuration information is costly in terms of performance.

Default values

60 seconds.

How to view or set

This parameter, <RefreshInterval=*xxxx*> (where *xxxx* is the number of seconds), is specified in the websphere_root/config/plug-in.xml file.

Our recommendation

Use the default. Content Manager is not directly affected by this parameter.

9.4.3 HTTP server maximum clients (MaxClients)

The MaxClients directive controls the maximum number of simultaneous connections or users that the Web server can service at any one time. If, at peak usage, your Web server needs to support 200 active users at once, you should set MaxClients to 220 (200 plus an extra 10% for load growth). Setting MaxClients too low can cause some users to believe that the Web server is not responding. You should have sufficient RAM in your Web server machines to support each connected client. For IBM HTTP Server 1.3 on UNIX, you should allocate about 1.5 MB MaxClients of RAM for use by the IBM HTTP Server. For IBM HTTP Server 1.3 on Windows, you should allocate about 300 KB MaxClients of RAM for use by the IBM HTTP Server. Some third-party modules can significantly increase the amount of RAM used per connected client.

Default values

150

How to view or set

1. Edit the IBM HTTP server file httpd.conf, which is located in the directory IBM HTTP Server_root_directory/conf.
2. Change the value of the `MaxClients` parameter.
3. Save the changes and restart the IBM HTTP server.

Our recommendation

Initially use the default and monitor your system performance. In general:

- Use a lower MaxClients value for a simple, CPU-intensive application.
- Use a higher MaxClients value for a more complex, database-intensive application with longer wait times.

9.4.4 Minimum spare server processes (MinSpareServers)

The MinSpareServers directive sets the desired minimum number of idle child server processes. An idle process is one that is not handling a request. If the number of the idle child server processes falls below the value specified for MinSpareServer, then the parent process will create new child processes.

Setting this directive to some value `m` ensures that you always have at least `n + m` httpd processes running when you have `n` active client requests.

Attention: This directive has no effect on Microsoft Windows.

Default values

5

How to view or set

1. Edit the IBM HTTP server file httpd.conf, located in the directory IBM_HTTP_Server_root_directory/conf.
2. Change the value of the `MinSpareServers` parameter.
3. Save the changes and restart the IBM HTTP server.

Our recommendation

Use the default. Monitor the CPU usage for creating and destroying HTTP requests.

You should have to tune this parameter *only* on very busy sites. Setting this parameter to a large number is almost always a bad idea.

9.4.5 Maximum spare server processes (MaxSpareServers)

The MaxSpareServers directive sets the desired maximum number of idle child server processes. An idle process is one that is not handling a request. If the number of idle processes exceeds the value specified for MaxSpareServers, then the parent process will kill the excess processes.

This is the maximum number of spare servers, not the maximum total number of client requests that can be handled at one time. If you wish to limit that number, see the MaxClients directive.

Attention: This directive has no effect when used with the Apache Web server on a Microsoft Windows platform.

Default values

10

How to view or set

1. Edit the IBM HTTP server file httpd.conf, located in the directory IBM_HTTP_Server_root_directory/conf.
2. Change the value of the `MaxSpareServers` parameter.
3. Save the changes and restart the IBM HTTP server.

Our recommendation

Use the default. Monitor the CPU usage for creating and destroying HTTP requests.

You should have to tune this parameter *only* on very busy sites. Setting this parameter to a large number is almost always a bad idea.

For optimum performance, specify the same value for the MaxSpareServers and the StartServers (see below) parameters. Specifying similar values reduces the CPU usage for creating and destroying httpd processes. It also preallocates and maintains the specified number of processes so that few processes are created and destroyed as the load approaches the specified number of processes (based on MinSpareServers).

9.4.6 Startup server processes (StartServers)

The StartServers directive sets the number of child server processes created on startup. The number of processes is dynamically controlled depending on the load, so there is usually little reason to adjust this parameter.

Attention: When running under Microsoft Windows, this directive has no effect. There is always one child that handles all requests. Within the child, requests are handled by separate threads. The ThreadsPerChild directive controls the maximum number of child threads handling requests, which will have an effect similar to the setting of StartServers on UNIX.

Default values

5

How to view or set

1. Edit the IBM HTTP server file httpd.conf, located in the directory IBM_HTTP_Server_root_directory/conf.
2. Change the value of the `StartServers` parameter.
3. Save the changes and restart the IBM HTTP server.

Our recommendation

For optimum performance, specify the same value for the MaxSpareServers (see above) and the StartServers parameters. Specifying similar values reduces the CPU usage for creating and destroying httpd processes. It also preallocates and maintains the specified number of processes so that few processes are created and destroyed as the load approaches the specified number of processes (based on MinSpareServers).

9.4.7 Maximum requests per child (MaxRequestPerChild)

Apache always creates one child process to handle requests. If it dies, another child process is created automatically. Within the child process, multiple threads handle incoming requests. The MaxRequestsPerChild directive sets the limit on the number of requests that an individual child server process will handle. After MaxRequestsPerChild requests, the child process will die. If MaxRequestsPerChild is zero, then the process will never expire.

Setting MaxRequestsPerChild to a non-zero limit has two beneficial effects:

- It limits the amount of memory that process can consume by (accidental) memory leakage.
- It helps to reduce the number of processes when the server load reduces by giving the processes a finite life time.

However, on Win32®, It is recommended that this be set to zero. If it is set to a non-zero value, when the request count is reached, the child process exits, and is re-spawned, at which time it rereads the configuration files. This can lead to unexpected behavior if you have modified a configuration file but are not expecting the changes to be applied yet. See also (ThreadsPerChild).

Default values

0 for Windows platforms

0000 for AIX platforms

How to view or set

1. Edit the IBM HTTP server file httpd.conf, located in the directory IBM_HTTP_Server_root_directory/conf.
2. Change the value of the `MaxRequestPerChild` parameter.
3. Save the changes and restart the IBM HTTP server.

Our recommendation

In general, do not force a server to exit after it has served some number of requests. This means that if there are no known memory leaks with Apache and Apache's libraries, set the value to zero for both Windows and AIX platforms.

If you want the server to exit after they have run for a long time (to help the system clean up after the process), set this to a large number, such as 10,000. In this example, each child server will exit after serving 10,000 requests, and another server will take its place.

9.4.8 Threads per child (ThreadsPerChild)

Apache always creates one child process to handle requests. If it dies, another child process is created automatically. Within the child process, multiple threads handle incoming requests. The ThreadsPerChild directive tells the server how many concurrent threads it should use at a time. It is the maximum number of connections the server can handle at once.

Attention: This directive has no effect on UNIX systems. UNIX users should look at StartServers and MaxRequestsPerChild.

Default values

50

How to view or set

To change:

1. Edit the IBM HTTP Server file httpd.conf, which is located in the IBM_HTTP_Server_root_directory/conf directory.
2. Change the value of the `ThreadsPerChild` parameter.
3. Save the changes and restart the IBM HTTP server.

Two ways to find out how many threads are being used under the current load:

- Use the Windows NT or 2000 Performance Monitor:
  a. Click **Start** → **Programs** → **Administrative Tools** → **Performance Monitor**.
  b. Click **Edit** → **Add to chart** and set:
     - Object: IBM HTTP Server
     - Instance: Apache
     - Counter: Waiting for connection

        - To calculate the number of busy threads, subtract the number waiting (Windows NT or 2000 Performance Monitor) from the total available (ThreadsPerChild).

- Use IBM HTTP Server server-status (this choice works on all platforms, not just Windows NT or 2000):

    a. Edit the IBM HTTP Server file httpd.conf as follows:

        - Remove the comment character "#" from the following lines:

            - `#LoadModule status_module modules/ApacheModuleStatus.dll`
            - `#<Location /server-status>`
            - `#SetHandler server-status`
            - `#</Location>`

    b. Save the changes and restart the IBM HTTP server.

    c. In a Web browser, go to `http://`*`yourhost`*`/server-status` and click **Reload** to update status. Or, if the browser supports refresh, go to `http://`*`yourhost`*`/server-status?refresh=5` to refresh every 5 seconds. You will see five requests currently being processed, and 45 idle servers.

Our recommendation

Set the value to the expected number of concurrent users. Monitor your system during normal workload to see whether this value should be changed.

Be sure that this value is high enough for your site if you get a lot of hits. Allow just enough traffic through to the application server to prevent bottlenecks.

9.4.9 Pending connections queue length (ListenBackLog)

When several clients request connections to the IBM HTTP server, and all threads are being used, a queue exists to hold additional client requests. The ListenBackLog directive sets the maximum length for the queue of pending connections. However, if you are using the default Fast Response Cache Accelerator (FRCA) feature, the ListenBackLog directive is not used because FRCA has its own internal queue.

This is often limited to a smaller number by the operating system. This varies from operating system to operating system. Also note that many operating systems do not use exactly what is specified as the backlog, but instead use a number based on (but normally larger than) what is set.

Default values

- 511 with FRCA disabled
- 1024 with FRCA enabled

How to view or set

For non-FRCA:

1. Edit IBM HTTP Server file httpd.conf, which is located in the IBM_HTTP_Server_root_directory/conf directory.
2. Change the `ListenBackLog` directive.
3. Save the changes and restart the IBM HTTP server.

Our recommendation

Generally, use the default.

9.5 WebSphere Application Server tuning

Each WebSphere Application Server process has several parameters that influence application performance. Each application server in WebSphere Application Server is comprised of an EJB container and Web container.

In this section, we discuss the performance tuning settings that are associated with the WebSphere Application Server.

9.5.1 Application server process priority

Improving the operating system process priority of the application server can help performance. On AIX systems, a smaller setting has a higher priority.

Default values

20

How to view or set

Using the WebSphere administrative console:

1. In the administrative console, select the application server you are tuning.
2. Under Additional Properties, click **Process Definition**.
3. Under Additional Properties, click **Process Execution**.
4. Specify the value in the Process Priority field and click **Apply**.
5. Save the changes and restart the application server.

To see the current process priority on AIX, type **`ps -efl`**.

Our recommendation

Lower the default value to zero. Monitor the system to see whether giving more priority to the application server affects the other components of the Content

Manager System such as DB2. Make small increment changes at a time. If the trade-off is not acceptable, then go back to the previous value. By improving the priority of an application server, noticeable improvements have been seen on AIX; but you always want to be able to balance performance of all components of Content Manager and avoid bottlenecks.

9.5.2 Web container maximum thread size

To route servlet requests from the Web server to the Web containers, the product establishes a transport queue between the Web server plug-in and each Web container. Use this configuration to specify the maximum number of threads that can be pooled to handle requests sent to the Web container. Requests are sent to the Web container through any of the HTTP transports.

Default values

50

How to view or set

1. In the administrative console, select the application server you are tuning.
2. Under Additional Properties, click **Web Container**.
3. Under Additional Properties, click **Thread Pool**.
4. Enter the desired maximum number of maximum threads in the Maximum Size field.
5. Select the **Growable Thread Pool** (see below) check box.
6. Click **Apply** or **OK**.
7. Click **Save**.

Our recommendation

Tivoli Performance Viewer displays a metric called Percent Maxed that determines the amount of time that the configured threads are in use. If this value is consistently in the double digits, the Web container could be a bottleneck and the number of threads should be increased.

9.5.3 Thread allocation beyond maximum

When this option is selected (enabled), more Web container threads can be allocated than specified in the maximum thread size field. When unselected, the thread pool cannot grow beyond the value specified for the maximum thread size.

Default values

Unchecked

How to view or set

1. In the administrative console, select the application server you are tuning.
2. Under Additional Properties, click **Web Container Service**.
3. Under Additional Properties, click **Thread Pool**.
4. Select the check box **Growable thread pool**.
5. Click **Apply** to ensure that the changes are saved.
6. Stop and restart the application server.

Our recommendation

This option is intended to handle brief loads beyond the configured maximum thread size. However, use caution when selecting this option because too many threads can cause the system to overload.

9.5.4 Max keep-alive connections (MaxKeepAliveConnections)

This parameter describes the maximum number of concurrent connections to the Web container that are allowed to be kept alive (that is, to be processed in multiple requests. The Web server plug-in keeps connections open to the application server as long as it can. However, if the value of this property is too small, performance is negatively affected because the plug-in has to open a new connection for each request instead of sending multiple requests through one connection. The application server might not accept a new connection under a heavy load if there are too many sockets in TIME_WAIT state.

If all client requests are going through the Web server plug-in and there are many TIME_WAIT state sockets for port 9080, the application server is closing connections prematurely, which decreases performance. The application server will close the connection from the plug-in, or from any client, for any of the following reasons:

- The client request was an HTTP 1.0 request when the Web server plug-in always sends HTTP 1.1 requests.
- The maximum number of concurrent keep-alives was reached. A keep-alive must be obtained only once for the life of a connection (that is, after the first request is completed, but before the second request can be read).
- The maximum number of requests for a connection was reached, preventing denial of service attacks in which a client tries to hold on to a keep-alive connection forever.
- A timeout occurred while waiting to read the next request or to read the remainder of the current request.

Default values

No default setting

How to view or set

1. Open the administrative console; select the application server you are tuning.
2. Under Additional Properties, click **Web Container**.
3. Under Additional Properties, click **HTTP Transports**.
4. Click the **port_number** link in the Host column.
5. Under Additional Properties, click **Custom Properties**.
6. Click **New**.
7. In the Name field, enter `MaxKeepAliveConnections`.
8. Enter the value in the Value field.
9. Click **Apply** or **OK**.
10. Click **Save**.

Our recommendation

The value should be at least 90% of the maximum number of threads in the Web container thread pool. If it is 100% of the maximum number of threads in the Web container thread pool, all of the threads could be consumed by keep-alive connections, leaving no threads available to process new connections.

9.5.5 Maximum requests for a connection (MaxKeepAliveRequests)

The maximum number of requests that are allowed on a single keep-alive (persistent) connection. This parameter can help prevent denial of service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance.

Set the value to zero to allow an unlimited amount.

Default values

No default setting

How to view or set

1. Open the administrative console. Select the application server you are tuning.
2. Under Additional Properties, click **Web Container**.
3. Under Additional Properties, click **HTTP Transports**.
4. Click the **port_number** link in the host column.
5. Click **Custom Properties** under **Additional Properties**.
6. Click **New**.
7. In the Name field, enter `MaxKeepAliveRequests`.
8. Enter the value in the Value field.
9. Click **Apply** or **OK**.
10. Click **Save**.

Our recommendation

Set this value high to get the maximum performance. A good starting value is 100. Monitor your system using Tivoli Performance Viewer to see whether this value is adequate for your environment.

9.5.6 URL invocation cache

The invocation cache holds information for mapping request URLs to servlet resources. A cache of the requested size is created for each thread. The number of threads is determined by the Web container maximum thread size setting. The default size of the invocation cache is 50. If more than 50 unique URLs are actively being used (each JavaServer Page is a unique URL), you should increase the size of the invocation cache.

> **Note:** A larger cache uses more of the Java heap, so you might need to increase maximum Java heap size. For example, if each cache entry requires 2 KB, maximum thread size is set to 25, and the URL invocation cache size is 100 then 5 MB of Java heap is required.

Default values

50

How to view or set

1. In the administrative console, click **Servers** → **Application servers** and select the application server you are tuning.
2. Under Additional Properties, click **Process Definition**.
3. Under Additional Properties, click **Java Virtual Machine**.
4. Under Additional Properties, click **Custom Properties**.
5. Specify InvocationCacheSize in the Name field and the size of the cache in the Value field. You can specify any number higher than zero for the cache size. Setting the value to zero disables the invocation cache.
6. Click **Apply** and then **Save** to save your changes.
7. Stop and restart the application server.

Our recommendation

Start with a value of 50. Monitor your system and increase this value as needed. For example, if more than 50 unique URLs are actively being used (each JSP is a unique URL), consider increasing this value.

9.5.7 Security

Security is a global setting. When security is enabled, the performance can decrease between 10 to 20 percent.

Default values

Disabled

How to view or set

1. In the **administrative** console click **Security** → **Global Security**.
2. Select **Enabled** and **Enforce Java 2 Security** setting.
3. Click **Apply** to ensure that the changes are saved.
4. Stop and restart the application server.

Our recommendation

When security is not needed, keep the default disable setting.

9.6 Java virtual machines tuning

Java virtual machines (JVM) offer several tuning parameters that affect the performance of WebSphere Application Servers and application performance.

In this section, we discuss the performance tuning settings associated with JVM.

9.6.1 Initial and maximum heap size for the JVM

These parameters set the initial and the maximum heap sizes for the JVM.

In general, increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory. After the heap begins swapping to disk, Java performance drastically suffers. Therefore, the maximum heap size must be low enough to contain the heap within physical memory. The physical memory usage must be shared between the JVM and the other applications, such as the database. For assurance, use a smaller heap, for example 64 MB, on machines with less memory.

Default values

- Initial heap size: 0 MB
- Maximum heap size 256 MB

How to view or set

1. Click **Servers** → **Application Servers**.

2. Click the application server you want to tune.
3. Under Additional Properties, click the **Process Definition**.
4. Under Additional Properties, click **Java Virtual Machine**.
5. Enter values in the General Properties field for the following fields: **Initial Heap Size** and **Maximum Heap Size**.
6. Save your changes.
7. Restart the application server.

Our recommendation

Try a maximum heap of 128 MB on systems with less than 1 GB of physical memory, 256 MB for systems with 2 GB memory, and 512 MB for larger systems. The starting point depends on the application.

For production systems where the working set size of the Java applications is not well understood, an initial setting of one-fourth the maximum setting is a good starting value. The JVM will then try to adapt the size of the heap to the working set of the Java application.

Monitor your system. When you have verified the normal size of your JVM heap size, change the initial heap size to this value using the Tivoli Performance Viewer.

10

Tuning TSM for Content Manager

This chapter covers tuning Tivoli Storage Manager (TSM) for Content Manager. It includes tuning server options that affect Content Manager performance and setting the system to automatically tune some of the server options for performance.

10.1 Introduction

Various tuning parameters affect Tivoli Storage Manager performance. The number of parameters that might be set in TSM is quite small. It is the tuning of the client, server, and network options that can become very complex.

Some of the factors that affect TSM performance are:

- Average client file size
- Percentage of files and bytes changed since last incremental backup
- Client/server hardware (CPUs, RAM, disk drives, network adapters)
- Client/server operating system
- Client/server activity (non-TSM workload)
- Server storage pool devices (disk, tape, optical)
- Network hardware, configuration, utilization, and reliability
- Communication protocol and protocol tuning
- Final output repository type (disk, tape, optical)

In the following sections, we cover the very basic aspects of the TSM configuration and tuning for performance. This is only a starting point for tuning TSM for Content Manager. For a complete list of parameters and the tuning features in TSM, refer to the following publications:

- *Tivoli Storage Manager for AIX - Administrator's Guide*, GC32-0768.
- *Tivoli Storage Manager Problem Determination Guide*; search at:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp

10.2 Automatic tuning of server options

Tuning a Tivoli Storage Manager environment to achieve the best performance requires experience, knowledge, and skill from experts, and is a time-consuming process. The current version of TSM can automate the tuning of some of the existing parameters and achieve more intelligent storage management.

TSM calculates performance figures, usually in throughput per unit of time, and changes a pre-set parameter that can be adjusted during the next set of iterations to test its effect on performance. The buffer pool is one of many parameters that affect performance. The buffer pool size (BUFPOOLSIZE) can be self-tuned by TSM.

For optimal performance and simplified tuning, we recommend the following action.

10.2.1 Self-tune buffer pool size (BUFPOOLSIZE)

The database buffer pool provides cache storage so that the database pages can remain in memory for longer periods of time. When the database pages remain in cache, the server can make continuous updates to the pages without requiring I/O operations to external storage. Although a database buffer pool can improve server performance, it will also require more memory.

For UNIX platforms, the buffer pool size may not exceed 10% of physical memory. For Windows NT/2000 platforms, the buffer pool size may not exceed 10% of physical memory and memory load may not exceed 80%.

An optimal setting for the database buffer pool is one in which the cache hit percentage is greater than or equal to 99%. Performance decreases drastically if buffer pool cache hit ratio decreases below 99%. To check the cache hit percentage, use the query `db format=detail`. Increasing the BUFPOOLSIZE parameter can improve the performance of many TSM server functions such as multi-client backup, storage pool migration, storage pool backup, expiration processing, and move data. If the cache hit percentage is lower than 99%, increase the size of the BUFPOOLSIZE parameter in the DSMSERV.OPT file. You need to shut down and restart the server for changes to take effect.

For most servers, if tuning the parameter manually, we recommend starting with a value of 32,768 KB, which equals 8,192 database pages. If you have enough memory, increase in 1 MB increments. A cache hit percentage greater than 99% is an indication that the proper BUFPOOLSIZE has been reached. However, continuing to raise BUFPOOLSIZE beyond that level can be very helpful. While increasing BUFPOOLSIZE, care must be taken not to cause paging in the virtual memory system. Monitor system memory usage to check for any increased paging after the BUFPOOLSIZE change. (Use the **`reset bufp`** command to reset the cache hit statistics.) If you are paging, buffer pool cache hit statistics will be misleading because the database pages will be coming from the paging file.

Automatic tuning through SELFTUNEBUFPOOLSIZE

For optimal performance and simplified tuning, we recommend having the server automatically tune the BUFPOOLSIZE by setting the SELFTUNEBUFPOOLSIZE option to Yes. The option is disabled by default.

Before expiration processing, the server resets the database buffer pool and examines the database buffer pool cache hit ratio. The server accounts for the amount of available storage and adjusts the buffer pool size as needed.

When the SELFTUNEBUFPOOLSIZE is set to Yes, the buffer pool cache hit ratio statistics are reset at the start of expiration. After expiration, the buffer pool size is increased if the cache hit ratio is less than 98%. The increase in the buffer pool size is in small increments and might change after each expiration. The change in the buffer pool size is not reflected in the server options file. You can check the current size at any time using the **`query status`** command. Use the **`setopt bufpoolsize`** command to change the buffer pool size.

Note: Although the value of BUFPOOLSIZE might be changed, the settings in the server options file are not changed. Issuing a **`query option`** command displays only what is set in the server options file.

This is a quick introduction of performance tuning TSM for Content Manager. If you want to explore more TSM tuning options, refer to the publications listed at the beginning of this chapter.

Part 4

Monitoring, troubleshooting, and maintenance

In this part, we cover monitoring tools at the operating system level, monitoring and analysis tools for DB2, monitoring tools for WebSphere Application Server, and performance tracing for Content Manager. In addition, we describe performance-related troubleshooting techniques based on a collection of real-life scenarios and a case study.

11

Performance monitoring, analysis, and tracing

Performance tuning is about adapting the system settings in order to achieve optimum system behavior in terms of response time, throughput, and resource consumption. It is essential to regularly monitor and maintain your system, and identify and resolve any performance problems that have occurred, or before they occur. In this chapter, we discuss the baseline measurement, monitoring tools, and techniques to use for performance monitoring, analysis, and tracing for a Content Manager environment.

We cover the monitoring tools at the operating system level.

DB2 monitoring tools covered in this chapter include: DB2 Memory Visualizer, DB2 Health Monitor and DB2 Health Center, db2pd, DB2 Performance Expert, DB2 Explain, and DB2 Design Advisor.

The WebSphere Application Server monitoring tools we cover include: Tivoli Performance Viewer, WebSphere JVM verbosegc, and Java Virtual Machine Profiler Interface.

11.1 Introduction

In the previous chapters, we introduced performance-related concepts, best practices for Content Manager system performance, and how to tune individual base products. In this chapter, we address monitoring, analysis, and tracing of a Content Manager system.

Monitoring and analysis of your system environment should be done after you initially set the system up and after you change any significant aspect of the system's configuration. Because many performance issues tend to escalate with increasing speed after they hit some thresholds, it is important to monitor your system on a regular basis. In many cases, this is done when there is a performance problem and you need to figure out the location of the bottleneck that caused the problem. Bottleneck hunting and problem analysis as well as problem fixing might require some time. We strongly recommend continuously monitoring your system.

We offer details about monitoring for each building block of a Content Manager environment because each requires different techniques and tools. In addition to the monitoring tools, we also include the discussion of analysis and techniques. For the purpose of effective tuning, it might not be sufficient to simply collect certain monitoring data. Instead, you might need to explicitly perform analysis procedures such as analysis of DB2 access plans for query execution.

The tools and analysis techniques enable you to acquire the necessary system information and system performance measurements.

For each section, we include reference materials to read for more detailed information. The intention of this chapter is to provide an overview with a list of the basic tools. We do not provide in-depth documentation about how to use these tools, because this is explained in great detail in the referenced materials.

11.2 Measuring the baseline

Performance problems are often reported immediately following some change to system hardware or software. Unless there is a pre-change baseline measurement with which to compare post-change performance, quantification of the problem is impossible.

Always measure the environment before and after each change. Monitor the system and maintain the system performance profile at regular intervals (for example, once a week or once a month) and save the output. When a problem is found, the previous history can be used for comparison. It is worth monitoring system performance and collecting a series of outputs in order to support the diagnosis of a possible performance problem.

Sample workload statistics to collect include:

- Number of desktop and Web client users
- Number of each frequently performed operation (search, view, import, doc-routing) per hour
- Average document size and number of pages
- Time duration of completing batch jobs such as load process
- Time duration of partial and full system backup

Server resource utilization statistics to collect include:

- CPU utilization
- Memory utilization and footprint
- Disk utilization and disk I/O utilization
- Network utilization

Collect data from various periods of the working day, week, or month when performance is likely to be an issue. Some of the workload peaks might be:

- In the middle of the morning for users
- During late night for batch jobs
- During massive document loads
- Over the weekend during full system backup

Use measurements to collect data for each of these peaks in workload, because a performance problem might appear during only one of these periods and not during other times.

Note: Always be aware that any monitoring activity has an impact on the performance of the system being monitored.

In the following sections, we discuss monitoring tools at operating system level for each of the Content Manager base products (DB2, WebSphere Application Server), and performance analysis and tracing techniques for Content Manager.

11.3 Monitoring tools at operating system level

Monitoring the OS from the perspective of tuning performance of a Content Manager environment means to look at certain operating system measurement counters that show use of OS resources. We specifically look at CPU utilization, memory consumption, disk I/O, and network I/O.

11.3.1 Monitoring tools for AIX

For the AIX operating system, various monitoring utilities and tools are available that provide information about internals of the operating system kernel, I/O devices, and processes running. Although individual tools differ significantly in the specific information they provide and the way they provide the information, all of these tools internally interact to the same interface that the AIX operating system provides. Before going into detail about any monitoring tool, we give an overview of the measurement counters that AIX provides and that, from our experience, have shown to be the most important for tuning a Content Manager environment.

Measurement counters

The measurement counters that AIX and other UNIX systems provide for system monitoring are commonly grouped into the following categories:

- CPU statistics

  For each individual CPU, percentage of time that the CPU is in one of the four states:

  – SYSTEM: executing in system mode; that is, executing some application in "kernel protected" mode or system call.

  – USER: executing in user mode; that is, executing some application in "unprotected" mode.

  – WAIT: for example, processor is waiting for some disk or network I/O to complete.

  – IDLE

  Note: If you see CPUs excessively being in a WAIT state, it means there is a bottleneck in some of your I/O devices. You should avoid this situation. Take a closer look at the measurement counters for disks and network.

- Memory usage statistics

  Several counters are performance related:

  – Total size of physical memory, and the amount of physical memory used

  – Total size of paging space, and the amount of paging space used

  – Size and usage of file system cache

Note: If your paging space is being used up to the limit during regular operating hours, this means that the system might not have enough paging space during the peak time. You should avoid this situation.

In addition, if you see a lot of I/O to your paging space, this increases WAIT-time and consumes capacity of your I/O devices. You should avoid this, too. You might need to add more physical memory to your hardware.

- Operating system kernel statistics:
  - Length of run queue: the number of threads waiting to be scheduled to have some processor assigned.
  - Load: the number of processes ready to be performed by some processor; typically this is referred to as an average over some preceding time interval.

  A long run queue typically goes hand in hand with a high load.

Note: If you constantly see a long run queue, it means that each thread has to wait for a longer time before being scheduled for some CPU. You should avoid this situation.

For the same reason, you should avoid a situation in which you see a constant high load on your system.

- Network statistics

  For each of the network interfaces:
  - I/O rate in terms of number of data packets and bytes in both directions (in and out)
  - Collisions and errors on this network interface

Note: If you see constant data transfer rates on any of your network interfaces that come close to the interface's capacity, it means this network interface is about to be a bottleneck. It is very likely that you observe a high value for `CPU in WAIT state`. You should avoid this situation.

- Disk I/O statistics

  For each of the disks attached to your computer:
  - I/O rate in terms of the number of data packets and bytes in both directions (in and out)

Note: If you see constant data transfer rates to any of your disks that come close to the disk’s capacity, this implies an impending bottleneck. It is very likely that you will observe a high value for `CPU in WAIT state`. You should avoid this situation.

If you have multiple disks attached to your computer, be sure that the disk I/O is about evenly distributed over all disks.

Tools (commands) to display measurement counters

AIX provides a set of commands that each display some of the previously described measurement counters. We do not go into detail about all of them. Instead, for the most basic ones, which focus on specific subsets of the measurement counters, we briefly describe the functionality that they provide and later focus on two commands, **`topas`** and **`nmon`**, which are more suited to display an integrated overview of all relevant system information.

In our overview, whenever we talk about some command being a UNIX or AIX command, this means that this command also is available in other UNIX systems, not only AIX. However, parameter syntax as well as output of these commands might vary depending on the specific UNIX system. Check the product manual for specific syntax.

The commands we describe here include:

- **`sar`**
- **`iostat`**
- **`vmstat`**
- **`filemon`**
- **`topas`**
- **`nmon`**

sar

The UNIX System Activity Report (SAR) command **`sar`** samples measurement counters for a single “group of counters” at a time, such as the CPU-related counters. It places its output on standard output. You use the command line parameters to let **`sar`** display these counters for repeated times. For example, the command **`sar -u 1 5`** samples CPU-counters five times with 1 second between each sample. This can produce the output shown in Example 11-1 on page 287.

Example 11-1 Sample output of sar -u 1 5 command

```
AIX red2 3 5 002F129D4C00    04/14/06

System configuration: lcpu=2

11:10:08    %usr    %sys    %wio   %idle
11:10:09       0       2       0      98
11:10:10       0       1       0      99
11:10:11       0       2       0      98
11:10:12       0       1       0      99
11:10:13       0       1       0      99

Average        0       2       0      98
```

You can use the command parameters to have **sar** display more than one group of counters, but **sar** displays each group in a separate line so that the output might not be that readable.

iostat

The UNIX **iostat** command can be used to monitor disk and TTY I/O activity. Similar to **sar**, **iostat** samples the operating system internal counters and reports these values. On AIX, the output of the simple command **iostat 1 5** is shown in Example 11-2.

Example 11-2 Sample output of iostat 1 5 command

```
System configuration: lcpu=2 drives=6 paths=4 vdisks=0

tty:      tin         tout    avg-cpu: % user % sys % idle % iowait
          0.0         50.0                2.6  11.2   63.8     22.4

Disks:         % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk1           0.0       0.0      0.0          0         0
hdisk2          38.8    1341.4    232.8       1372       184
hdisk3           6.0     182.8     27.6        172        40
hdisk0           0.0       0.0      0.0          0         0
cd0              0.0       0.0      0.0          0         0
cd1              0.0       0.0      0.0          0         0

... < repeat counters 5 times >
```

vmstat

The UNIX command **vmstat** can be used to monitor memory usage. We do not go into detail here, but Example 11-3 shows sample output of **vmstat 5 3**.

Example 11-3 Sample output of vmstat 5 3 command

```
db2inst1@[/home/db2inst1]vmstat 5 3

System configuration: lcpu=2 mem=8192MB

kthr    memory              page                    faults        cpu
----- ------------ ------------------------ -------------- -----------
 r  b    avm   fre  re  pi  po  fr   sr  cy  in    sy   cs us sy id wa
 3  0 954201  1035   0   0   0  25   43   0 252  9989 2916  6  2 85  7
 1  0 954355  1416   0   0  27 129  202   0 229 10872 2448  4  6 84  7
```

filemon

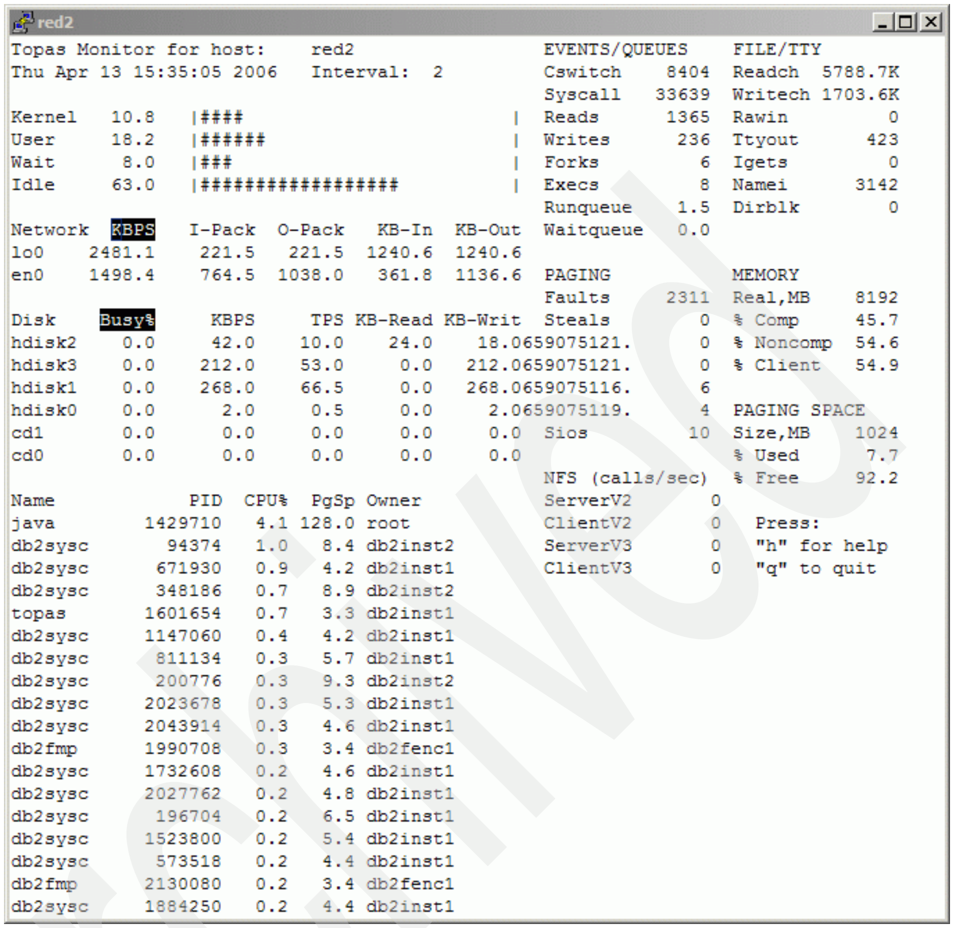
The UNIX command **filemon** monitors the performance of the file system and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes. This command is valuable especially if the table space containers of the Content Manager databases are system-managed (for information, refer to “Containers” on page 26).

topas

The **topas** command on AIX shows the information displayed by **sar**, **iostat**, and **vmstat**. The tool reports local system statistics such as CPU use, CPU events and queues, memory and paging use, disk performance, network performance, and NFS statistics. It also reports the top hot processes of the system. Hot processes are those processes that use large amount of CPU time. All information displayed by **topas** is real time.

When a bottleneck is identified, whether it is CPU, memory, or I/O bottleneck, you can use additional tools to further investigate the cause.

Figure 11-1 on page 289 displays a system status using **topas**.



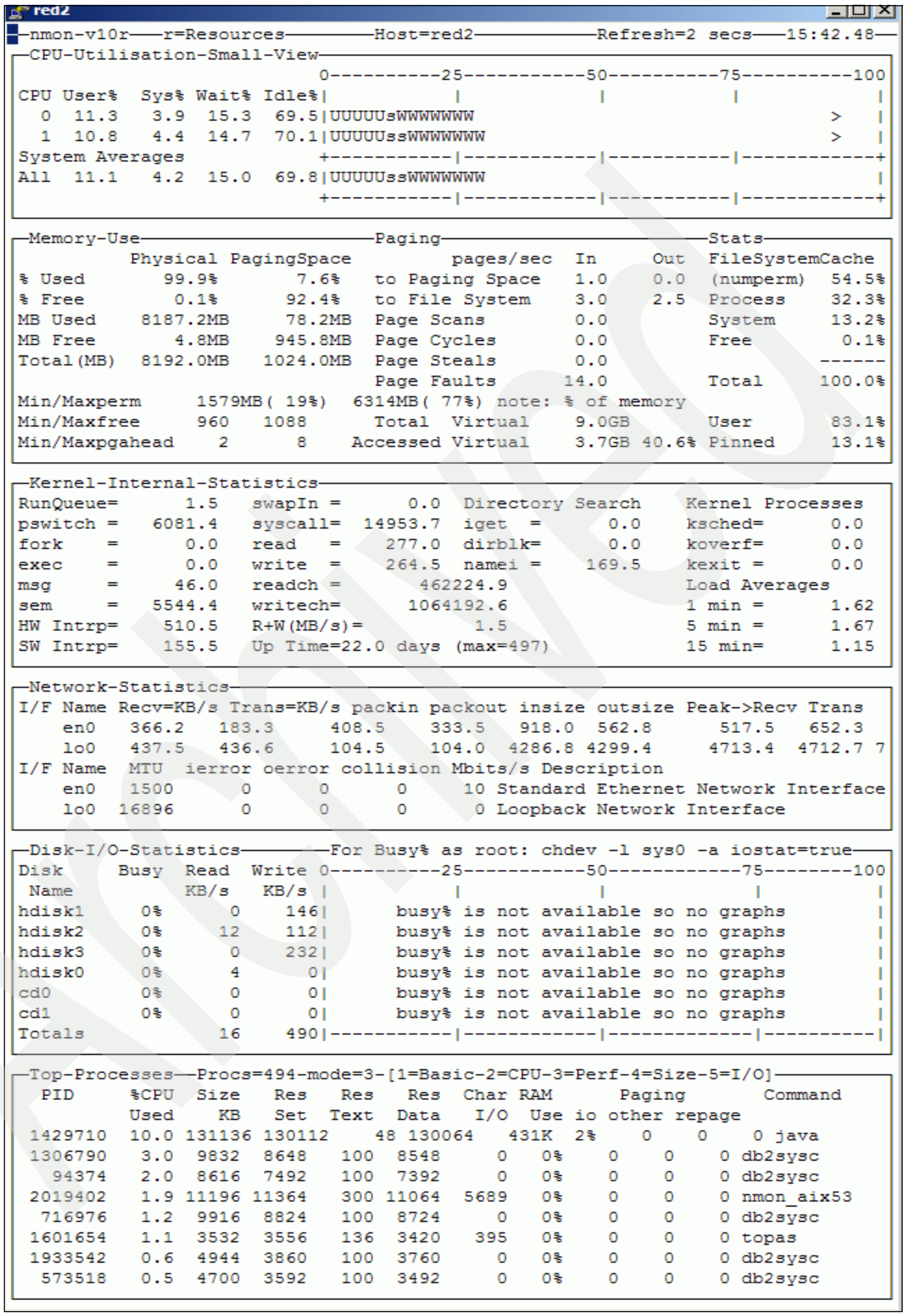
```
Topas Monitor for host:    red2                 EVENTS/QUEUES    FILE/TTY
Thu Apr 13 15:35:05 2006   Interval:  2         Cswitch     8404 Readch  5788.7K
                                                Syscall    33639 Writech 1703.6K
Kernel    10.8   |####                       |  Reads       1365 Rawin         0
User      18.2   |######                     |  Writes       236 Ttyout      423
Wait       8.0   |###                        |  Forks          6 Igets         0
Idle      63.0   |###################        |  Execs          8 Namei      3142
                                                Runqueue     1.5 Dirblk        0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out  Waitqueue    0.0
lo0    2481.1    221.5   221.5  1240.6  1240.6
en0    1498.4    764.5  1038.0   361.8  1136.6  PAGING           MEMORY
                                                Faults      2311 Real,MB    8192
Disk    Busy%     KBPS     TPS KB-Read KB-Writ  Steals         0 % Comp     45.7
hdisk2    0.0     42.0    10.0    24.0    18.0659075121.       0 % Noncomp  54.6
hdisk3    0.0    212.0    53.0     0.0   212.0659075121.       0 % Client   54.9
hdisk1    0.0    268.0    66.5     0.0   268.0659075116.       6
hdisk0    0.0      2.0     0.5     0.0     2.0659075119.       4 PAGING SPACE
cd1       0.0      0.0     0.0     0.0     0.0  Sios          10 Size,MB    1024
cd0       0.0      0.0     0.0     0.0     0.0                   % Used      7.7
                                                NFS (calls/sec)  % Free     92.2
Name            PID  CPU%  PgSp Owner           ServerV2       0
java        1429710   4.1 128.0 root            ClientV2       0   Press:
db2sysc       94374   1.0   8.4 db2inst2        ServerV3       0   "h" for help
db2sysc      671930   0.9   4.2 db2inst1        ClientV3       0   "q" to quit
db2sysc      348186   0.7   8.9 db2inst2
topas       1601654   0.7   3.3 db2inst1
db2sysc     1147060   0.4   4.2 db2inst1
db2sysc      811134   0.3   5.7 db2inst1
db2sysc      200776   0.3   9.3 db2inst2
db2sysc     2023678   0.3   5.3 db2inst1
db2sysc     2043914   0.3   4.6 db2inst1
db2fmp      1990708   0.3   3.4 db2fenc1
db2sysc     1732608   0.2   4.6 db2inst1
db2sysc     2027762   0.2   4.8 db2inst1
db2sysc      196704   0.2   6.5 db2inst1
db2sysc     1523800   0.2   5.4 db2inst1
db2sysc      573518   0.2   4.4 db2inst1
db2fmp      2130080   0.2   3.4 db2fenc1
db2sysc     1884250   0.2   4.4 db2inst1
```

Figure 11-1 topas display

nmon

The command **nmon** is a very convenient tool that provides a concise overview of a wide range of operating system measurement counters. **nmon** is not part of the AIX operating system and is not officially supported by IBM, but it is freely available through a search at `http://www.ibm.com/developerworks`. It is also available for the Linux operating system.

nmon is designed to be used as an interactive tool. It can continuously report the measurement counters to a file. Each group of counters can be toggled on and off, so you can easily tailor your monitoring view to your needs. Figure 11-2 on page 290 shows a screen capture of **nmon** displaying important groups of measurement counters.



```
red2
-nmon-v10r----r=Resources----------Host=red2------------Refresh=2 secs---15:42.48-
 CPU-Utilisation-Small-View
                            0----------25-----------50----------75----------100
CPU User%  Sys% Wait% Idle%|           |            |           |            |
  0  11.3   3.9  15.3  69.5|UUUUUsWWWWWWW                                 >  |
  1  10.8   4.4  14.7  70.1|UUUUUssWWWWWWWW                               >  |
System Averages            +-----------|------------|-----------|------------+
All  11.1   4.2  15.0  69.8|UUUUUssWWWWWWWW                                  |
                           +-----------|------------|-----------|------------+

 Memory-Use---------------------Paging-------------------------Stats-
          Physical PagingSpace         pages/sec  In     Out   FileSystemCache
% Used       99.9%        7.6%   to Paging Space  1.0    0.0   (numperm)  54.5%
% Free        0.1%       92.4%   to File System   3.0    2.5   Process    32.3%
MB Used   8187.2MB      78.2MB   Page Scans       0.0          System     13.2%
MB Free      4.8MB     945.8MB   Page Cycles      0.0          Free        0.1%
Total(MB) 8192.0MB    1024.0MB   Page Steals      0.0                     ------
                                 Page Faults     14.0          Total     100.0%
Min/Maxperm     1579MB( 19%)  6314MB( 77%) note: % of memory
Min/Maxfree     960    1088      Total  Virtual   9.0GB        User       83.1%
Min/Maxpgahead    2       8    Accessed Virtual   3.7GB 40.6% Pinned     13.1%

 Kernel-Internal-Statistics
RunQueue=       1.5   swapIn =       0.0  Directory Search    Kernel Processes
pswitch =    6081.4   syscall=   14953.7  iget   =      0.0   ksched=       0.0
fork    =       0.0   read    =    277.0  dirblk=       0.0   koverf=       0.0
exec    =       0.0   write   =    264.5  namei =     169.5   kexit =       0.0
msg     =      46.0   readch =        462224.9                Load Averages
sem     =    5544.4   writech=       1064192.6                1 min =      1.62
HW Intrp=     510.5   R+W(MB/s)=           1.5                5 min =      1.67
SW Intrp=     155.5   Up Time=22.0 days (max=497)             15 min=      1.15

 Network-Statistics
I/F Name Recv=KB/s Trans=KB/s packin packout insize outsize Peak->Recv Trans
    en0   366.2    183.3      408.5    333.5   918.0  562.8      517.5   652.3
    lo0   437.5    436.6      104.5    104.0  4286.8 4299.4     4713.4  4712.7 7
I/F Name  MTU  ierror oerror collision Mbits/s Description
    en0   1500      0      0       0      10 Standard Ethernet Network Interface
    lo0  16896      0      0       0       0 Loopback Network Interface

 Disk-I/O-Statistics-------For Busy% as root: chdev -l sys0 -a iostat=true-
Disk      Busy  Read  Write 0----------25-----------50------------75--------100
 Name           KB/s   KB/s |           |            |             |         |
hdisk1      0%     0    146|      busy% is not available so no graphs        |
hdisk2      0%    12    112|      busy% is not available so no graphs        |
hdisk3      0%     0    232|      busy% is not available so no graphs        |
hdisk0      0%     4      0|      busy% is not available so no graphs        |
cd0         0%     0      0|      busy% is not available so no graphs        |
cd1         0%     0      0|      busy% is not available so no graphs        |
Totals            16    490|-----------|------------|-------------|----------|

 Top-Processes---Procs=494-mode=3-[1=Basic-2=CPU-3=Perf-4=Size-5=I/O]-
  PID     %CPU  Size   Res    Res   Res  Char RAM      Paging       Command
          Used    KB   Set   Text  Data   I/O  Use io other repage
 1429710  10.0 131136 130112    48 130064  431K  2%    0     0     0 java
 1306790   3.0   9832   8648   100   8548     0  0%    0     0     0 db2sysc
   94374   2.0   8616   7492   100   7392     0  0%    0     0     0 db2sysc
 2019402   1.9  11196  11364   300  11064  5689  0%    0     0     0 nmon_aix53
  716976   1.2   9916   8824   100   8724     0  0%    0     0     0 db2sysc
 1601654   1.1   3532   3556   136   3420   395  0%    0     0     0 topas
 1933542   0.6   4944   3860   100   3760     0  0%    0     0     0 db2sysc
  573518   0.5   4700   3592   100   3492     0  0%    0     0     0 db2sysc
```

Figure 11-2   nmon display

11.3.2 Monitoring tools for Windows

For the Windows operating system, you can use Windows Task Manager and other tools to monitor system activities. We introduce the following tools here:

- Windows Task Manager
- Windows Performance Monitor

Windows Task Manager

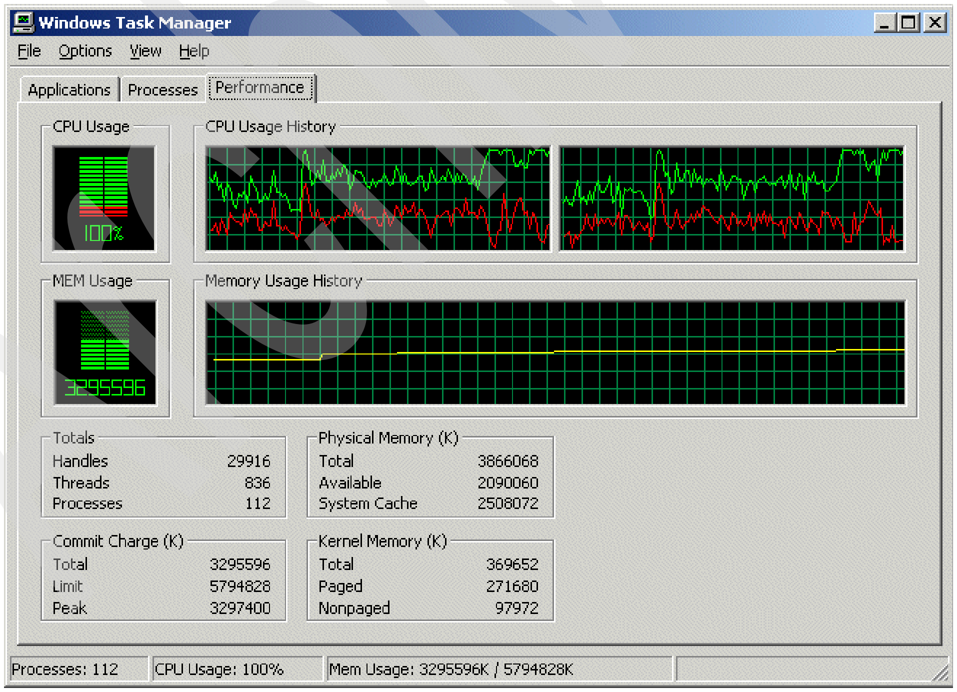
The Windows Task Manager provides basic system performance monitoring functionality.

To start the Task Manager, right-click an empty space on the taskbar and click **Task Manager**.

- Click the **Performance** tab.

  As shown in Figure 11-3), the Performance tab displays:

  – Graphical representation of CPU and memory
  – The number of handles, threads, and processes running information
  – The physical, kernel, and commit memory information
  – The processes that are currently running on your computer



Totals
Handles 29916
Threads 836
Processes 112

Physical Memory (K)
Total 3866068
Available 2090060
System Cache 2508072

Commit Charge (K)
Total 3295596
Limit 5794828
Peak 3297400

Kernel Memory (K)
Total 369652
Paged 271680
Nonpaged 97972

Processes: 112 CPU Usage: 100% Mem Usage: 3295596K / 5794828K

Figure 11-3 Windows Task Manager: Performance tab

If you have more than one CPU in the system and would like to see a graphical representation for each CPU:

a. Select **View** → **CPU History**.
b. Select **One Graph Per CPU**.

- Click the **Processes** tab (Figure 11-4) to see information about the processes that are currently running on your computer:
  - CPU usage and CPU time
  - Memory usage and peak memory usage
  - Numbers of handles and threads count
  - I/O reads and writes
  - Virtual memory size
  - Paged and non-paged pool

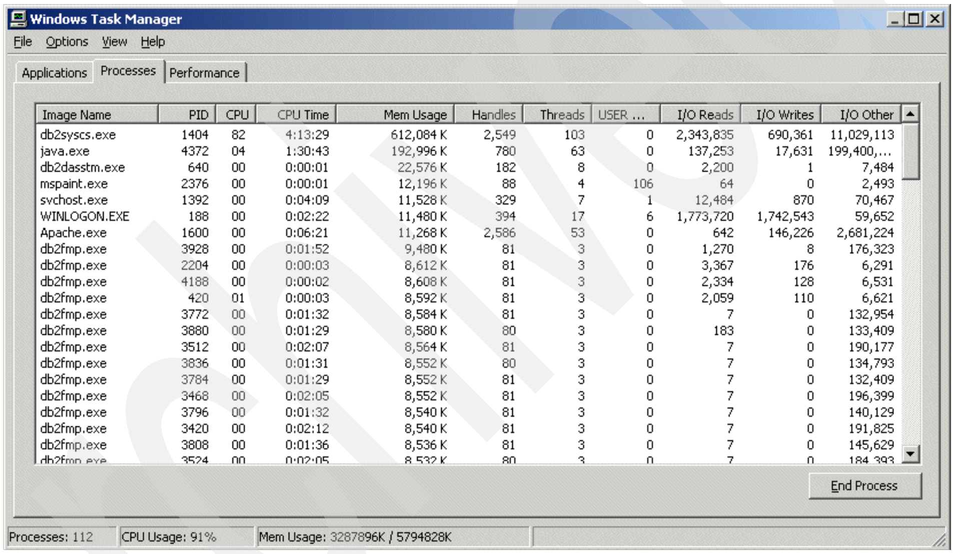


Image Name	PID	CPU	CPU Time	Mem Usage	Handles	Threads	USER ...	I/O Reads	I/O Writes	I/O Other
db2syscs.exe	1404	82	4:13:29	612,084 K	2,549	103	0	2,343,835	690,361	11,029,113
java.exe	4372	04	1:30:43	192,996 K	780	63	0	137,253	17,631	199,400,...
db2dasstm.exe	640	00	0:00:01	22,576 K	182	8	0	2,200	1	7,484
mspaint.exe	2376	00	0:00:01	12,196 K	88	4	106	64	0	2,493
svchost.exe	1392	00	0:04:09	11,528 K	329	7	1	12,484	870	70,467
WINLOGON.EXE	188	00	0:02:22	11,480 K	394	17	6	1,773,720	1,742,543	59,652
Apache.exe	1600	00	0:06:21	11,268 K	2,586	53	0	642	146,226	2,681,224
db2fmp.exe	3928	00	0:01:52	9,480 K	81	3	0	1,270	8	176,323
db2fmp.exe	2204	00	0:00:03	8,612 K	81	3	0	3,367	176	6,291
db2fmp.exe	4188	00	0:00:02	8,608 K	81	3	0	2,334	128	6,531
db2fmp.exe	420	01	0:00:03	8,592 K	81	3	0	2,059	110	6,621
db2fmp.exe	3772	00	0:01:32	8,584 K	81	3	0	7	0	132,954
db2fmp.exe	3880	00	0:01:29	8,580 K	80	3	0	183	0	133,409
db2fmp.exe	3512	00	0:02:07	8,564 K	81	3	0	7	0	190,177
db2fmp.exe	3836	00	0:01:31	8,552 K	80	3	0	7	0	134,793
db2fmp.exe	3784	00	0:01:29	8,552 K	81	3	0	7	0	132,409
db2fmp.exe	3468	00	0:02:05	8,552 K	81	3	0	7	0	196,399
db2fmp.exe	3796	00	0:01:32	8,540 K	81	3	0	7	0	140,129
db2fmp.exe	3420	00	0:02:12	8,540 K	81	3	0	7	0	191,825
db2fmp.exe	3808	00	0:01:36	8,536 K	81	3	0	7	0	145,629
db2fmp.exe	3524	00	0:02:05	8,532 K	80	3	0	7	0	184,393

Processes: 112 CPU Usage: 91% Mem Usage: 3287896K / 5794828K

Figure 11-4 Windows Task Manager: Processes tab

- To configure the system to display different column information on the Processes tab:

a. Select **View** → **Select Columns**.

b. In the opened Select Columns dialog box, select the columns of interest for performance tuning. Click **OK**.

For more information, refer to the *Windows 2000 Professional Resource Guide*, which is part of the Windows 2000 Professional Resource Kit.

Windows Performance Monitor

The Windows Performance Monitor is the best tool for keeping track of system resource utilization on Win2000/XP servers.

Note: The white paper *Content Manager V8.3 Performance Monitoring Guide* provides a good description of this tool. For your convenience, we extracted its content in this section.

Windows Performance Monitor is part of the Administrative tools in the Windows Control Panel. It enables you to log counter and event trace data. Counters are specific data items associated with a performance object. A performance object is an individual resource or service that can be measured. Examples of performance objects include processor, memory, paging file, and physical disk. Examples of counters for the processor performance objects include % of processor time and interrupts per second. Performance objects that are critical to monitor on Content Manager servers are:

- Processor
- Memory
- Network interface
- Paging file
- Physical disk
- Process

Creating counter logs to monitor Content Manager servers

You need to create a counter log for each of your Content Manager servers before you can use the counter log to view and monitor system activities.

To create a counter log for a server, follow these steps:

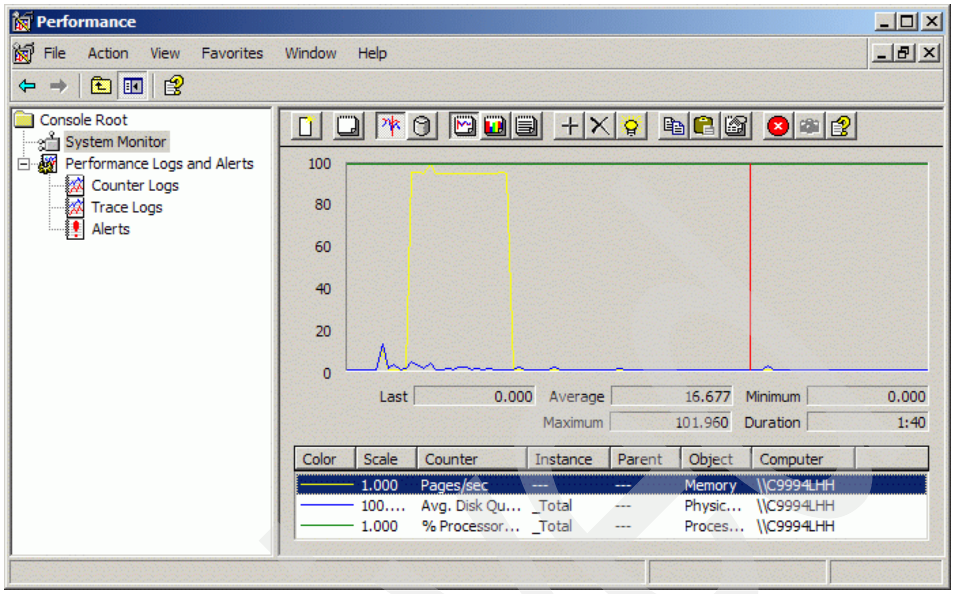
1. Make sure all Content Manager related services, such as programs, are started prior to starting the Performance tool. The tool will allow you only to monitor processes running at the time to define your counter log.
2. Open the Performance tool by clicking **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Performance**.
3. Click the **Tree** tab and expand **Performance Logs and Alerts**.
4. Right-click **Counter Logs** and select **New Log Settings**. Enter a name for this counter log, such as `CM Server Counter Log`. You will see that the current counter log is stored in a location such as C:\PerfLogs\.
5. Click **Add** to add a counter. This opens the Select Counters window. You can select counters from the local computer or from other computers on your network. This enables you to monitor all Content Manager servers on your network from one central location.

6. Select the **Processor** in the Performance object field. Click the radio button for **All counters** and **All instances**. Click **Add**. Continue this for the memory, network interface, paging file, physical disk, and process performance objects. For each performance object, select all counters and instances. The only exception to this will be for network interface. You may have multiple network interface cards, including a loopback adapter. Select only the active network interface card. Sampling the data every 15 seconds should be sufficient for basic monitoring purposes. If you need more information about these counters, use the **Explain** button.
7. When you are finished, click the **Schedule** tab. Select when you want the monitor to run. You might want to create two versions of the counter log: one that runs daily at peak hours and a second one that runs for the course of an entire week a few times per year. Click **OK** when you are done.

Viewing a counter log

After the counter logs are created, you can view them by following these steps:

1. From the Performance window, click **System Monitor**. There is an icon of a disk drive, which is for viewing log files.
2. Select your Content Manager server log file from the directory it is stored in. When you click **Open**, you will notice that no information is shown on the chart. You have to select which items you want to view (you can view counters only for performance objects you originally selected).
3. Right-click on the chart and select **Add Counters**. You can now choose which counters you want to view. (Trying to view all counters at once would make interpreting the chart virtually impossible). Figure 11-5 on page 295 shows an example of the percent of processor used and the percent of idle disk time.



Last 0.000 Average 16.677 Minimum 0.000
Maximum 101.960 Duration 1:40

Color	Scale	Counter	Instance	Parent	Object	Computer
	1.000	Pages/sec	---	---	Memory	\\C9994LHH
	100....	Avg. Disk Qu...	_Total	---	Physic...	\\C9994LHH
	1.000	% Processor...	_Total	---	Proces...	\\C9994LHH

Figure 11-5 Viewing a counter log

11.4 Monitoring and analysis tools for DB2

In 2.1.3, “Performance-related concepts” on page 39, we mentioned a variety of performance-related concepts in DB2. Each of these concepts could be the source of some performance bottleneck if the system is not configured appropriately. In this section, we go into the details of which monitoring data to look at to conclude that certain configuration aspects should be modified.

We focus on the following concepts of DB2:

- Access plan
- Indexes
- Clustering
- Multi-dimensional clustering
- Materialized query tables
- Table space types
- I/O servers
- Coordinator agents
- Data placement
- Locking, lock escalation, and deadlocks

For each of these concepts, we describe which performance-related measurement data you may collect and which tool you may use to do this. Many different tools can be used for this purpose. In this section, we focus on the tools that either come with the DB2 database system, are members of the DB2 family of products, or are part of the underlying operating system.

Because of the wide variety of tools that are available and the diversity of data to collect, we typically stay at a generic level of description and present screen captures to demonstrate the nature of the data collected. We do not go into detail about how to use these tools, and instead refer to documentation that can be studied for details.

Much of the performance-related data may be collected by more than just one monitoring tool, so we start with an overview of the individual techniques and tools for monitoring, and close with a summary of the tools that can be used to monitor the various performance-related concepts.

11.4.1 Basic DB2 monitoring, logging, and tracing interfaces

The retrievable performance-related data in DB2 relies heavily on mechanisms and interfaces that DB2 provides to access this data. DB2 keeps track of its own operation, performance, and the applications using it.

You can access this information using two built-in DB2 monitoring interfaces:

- Snapshot monitoring
- Event monitoring

Each interface provides a slightly different view of the data and differs from the other in the way data is accessed. In this section, we give an overview of these interfaces and describe what to consider when deciding which interface to use for your purposes.

DB2 also provides logging and tracing interfaces. You can configure how much information you want DB2 to provide. These files are useful for troubleshooting. You can also find valuable hints for tracking down performance problems. From the performance aspect, we discuss:

- DB2 diagnostic file: db2diag.log
- DB2 CLI and JDBC trace

Snapshot monitoring

As the name already suggests, you can use the snapshot monitor to capture information about the database and connected applications *at a specific time*. Snapshots are useful for determining the status of a database system. Taken at regular intervals, they are also useful for observing trends and foreseeing potential problems.

The data that can be captured using snapshot monitoring is structured in *categories* (also known as types of monitoring elements in DB2 documentation). Because snapshot monitoring implies a certain overhead in database manager processing, snapshot monitoring is not activated by default for most of these categories. Snapshot monitoring for each category can be toggled on and off individually (Table 11-1).

Table 11-1 DB2 snapshot categories

Snapshot category	**Switch**	**Default**
Buffer pool activity information	BUFFERPOOL	OFF
Lock wait, and time-related lock information	LOCK	OFF
Sorting information	SORT	OFF
SQL statement information	STATEMENT	OFF
Table activity information	TABLE	OFF
Times and timestamp information	TIMESTAMP	ON
Unit of work information	UOW	OFF

Before capturing a snapshot or using an event monitor, determine what data you need the database manager to gather and set only the corresponding switches. The following DB2 command can be used to toggle on all switches:

```
UPDATE MONITOR SWITCHES USING BUFFERPOOL ON LOCK ON SORT ON STATEMENT
ON TIMESTAMP ON TABLE ON UOW ON
```

To check the status of snapshot monitoring switches, use this DB2 command:

```
GET MONITOR SWITCHES
```

When using snapshot monitoring, consider these facts:

- Snapshot monitoring consumes some system resources and can affect general system performance. We recommend switching it on *if and only if* monitoring actually is done.
- The database manager uses some internal memory heap to store snapshot data. The size of this heap is always limited and can be configured using the database manager configuration parameter MON_HEAP_SZ. The amount of data covered by some snapshot is determined by the load on the database system as well as by the amount of memory that you allow the database manager to use for monitoring. You might want to increase the default setting before starting to activate a snapshot monitor.

- Because the amount of data for snapshot monitoring is limited, there is no guarantee that a specific event that you are looking for is covered by a snapshot. Similarly, in general, there is no way to determine the amount of overlap that might exist between two subsequent snapshots. Thus, a snapshot can give you a good understanding of the current status of the system, but it is not suited to completely cover a given time span.
- Activation and deactivation of snapshot monitoring is done per user connection. This means that after activating snapshot monitoring on just one database connection, you will see snapshot data only in this specific connection. However, the snapshot data will cover all events, not just those that correspond to this connection. You can set some database manager configuration parameters to change the default setting for monitor switches. For each monitor switch, such as BUFFERPOOL, the corresponding configuration parameter resembles the switch name prefixed with DFT_MON_ (for example, DFT_MON_BUFPOOL). After instance activation, this setting applies to each newly opened database connection.

DB2 provides access to snapshot monitor data in three different ways:

- You can capture a snapshot using a DB2 command to extract the data.
- You can use the SQL table functions to query snapshot data within an SQL statement such as:

  ```
  SELECT * FROM TABLE(SNAPSHOT_APPL('SAMPLE', -1)) AS SNAPSHOT_APPL
  ```

  This allows for much more sophisticated data analysis and filtering but requires an advanced level of SQL skill. We do not offer further details here.
- DB2 provides APIs for programming languages such as C and C++. Most monitoring tools internally base on these APIs. We do offer further details here.

We discuss extracting snapshot monitor data using a native, built-in DB2 command. This command comes with different flavors reflecting the individual categories of snapshot data. We give an overview of those command variants that provide the most helpful information when monitoring a database environment. For each of them, we list the most valuable information with respect to performance-related system status that you find in the output of the command.

At the database manager level, the following variants are most prominent:

- GET SNAPSHOT FOR DATABASE MANAGER

  Besides information on the database manager configuration, this command provides information about the following database manager related aspects:

  - Allocation and creation of agents in response to clients connecting to the database

  Decide whether your environment settings for NUM_POOLAGENTS, MAX_COORDAGENTS, and MAXAGENTS fit the requirements.
  - Size and usage of various memory segments such as monitor heap

    Conclude whether the memory layout in your database needs tuning.

- GET SNAPSHOT FOR ALL APPLICATIONS

  This command provides detailed information about each individual database connection, specifically the following aspects:

  - Resource consumption in terms of CPU time, buffer pool access, table I/O, and others

    You can use this information to identify the trouble connection, analyze it individually, and treat it.

  - The last SQL statement issued on this connection

    In case of a hanging application, this might provide the clue. It can be a complex statement that needs individual analysis, or this statement generates a lock on something already locked by a different application.

  - Detailed information about various memory pools assigned to the application and their actual use

    From this, you may conclude whether the configuration of these memory pools needs tuning.

- GET SNAPSHOT FOR ALL BUFFERPOOLS

  This command lists information about the use of each individual buffer pool:

  - Number of reads and writes on this buffer pool, logical as well as physical

    From these numbers, you can derive the buffer pool hit ratio. This ratio is defined as:

    ```
    (1 - (number of physical reads / number of logical reads)) * 100
    ```

    As a general rule, if this ratio is below about 90-95% for index buffer pools or below 75-80% for data buffer pools, this can indicate that a buffer pool is configured too small and should be increased in size.

    Refer to 8.5.1, “Buffer pool hit ratios” on page 197 for more information about the hit ratios.

  - Table spaces that the buffer pool is assigned to

    You can see whether your buffer pool is assigned to more than just the table space you thought of.

For snapshot data specific to your database, replace <myDbName> with the name of the database in the following commands:

- GET SNAPSHOT FOR DATABASE ON <myDbName>

  This command provides information about a specific database:

  – Accumulated information about buffer pools in the database and their use
  – Accumulated information about the statements performed against the database
  – Detailed information about various memory pools assigned to the database and their actual use

    From this, you can conclude whether the configuration of these memory pools needs tuning.

- GET SNAPSHOT FOR APPLICATIONS ON <myDbName>

  This command is similar to the analogous command at database manager level except that it lists only connections for the specified database.

- GET SNAPSHOT FOR TABLES ON <myDbName>

  This command provides information about the following aspects of tables in the given database:

  – The actual size of each table that has been accessed since activation of snapshot monitoring
  – The I/O hit of the table spaces

  From this, you can identify heavily accessed tables and use this as a starting point for further analysis. If the reason is not poor SQL or poor application logic (which is likely in most of the situations), one tuning option is to place this table in a different table space with a dedicated buffer pool assigned to it.

- GET SNAPSHOT FOR TABLESPACES ON <myDbName>

  This command provides information about the following aspects of table spaces in the given database:

  – The actual size of each table space in the specified database
  – The table space containers that are associated with the table space
  – The I/O hit of the table spaces

  From this, you can identify heavily accessed table spaces and use this as a starting point for further analysis and tuning. When further analyzing physical I/O, special focus should be on the potential disk I/O bottlenecks.

- GET SNAPSHOT FOR LOCKS ON <myDbName>

  This command provides information about applications holding and requesting locks in the database, the type of locks and database tables, or the indexes that these locks refer to.

  This snapshot data is especially valuable when tracking unexpected locks in concurrent applications.
- GET SNAPSHOT FOR BUFFERPOOLS ON <myDbName>

  This command is similar to the analogous command at database manager level except that it lists only the buffer pools for the specified database.
- GET SNAPSHOT FOR DYNAMIC SQL ON <myDbName>

  This command provides valuable information about the SQL statements that were run in the particular database.

For a complete list of the command parameters, refer to *DB2 UDB Command Reference V8.2*, SC09-4828.

The output of each of the listed command variants differ significantly.

As an example, Example 11-4 provides a section of the output using the DB2 command GET SNAPSHOT FOR ALL BUFFERPOOLS.

Example 11-4 Sample snapshot data for buffer pools

```
Bufferpool Snapshot

Bufferpool name                           = ICMLSMAINBP32
Database name                             = ICMNLSDB
Database path                             = /dbdata/ls/db2inst1/NODE0000/SQL00001/
Input database alias                      = ICMNLSDB
Snapshot timestamp                        = 04/05/2006 11:42:04.496972

Buffer pool data logical reads            = 168730364
Buffer pool data physical reads           = 30371445
Buffer pool temporary data logical reads  = 37605
Buffer pool temporary data physical reads = 0
Buffer pool data writes                   = 280
Buffer pool index logical reads           = 342506032
Buffer pool index physical reads          = 71926240
Buffer pool temporary index logical reads  = 0
Buffer pool temporary index physical reads = 0
Total buffer pool read time (milliseconds) = 1668
Total buffer pool write time (milliseconds)= 2727
Asynchronous pool data page reads         = 0
Asynchronous pool data page writes        = 183
Buffer pool index writes                  = 499
Asynchronous pool index page reads        = 0
Asynchronous pool index page writes       = 403
Total elapsed asynchronous read time      = 0
Total elapsed asynchronous write time     = 1228
Asynchronous data read requests           = 0
Asynchronous index read requests          = 0
...
```

This output displays important information about the usage of buffer pool ICMLSMAINBP32 such as the buffer pool hit ratio, which is close to 82% for data (168730364 logical data reads causing 30371445 physical reads) and 79% for index. This is below the recommended value of 85% discussed in 8.5, "Buffer pool tuning" on page 196, so we should think of increasing the buffer pool size.

Example 11-5 shows which performance related data you can retrieve when using snapshot data for DYNAMIC SQL statements. We do not list all parts of the SQL statement (indicated by "...") because these details are not relevant here.

Example 11-5 Sample snapshot data for DYNAMIC SQL statements

```
...
Number of executions                = 63
Number of compilations              = 1
Worst preparation time (ms)         = 12
Best preparation time (ms)          = 12
Internal rows deleted               = 0
Internal rows inserted              = 0
Rows read                           = 11976
Internal rows updated               = 0
Rows written                        = 0
Statement sorts                     = 63
Statement sort overflows            = 0
Total sort time                     = 0
Buffer pool data logical reads      = 5197
Buffer pool data physical reads     = 0
Buffer pool temporary data logical reads   = 0
Buffer pool temporary data physical reads  = 0
Buffer pool index logical reads     = 735
Buffer pool index physical reads    = 0
Buffer pool temporary index logical reads  = 0
Buffer pool temporary index physical reads = 0
Total execution time (sec.ms)       = 0.072253
Total user cpu time (sec.ms)        = 0.001637
Total system cpu time (sec.ms)      = 0.000310
Statement text                      = SELECT TMP.COMPONENTID, TMP.RTARGETITEMID,
TMP.USERID FROM (SELECT ... ) TMP WHERE ... ORDER BY TMP.PRIORITY
DESC,TMP.WORKNODEORDER ,TMP.LASTACCTIME ASC WITH UR
```

When looking at the snapshot for SQL statements, you specifically search for individual statements that show one of the following characteristics:

- A large number of rows read

  This could indicate that the application retrieves more rows than actually needed. You should look into the application. Another possibility is that the table might benefit from an additional index.

- A high value for total execution or CPU time per statement execution

  This can indicate that this statement is expensive for DB2 to process. Statistics might not be up to date, so DB2 does not choose the correct access plan. Otherwise, we recommend analyzing this query further using techniques described n 11.4.3, “DB2 analysis tools” on page 328. This way, we can determine whether an index is missing or the SQL statement is expensive by its very nature.

- A large number of compilations

  This might indicate that the package cache should be increased. This refers to configuration parameter PCKCACHESZ at database level.

  The CATALOGCACHE_SZ is another configuration parameter that might allow more catalog information to be stored in memory, reducing the need for catalog I/O during compilation.

- A large number of sorts per statement execution

  This also might indicate that some index is missing. Proceed with further analysis of this statement as described earlier.

- A large number of sort overflows

  This might indicate that the execution of this statement internally involves sorts on huge amounts of data that do not fit into the sort heap of the database. You could increase the setting for configuration parameter SORTHEAP at database level or you may further analyze, if sorting can be avoided by some extra index on the data.

  Note that if the SORTHEAP is too large, then multiple concurrent sorts might consume most of the memory and starve other concurrent sorts or hash joins.

In their output, most of the command variants also include configuration information that is not directly related to any events in the database. This is displayed even if snapshot monitoring is not switched on at all. This way, the snapshot for BUFFERPOOLS, for example, also displays information about the size of buffer pools and about the table spaces using this buffer pool.

Event monitoring

The event monitoring interface of DB2 enables you to access information about events in the database system. Similar to snapshot monitoring, event monitoring structures the information it provides using categories. With event monitoring, these categories are called *event types*. Event monitoring can be activated and deactivated independently for each event type. Table 11-2 lists the individual event types that DB2 distinguishes, and describes when these events are reported and what information is collected with each event.

Table 11-2 DB2 event types and information related to event types

Event type	When data is collected	Available information
DEADLOCKS	Detection of a deadlock	Applications involved, and locks in contention.
DEADLOCKS WITH DETAILS	Detection of a deadlock	Comprehensive information regarding applications involved, identification of participating statements, and a list of locks being held.
STATEMENTS	End of SQL statement	Statement start and stop time, CPU used, text of dynamic SQL, return code of SQL statement, and other metrics such as fetch count.
TRANSACTIONS	End of unit of work	End of the unit of work (UOW) start and stop time, previous UOW time, CPU consumed, and locking and logging metrics.
CONNECTIONS	End of connection	All application level counters.
DATABASE	Database deactivation	All database level counters.
BUFFERPOOLS	Database deactivation	Counters for buffer pool, prefetchers, page cleaners, and direct I/O for each buffer pool.
TABLESPACES	Database deactivation	Counters for buffer pool, prefetchers, page cleaners, and direct I/O for each table space.
TABLES	Database deactivation	Rows read or written for each table.

None of the event monitors is activated by default and several decisions should be made before creating and activating an event monitor.

Because event monitoring is an ongoing process that records event data as long the monitor is active, you must provide DB2 with information about where you want these event records to be stored. DB2 offers the following options:

- DB2 can automatically put the event records *in database tables*. These tables are created automatically by DB2, but it is up to you to provide table spaces that are prepared to accept the expected data volume. This method enables powerful analysis of event monitor data with DB2 itself.
- DB2 can post the event records *to named pipe*. It is up to you to set up an application to read the event records from this pipe.
- DB2 can put the event records *in a file* in the file system. It is up to you to provide an adequate file system.

Note that event monitors, especially statement event monitors, can generate huge volumes of data. All three storage methods can consume significant amounts of space if an event monitor runs for a long period of time. In general, it is a good idea to keep track of how much space is being consumed by an event monitor, and to deactivate the monitor if space consumption grows too quickly.

An event monitor must be created before it can be used. Due to the wide variety of flavors of event monitors and options to choose from, we do not offer all details of the command syntax for creating an event monitor. Instead, we give two examples; for details, refer to *DB2 UDB Command Reference V8.2*, SC09-4828.

The following DB2 command creates an event monitor for deadlock events with details. Events are reported to database tables:

```
CREATE EVENT MONITOR dlMon FOR DEADLOCKS WITH DETAILS WRITE TO TABLE
```

In general, more than one table is used to store the event information. You find these tables if you look for tables with a suffix such as _<*monitor name*>. In the case of a DEADLOCK event monitor, the tables CONNHEADER_DLMON, DEADLOCK_DLMON, DLCONN_DLMON, and CONTROL1_DLMON are used to store the event information. They can be queried to perform further analysis of the events.

The following command creates an event monitor for statement events. Events are reported to files in the specified directory (on UNIX):

```
CREATE EVENT MONITOR stMon FOR STATEMENTS WRITE TO FILE '/tmp/stEvents'
```

After creating the event monitor, it is in a deactivated state. To activate it, use the following command:

```
SET EVENT MONITOR <monName> STATE 1
```

In this command, <monName> is the name of the monitor that you used when creating it. Set state to 0 to deactivate the monitor.

Because DB2 uses internal memory heaps to store snapshot data, you can explicitly flush it to force event records to be written to their destination. Use the following command to do this:

```
SET FLUSH EVENT MONITOR <monName>
```

Event monitors are automatically flushed when they are deactivated.

When they are written to a named pipe or file, event records are in a binary format and must be converted using some programs. DB2 provides the command **db2evmon** for this purpose. It may be used as follows:

```
db2evmon -path '/tmp/dlevents'
db2evmon -db 'myDb' -evm 'stMon'
```

Example 11-6 shows the information that can be taken from an event log for buffer pool activity. This information is analogous to snapshot data for buffer pool and you should interpret it as described in “Snapshot monitoring” on page 296.

Example 11-6 Sample event log for buffer pool events

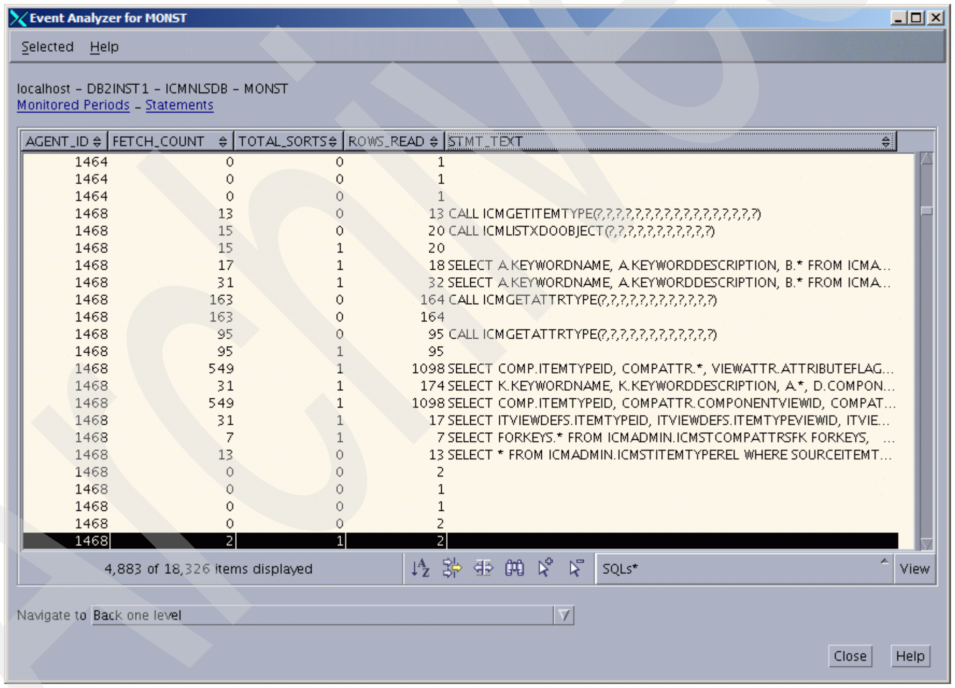
```
3) Bufferpool Event ...
  Bufferpool Name: IBMDEFAULTBP
  Database Name: ICMNLSDB
  Database Path: /dbdata/ls/db2inst1/NODE0000/SQL00001/

 Buffer Pool Statistics:
  Buffer pool logical data page reads: 850
  Buffer pool physical data page reads: 192
  Buffer pool data page writes: 4
  Buffer pool logical index page reads: 1323
  Buffer pool physical index page reads: 168
  Buffer pool index page writes: 1
  Buffer pool read time (milliseconds): 534
  Buffer pool write time (milliseconds): 19
  Files closed: 0
  Buffer pool asynch data page reads: 38
  Buffer pool asynch data page read reqs: 2
  Buffer pool asynch data page writes: 0
  Buffer pool asynch index page reads: 0
  Buffer pool asynch index page read reqs: 0
  Buffer pool asynch index page writes: 0
  Buffer pool asynch read time: 3
  Buffer pool asynch write time: 0
...
```

For event monitor data that is put in database tables, DB2 provides the interactive event analyzer with a graphical interface to browse through the data.

Open the interactive event analyzer by typing `db2eva` at the command line. It comes with built-in filtering mechanisms and enables you to customize the columns of event data that you are interested in.

Figure 11-6 shows the event analyzer based on an example for statement event data.



localhost - DB2INST1 - ICMNLSDB - MONST
Monitored Periods - Statements

AGENT_ID	FETCH_COUNT	TOTAL_SORTS	ROWS_READ	STMT_TEXT
1464	0	0	1	
1464	0	0	1	
1464	0	0	1	
1468	13	0	13	CALL ICMGETITEMTYPE(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)
1468	15	0	20	CALL ICMLISTXDOOBJECT(?,?,?,?,?,?,?,?,?,?,?,?)
1468	15	1	20	
1468	17	1	18	SELECT A.KEYWORDNAME, A.KEYWORDDESCRIPTION, B.* FROM ICMA...
1468	31	1	32	SELECT A.KEYWORDNAME, A.KEYWORDDESCRIPTION, B.* FROM ICMA...
1468	163	0	164	CALL ICMGETATTRTYPE(?,?,?,?,?,?,?,?,?,?,?,?,?)
1468	163	0	164	
1468	95	0	95	CALL ICMGETATTRTYPE(?,?,?,?,?,?,?,?,?,?,?,?,?)
1468	95	1	95	
1468	549	1	1098	SELECT COMP.ITEMTYPEID, COMPATTR.*, VIEWATTR.ATTRIBUTEFLAG...
1468	31	1	174	SELECT K.KEYWORDNAME, K.KEYWORDDESCRIPTION, A.*, D.COMPON...
1468	549	1	1098	SELECT COMP.ITEMTYPEID, COMPATTR.COMPONENTVIEWID, COMPAT...
1468	31	1	17	SELECT ITVIEWDEFS.ITEMTYPEID, ITVIEWDEFS.ITEMTYPEVIEWID, ITVIE...
1468	7	1	7	SELECT FORKEYS.* FROM ICMADMIN.ICMSTCOMPATTRSFK FORKEYS, ...
1468	13	0	13	SELECT * FROM ICMADMIN.ICMSTITEMTYPEREL WHERE SOURCEITEMT...
1468	0	0	2	
1468	0	0	1	
1468	0	0	1	
1468	0	0	2	
1468	2	1	2	

4,883 of 18,326 items displayed

Figure 11-6 Using db2eva to analyze event data captured in tables

With respect to performance tuning, the most valuable information that can be taken from event monitoring is that of SQL statements as they are processed by the database system. You can capture the statements with performance-related information such as compilation or execution time. This information can then be used to identify critical statements that require further analysis by using one of the analysis tools described in 11.4.3, “DB2 analysis tools” on page 328.

Note: The major difference between snapshot monitoring and event monitoring is that *snapshot monitoring provides status information* and *event monitoring records individual events.*

When using event monitoring, consider these facts:

- Similar to snapshot monitoring, event monitoring consumes system resources and has an impact on general system performance. We recommend switching it on *if and only if* monitoring actually is done. The overhead of event monitors is significantly higher than snapshot monitoring.
- Event monitoring to a file or to a table may be done in a *blocked* or *unblocked* mode. The blocked mode is the default. In this mode DB2 causes the agent to wait for the event buffer to be flushed before it continues. This avoids a situation where DB2 runs out of event buffers and is the only way to guarantee that no event data lost, but depending on the event type that you are monitoring, this might reduce throughput in your system significantly.
- Event monitors can be targeted against a particular DB2 application, which dramatically reduces the overhead.

DB2 diagnostic file: db2diag.log

This file is configured and maintained at the database manager level and reports events during the database manager’s processing. It resides in the database instance home directory. Although it is a readable text file, we recommend using the DB2 command **`db2diag`** to look at it because it conveniently reformats and filters the data. The level of details reported in this file can be configured with the database manager configuration parameter DIAGLEVEL, which comes with a default value of 3. For more detailed information, this value must be increased.

For performance tuning purposes, you might find messages in this file that indicate shortage of system resources (for example, messages that indicate that a certain heap was resized in order to be able to complete some operation).

DB2 CLI and JDBC trace

The DB2 CLI traces client-side facility to record all calls an application makes to the CLI interface. You can explicitly activate it using the following commands with a valid file path for <myFilePath>:

```
UPDATE CLI FGR FOR SECTION COMMON USING Trace 1
UPDATE CLI FGR FOR SECTION COMMON USING TraceFileName <myFilePath>
UPDATE CLI FGR FOR SECTION COMMON USING TraceFlush 1
```

If your application does not provide log options to figure out what it is doing internally, this trace output gives you a detailed report about its interactions with the database including timestamps and performance data.

If your application uses JDBC instead of CLI to connect to the database, use the following commands to activate tracing:

```
UPDATE CLI FGR FOR SECTION COMMON USING JdbcTrace 1
UPDATE CLI FGR FOR SECTION COMMON USING JdbcTraceFileName <myFilePath>
UPDATE CLI FGR FOR SECTION COMMON USING JdbcTraceFlush 1
```

11.4.2 DB2 monitoring tools

In this section, we provide an overview of the various tools that are available for monitoring a DB2 database system. These tools differ in their individual focus and the way they present the data to users.

We discuss the following tools:

- DB2 Memory Visualizer
- DB2 Health Monitor and DB2 Health Center
- db2pd: Monitor and troubleshoot DB2
- DB2 Performance Expert
- DB2 Explain
- DB2 Design Advisor

DB2 Memory Visualizer

DB2 comes with several memory-related performance concepts as outlined in 2.1.3, “Performance-related concepts” on page 39. Although you can retrieve data related to the use of different types of memory by using snapshot monitoring, for tuning purposes it is easier if you can monitor the performance data online and have the usage of the individual memory components graphically visualized while your database actually is hit by the load.

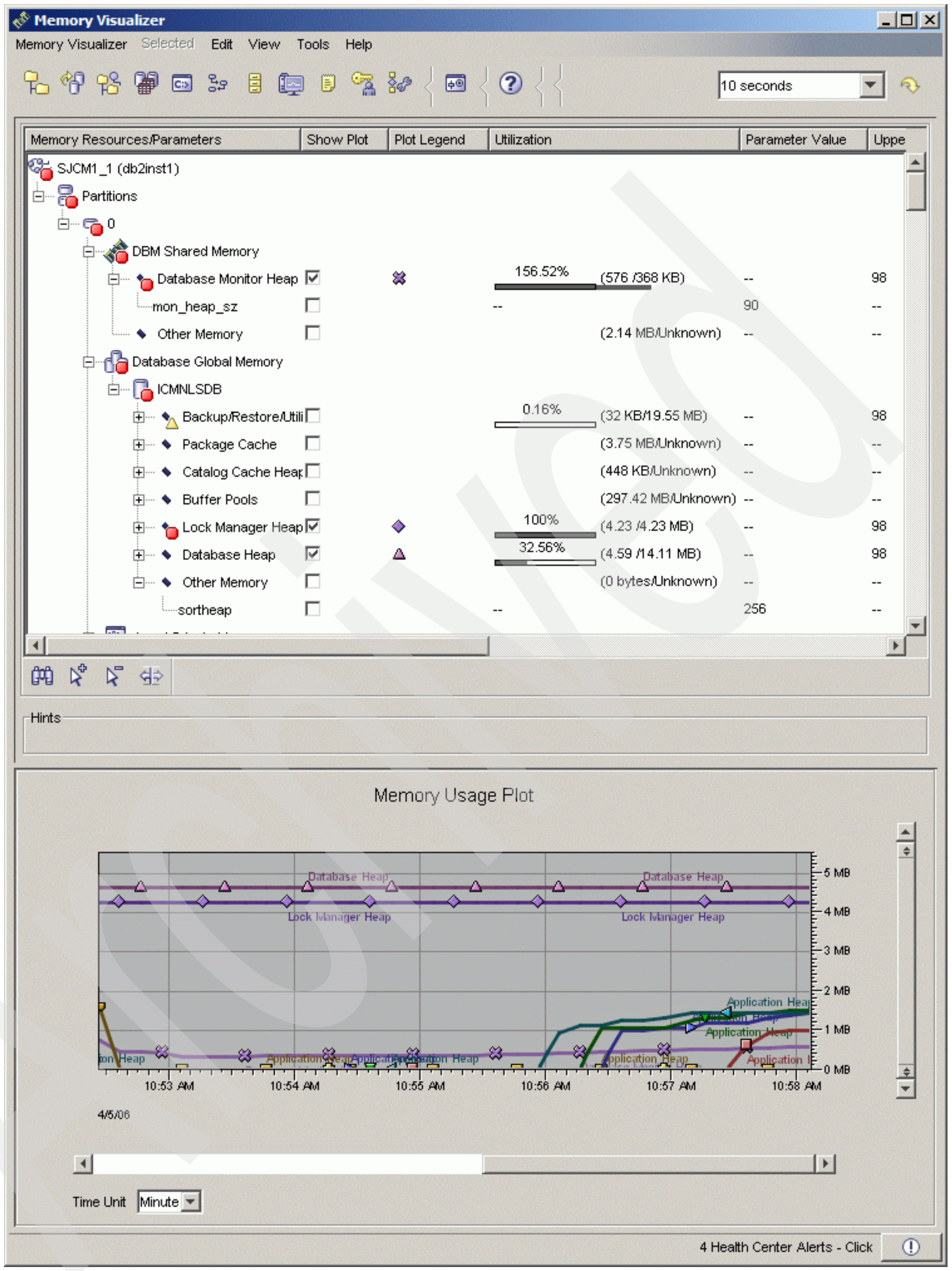
DB2 comes with the *DB2 Memory Visualizer*, a tool that enables you to select interactively among the individual memory components what you want to focus on and graphically displays their use over time. This gives you a convenient option to graphically visualize memory-related performance for a single database instance and all its databases including the connections that are currently active on these databases. This way, you can easily identify and troubleshoot memory-related performance problems in a single database instance. If you need to monitor two different database instances simultaneously, you can open two separate Memory Visualizer windows. You cannot automatically interrelate the data displayed in both windows.

DB2 Memory Visualizer uses configurable thresholds to determine whether a specific type of memory is in a critical state and might require some tuning. This state is visualized and it focuses your attention directly to the hot spots. The tool uses a tree navigation to enable you to browse easily through the individual elements of the DB2 memory.

To open DB2 Memory Visualizer, either type `db2memvis` at the command line or use DB2 Control Center as follows:

1. In the object view panel of the control center, browse the object hierarchy, and select the database instance you want to monitor.
2. Right-click and select **View memory usage** from the context menu.

Figure 11-7 on page 311 visualizes a situation during a heavy load phase of the Content Manager Library Server database. In this figure, red markers highlight the memory elements that might be considered in an alarm state. For example, the Database Monitor Heap is in an alarm state. This is because we had snapshot monitoring switched on during this load. This is not required for visualizing memory, but DB2 Memory Visualizer shows that all Database Monitor Heap is used up by snapshot monitoring at the time.



Memory Resources/Parameters	Show Plot	Plot Legend	Utilization	Parameter Value	Uppe
SJCM1_1 (db2inst1)					
Partitions					
0					
DBM Shared Memory					
Database Monitor Heap	☑	✖	156.52% (576 /368 KB)	--	98
mon_heap_sz	☐		--	90	--
Other Memory	☐		(2.14 MB/Unknown)	--	--
Database Global Memory					
ICMNLSDB					
Backup/Restore/Utili	☐		0.16% (32 KB/19.55 MB)	--	98
Package Cache	☐		(3.75 MB/Unknown)	--	--
Catalog Cache Heap	☐		(448 KB/Unknown)	--	--
Buffer Pools	☐		(297.42 MB/Unknown)	--	--
Lock Manager Heap	☑	◆	100% (4.23 /4.23 MB)	--	98
Database Heap	☑	▲	32.56% (4.59 /14.11 MB)	--	98
Other Memory	☐		(0 bytes/Unknown)	--	--
sortheap	☐		--	256	--

Figure 11-7 Monitoring a DB2 database instance using DB2 Memory Visualizer

DB2 Health Monitor and DB2 Health Center

DB2 Health Monitor is a server-side tool that adds management-by-exception capability by constantly monitoring the health of an instance and its active databases. The health monitor also has the capability to alert a database administrator (DBA) of potential system health issues. The health monitor proactively detects issues that might lead to hardware failure, or to unacceptable system performance or capability. The proactive nature of the health monitor enables users to address an issue before it becomes a problem. In this way, the health monitor is more an operations support tool focusing on performance aspects than a tool that can be used to analyze and drill down a performance when the problem is there.

The health monitor checks the state of your system using health indicators to determine whether an alert should be issued. Preconfigured actions can be taken in response to alerts. The health monitor can also log alerts in the administration notification log and send notifications by e-mail or pager. This management-by-exception model frees valuable DBA resources by generating alerts to potential system health issues without requiring active monitoring.

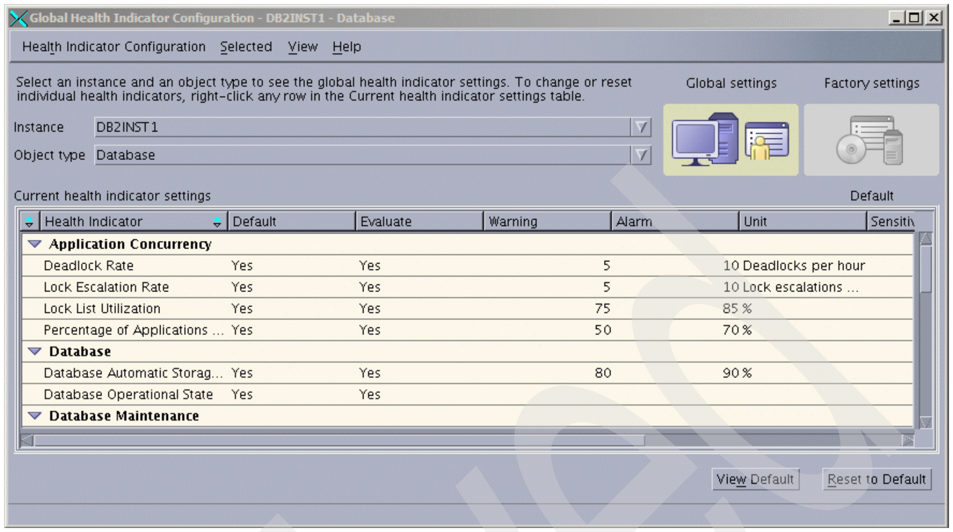
The health monitor gathers information about the health of the system using interfaces that do not impose a performance penalty. It does not turn on any snapshot monitor switches to collect information.

The *DB2 Health Center* is the graphical user interface that can be used to configure the settings of the health monitor as well as to browse the monitored database objects and inspect their health state as well as their history of alert messages.

Health monitor settings can be configured independently for each database instance. If required, thresholds can be adapted to indicate the health state of database objects or to trigger alarm notifications. With health monitor, you can monitor the state of the database manager as well as that of databases, table spaces, and table space containers.

To open DB2 Health Center, either type `db2hc` at the command line or use DB2 Control Center and select **Tools** → **Health Center**.

Figure 11-8 on page 313 illustrates how to configure a health monitor of a database using DB2 Health Center.



Global Health Indicator Configuration - DB2INST1 - Database
Health Indicator Configuration Selected View Help
Global settings
Factory settings
View Default
Reset to Default

Select an instance and an object type to see the global health indicator settings. To change or reset individual health indicators, right-click any row in the Current health indicator settings table.

Instance DB2INST1

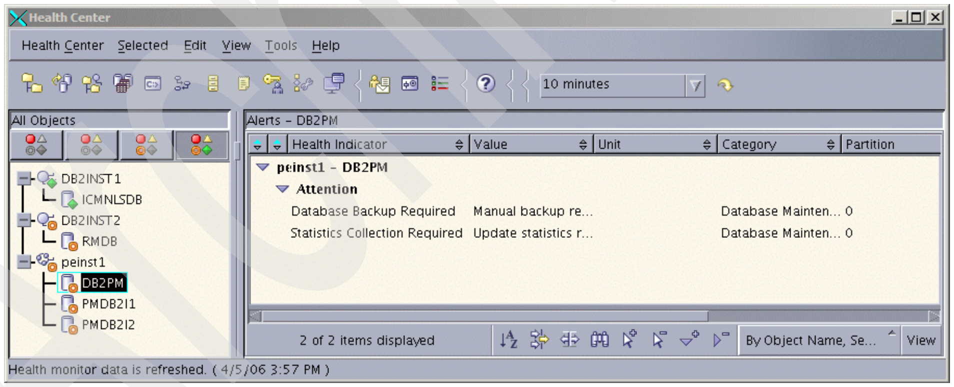
Object type Database

Current health indicator settings — Default

Health Indicator	Default	Evaluate	Warning	Alarm	Unit	Sensitiv
Application Concurrency						
Deadlock Rate	Yes	Yes	5	10	Deadlocks per hour	
Lock Escalation Rate	Yes	Yes	5	10	Lock escalations ...	
Lock List Utilization	Yes	Yes	75	85	%	
Percentage of Applications ...	Yes	Yes	50	70	%	
Database						
Database Automatic Storag...	Yes	Yes	80	90	%	
Database Operational State	Yes	Yes				
Database Maintenance						

Figure 11-8 Configuring DB2 Health Monitor using DB2 Health Center

Figure 11-9 demonstrates how to monitor database instance health and alerts using DB2 Health Center.



Health Indicator	Value	Unit	Category	Partition
peinst1 - DB2PM				
Attention				
Database Backup Required	Manual backup re...		Database Mainten...	0
Statistics Collection Required	Update statistics r...		Database Mainten...	0

Health monitor data is refreshed. (4/5/06 3:57 PM)

Figure 11-9 Monitoring health using DB2 Health Center

db2pd: Monitor and troubleshoot DB2

`db2pd` is more for troubleshooting than monitoring. We nevertheless include this tool in our discussion because it may be used to further analyze and drill down any problem that shows up in the monitoring tool. **`db2pd`** is a pure command line tool that enables you to inspect the various database objects, even internal objects that you otherwise do not see. You can inspect objects such as:

- Various shared operating system resources such as shared memory, message queues, and semaphores, depending on the underlying OS
- Details of applications currently connected to the database and their transactions
- Locks on database objects held by applications
- Table space and buffer pool allocation and usage
- Table space container details
- SQL statements held in the statement cache and their usage
- Packages loaded in memory
- Memory pools and their usage

Each object may be queried individually using specific command line parameters. For details about the command syntax, refer to *DB2 UDB Command Reference V8.2*, SC09-4828. As a starting point to work with, we recommend using the following command, which gives you probably the most detailed and comprehensive level of information for a database:

```
db2pd -database <myDatabase>
```

In this syntax, <myDatabase> stands for the name of the database that you want to analyze. You can add parameters such as `-repeat <secsToWait> <repeatCount>` to this command to have it repeated every <secsToWait> seconds for <repeatCount> repeats. With this repeat mode, you can monitor the behavior of your database system while it is under load.

The output of this command is lengthy. Example 11-7 displays a subset of the messages produced by this command.

Example 11-7 Output of command db2pd

```
Instance db2inst1 uses 32 bits and DB2 code release SQL08022
with level identifier 03030106
Informational tokens are DB2 v8.1.1.89, OD_14086, U800789_14086, FixPak 9.

Operating System Information:
<... information about CPU, memory, share ressources (queues, memory, semaphores) and
load ...>

Database Partition 0 -- Database ICMNLSDB -- Active -- Up 0 days 00:00:59

Applications:
```

```
Address    AppHandl [nod-index] NumAgents  CoorPid    Status <...>
0x30173F60 518      [000-00518] 1          1204322    UOW-Waiting <...>
0x30173DC0 519      [000-00519] 1          1298522    UOW-Waiting <...>

Transactions:
Address    AppHandl [nod-index] TranHdl    Locks      State   Tflag <...>
0x40275580 503      [000-00503] 2          0          READ    0x00000000 <...>
0x40276000 504      [000-00504] 3          0        READ    0x00000000 <...>
<...>

BufferPools:
First Active Pool ID      1
Max Bufferpool ID         5
Max Bufferpool ID on Disk 5
Num Bufferpools           9

Address    Id         Name              PageSz     PA-NumPgs  BA-NumPgs <...>
0x402DA590 1          IBMDEFAULTBP      4096       1000       0 <...>
0x402DA850 2          ICMLSFREQBP4      4096       1000       0  <...>
<...>

Logs:
<...>

Database Partition 0 -- Database ICMNLSDB -- Active -- Up 0 days 00:00:59

Locks:
Address    TranHdl    Lockname                   Type       Mode Sts <...>
0x4032D960 17         535953534832303028EFECDC41 Internal P ..S  G <...>
0x4032C128 17         49434D504C53524424CBB4D241 Internal P ..S  G <...>
<...>

Tablespaces:
Address    Id    Type Content AS  AR  PageSize   ExtentSize Auto <...>
0x455B92A0 0     SMS  Any     No  No  4096       32         Yes <...>
0x407540B0 1     SMS  SysTmp  No  No  4096       32         Yes <...>

Containers:
Address    TspId ContainNum Type <...>
0x455B98F0 0     0          Path <...> /dbdata/<...>/SQLT0000.0
0x455BBA70 1     0          Path <...> /dbdata/<...>/SQLT0001.0
<...>

Dynamic Cache:
Current Memory Used           1275469
```

```
Total Heap Size               6356992
Cache Overflow Flag           0
Number of References          2120
Number of Statement Inserts   86
Number of Statement Deletes   0
Number of Variation Inserts   85
Number of Statements          86

Dynamic SQL Statements:
Address    AnchID StmtUID NumEnv NumVar NumRef NumExe Text
0x456A3600 5      1       1      1      16     32     CALL ICMGETATTRTYPE(<..>)
0x45746B70 26     1       1      1      13     13     SELECT
                                                        LINKS.SOURCEITEMID,<..>

Dynamic SQL Environments:
Address    AnchID StmtUID    EnvID Iso QOpt Blk
0x456A36D0 5      1          1     CS  2    B
0x4574DB30 6      1          1     CS  2    B
<...>

Dynamic SQL Variations:
Address    AnchID StmtUID EnvID VarID NumRef Typ Lockname
0x456A3890 5     1       1      1     16     13  000000010000000100010<...>

Static Cache:
Current Memory Used           1275469
Total Heap Size               6356992
Cache Overflow Flag           0
Number of References          2398
Number of Package Inserts     29
Number of Section Inserts     121

Packages:
Address    Schema   PkgName  Version  UniqueID NumSec UseCount NumRef Iso <...>
0x4574F020 ICMADMIN ICMPLSGD          8A72AWIV 2      0        80     CS  <...>
0x457D41C0 ICMADMIN ICMPLSCB          eAK1AWIV 0      0        25     CS  <...>
<...>

Sections:
Address    Schema   PkgName  UniqueID SecNo NumRef UseCount StmtType Cursor<..>
0x4574F250 ICMADMIN ICMPLSGD 8A72AWIV 1     80     0        6        NO   <...>

Memory Pools:
Address    MemSet   PoolName Id Overhead LogSz LogUpBnd  LogHWM PhySz  <...>
0x40000A84 ICMNLSDB utilh    5  0        3496  20496384  3496   16384  <...>
```

```
0x400009E8 ICMNLSDB undefh   59 0        0     12345678  0      0      <...>
0x4000094C ICMNLSDB pckcacheh  7 0       12769 Unlimited 12769  13120  <...>
<...>

Memory Sets:
Name   Address    Id Size Key DBP Type Ov OvSize
App518 0x00000000 0 0     0x0 0    0   N  0
App519 0x00000000 0 0     0x0 0    0   N  0
App520 0x00000000 0 0     0x0 0    0   N  0
<...>

Database Configuration Settings:
Description                    Memory Value         Disk Value
DB configuration release level 0xa00                0xa00
Database release level         0xa00                0xa00
Database territory             US                   US
Database code page             819                  819
Database code set              ISO8859-1            ISO8859-1
Database country/region code   1                    1
<...>
DBHEAP (4KB)                   2400                 2400
DATABASE_MEMORY (4KB)          40312                40312
CATALOGCACHE_SZ (4KB)          800                  800
<...>

Catalog Cache:
Configured Size        3276800
Current Size           616780
Maximum Size           4294950912
High Water Mark        655360

SYSTABLES:
Address    Schema   Name               Type TableID TbspaceID LastRefID  <...>
0x45769310 ICMADMIN ICMUT01009001      T    27      3         1441812    <...>
<...>

Table Reorg Stats:
Address   TbspaceID TableID   TableName       Start              End
PhaseStart               MaxPhase   Phase     CurCount   MaxCount   Type    Status
Completion IndexID    TempSpaceID
<...>
```

The output of the **db2pd** command provides information about many internal details. We do not discuss all of them here. Instead, we highlight certain pieces of information that you should look at, or that you otherwise cannot retrieve using the tools that we discuss in this chapter:

- When looking at the application details, you might find applications in a lock-wait state. In a situation where you suspect performance problems due to concurrency problems in your applications, you can determine which locks are held by each of the involved applications and which SQL statements these applications are currently working on.
- This command lists the active statements. If you observe a long-running statement in your database that consumes all of your system resources, you can use this command to identify this statement and perform further analysis. You can identify the corresponding application to force it off the database using the DB2 command **`FORCE APPLICATION(<applId>)`**.

To read and successfully analyze the output of the **`db2pd`** command, you should have a solid understanding of the DB2 architecture and memory model.

DB2 Performance Expert

The monitoring tools that we discussed so far focus on a single database instance or on a single database. As we know from 1.2, "Content Manager system components" on page 4, a Content Manager environment comes with not just one database but with two databases, the Library Server database and the Resource Manager database. It might be placed on different hardware. In this situation, the standard monitoring tools that come with DB2 cannot provide you an integrated view on both databases.

This is one main aspect where DB2 Performance Expert (PE) significantly adds value in a Content Manager environment and differs from the other monitoring tools we discussed so far. With this tool, you can monitor multiple databases simultaneously in a distributed environment.

DB2 Performance Expert architecture overview

Figure 11-10 on page 319 shows how Performance Expert integrates into the Content Manager environment.

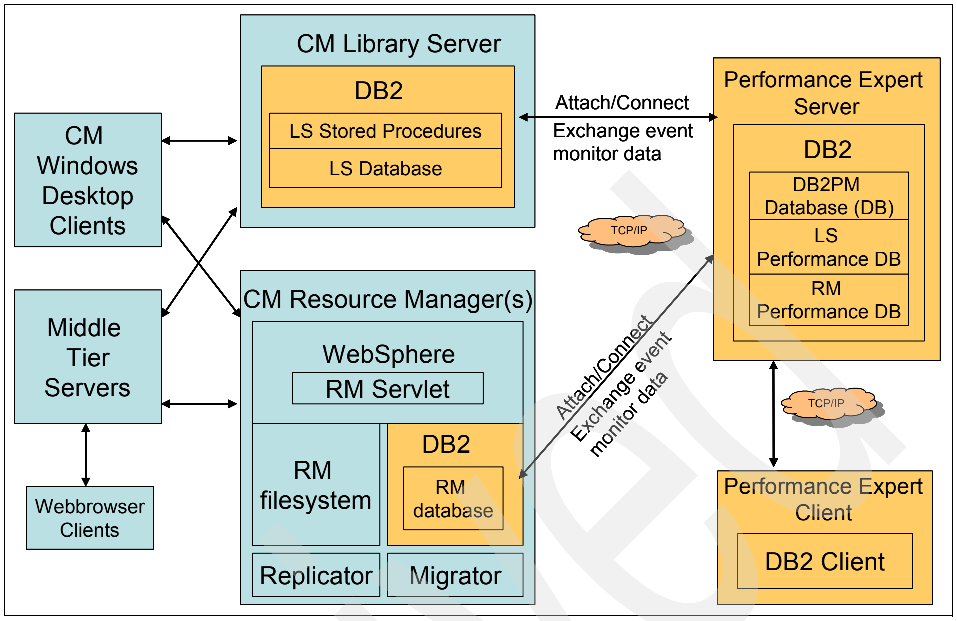


Figure 11-10 DB2 Performance Expert architecture overview

A typical DB2 Performance Expert (PE) setup in a Content Manager environment is made up of two individual components:

- The *DB2 Performance Expert Server* connects to all monitored databases and instances through standard DB2 interfaces. No agent component is involved in collecting the data. For each monitored database instance, the Performance Expert Server internally creates and maintains a separate performance database to store the performance-related data of this instance and all databases in this instance.

  Note: Although no agent is required to collect data for Performance Expert, it includes a component, DB2 Performance Expert Agent. This is used for a different purpose and we do not include it in our discussion.

- The *DB2 Performance Expert Client* provides a GUI to view the performance data that is collected by the Performance Expert Server. Use the Performance Expert Client to configure the Performance Expert Server and to browse and analyze data collected by the Performance Expert Server.

There are certain aspects of the Performance Expert architecture and concepts that you should know when using Performance Expert for monitoring your Content Manager environment:

- The Performance Expert Client does not communicate directly with the monitored databases; it only communicates with the Performance Expert Server. This keeps setup of your Performance Expert Client environment simple because it does not require setting up connections to all monitored databases.
- The Performance Expert Server uses DB2 Snapshot and Event Monitors to collect DB2 performance data for online monitoring, short-term history, long-term history, and exception processing.
- When activated, Performance Expert-monitoring of a database has impact on the performance and general behavior of the system. To reduce overhead on the monitored DB2 instance, the Performance Expert Server uses snapshot instead of event monitoring whenever possible. However, as with any snapshot or event monitoring, you should have Performance Expert collect only the data that you really need for your purposes.
- In addition to monitoring DB2-related performance data, the Performance Expert Server also supports collecting data at operating system level (for AIX, Solaris, and Linux platforms). This is done using the Common Information Model Object Manager (CIMOM), an object management engine that resides between the managed system and the management application.

  The Common Information Model (CIM) is a standard model developed by the Distributed Management Task Force (DMTF) for representing managed elements (such as computers) in an IT environment and relationships between them.

The scope of this book does not enable us to go into all details about DB2 Performance Expert. Refer to *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470, which contains specific sections about monitoring a Content Manager environment and gives detailed guidance for using Performance Expert in this context. In addition, we recommend the technical white paper *Monitoring IBM DB2 Content Manager V8.3 Enterprise Edition with DB2 Performance Expert for MP V2.2*, which you can download at:

`ftp://ftp.software.ibm.com/software/data/db2imstools/whitepapers/wp-cmpev22.pdf`

Note: Although DB2 Performance Expert is a member of the DB2 product family, it does not ship with DB2 UDB. DB2 Performance Expert is a separate product that must be acquired and licensed.

Using PE to monitor Content Manager databases

In a Content Manager environment, with DB2 Performance Expert, you can view your Library Server and Resource Manager databases from a single user interface, the Performance Expert Client, and monitor applications, system statistics, system parameters, and the operating system. This can be done in real time and in historical mode. In addition to collecting performance metrics based on snapshot and event data, Performance Expert raises exceptions to anticipate emerging DB2 performance and availability problems and enables you to generate reports that show buffer pool, database, or SQL activity over time in order to identify trends or workload peak times.

The value and benefit of DB2 Performance Expert for monitoring a Content Manager environment can be summarized as follows:

- Instead of various monitoring tools that have to be used individually for each database and instance, Performance Expert gives you a single interface to all database-related monitoring aspects.
- In addition to online monitoring, you can store performance-related data and run historical analysis, identify trends, and apply corrective steps early if required.
- You can even integrate monitoring at operating system level (for AIX, Solaris, and Linux platforms).
- The Performance Expert Client comes with a rich set of predefined reports and graphical data views for sophisticated analysis. You can individually modify them or develop your own reports and views.
- Some of the visualized reports, also known as *System Health Graphs*, are designed to show how critical performance counters change over time. You can use this information to verify that your Library Server or Resource Managers are running healthy or to detect performance problems immediately.
- The user interface of the Performance Expert Client can be adapted to the preferred reports and views for which you need immediate and direct access.
- Monitoring by exception: The Performance Expert Server can periodically compare the warning and problem thresholds that are set on DB2 snapshot values or calculated values out of snapshot values against the actual snapshot values. Exceptions are raised on the Performance Expert Client if a threshold is exceeded. Thresholds on different snapshot counters can be combined in threshold sets. Performance Expert offers predefined threshold sets that can be applied as is or adapted to your own needs.

  You can use Performance Expert as an active health monitoring system that incorporates much of the functionality of DB2 Health Monitor, but with all the databases in your environment in an integrated and homogeneous fashion.

- The Performance Expert Client provides cross-functionality with other DB2 tools. If you identify a troubled query in Performance Expert, you can directly launch DB2 Visual Explain (described in more detail in "DB2 Explain" on page 328) from the Performance Expert Client so that you have a direct and quick access to the queries access plan.

In addition to these general features that you can use in any database environment, Performance Expert provides specific support for Content Manager environments:

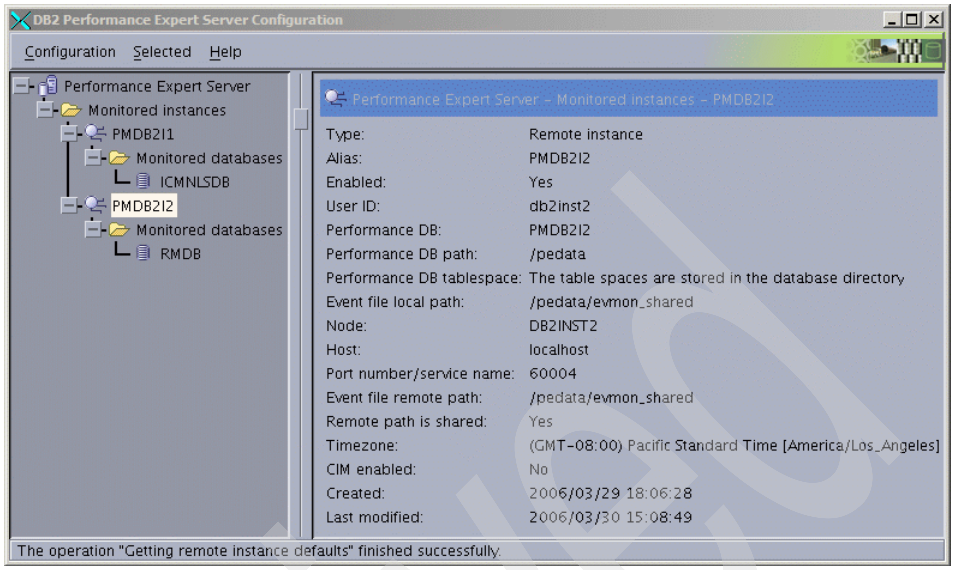
- Since DB2 Performance Expert V2.2. FP1, some Content Manager specific performance visualizations are added.
- As of the same version, Performance Expert comes with a predefined set of Content Manager-specific thresholds that you can use to monitor as well as for exception processing.

Setting up PE Server to monitor a Content Manager environment

To start monitoring your Content Manager environment with Performance Expert, configure the Performance Expert Server so that it knows about your Content Manager databases. At this point, we assume that you followed the install instructions for Performance Expert so that you (as user *root*) on your system processed the following steps:

- You created (or choose an existing) database instance where you want Performance Expert to put all of its databases.
- You enabled this database instance to act as a Performance Expert Server (by using the command **`pecentralize.sh`**).

The following steps are required to link your Performance Expert Server to your Content Manager databases. These steps must be processed as owner of the Performance Expert Server database instance using the command **`peconfig`**, which can be run in a GUI mode or in a command line mode (for example, if you do not have a graphical user interface available in your production system). Figure 11-11 on page 323 demonstrates use of the graphical interface. The command line mode essentially follows the same interaction principle, but it interactively asks you for the required instance and database information.


DB2 Performance Expert Server Configuration
Configuration
Selected
Help
Performance Expert Server
Monitored instances
PMDB2I1
Monitored databases
ICMNLSDB
PMDB2I2
Monitored databases
RMDB
Performance Expert Server - Monitored instances - PMDB2I2

Type:	Remote instance
Alias:	PMDB2I2
Enabled:	Yes
User ID:	db2inst2
Performance DB:	PMDB2I2
Performance DB path:	/pedata
Performance DB tablespace:	The table spaces are stored in the database directory
Event file local path:	/pedata/evmon_shared
Node:	DB2INST2
Host:	localhost
Port number/service name:	60004
Event file remote path:	/pedata/evmon_shared
Remote path is shared:	Yes
Timezone:	(GMT-08:00) Pacific Standard Time [America/Los_Angeles]
CIM enabled:	No
Created:	2006/03/29 18:06:28
Last modified:	2006/03/30 15:08:49

The operation "Getting remote instance defaults" finished successfully.

Figure 11-11 Configuring PE Server to monitor your Content Manager databases

To link your Performance Expert Server to your Content Manager databases:

1. From the left navigation panel, right-click **Monitored instances** to add a new instance.
2. This opens a multi-step wizard that asks for the connection information displayed in the right panel. (For details, refer to *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470.)
3. After you have successfully configured your database instances, right-click **Monitored Databases** under the database instance you created. You can choose to add either all of the databases or a specific one to the Performance Expert Server configuration.
4. For each instance you add to the Performance Expert Server configuration, you can individually enable or disable Performance Expert monitoring by right-clicking the instance and select **Disable Instance** or **Enable Instance**.

You must start the Performance Expert Server to collect monitoring data and before any Performance Expert Client can connect to the server.

- To start the Performance Expert Server, enter the command **`pestart`** as owner of the Performance Expert Server's database instance.
- To stop the Performance Expert Server, enter the command **`pestop`** as owner of the Performance Expert Server's database instance.

Monitoring a Content Manager environment under load: an example

When you have configured and started your Performance Expert Server to monitor your Content Manager environment, use the Performance Expert Client to look at the monitoring data.

To open the Performance Expert Client window:

1. Enter the command `db2pe` at command line.
2. If this is the first time, the Performance Expert Client asks whether you want classic design or extended design. Select one of your choice.
3. Before you can look at monitoring data collected by a Performance Expert Server, you have to configure the connection to this Performance Expert Server. Select **Monitor** → **New DB2 System**.
4. Repeat this step if you want to have your Performance Expert Client simultaneously look at multiple Performance Expert Servers.

Performance Expert comes with a set of predefined monitoring view and threshold settings specifically developed for a Content Manager environment. However, you may always define your own views on the data and specify which counters you want to have displayed in your view. For details, refer to *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470.

Figure 11-12 on page 325 displays a snapshot of our Performance Expert Client that was taken while the Performance Expert Server collected data from our Content Manager databases during some Content Manager load process. In this figure, we display data for only the Library Server database.

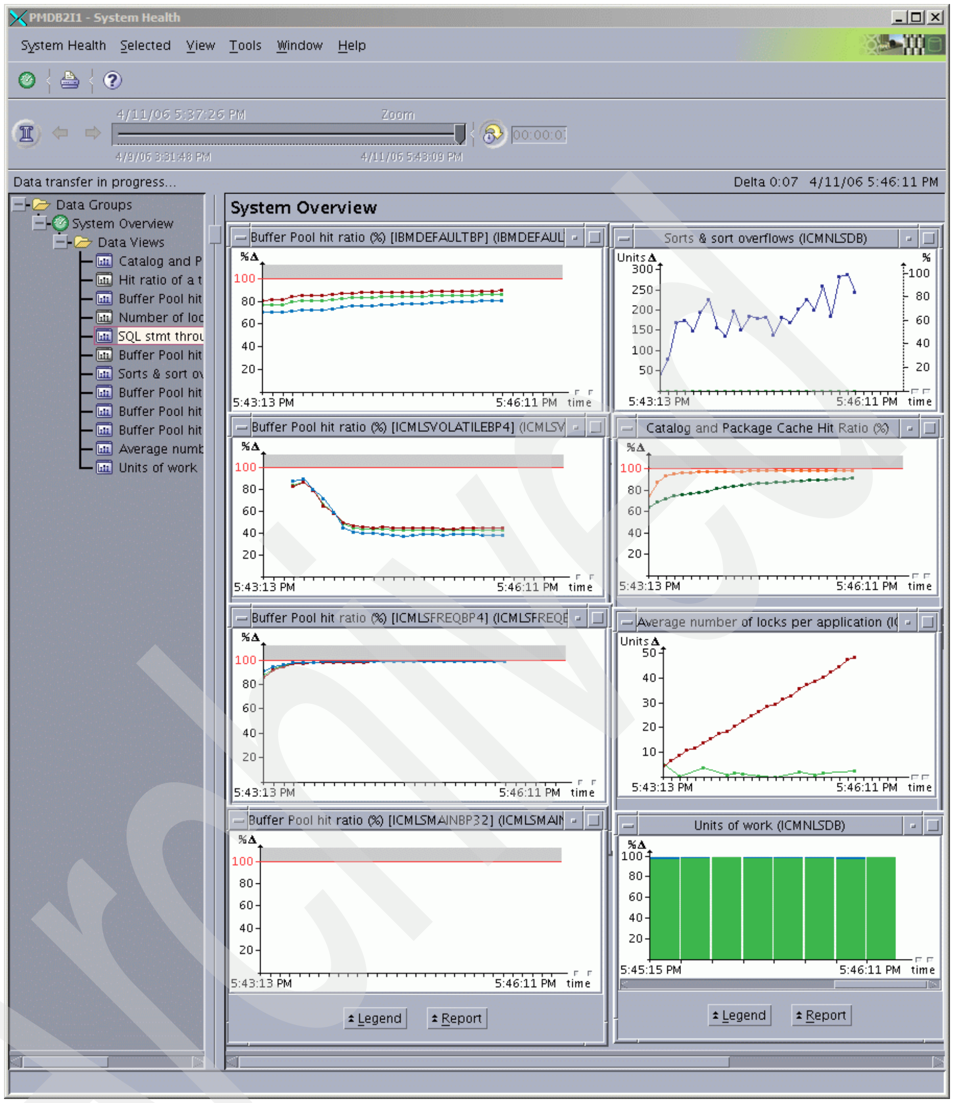


Figure 11-12 Example of monitoring a Content Manager environment under load

In the System Overview panel of the Performance Expert Client, we placed eight graphical views on individual monitoring counters. They are:

- Buffer pool hit ratio for the each of the buffer pools (IBMDEFAULTBP, ICMLSVOLATILEBP4, ICMLSFREQBP4, ICMLSMAINBP32). They are the left four graphical views from top to bottom. The graph displays index ratio for data and index separately as well as the accumulated hit ratio.

- Number of sort operations in the database as well as number of sort overflows. This is the top-right view.
- Catalog and package cache hit ratio. This is the second right-side view.
- Number of active database connections and average number of locks per connection. This is the third right-side view.
- Relative number of transactions, distinguishing by “committed transactions” versus “transactions rolled back.” This is the bottom-right view.

The snapshot displays the startup phase of our Content Manager load generator simulating a maximum number of 100 users.

The information that immediately can be taken from the visualization of monitoring data is:

- Buffer pool hit ratio for all buffer pools stabilizes at a point in time when about 30 simulated users were connected.
- Buffer pool ICMLSMAINBP32 is never hit by our Content Manager load application.
- The buffer pool ICMLSFREQBP4 hit ratio climbs up very soon to close to 100%. This buffer pool does not need any tuning.
- The buffer pool IBMDEFAULTBP hit ratio slowly climbs up to about 80% and does not tend to significantly increase any more. This is slightly below the recommended threshold value of 85%, so we should moderately increase the size of this buffer pool.
- The buffer pool ICMLSVOLATILEBP4 hit ratio soon stabilizes at a level slightly below 50%. This is much below the threshold value, indicating that this buffer pool might need to be increased by some factor.
- Number of sort operations slightly increases over time as more users get connected. No sort overflows are reported so that sort heap seems to have an adequate size. No indication for any tuning can be taken from this view.
- Catalog and package cache hit ratio constantly climbs up to 90% and more. This does not indicate any need for tuning.
- The average number of locks held by each application stays between 1 and 5 and is about constant. No need for tuning can be derived from this.
- About 100% of all transactions are successfully committed. No indication can be taken from this that locking is a problem here.

As can be seen from the “Number of active database connections” view, the snapshot was taken at a time when about 50 of the 100 simulated users were connected to Content Manager.

Figure 11-13 shows a snapshot of the same Content Manager environment hit by the same load after increasing the size of buffer pool ICMLSVOLATILEBP4 by a factor of 5.

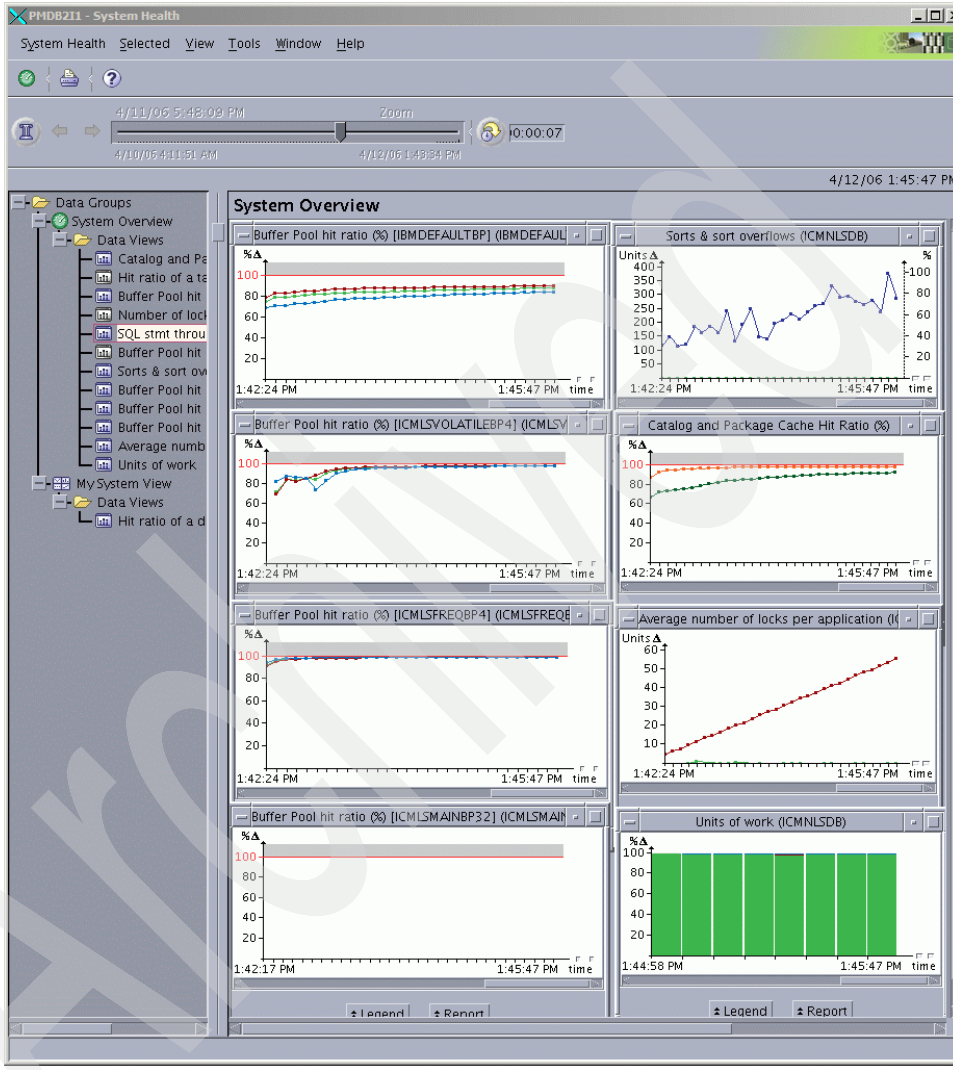


Figure 11-13 Monitoring the Content Manager environment after buffer pool tuning

As expected, hit ratio of the buffer pool in question significantly improved and is now close to 100%. In addition, average number of locks simultaneously held by applications decreased and is almost constantly below 1. This is likely a side effect of increased buffer pool hit ratio since this means reduced disk I/O and avoidance of I/O wait when an application accesses data. This in turn leads to an increase of transaction throughput and a reduction of transaction length (measured in time units). Thus, locks are not held as long as before.

In this example, we just monitored the Library Server database. Without going into further details, we note that Performance Expert enables you to open windows that provide graphical views to simultaneously monitor the Library Server database and the Resource Manager database.

Recommendations for using PE to monitor CM databases

DB2 Performance Expert Server is a sophisticated monitoring and analysis application that is built on DB2. This means that it adds a certain load to the server machine.

When using it, we recommend avoiding resource contention and keeping monitoring completely separated from your Content Manager server workload. You should install the Performance Expert server on a different machine than the Library Server and the Resource Manager.

11.4.3 DB2 analysis tools

In 11.4.2, “DB2 monitoring tools” on page 309, we give an overview of the tools that you can use to monitor a database environment in order to identify a problem if there is any. You might not be able to retrieve sufficient information from the monitoring tool to solve some problems. In some cases, you might need to perform further analysis at the operating system level (for example, if it comes to the impact of low buffer pool hit ratio on disk I/O) and in other cases, you may need to use DB2 analysis tools. This is typically the case when you figure out that execution of a specific SQL statement is very slow and consumes unreasonably high CPU or I/O resources in your database system. In the DB2 Explain section, we describe recommended steps toward identifying the cause of the problem and helping you find a way to solve it.

We discuss two DB2 analysis tools here:

- DB2 Explain
- DB2 Design Advisor

DB2 Explain

As described in 2.1.3, “Performance-related concepts” on page 39, DB2 involves the SQL query optimizer to evaluate and compare the cost of access plans to finally choose the access plan that appears to be most efficient. Nevertheless, even when using this best access plan, the query execution might turn out to be very expensive. This is typically observed by the user as long response time or by the database administrator as a high load on disks and network. Using the monitoring tools described earlier in this chapter, you will be able to identify this query but you will not be pointed exactly to the reason why this statement turns out to be so expensive. DB2 provides an EXPLAIN facility that enables you to analyze the access plan that the optimizer chooses. It provides detailed information about this access plan and it is likely that it directly points you to the cause of the costs. In general, there are two options for tuning the query:

- One is that the analysis recommends reformulating the SQL query, possibly because the developer used a semantically incorrect statement. (True, it is not unusual to figure out that a query is semantically incorrect by analyzing its access plan.)
- The other option is that the analysis shows that some performance increasing database objects such as indexes should be added.

There might be the possibility that the analysis shows that the query in its very nature is logically complex and cannot be improved that easily. In this case, you might need to restructure your application or data model, or introduce redundancy in the data.

In order to use DB2 Explain, it is important to understand a few concepts of how cost estimation of access plans is done. The cost model of the DB2 query optimizer mainly uses two factors that contribute to the cost of query execution. One is the disk I/O required to access the data in the tables and the other is CPU to process this data. Other factors such as communication costs in a partitioned environment also come into play. When evaluating the total cost of access plans, DB2 uses a virtual measurement known as *timeron* that accumulates both cost factors. This cannot be converted in response times or number of megabytes read from disk, but it reasonably reflects the total resource consumption so that DB2 uses this value as the decision criterion for the selection of the best access plan. DB2 makes estimations based on the characteristics of the disks, CPU, and most important, the population and distribution of data in tables, and the database catalog statistics.

The second concept that you should know about is the basic database operators that form an access plan such as join, union, sort, table scan, and index scan. For many of these operators, DB2 enumerates through many alternatives and combinations of different access plans. Joins, for example, can be of different varieties such as “nested loop join” and “merge scan join.” These operators take one or more streams or tables as input and produce an output stream. Some operators such as union might take more than two operands.

Based on these concepts, an access plan can essentially be viewed as a tree where certain database objects such as tables and indexes form the leaves of the tree and operators form the inner nodes.

During normal database operation, DB2 keeps access plans as a purely internal concept and does not display them to a user. But for analysis purposes, it provides a user command that explicitly displays an access plan to the user. This command comes in both a command line version and a version with a graphical user interface.

Both place a representation of the access plan in a set of explain-tables in your database. (Actually there is a third, slightly different variant of the DB2 explain functionality that comes with the command **db2expln**. It explains the access plans in static SQL precompiled packages. Refer to *DB2 UDB Command Reference V8.2*, SC09-4828 for more information.)

For the command line version of DB2 Explain, these tables must exist before you use DB2 Explain. (You can find the DDL statements for these tables in a file EXPLAIN.DDL. On UNIX systems, the file resides in the sqllib/misc subdirectory in the database instance owners' home directory. In a Windows environment, this subdirectory should be found in the directory that the DB2PATH environment variable points to. This file contains help that describes how to use it.)

The GUI version creates these tables implicitly if they do not exist. Although it is important to know that these tables should be there, consider them as a pure internal concept because DB2 provides other, more readable and understandable interfaces to look at access plans.

Note: Before you start to analyze an access plan, be sure that the database catalog statistics are up to date (as part of your regular maintenance), at least for the tables referenced by the query. Otherwise, DB2 might choose the wrong access plan and the DB2 cost estimation might not match reality.

DB2 Visual Explain: the GUI version of DB2 Explain

The graphical user interface of DB2 Explain is integrated closely into the DB2 Control Center. It is also known as *DB2 Visual Explain*. In order to run it, proceed as follows:

1. In the Object View of the DB2 Control Center, locate the database you want to work with.
2. Right-click on this database and select **Explain SQL**. This opens the DB2 Visual Explain window.

You can either type the SQL statement into the window or have the statement imported from a file. The tool then opens two independent windows as shown in Figure 11-14 on page 331: one overview window displaying the full access plan as a tree and one window displaying the details of the elements of the access plan. The overview window is useful especially when you navigate in complex access plans that do not fit on a single detail window on your screen. You can drag the smaller detail box within the overview window. It is the content of this detail box that appears in the Visual Explain window.

You can configure the user interface, such as the level of detail that is displayed with each individual node. In Figure 11-14, the graph for each node includes the costs for the sub-tree of the access plan that this node spans.

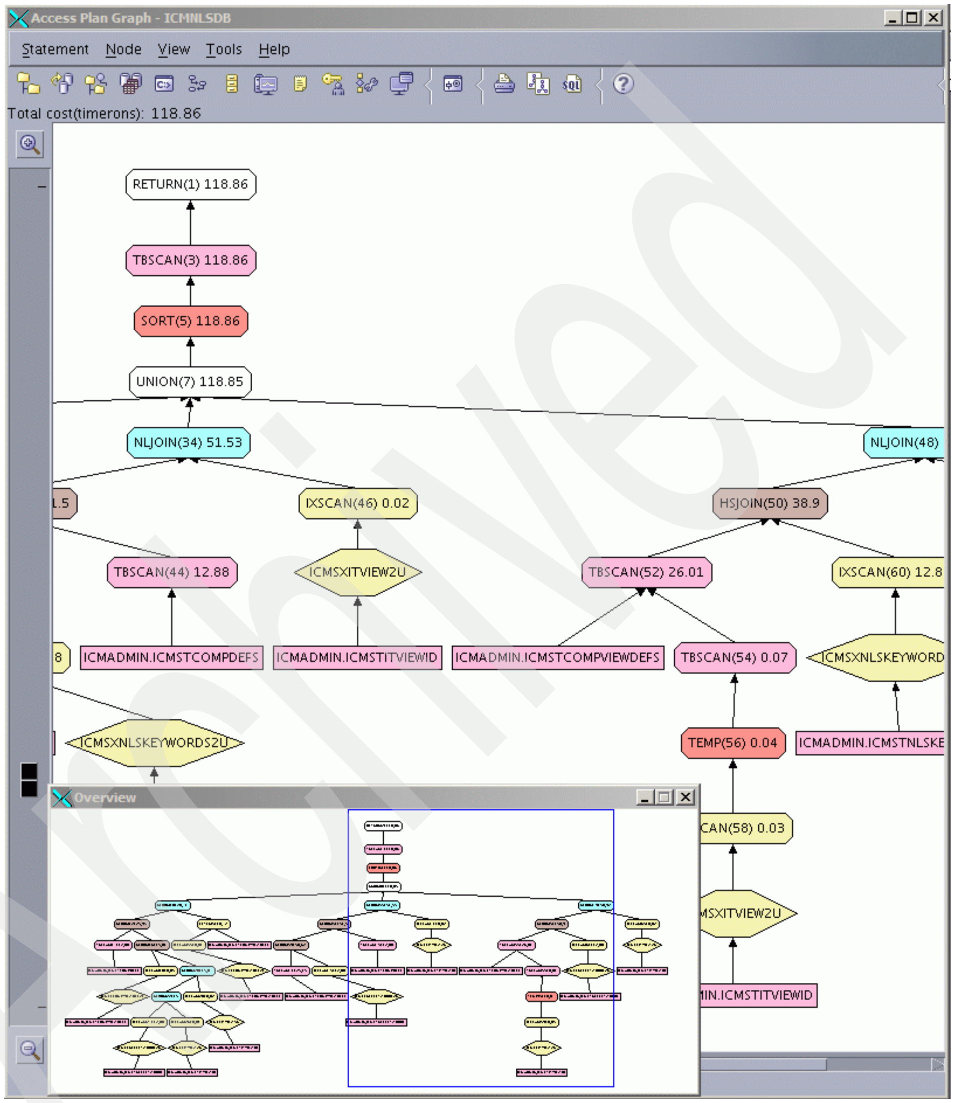


Figure 11-14 Using DB2 Visual Explain to analyze an access plan

In this tree representation of the access plan, the tree's root is displayed at the top, representing the full query. Sub-trees represent sub-queries that contribute to the queries result. Execution of this access plan is performed bottom-up, starting with the leaves representing tables and indexes.

Although navigating in this graph is quite straightforward, it might require some experience to find the things that you are looking for, namely the reason why a query is so expensive. In general, analysis of an access plan with DB2 Visual Explain for performance tuning focuses on searching for individual nodes in the access plan that are critical because they contribute significantly to the costs of the access plan and thus provide the biggest potential for optimization.

Here are a few general rules to guide you in your analysis of the access plan:

- You should *not* take the coloring schema for nodes in the tree that DB2 Visual Explain applies as a strict rule for differentiating critical and uncritical nodes in the tree. A dark red node such as a SORT-operator node could be something you might want to look at; but in many cases, an isolated SORT is absolutely fine, especially if this SORT operator is close to the root (at the top) of the access plan or if the SORT is done on a small data volume.

  Similarly, a TBSCAN (table scan) can be a critical node you might want to look at; but it always depends on what is done with the result of the table scan. If the table scan forms just the result of the query as shown at the top level in Figure 11-14 on page 331, this is fine. However, if two table scans on large input streams feed an NLJOIN (nested loop join), you definitely want to look at this.

  There is no absolute rule that says that a certain operator type in an access plan is the critical point that provides the key to optimization. You always need to look at the details of why and what.

- A certain rule has been proven to be very successful in identifying critical nodes in the access plan: "If the cost of an inner node in the access plan is significantly higher than (a good guess is two times as high as) the sum of the cost of the child nodes, this node very likely focuses on a hot spot in the access plan." This does not necessarily mean that this node itself is the critical point; one of the child nodes might be causing a problem. For example, if you see an NLJOIN that appears to be critical in this sense, it might be because there was no usable index on the operands that allowed the optimizer to use a more efficient join. This way, the problem is caused by the sub-trees underneath this node.

- Certain nodes in the access plan might be examined carefully when they appear deep inside the access plan because of their impact on the usability of indexes even when such indexes exist on database tables. For example, a UNION operator that takes multiple operands as input produces a single output table. No matter whether there are any indexes on the tables below the UNION operator, the optimizer will not be able to make use of these indexes above the UNION operator.

Although analyzing a given problem SQL query and identifying its hot spots is a rather individual and query-specific procedure, with DB2 Visual Explain, you very quickly receive a good and visual understanding of where the problem is.

For completeness, here are some scenarios that from our experience are typical for queries with unacceptable performance. Some of them can be avoided easily by reformulating the query.

- If the query uses OUTER JOINs instead of INNER JOINs, indexes on tables under certain circumstances might not be usable above this join in the access plan. This could result in expensive join operators as described above.

  Although in general the semantics of both join types is not equivalent, in some cases developers of SQL statements carelessly use OUTER joins in situations where this does not reflect the requirements. Depending on the where clauses, sometimes OUTER JOINs might even be equivalently replaced by INNER JOINs. The DB2 optimizer cannot detect this in all cases. Thus, OUTER JOINs should be carefully examined.
- As described above, certain operators such as UNION, DISTINCT, and GRPBY in general make it impossible for the optimizer to use indexes on tables below this operator. If these operators appear deep inside the query tree, this might result in some cost explosion.

  It is common that UNION and DISTINCT operators are used quite carelessly by developers of SQL statements and are used where it is not required. This can easily be avoided.
- In many typical cases, you will see that the cost of an access plan is caused by expensive leaf nodes in the tree. This is typical for a situation when an index might be introduced to increase performance. In “DB2 Design Advisor” on page 334, we discuss how to determine which index might help.

DB2 Explain: the command line version

To use the command line interface to explain an access plan, you can use the following DB2 command:

```
EXPLAIN PLAN FOR <mySqlStatement>
```

Here <mySqlStatement> stands for the SQL statement that you want to analyze. After successful execution of this statement, all explain tables are populated with information about the access plan. In a second step, you can generate a readable representation of this plan in a text file using the following command:

```
db2exfmt -d <myDb> -e % -w -1 -s % -n % -t -# 0
```

In this sequence, <myDb> stands for the name of the database that you are working with. (For more detailed description of the parameter semantics, refer to *DB2 UDB Command Reference V8.2*, SC09-4828.) This command displays a graph representation of the access plan with all available detail information about standard output. This graph resembles the tree representation of the access plan as we know it from DB2 Visual Explain. It displays the same detail information, but it might be more difficult to read and understand it because we cannot browse through the tree and unfold the detail information just as we need it. Everything is simply included in the output.

Note: Whenever we mention executing a DB2 command, it should always be run in the DB2 command interpreter. If we speak simply about a command, then the command is run in the operating system's command interpreter.

DB2 Design Advisor

We have seen how to analyze an individual query and drill down to the cause of a performance problem. This analysis might let you look for objects that can be added in the database to increase performance. The typical and well-known database object that could help in this way is an index. As described in 2.1.3, "Performance-related concepts" on page 39, with DB2 Version 8, indexes might not be the only database object of this kind. There are now also MQTs (Materialized Query Tables) and the MDC (Multi-Dimensional Clustering) feature. DB2 provides the *DB2 Design Advisor* (formerly known as DB2 Index Advisor) as a tool that advises you about which database objects to consider to add to improve performance. The DB2 Design Advisor supports recommendations for configuring indexes, MQTs, and MDC, and partitioning strategies in a data partitioning feature (DPF) environment.

When using the Design Advisor, it always refers to an existing database. This means that the recommendations that it might come up with apply to this database and will not be valid for another database. In addition, the advice that you receive from this tool always refers to a workload as a set of SQL statements that you need to provide as input to the tool. DB2 Design Advisor then identifies database objects, such as indexes, which might improve performance of the workload, then virtually adds these objects to the database and estimates the performance impact for each of the SQL statements in the workload. In general, it evaluates multiple alternative sets of additional database objects. If the workload is not representative of the actual workload, take this into account before applying the recommendations. Performance and cost estimation is done based on the DB2 database catalog statistics. DB2 Design Advisor not only provides a recommendation for the set of additional database objects that promise best performance, but also estimates the size of these objects in terms of disk space.

Similar to DB2 Explain, DB2 Design Advisor internally places the information that it derives into certain tables in the database.

Important: Before you use DB2 Design Advisor, be sure that the database catalog statistics are up to date. Otherwise, suggestions and estimations about performance improvements that this tool derives might not accurate.

As with many other tools, the DB2 Design Advisor comes in two different versions:

- Command line
- Graphical user interface

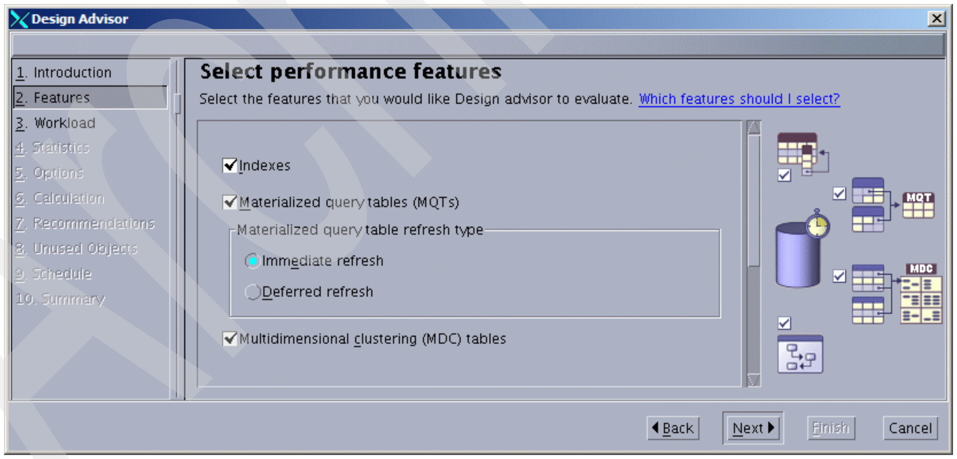
For convenience, we begin our overview with the GUI version, which is integrated into the DB2 Control Center.

DB2 Design Advisor GUI version

To open the DB2 Design Advisor:

1. In the Object View of the DB2 Control Center, locate the database you work with.
2. Right-click on this database and select **Design advisor**. This opens the DB2 Design Advisor wizard.

The DB2 Design Advisor wizard leads you through a sequence of steps. One of the first steps is to specify which of the database objects, index, MQT, and MDC, you want the advisor to consider (Figure 11-15).



Select performance features

Select the features that you would like Design advisor to evaluate. Which features should I select?

[✓] Indexes

[✓] Materialized query tables (MQTs)

Materialized query table refresh type

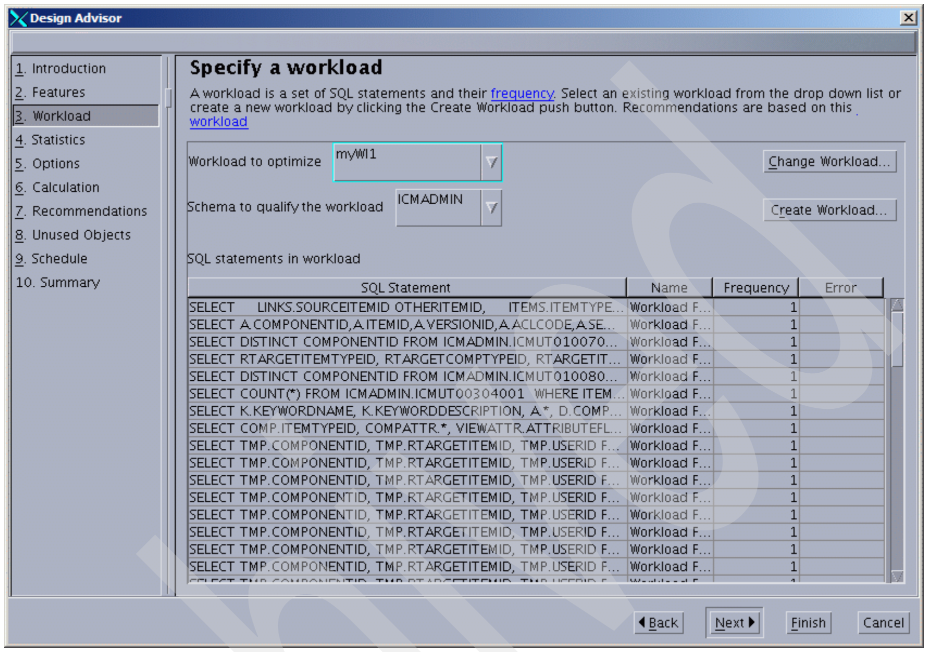
[● Immediate refresh]

○ Deferred refresh

[✓] Multidimensional clustering (MDC) tables

Figure 11-15 Using DB2 Design Advisor: specifying the scope

In subsequent steps, specify the workload that you want to use (Figure 11-16), whether you want to update statistics first, and how much of the system resources you allow the advisor to use during its estimation.



Specify a workload

A workload is a set of SQL statements and their frequency. Select an existing workload from the drop down list or create a new workload by clicking the Create Workload push button. Recommendations are based on this workload

Workload to optimize: myWl1

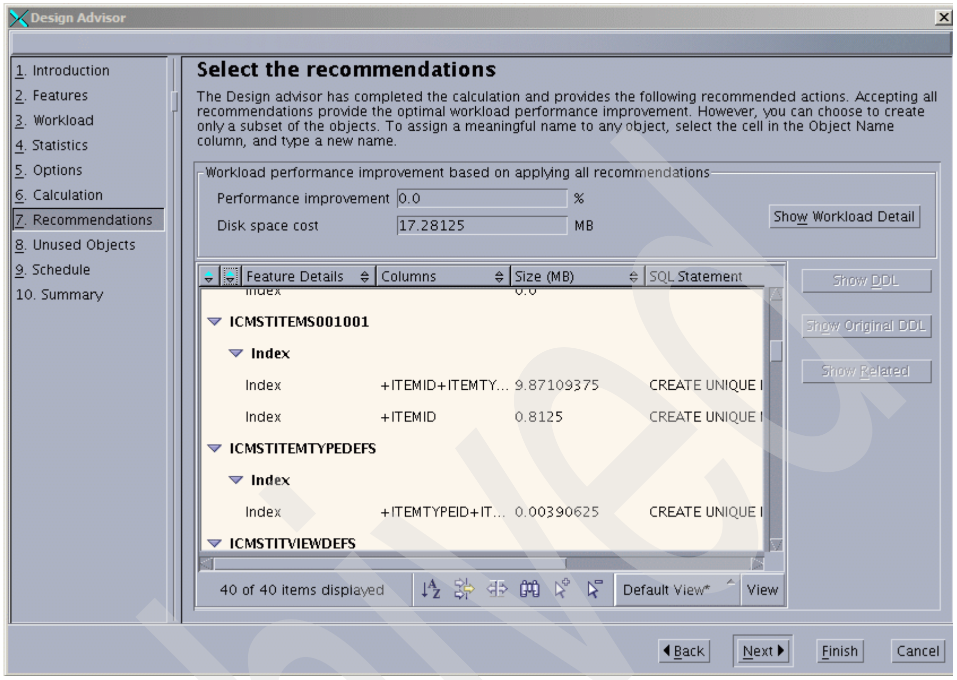
Schema to qualify the workload: ICMADMIN

SQL statements in workload

SQL Statement	Name	Frequency	Error
SELECT LINKS.SOURCEITEMID OTHERITEMID, ITEMS.ITEMTYPE...	Workload F...	1	
SELECT A.COMPONENTID,A.ITEMID,A.VERSIONID,A.ACLCODE,A.SE...	Workload F...	1	
SELECT DISTINCT COMPONENTID FROM ICMADMIN.ICMUT010070...	Workload F...	1	
SELECT RTARGETITEMTYPEID, RTARGETCOMPTYPEID, RTARGETIT...	Workload F...	1	
SELECT DISTINCT COMPONENTID FROM ICMADMIN.ICMUT010080...	Workload F...	1	
SELECT COUNT(*) FROM ICMADMIN.ICMUT00304001 WHERE ITEM...	Workload F...	1	
SELECT K.KEYWORDNAME, K.KEYWORDDESCRIPTION, A.*, D.COMP...	Workload F...	1	
SELECT COMP.ITEMTYPEID, COMPATTR.*, VIEWATTR.ATTRIBUTEFL...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	
SELECT TMP.COMPONENTID, TMP.RTARGETITEMID, TMP.USERID F...	Workload F...	1	

Figure 11-16 DB2 Design Advisor: specifying the workload

At the end, Design Advisor returns its recommendations (Figure 11-17).


Design Advisor

1. Introduction
2. Features
3. Workload
4. Statistics
5. Options
6. Calculation
7. Recommendations
8. Unused Objects
9. Schedule
10. Summary

Select the recommendations

The Design advisor has completed the calculation and provides the following recommended actions. Accepting all recommendations provide the optimal workload performance improvement. However, you can choose to create only a subset of the objects. To assign a meaningful name to any object, select the cell in the Object Name column, and type a new name.

Workload performance improvement based on applying all recommendations

Performance improvement 0.0 %

Disk space cost 17.28125 MB

Feature Details	Columns	Size (MB)	SQL Statement
ICMSTITEMS001001			
Index			
Index	+ITEMID+ITEMTY...	9.87109375	CREATE UNIQUE I
Index	+ITEMID	0.8125	CREATE UNIQUE I
ICMSTITEMTYPEDEFS			
Index			
Index	+ITEMTYPEID+IT...	0.00390625	CREATE UNIQUE I
ICMSTITVIEWDEFS			

40 of 40 items displayed

Figure 11-17 DB2 Design Advisor: recommendations

DB2 Design Advisor command line version

The command line version of DB2 Design Advisor provides the same functionality as the GUI version. You must pass all information that you interactively provide to the GUI version via command line parameters to the command line version. We do not go into all of the details of the command syntax. Refer to *DB2 UDB Command Reference V8.2*, SC09-4828.

We demonstrate the Design Advisor using a very simple single-statement workload. For this, we put the SQL statement `SELECT ITEMTYPEVIEWID from icmadmin.ICMUT00400001;` into the file myWorkLoad. To have DB2 Design Advisor evaluate this in database ICMNLSDB, we use the following command:

```
db2advis -d ICMNLSDB -i myWorkLoad
```

This eventually outputs as shown in Example 11-8. (Some lines are omitted.)

Example 11-8 DB2 Visual Explain: command line output

```
execution started at timestamp 2006-04-07-15.47.20.274518
Recommending indexes...
total disk space needed for initial set [   0.028] MB
total disk space constrained to         [  67.338] MB
Trying variations of the solution set.
Optimization finished.
  1  indexes in current solution
 [184.0000] timerons  (without recommendations)
 [ 91.0000] timerons  (with current solution)
 [50.54%] improvement
--
-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1],    0.028MB
   CREATE INDEX "DB2INST1"."IDX604080215500000" ON "ICMADMIN"."ICMUT00400001"
("ITEMTYPEVIEWID" ASC) ALLOW REVERSE SCANS ;
   COMMIT WORK ;
   RUNSTATS ON TABLE "ICMADMIN"."ICMUT00400001" FOR INDEX
"DB2INST1"."IDX604080215500000" ;
   COMMIT WORK ;

...

7 solutions were evaluated by the advisor
DB2 Workload Performance Advisor tool is finished.
```

DB2 Design Advisor recommendations

Although DB2 Design Advisor appears as an efficient tool that is easy to use to identify performance improvements at SQL statement level, we strongly recommend not blindly following the advice you receive from this tool. In most cases, adding any of the recommended database objects might incur some negative performance impacts that cannot be estimated and quantified by the Design Advisor. Keep the following points in mind:

- Creating an index implies some performance overhead for updates. If this table is frequently hit by updates, you might refrain from adding this index.
- Adding an index under certain circumstances can increase the risk of lock conflicts between concurrent applications.
- An MQT implies redundancy in the data. Logically, your applications probably require that this redundant data is kept in sync. If this synchronization is done automatically during update, it implies a performance overhead for updates.

On the other hand there is no guarantee that the Design Advisor will come up with the most efficient index to support your queries.

It might be tempting to just feed the Design Adviser with the workload you have and let it come back with proposals for additional database objects. You should always be aware that there are some negative side effects.

After implementing your index in your database, verify that the optimizer actually chooses this index.

11.5 Monitoring tools for WebSphere Application Server

WebSphere Application Server collects running application data through the Performance Monitoring Infrastructure (PMI). PMI is a set of packages and libraries designed to assist with gathering, delivering, processing, and displaying performance data in WebSphere Application Server runtime components. Gathered data can be monitored and analyzed with a variety of tools. You can:

- Monitor performance data with Tivoli Performance Viewer.

  This tool is included with WebSphere Application Server.
- Monitor performance data with other Tivoli monitoring tools:
  - IBM Tivoli Monitoring for Web Infrastructure

    Provides best-practice monitoring of the key elements of WebSphere Application Server. This is the inside-out view, enabling administrators to quickly address problems before they affect users. Using Tivoli's advanced monitoring technology and predefined WebSphere best practices, this tool quickly identifies problems, notifies appropriate personnel, and helps provide a solution. Health console displays real-time monitoring data. The same information can be uploaded to a common data warehouse for historical reporting.
  - IBM Tivoli Monitoring for Transaction Performance

    Provides a unique monitoring perspective from that of the users. This is the outside-in view that verifies that end-to-end components provide a positive user experience. Tivoli Monitoring for Transaction Performance monitors performance of the actual and synthetic transactions, as well as verifying that the delivered content meets predefined guidelines.
- Monitor performance data with user-developed monitoring tools.

  Write your own applications to monitor performance data.

11.5.1 Tivoli Performance Viewer

For this book, we used the Tivoli Performance Viewer, previously called Resource Analyzer. The Tivoli Performance Viewer is a stand-alone runtime performance monitor for WebSphere Application Server Version 5.1. It retrieves performance data by periodically polling the PMI service of the monitoring application server, and uses the PMI Java client to provide graphical display and summary reports of collected data.

Tivoli Performance Viewer enables you to:

- View data in real time or in the logs.
- Record data in a log and replay the log later.
- View data in chart form and tabular forms; given visual comparison of multiple counters, each can be scaled independently to enable meaningful graphs.
- Compare data for single resources to aggregate of resources across a node.

Using Tivoli Performance Viewer

Using the Tivoli Monitoring Viewer, you can monitor system performance, detect performance trends, determine the efficiency of resources configurations, and estimate the load on application servers and average response time for clients.

To use Tivoli Performance Viewer, follow these steps:

1. Enable PMI services through the WebSphere Application Server Administrative Console.

   You must do this first to monitor performance data through the PMI interfaces:

   a. Click **Start** → **Programs** → **IBM WebSphere** → **Application Server v5.1** → **Administrative Console**.

   b. Expand **Servers** folder in the console navigation tree on the left side.

   c. Click **Application Server** under the Servers.

   d. Click your WAS server running the Content Manager. By default, it is **server1**.

   e. If you are not at the **Configuration** tab, click it.

   f. Click **Performance Monitoring Service**. If you do not see the entry, scroll down the right side of the window until you see it.

   g. Check the **Startup** box.

   h. (Optional) Select the PMI modules and levels to set the initial specification level field.

   i. Click **Apply** or **OK**, and click **Save** twice.

   j. Restart the WebSphere Application Server.

   Note: The changes you make will not take effect until you restart the WebSphere Application Server.

2. Collect data.

   The monitoring levels that determine which data counters are enabled can be set dynamically, without restarting the server, in any of these ways:

   – Enable data collection through the administrative console.

   – Enable performance monitoring services through Tivoli Performance Viewer.

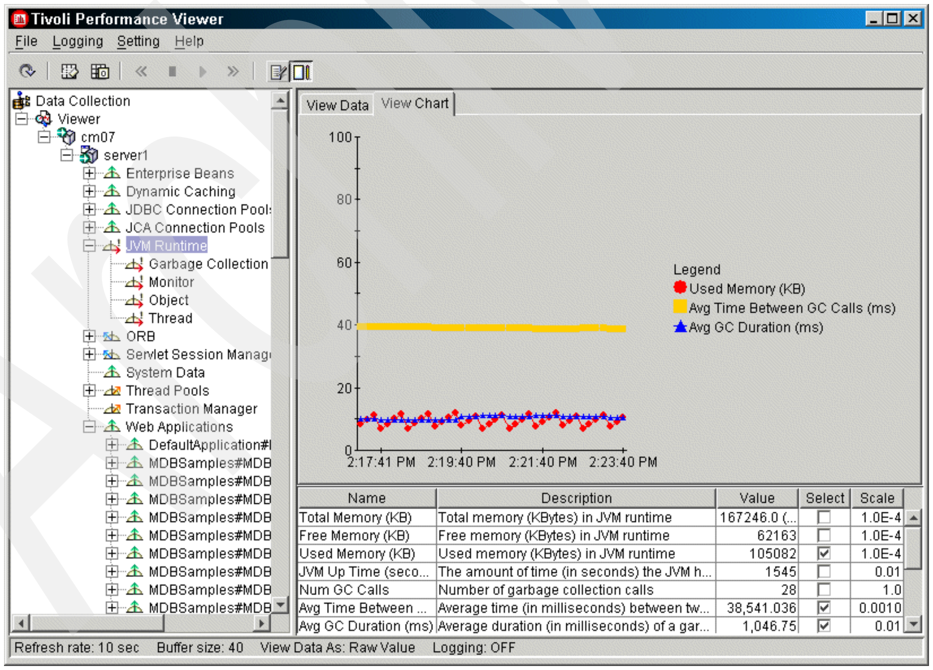
   – Enable performance monitoring services using the command **`wsadmin`**.

3. Monitor and analyze performance data.

   The Tivoli Performance Viewer provides information about a wide range of performance data for two kinds of resources:

   – Application resources such as enterprise beans and servlets
   – WebSphere runtime resources such as Java virtual machine (JVM)

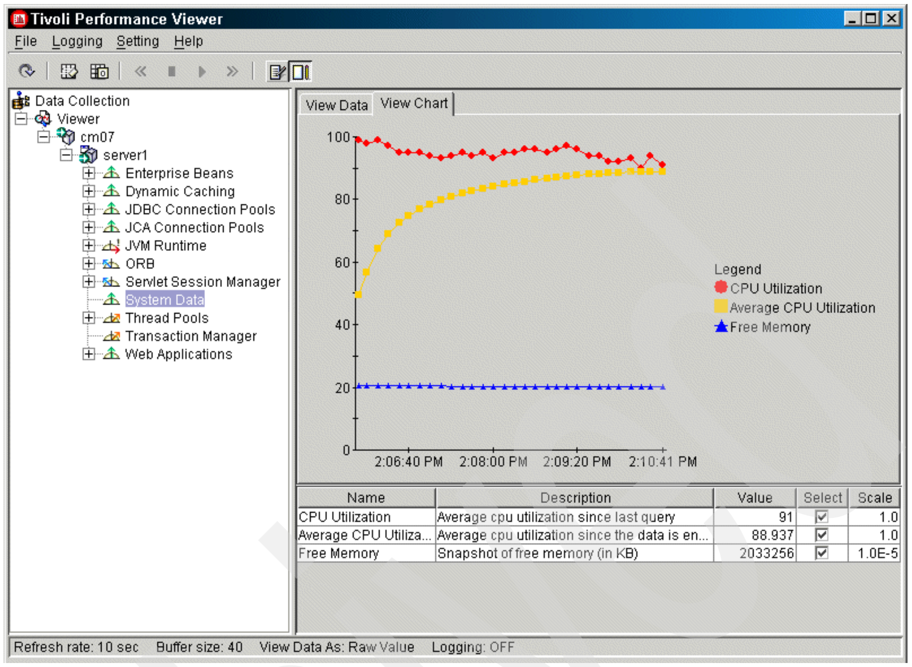
For Content Manager system performance tuning, we monitored information about JVM runtime, system data, and the servlets of the applications on the Tivoli Performance Viewer. Figure 11-18 and Figure 11-19 on page 342 show some of the information that you need to monitor.



Name	Description	Value	Select	Scale
Total Memory (KB)	Total memory (KBytes) in JVM runtime	167246.0 (...	☐	1.0E-4
Free Memory (KB)	Free memory (KBytes) in JVM runtime	62163	☐	1.0E-4
Used Memory (KB)	Used memory (KBytes) in JVM runtime	105082	✓	1.0E-4
JVM Up Time (seco...	The amount of time (in seconds) the JVM h...	1545	☐	0.01
Num GC Calls	Number of garbage collection calls	28	☐	1.0
Avg Time Between ...	Average time (in milliseconds) between tw...	38,541.036	✓	0.0010
Avg GC Duration (ms)	Average duration (in milliseconds) of a gar...	1,046.75	✓	0.01

Refresh rate: 10 sec   Buffer size: 40   View Data As: Raw Value   Logging: OFF

Figure 11-18 Tivoli Performance Viewer: JVM runtime



Name	Description	Value	Select	Scale
CPU Utilization	Average cpu utilization since last query	91	✓	1.0
Average CPU Utiliza...	Average cpu utilization since the data is en...	88.937	✓	1.0
Free Memory	Snapshot of free memory (in KB)	2033256	✓	1.0E-5

Refresh rate: 10 sec Buffer size: 40 View Data As: Raw Value Logging: OFF

Figure 11-19 Tivoli Performance Viewer: System data

For more information about WebSphere Application Server performance monitoring, tuning, and usage of the Tivoli Performance Viewer tool, refer to *IBM WebSphere Application Server, Version 5 - Monitoring and Troubleshooting* at:

http://www.ibm.com/software/webservers/appserv/was/library

11.5.2 WebSphere JVM verbosegc

For WebSphere applications, such as Resource Manager, eClient Server, or a customized mid-tier application, monitoring the memory usage is critical in tuning the performance. The -verbose:gc option in the JVM setting logs garbage collection output to the stderr.log of the application server. Verbose gc outputs are similar to Example 11-9.

Example 11-9 Verbose gc output sample

```
<AF[43]: Allocation Failure. need 524 bytes, 0 ms since last AF>
<AF[43]: managing allocation failure, action=0 (1057927024/1073740288)>
<GC(43): GC cycle started Wed Jul 06 10:58:42 2005
<GC(43): freed 6294968 bytes, 45% free (1064221992/1073740288), in 61 ms>
<GC(43): mark: 53 ms, sweep: 8 ms, compact: 0 ms>
```

```
<GC(43): refs: soft 0 (age >= 32), weak 0, final 23, phantom 0>
<AF[43]: completed in 75 ms>
```

A healthy JVM under a workload should keep the free memory percentage somewhat constant. It decreases then increases after garbage collection is done. If the `45% free (1064221992/1073740288)` percentage goes down over time to near zero, the JVM might hang or report an "out of memory" error, an indication that your application either has a large memory footprint or has memory leaks. More debugging is needed to determine what is causing the problem.

11.5.3 Java Virtual Machine Profiler Interface

The Tivoli Performance Viewer leverages a Java Virtual Machine Profiler Interface (JVMPI) to enable more comprehensive performance analysis. This profiling tool enables the collection of information, such as data about garbage collection, and the JVM API that runs the application server.

JVMPI is a two-way function call interface between the JVM API and an in-process profiler agent. The JVM API notifies the profiler agent of various events, such as heap allocations and thread starts. The profiler agent can activate or deactivate specific event notifications based on the needs of the profiler.

JVMPI supports partial profiling by enabling the user to choose which types of profiling information to collect and to select certain subsets of the time during which the JVM API is active. JVMPI moderately increases performance impact.

Enabling Java Virtual Machine Profiler Interface data reporting

To enable JVMPI data reporting for each individual application server:

1. Open the administrative console.
2. Click **Servers** → **Application Servers** in the console navigation tree. Select the application server for which JVMPI must be enabled.
3. Click **Process Definition**. Then click **Java Virtual Machine**.
4. Type `-XrunpmiJvmpiProfiler` in the Generic JVM arguments field.
5. Click **Apply** or **OK**. Click **Save**.
6. This returns you to the window from step 2. Click the **Configuration** tab.

   When in the Configuration tab, settings will apply when the server is restarted. When in the Runtime Tab, settings apply immediately. Note that Performance Monitoring Service can be enabled only in the Configuration tab.
7. Click **Performance Monitoring Service** and select the **Startup** checkbox.
8. Set initial specification level to **Custom** and `jvmRuntimeModule=X`.

9. Click **Apply** or **OK**. Click **Save**.
10. Start the application server, or restart the application server if it is currently running. Refresh the Tivoli Performance Viewer if you are using it. The changes you make will not take effect until you restart the application server.

11.6 Performance tracing for Content Manager

Several performance tracing options are available for Content Manager. They are used to trace ICM connector performance, Library Server, and Resource Manager performance.

11.6.1 ICM connector performance tracing

The log configuration file, cmblogconfig.properties in the %cmgmt% directory, controls the trace level for ICM connector. The log priority setting key, DKLogPriority, can be used to turn on performance tracing and is applicable on the basis of per-JVM instance.

DKLogPriority has the following possible values:

DISABLE	Disable logging.
FATAL	Log fatal messages.
ERROR	Log fatal and error messages.
PERF	Log fatal, error, and performance messages.
INFO	Log fatal, error, performance, and information messages.
TRACE_NATIVE_API	Log all of the above, and TRACE_NATIVE_API messages.
TRACE_ENTRY_EXIT	Log all of the above, and TRACE_ENTRY_EXIT messages.
TRACE	Log all of the above, and trace messages.
DEBUG	Log all of the above, and debug messages.

The traced data will go to a file specified by DKLogOutputFileName in the configuration file. The default value for the output file is dklog.log.

To enable ICM connector performance trace, follow these steps:

1. Open the cmblogconfig.properties file (in %cmgmt% directory) for edit using Notepad, vi, or another applicable editor.

2. Set `DKLogPriority=TRACE`.
3. Stop and restart your system.

To turn off the ICM connector performance trace:

1. Open the cmblogconfig.properties file for edit using Notepad, vi, or another available editor.
2. Set `KDKLogPriority=ERROR`.
3. Stop and restart your system.

Note: Log configuration settings (key and value) are case sensitive. Trailing blanks following the value fields are not allowed.

See the comment section in the configuration file for more details.

11.6.2 Library Server performance tracing

To enable Library Server performance tracing, run the script in Example 11-10.

Example 11-10 Sample DB2 script to enable Library Server performance trace

```
db2 connect to icmnlsdb user icmadmin
db2 update icmstsyscontrol set tracelevel=-8
db2 update icmstsyscontrol set keeptraceopen=1
db2 connect reset
```

The trace data will go into the file specified in the system administration client. On Windows, the default is c:\ICMServer.log.

To turn off the Library Server performance trace, run the script in Example 11-11.

Example 11-11 Sample DB2 script to turn off Library Server performance trace

```
db2 connect to icmnlsdb user icmadmin
db2 update icmstsyscontrol set tracelevel=0
db2 update icmstsyscontrol set keeptraceopen=0
db2 connect reset
```

11.6.3 Resource Manager performance tracing

Resource Manager logging is controlled by the XML file, icmrm_loggin.xml, in the WebSphere\AppServer\InstalledApps\ICMRM_App.ear\icmrm.war directory. The priority value in the XML file can be used to turn on performance tracing.

The priority value has the following possible values:

FATAL	Only logs if the servlet is terminating unexpectedly.
ACTION	Logs messages that describe an action the System Administrator has to take. These are not errors but conditions such as low space.
ERROR	Logs a request that was unable to be fulfilled or an internal error.
WARN	Logs warnings of unexpected behavior.
INFO	Logs informational start and stop messages.
BEGINEND	Time markers begin and end for performance measures.
REQUEST	Logs detailed information about the incoming request.
RESPONSE	Logs detailed information about the outgoing response.
TRACE	Logs general flow messages.
DEBUG	Logs detailed debugging information, plus all other levels.

The traced data will go to ibmrm.logfile in the \WebSphere\AppServer\logs directory. You can direct it to another file within this XML file:

```
<param name="File"
value="C:/WEBSPH~1/APPSER~1/logs/icmrm/icmrm.logfile"/>
```

To enable Resource Manager performance trace:

1. Open the icmrm_loggin.xml file (in the WebSphere\AppServer\InstalledApps\ICMRM_App.ear\icmrm.war directory) to edit using Notepad or another applicable editor.
2. Set `<priority value=”BEGINEND”>`
3. Stop and restart your system.

To turn off the Resource Manager performance trace:

1. Open the icmrm_loggin.xml file to edit using Notepad or another applicable editor.
2. Set `<priority value=”ERROR”>`
3. Stop and restart your system.

12

Troubleshooting performance problems

This chapter covers troubleshooting Content Manager performance problems based on a collection of real-life scenarios.

We start with troubleshooting basics. Then we address the most common performance issues including slow logon to the Content Manager system, slow search, slow import, slow migrator, and slow administration using the system administration client. In addition, we recommend common rules to help you avoid performance issues and describe the tools you can use to analyze the log files.

12.1 Troubleshooting basics

The key to successfully and efficiently resolving performance problems in a Content Manager system is to have an organized, disciplined process that focuses on identifying and relieving the system's performance bottlenecks. Without an organized process focused on identifying bottlenecks, it is very easy to spend a lot of time and energy tweaking various parameters, ending up with a system in a worse state than before with no good record of what was changed and why. To be successful, you must collect the right data and find the real bottleneck. You will then be able to address that bottleneck, tune the system, and improve the performance of your Content Manager system.

Before beginning any full performance troubleshooting process, first ensure that the database statistics for the Library Server and Resource Manager databases are up to date. Content Manager is a database application. It depends heavily on accurate database statistics for optimizing all operations. Updating database statistics on the Library Server and Resource Manager databases is quick and easy to do, and often turns out to be the only fix needed.

Performance problems are often caused by changes in the system. The changes that might affect system performance include:

- Hardware changes

  Example: Adding new disks or changing hardware configurations
- Operating system changes

  Example: Installing PTFs or changing parameters
- Software changes

  Example: Adding fix packs for DB2 and WebSphere, or configuration changes
- Application changes

  Example: Installing or upgrading new versions and fixes, configuration changes, installing new modules, or adding new users or new content
- Performance tuning

  Example: Tuning changes in operating system, network, DB2, WebSphere, TSM, or the application
- Any changes

Is it a performance problem?

When a user complains about a problem or you experience a problem firsthand, you must first determine the nature of the problem: whether it is a functional problem or a performance problem. When your hardware, network, or application does not behave correctly, it is a functional problem (for example, if your Content

Manager application has a memory leak). Sometimes, functional problems lead to performance problems (an application memory leak eventually slowing down the overall system performance). In such cases, it is important to determine the root cause of the problem and fix it rather than focus on tuning your system. Using the example of application memory leak, you need to address the problem to the development team and ask them to eliminate the memory leak.

Asking questions

Before jumping into collecting data and analyzing data, you should get as much detailed information as possible about the problem experienced by users. Questions to ask users or to research on your own include:

- Is this an across-the-board problem, or does it affect only certain operations or users?
- What operations are slow — the time it takes to search for a document or to display a document, for example?
- Can the problem be reproduced by exercising a certain task or sequence of events?
- Is the slow performance intermittent (does it disappear from time to time)? Does it occur at a certain time of day or in relation to certain tasks?
- When did the problem start occurring? Was the situation the same when the system was first installed or went into production? (Is this a regression or were the performance objectives never met?) Did anything change on the system before the problem occurred (such as adding more users or loading additional legacy content to the system)?
- Is this a single-user performance problem or multi-user scalability problem?
- If remote users are complaining about document retrieval, do local users performing the same document retrieval experience the same slowness (network versus server issue)?
- If this is network related, how are the network segments configured (including bandwidth such as 10 Mbps or 9600 baud)? Are there any routers between the client and server? Are the objects cached?
- What other applications are running on the system, and are those applications involved in the performance issue?
- What is the impact of the performance problem on the users?

After understanding the nature of the problem, we need to find the location of the bottleneck or what area causes the problem. Several actions can be taken:

- Use monitoring tools and programs to check system performance:
  - Obtain workload metrics to compare with the baseline.

  - Obtain server resource utilization (CPU, network, memory, and disk) with the baseline.
- Use Content Manager performance trace to start tracing the problem.
- Review last changes to the system.
- Review historic performance parameter changes since production.

If the problem is DB2 performance:

- Have you run **runstats/rebind** lately? If not, it is time to run them and make sure you perform this task on weekly or monthly basis in future.
- Less often than **runstats**/**rebind**, consider reorganizing certain tables in the Library Server and Resource Manager databases to improve performance. Use **reorgchk current statistics** to identify tables that might benefit from reorganization. Perform **reorg** only on the tables that will benefit from it.

Note: Always keep the following pointers in mind:

- Run **reorgchk** whenever you run **runstats** to see whether any tables need to be reorganized.
- Running **reorg** affects performance and the task might take long time. Always plan ahead as when and what tables to **reorg**.
- Running the **reorg** task should be performed only by an experienced database administrator. We *strongly recommend* that before you proceed, you read "Table Management" in *IBM DB2 Universal Database Version 8 Administration Guide Performance*, SC09-4821.

12.2 Content Manager troubleshooting overview

Users might experience slow performance when they log on to a Content Manager system using the Content Manager client, search a document, import a new document, migrate the document from one storage device to another, or manage the system in system administration client. These actions involve the Library Server, Resource Manager application, Resource Manager processes, system administration client, and other system components.

Library Server is the key component of Content Manager. It stores, manages, and provides access to the content stored on a Resource Manager. Library Server relies on a relational database to perform parametric, text, and combined parametric-text searches. It is accessed using SQL or a relational database client. The Library Server log file (ICMSERVER.LOG by default) contains performance data. If a user experienced slow performance when taking the actions above, the Content Manager administrator should check the ICMSERVER.LOG file for any performance bottleneck on the Library Server. A Library Server trace is the most important step in troubleshooting a performance problem.

Resource Manager stores objects for Content Manager. It includes the Resource Manager database, Resource Manager application, and some Resource Manager processes. The Resource Manager application runs on a WebSphere Application Server. Most of the performance issues on the Resource Manager application can be resolved by tuning the WebSphere Application Server and Tivoli Storage Manager (TSM). The Resource Manager log file also contains some performance data.

There are several Resource Manager processes. For example, the migrator, one of the Resource Manager processes, checks the migration policies in the Content Manager system and moves objects to the next storage class when the objects are scheduled to move. The migrator is also responsible for deleting from the file system the items that have been marked for deletion by users' explicit action. The Resource Manager processes are Java-based applications that talk to the Library Server database and Resource Manager database using JDBC drivers. The performance issues related to the Resource Manager processes can be resolved by tuning some SQL statements in these processes, or tuning TSM (if you are using it).

The system administration client communicates with the Library Server and Resource Managers. With it, you can perform tasks such as defining the data model, defining users and their access to the system, and managing storage. Performance issues that are related to the system administration client can be resolved by tuning the Library Server database, the SQL involved in the administration tasks, and other actions.

We discuss performance troubleshooting in the order of the following actions:

- Log on to Content Manager client.
- Search a document.
- Import a new document.
- Migrate a document from one storage device to another.
- Manage the system using the system administration client.
- API programming.

The troubleshooting scenarios that are described in the following sections are collected from real-life experiences. We include these scenarios to help you better understand some of the processes you might have to go through to troubleshoot performance problems in a Content Manager system.

12.3 Slow logon

When users log on to the Windows client, eClient, or their own custom Content Manager client (using Information Integrator for Content APIs), they might experience slow performance. For example, in one scenario, users need approximately 40 seconds to log on to the eClient. Some possible causes are:

- Slow DB2 UDB connection between client and server machines.
- Slow response from the LDAP server (such as the Microsoft Active Directory® Server and IBM Tivoli Directory Server) if you enabled Content Manager with the LDAP server.
- Slow response from DNS or network.
- A DB2 UDB performance issue.
- Slow response from Content Manager's stored procedure call (such as ICMLOGON), especially with some bad performance SQL statements.

Before we discuss these possibilities, we briefly explain the logon process. Figure 12-1 illustrates the logon process. Content Manager clients are based on the Information Integrator for Content APIs. When a user tries to log on to the Content Manager client, the client calls the Information Integrator for Content APIs. The APIs then talk to the DB2 UDB Client, which then calls the stored procedures (such as ICMLOGON) on the DB2 UDB server. If you integrate Content Manager with the LDAP servers, the ICMLOGON calls a user exit first to authenticate the user ID and password with the LDAP server. If the LDAP server cannot verify this user ID and password, the ICMLOGON stored procedure will check it by itself. The logon process is also doing other jobs, such as getting item type information from the Library Server.

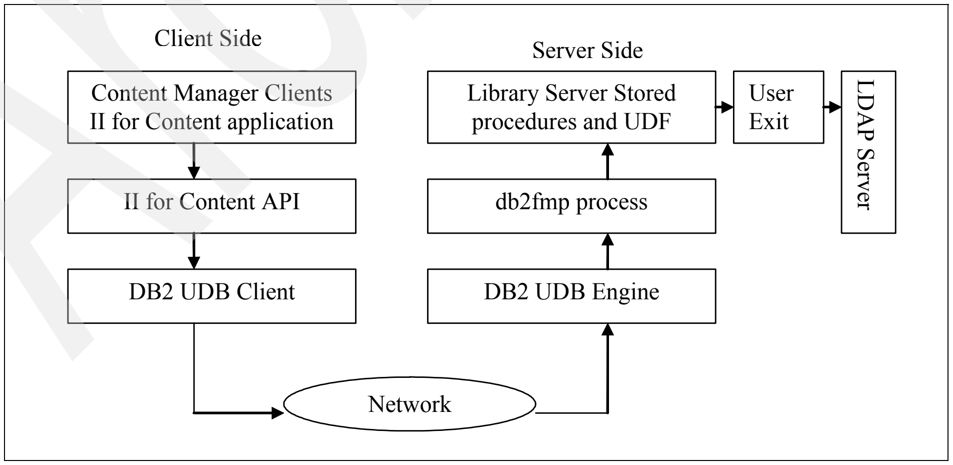


Figure 12-1 Logon process

Content Manager consists of stored procedures and UDFs, so before the Information Integrator for Content API can call the stored procedures, the API must connect successfully to the Library Server database.

Whatever the cause, we need to check the ICMSERVER.LOG with level 15, which can tell us which is used most of the time: the stored procedure call or the user exit calls (including the LDAP server).

Note: Trace level 8 is for performance, but trace level 15 can give us information besides performance.

The Library Server performance trace can be controlled from the Content Manager system administration client. Starting or stopping this trace will begin immediately.

The Library Server performance trace can also be started and stopped from the following script:

```
db2 connect to ICMNLSDB user icmadmin using <password>
db2 update icmstsyscontrol set tracelevel=15
db2 connect reset
```

Trace data will go into the file specified in the system administration client (called ICMSERVER.LOG by default). To turn off the trace, run the following script:

```
db2 connect to icmnlsdb user icmadmin using <password>
db2 update icmstsyscontrol set tracelevel=0
db2 connect reset
```

This setting is applicable for all Library Server actions including search and import.

Example 12-1 shows sample output of the Library Server log with trace level 15.

Example 12-1 ICMSERVER.LOG sample content

```
ICMPLSLG ICMLOGON                    01856 05/30 17:39:24.193 GMT ; ... ICMADMIN  1804 msec
...
ICMPLSMM ICMLISTMIMETYPE             00259 05/30 17:39:24.704 GMT ; ... ICMADMIN  266 msec
...
ICMPLSGA ICMGETATTRTYPE              00412 05/30 17:39:25.024 GMT ; ... ICMADMIN  240 msec
...
ICMPLSGT ICMGETITEMTYPE              00919 05/30 17:39:28.489 GMT ; ... ICMADMIN  3404
msec
```

The second field is the name of the Library Server operation, and you can find the times spent on the operation in milliseconds in the end of the row. This information shows whether a performance issue is due to time spent within the Library Server or some other component of the Content Manager system.

12.3.1 Typical log information for logon process

We now examine the information logged in to the ICMSERVER.LOG file, with trace level set to 15, with a Content Manager administrator (ICMADMIN by default), an LDAP user ID, and an internal Content Manager user ID logged on to the Content Manager Windows client. The main difference is in the ICMLOGON stored procedure call.

Using trace level 15 in all of the ICMSERVER.LOG files produces this information:

- How long each stored procedure is run
- The user ID who tried to log on to the client and the database ID that was used to connect to the Library Server database
- Whether the LDAP server authenticated the user ID and password successfully
- How long the Content Manager server talked to the LDAP server
- Which SQL statement might be used most often

Log on as the system administrator (ICMADMIN)

In this case, we logged on to the Content Manager system using the default Content Manager system administrator, ICMADMIN. Example 12-2 shows the extracted information from the ICMSERVER.LOG file. Both the UserID and DB2UserID are ICMADMIN. The client connected to the Library Server database as ICMADMIN and called the ICMLOGON stored procedure. The entire ICMLOGON stored procedure used 1.804 seconds. After that, the client called other stored procedures (for example, ICMLISTMIMETYPE) to finish the logon process. You can add all msec to calculate how long the stored procedures ran. If a user experienced 2 minutes to log on to the client, but the stored procedures only used 5 seconds from your calculation, then the issue should not be at the Content Manager server side. It might be caused by the database connection, network, or some other areas.

Example 12-2 ICMSERVER.LOG: Log on with system user ID, ICMADMIN

```
ICMPLSLG ICMLOGON                         00586 05/30 17:39:22.781 GMT ;30173922541325
ICMADMIN Entr
......
ICMPLSLG ICMLOGON                         00787 05/30 17:39:22.781 GMT ;30173922541325
ICMADMIN Input
   UserID            <ICMADMIN>
   DB2UserID         <ICMADMIN>
   Language          <ENU>
   Application       <ICM Connector>
   LogonOption        0
......
```

```
ICMPLSLG callUserExit              02103 05/30 17:39:23.372 GMT ;30173922541325  ICMADMIN
Return rc=0
......
ICMPLSLG ICMLOGON                  01856 05/30 17:39:24.193 GMT ; ... ICMADMIN 1804 msec
ICMPLSLG ICMLOGON                  01857 05/30 17:39:24.193 GMT ;30173922541325  ICMADMIN
Exit rc=0 reason=0 extrc=0 extreason=0
......
ICMPLSMM ICMLISTMIMETYPE           00259 05/30 17:39:24.704 GMT ; ... ICMADMIN  266 msec
ICMPLSGA ICMGETATTRTYPE            00412 05/30 17:39:25.024 GMT ; ... ICMADMIN  240 msec
ICMPLSGT ICMGETITEMTYPE            00919 05/30 17:39:28.489 GMT ; ... ICMADMIN 3404 msec
ICMPLSLX ICMLISTXDOOBJECT          00335 05/30 17:39:28.779 GMT ; ... ICMADMIN  265 msec
ICMPLSGA ICMGETATTRTYPE            00412 05/30 17:39:28.900 GMT ; ... ICMADMIN  117 msec
ICMPLSLK ICMLISTNLSKEYWRD          00860 05/30 17:39:29.220 GMT ; ... ICMADMIN  102 msec
```

Log on to the system as an LDAP user ID

In this case (Example 12-3), we logged on to the Content Manager system with an LDAP user ID: SAM. The client connected to the Library Server database using the ICMCONCT user ID. Then, it called ICMLOGON. The ICMLOGON called user exit ICMXLSLG.DLL, which authenticated user ID SAM and his password with the LDAP server. In the log file, if the user exit returned RC=0, Reason=1, ExtRC=0, and ExtReason=0, then it means that the LDAP server successfully authenticated this user ID and password.

Example 12-3 Log on to the system using an LDAP user ID, User exit return

```
ICMPLSLG callUserExit                     02097 05/30 17:42:05.955 GMT
;30174205175681 ? SAM User Exit returned
   RC             0
   Reason         1
   ExtRC          0
   ExtReason      0
```

Also, you can calculate how long the Content Manager server talked to the LDAP server: 17:42:05.955 - 17:42:05.905 = 0.04 seconds. The first timestamp is when the user exit returned and the second timestamp is just before the ICMLOGON called the user exit (ICMPLSLG ICMLOGON ... Input). See Example 12-4.

Example 12-4 Log on to the system using an LDAP user ID, extracted information

```
ICMPLSLG ICMLOGON                       00586 05/30 17:42:05.905 GMT ;30174205175681 ? SAM
Entry
......
ICMPLSLG ICMLOGON                       00787 05/30 17:42:05.905 GMT ;30174205175681 ? SAM
Input
   UserID            <SAM>
   DB2UserID         <ICMCONCT>
   Language          <ENU>
   Application       <ICM Connector>
   LogonOption        0
......
ICMPLSLG callUserExit                    01935 05/30 17:42:05.905 GMT ;30174205175681 ? SAM
Entry
.....
ICMPLSLG callUserExit                    02097 05/30 17:42:05.955 GMT ;30174205175681 ? SAM
User Exit returned
   RC              0
   Reason          1
   ExtRC           0
   ExtReason       0
......
ICMPLSLG ICMLOGON                    01856 05/30 17:42:05.955 GMT ; ...  SAM     63 msec
ICMPLSMM ICMLISTMIMETYPE             00259 05/30 17:42:05.975 GMT ; ...  SAM      3 msec
ICMPLSGA ICMGETATTRTYPE              00412 05/30 17:42:05.995 GMT ; ...  SAM      3 msec
ICMPLSGT ICMGETITEMTYPE              00919 05/30 17:42:06.126 GMT ; ...  SAM    102 msec
ICMPLSLX ICMLISTXDOOBJECT            00335 05/30 17:42:06.136 GMT ; ...  SAM      3 msec
ICMPLSGA ICMGETATTRTYPE              00412 05/30 17:42:06.146 GMT ; ...  SAM      2 msec
ICMPLSLK ICMLISTNLSKEYWRD            00860 05/30 17:42:06.306 GMT ; ...  SAM      1 msec
```

Log on as a Content Manager internal user ID

In this case, John is a Content Manager internal user ID, therefore the client connected to the Library Server database using the connection ID (ICMCONCT). Then the client called ICMLOGON. Example 12-5 shows the extracted information from the ICMSERVER.LOG file.

Example 12-5 Log on to the system using a Content Manager internal user ID

```
ICMPLSLG ICMLOGON                        00586 05/30 18:39:07.806 GMT ;30183907123688 ? JOHN
Entry
......
ICMPLSLG ICMLOGON                        00787 05/30 18:39:07.806 GMT ;30183907123688 ? JOHN
Input
   UserID            <JOHN>
   DB2UserID         <ICMCONCT>
   Language          <ENU>
   Application       <ICM Connector>
   LogonOption        0
......
ICMPLSLG callUserExit                    02097 05/30 18:39:07.846 GMT ;30183907123688 ? JOHN
User Exit returned
   RC              0
   Reason          0
   ExtRC           10
   ExtReason       0
......
ICMPLSLG ICMLOGON                    01856 05/30 18:39:07.896 GMT ; ...  JOHN    95 msec
```

```
ICMPLSMM ICMLISTMIMETYPE           00259 05/30 18:39:07.906 GMT ; ...  JOHN     4 msec
ICMPLSGA ICMGETATTRTYPE            00412 05/30 18:39:07.926 GMT ; ...  JOHN     3 msec
ICMPLSGT ICMGETITEMTYPE            00919 05/30 18:39:08.056 GMT ; ...  JOHN    98 msec
ICMPLSLX ICMLISTXDOOBJECT          00335 05/30 18:39:08.066 GMT ; ...  JOHN     3 msec
ICMPLSGA ICMGETATTRTYPE            00412 05/30 18:39:08.076 GMT ; ...  JOHN     2 msec
ICMPLSLK ICMLISTNLSKEYWRD          00860 05/30 18:39:08.236 GMT ; ...  JOHN     1 msec
```

12.3.2 DB2 UDB connection

In the logon process, the client needs to connect to the Library Server database successfully and then calls the stored procedures. If the database connection is very slow, the whole logon process will be very slow as well.

For example, in one scenario, some user IDs logged on to the system very slowly (around 1-5 minutes), while others were as usual. We set the trace level to 15 and examined the ICMSERVER.LOG file, which included the data when a "good" ID and a "bad" ID logged on to the system. A good ID means that this ID has no logon performance issues, and a bad ID has a logon performance issue. For this scenario, TB774 was the good ID and T4834 was the bad ID.

Example 12-6 shows the extracted information from the ICMSERVER.LOG file.

Example 12-6 ICMSERVER.log with both good ID and bad ID log on to the system

```
ICMPLSLG ICMLOGON                         00525 02/22 17:00:08.717
?22170008716468 Input
   UserID          <TB774>
   DB2UserID       <TB774>
......

ICMPLSLG ICMLOGON                         00525 02/22 17:04:18.158
?22170418157471 Input
   UserID          <T4834>
   DB2UserID       <ICMCONCT>
```

Because user TB774 connected to Library Server database (DB2UserID <TB774> in the log file), we assumed that user TB774 is an operating system user ID. User T4834 used ICMCONCT to connect to the Library Server database (DB2UserID <ICMCONCT> in the log file), so we assumed that user T4834 is a Content Manager internal user ID or an LDAP user ID.

With further examination in this case, we found the following information:

- Users who had no performance issues were the ones using the operating system user IDs, such as TB774.
- Those operating system user IDs have the database connection privilege.

- The operating system IDs connected to the Library Server database using their own user IDs instead of ICMCONCT.
- Users who had performance issues were Content Manager internal user IDs.

Then, we calculated the total time the stored procedures used for logon process on user ID TB774 and T4834. We found that TB774 spent more time on stored procedures execution than T4834. That is, the bad one finished the stored procedure calls more quickly than the good one.

For the good one (the operating system user ID), we extracted the information from the ICMSERVER.LOG file as shown in Example 12-7.

Example 12-7 ICMSERVER.LOG with good user ID

```
ICMPLSLG ICMLOGON                    01393 02/22 17:00:08.726 ?22170008716468    10 msec
ICMPLSLS ICMLISTCNTRLPARM            00271 02/22 17:00:09.126 ?22170008716468     2 msec
ICMPLSLK ICMLISTNLSKEYWRD            00499 02/22 17:00:09.250 ?22170008716468     1 msec
ICMPLSLP ICMLISTPRIVILEGE            00374 02/22 17:00:09.552 ?22170008716468     3 msec
ICMPLSGA ICMGETATTRTYPE              00286 02/22 17:00:09.798 ?22170008716468     3 msec
ICMPLSGT ICMGETITEMTYPE              00588 02/22 17:00:10.140 ?22170008716468    83 msec
ICMPLSLX ICMLISTXDOOBJECT            00226 02/22 17:00:10.404 ?22170008716468     1 msec
ICMPLSGA ICMGETATTRTYPE              00286 02/22 17:00:10.719 ?22170008716468     1 msec
ICMPLSCL ICMLISTCOLL                 00486 02/22 17:00:11.860 ?22170008716468     2 msec
ICMPLSLK ICMLISTNLSKEYWRD            00499 02/22 17:00:12.013 ?22170008716468     1 msec
ICMPLSQU ICMSEARCH                   00228 02/22 17:00:12.261 ?22170008716468    16 msec
ICMPLSCL ICMLISTCOLL                 00486 02/22 17:00:12.489 ?22170008716468     2 msec
ICMPLSLK ICMLISTNLSKEYWRD            00499 02/22 17:00:12.652 ?22170008716468     2 msec
ICMPLSLK ICMLISTNLSKEYWRD            00499 02/22 17:00:12.899 ?22170008716468     1 msec
ICMPLSLC ICMLISTCOMPILEDACL          00337 02/22 17:00:13.160 ?22170008716468    27 msec
ICMPLSMM ICMLISTMIMETYPE             00181 02/22 17:00:13.439 ?22170008716468     2 msec
ICMPLSQU ICMSEARCH                   00228 02/22 17:00:13.697 ?22170008716468    13 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:00:13.800 ?22170008716468    17 msec
ICMPLSQU ICMSEARCH                   00228 02/22 17:00:15.512 ?22170008716468    10 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:00:15.610 ?22170008716468    14 msec
ICMPLSQU ICMSEARCH                   00228 02/22 17:00:16.445 ?22170008716468    11 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:00:16.674 ?22170008716468    25 msec
ICMPLSQU ICMSEARCH                   00228 02/22 17:00:32.175 ?22170008716468  5259 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:00:33.797 ?22170008716468    60 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:00:36.913 ?22170008716468   231 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:00:39.065 ?22170008716468    52 msec
ICMPLSCO ICMCHECKOUTITEM             00407 02/22 17:01:23.414 ?22170008716468    41 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:01:23.615 ?22170008716468    30 msec
ICMPLSGD ICMGETDOCTOC                00598 02/22 17:01:23.862 ?22170008716468     8 msec
ICMPLSGI ICMGETITEM                  03832 02/22 17:01:24.068 ?22170008716468    22 msec
ICMPLSLM ICMLISTRESOURCEMGR          00788 02/22 17:01:24.169 ?22170008716468    13 msec
ICMPLSCI ICMCHECKINITEM              00387 02/22 17:03:30.651 ?22170008716468    22 msec
ICMPLSLF ICMLOGOFF                   00176 02/22 17:03:34.061 ?22170008716468     1 msec
```

For the bad one (using the Content Manager internal user ID), we extracted the information from the ICMSERVER.LOG file as shown in Example 12-8.

Example 12-8 ICMSERVER.LOG with bad user ID

```
ICMPLSLG ICMLOGON                  01393 02/22 17:04:18.179 ?22170418157471    22 msec
ICMPLSLS ICMLISTCNTRLPARM          00271 02/22 17:04:18.577 ?22170418157471     2 msec
ICMPLSLK ICMLISTNLSKEYWRD          00499 02/22 17:04:18.663 ?22170418157471     1 msec
ICMPLSLP ICMLISTPRIVILEGE          00374 02/22 17:04:19.026 ?22170418157471     4 msec
ICMPLSGA ICMGETATTRTYPE            00286 02/22 17:04:19.277 ?22170418157471     3 msec
ICMPLSGT ICMGETITEMTYPE            00588 02/22 17:04:19.618 ?22170418157471    92 msec
ICMPLSLX ICMLISTXDOOBJECT          00226 02/22 17:04:21.469 ?22170418157471     3 msec
ICMPLSGA ICMGETATTRTYPE            00286 02/22 17:04:21.920 ?22170418157471     5 msec
ICMPLSCL ICMLISTCOLL               00486 02/22 17:04:23.031 ?22170418157471     1 msec
ICMPLSLK ICMLISTNLSKEYWRD          00499 02/22 17:04:23.183 ?22170418157471     2 msec
ICMPLSQU ICMSEARCH                 00228 02/22 17:04:23.432 ?22170418157471    16 msec
ICMPLSCL ICMLISTCOLL               00486 02/22 17:04:23.779 ?22170418157471     2 msec
ICMPLSLK ICMLISTNLSKEYWRD          00499 02/22 17:04:23.948 ?22170418157471     3 msec
ICMPLSLK ICMLISTNLSKEYWRD          00499 02/22 17:04:24.215 ?22170418157471     2 msec
ICMPLSLC ICMLISTCOMPILEDACL        00337 02/22 17:04:24.472 ?22170418157471    27 msec
ICMPLSMM ICMLISTMIMETYPE           00181 02/22 17:04:24.751 ?22170418157471     3 msec
ICMPLSQU ICMSEARCH                 00228 02/22 17:04:25.001 ?22170418157471    13 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:25.105 ?22170418157471    22 msec
ICMPLSQU ICMSEARCH                 00228 02/22 17:04:25.514 ?22170418157471    14 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:25.615 ?22170418157471    20 msec
ICMPLSQU ICMSEARCH                 00228 02/22 17:04:26.036 ?22170418157471    18 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:26.775 ?22170418157471    29 msec
ICMPLSQU ICMSEARCH                 00228 02/22 17:04:33.014 ?22170418157471  1208 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:34.120 ?22170418157471    58 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:37.029 ?22170418157471   229 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:39.456 ?22170418157471    50 msec
ICMPLSCO ICMCHECKOUTITEM           00407 02/22 17:04:47.114 ?22170418157471     3 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:47.294 ?22170418157471    23 msec
ICMPLSGD ICMGETDOCTOC              00598 02/22 17:04:47.537 ?22170418157471     7 msec
ICMPLSGI ICMGETITEM                03832 02/22 17:04:47.715 ?22170418157471    18 msec
ICMPLSLM ICMLISTRESOURCEMGR        00788 02/22 17:04:47.803 ?22170418157471     3 msec
ICMPLSCI ICMCHECKINITEM            00387 02/22 17:05:51.017 ?22170418157471    26 msec
ICMPLSLF ICMLOGOFF                 00176 02/22 17:05:53.293 ?22170418157471    35 msec
```

Based on the data above, the delay is not from the Content Manager Server side. With further investigation, we took a CLI trace on the Windows client side for the good user ID and the bad user ID. We found that:

- ICMONCT spent about 22 seconds connecting to the Library Server database:

```
[02-22-2006 18:36:04.254510] SQLDriverConnect(
 szConnStrOut="DSN=ICMNLSDB;UID=ICMCONCT;PWD=********;",
pcbConnStrOut=39
 )
 [02-22-2006 18:36:04.254578]     <--- SQL_SUCCESS   Time elapsed -
+2.283177E+001 seconds
```

- TB774 (the operating system user ID) spent about 0.3 seconds connecting to the Library Server database:

```
[02-22-2006 18:40:54.088985] SQLDriverConnect(
 szConnStrOut="DSN=ICMNLSDB;UID=tb774;PWD=********;",
pcbConnStrOut=36 )
 [02-22-2006 18:40:54.089051]     <--- SQL_SUCCESS   Time elapsed -
+3.239490E-001 seconds
```

We found more performance issues with ICMCONCT in the CLI trace as follows:

- ICMCONCT spent 44 seconds allocating a connection handle:

```
[02-22-2006 18:38:04.181760] SQLAllocHandle(
fHandleType=SQL_HANDLE_DBC,
 hInput=0:1, phOutput=&0012debc )
 [02-22-2006 18:38:04.181819]     ---> Time elapsed - +4.469554E+001
seconds
```

- ICMCONCT spent 28 seconds allocating a statement handle:

```
[02-22-2006 18:37:18.418062] SQLAllocHandle(
 fHandleType=SQL_HANDLE_STMT, hInput=0:1, phOutput=&0012f624 )
 [02-22-2006 18:37:18.418115]     ---> Time elapsed - +2.814248E+001
seconds
```

We thought that the performance issue was related to the connection of the ICMCONCT user ID. The ICMCONCT used much more time connecting to the Library Server than any other operating system user ID. As we mentioned earlier, before the Information Integrator for Content API can call to the stored procedure, the API has to connect successfully to the Library Server database. That explains why the entire Content Manager logon was slow on the Content Manager internal user IDs: because the Content Manager internal user IDs used ICMCONCT to connect to the Library Server database, and the operating system user IDs used their own IDs to connect to the Library Server database.

From there, we worked with the AIX administrator, who did not find any difference between ICMCONCT and any other operating system user ID (such as TB774). Both user IDs belonged to same group and had same setting.

We also provided a Java program to double-check the database connection performance between the client machine and server machine using ICMCONCT and TB774. As expected, ICMCONCT spent more time connecting the Library Server database (53273 versus 2654 msec). Example 12-9 on page 361 shows the detailed test data.

Example 12-9 Testing logon using icmconct user ID and TB774

```
C:\>java DB2LogonLoop icmnlsdb ICMCONCT password 10
 Try to Logon 10 Users
 try to connect to icmnlsdb
 User #0 connected in 22404 ms
 try to connect to icmnlsdb
 User #1 connected in 4247 ms
 try to connect to icmnlsdb
 User #2 connected in 2905 ms
 try to connect to icmnlsdb
 User #3 connected in 2974 ms
 try to connect to icmnlsdb
 User #4 connected in 4117 ms
 try to connect to icmnlsdb
 User #5 connected in 3155 ms
 try to connect to icmnlsdb
 User #6 connected in 3134 ms
 try to connect to icmnlsdb
 User #7 connected in 3025 ms
 try to connect to icmnlsdb
 User #8 connected in 3345 ms
 try to connect to icmnlsdb
 User #9 connected in 3967 ms
 connected 10 Users in 53273 ms

C:\>java DB2LogonLoop icmnlsdb TB774 password 10

 For an other OS user ID,
 Try to Logon 10 Users
 try to connect to icmnlsdb
 User #0 connected in 290 ms
 try to connect to icmnlsdb
 User #1 connected in 261 ms
 try to connect to icmnlsdb
 User #2 connected in 270 ms
 try to connect to icmnlsdb
 User #3 connected in 260 ms
 try to connect to icmnlsdb
 User #4 connected in 261 ms
 try to connect to icmnlsdb
 User #5 connected in 260 ms
 try to connect to icmnlsdb
 User #6 connected in 261 ms
 try to connect to icmnlsdb
```

```
User #7 connected in 270 ms
try to connect to icmnlsdb
User #8 connected in 260 ms
try to connect to icmnlsdb
User #9 connected in 261 ms
connected 10 Users in 2654 ms
```

We also ran the same Java program on the DB2 UDB server (AIX machine). Both user IDs (ICMCONCT and TB774) were very fast to connect to the Library Server database: about 960 msec total.

We applied the same fix pack on the client machine as that on DB2 UDB server. However, the ICMCONCT user ID still had the same DB2 connection performance issue. We checked the network between the client machine and server machine, and did not find any clues.

Finally, we found that if we cataloged the Library Server database using `AUTHENTICATION SERVER` on the client machine, the ICMCONCT connected to Library Server database as fast as any other operating system user IDs, meaning that the entire Content Manager logon performance issue was gone.

About DB2CON

The cmbicmenv.ini file has the database connect information: the Library Server name, user ID, and user password. The default user ID is ICMCONCT.

The cmbicmsvsr.ini file, which has the data source information, contains the list of library servers, the rep type, whether the database is remote or local, and catalog information if it is remote. The cmbicmsrvr.ini file is used by the Content Manager clients to find connection information for the Library Server databases In this file, there is one line:

```
ICMSERVERREPTYPE=DB2
```

Suppose you tried to log on to Content Manager client as SAM. Under this configuration:

1. The Information Integrator for Content API tried to connect to the Library Server database as the user SAM and its password.
2. If the connection failed, the API caught the error message and got the connection ID (ICMCONCT by default) and its password from the cmbicmenv.ini file to connect to the Library Server database.
   - If the connection failed again, the API threw out an error message.
   - If the connection succeeded, the API called ICMLOGON to continue the logon process.

If you know that your users are all Content Manager internal user IDs or LDAP user IDs (and have no DB2 connection privileges), you can change ICMSERVERREPTYPE to DB2CON. This avoids the first step mentioned above, and goes directly to cmbicmenv.ini, and connects to the Library Server database using the user ID and password in the cmbicmenv.ini file, then calls ICMLOGON. It improves the performance of the entire logon process, too.

About the Java test program

We wrote a Java program to test the DB2 UDB connection. Example 12-10 shows the Java program code.

Example 12-10 Java program source code used to test the DB2 UDB connection

```
import java.sql.*;
import java.net.*;
import java.math.*;
import java.io.*;

public class DB2LogonLoop
{
  static {
    try {
      Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public static BufferedReader inbuffer = new BufferedReader(new 
InputStreamReader(System.in));
  public static void main(String argv[]) throws Exception
  {
    long logonTime = 0;
    long logonTimeAll = 0;
    int i = 0;
    int MAX_CONNECTIONS = 10000;
    Connection con[] = new Connection[MAX_CONNECTIONS];

    if (argv.length < 4)
    {
      System.out.println("Usage: " );
      System.out.println("  java DB2LogonLoop <database> <userid> <pw> 
<#Users>" );
      System.exit(0);
    }
    else
    {
      int number = (new Integer(argv[3])).intValue();
      System.out.println("Try to Logon "+number+" Users");

      for (i=0;i<number;i++)
      {
        System.out.println("try to connect to " + argv[0]);
        logonTime=System.currentTimeMillis();
        String url; url = "jdbc:db2:" + argv[0];
        con[i] = DriverManager.getConnection(url, argv[1], argv[2]);
        logonTime=System.currentTimeMillis()-logonTime;
        logonTimeAll=logonTimeAll+logonTime;
        System.out.println("User #"+i+" connected in "+logonTime+"
ms");
      }
    }
    System.out.println("connected "+i+" Users in "+logonTimeAll+" ms
\n");

    System.out.println("press <Enter> to disconnect and Exit");
    inbuffer.readLine();

    for (int j=0;j<i;j++)
    {
      con[j].close();
    }
  }
 }
```

12.3.3 Network/firewall issue

There might be a network or firewall between the Content Manager clients and the server. Sometimes, the network or firewall delays the entire logon process.

In one scenario, some users could log on to their Windows clients quickly, but others had some performance issues when logging on to their Windows clients. These Windows clients are in different physical locations, and the Content Manager server is on the remote site.

We need to ask several questions in this scenario.

User ID issue?

We use ICMADMIN to log on to all of the good and bad Windows clients. A good Windows client means that there is no performance issue when logging on to the system. A bad Windows client means that the user experiences a performance issue when logging on to the Content Manager. We discover that ICMADMIN experienced the same performance issue on bad Windows client machines. We concluded that the issue should not be on the Content Manager server side; it is not related to the specific user IDs.

Same Windows client and DB2 Client software?

We check to see whether the clients are using the same fix pack level for the Windows client and DB2 client software. In this scenario, both the Windows client and DB2 client software are at the same fix pack level for the good and bad clients. This removed the possibility that the software itself was the cause.

Different locations?

We check to see whether the bad Windows clients and good Windows client are on different sites or different networks. We discover that the clients are physically located in different cities and all bad performances were from two cities. Now, we suspect it is a network or firewall issue.

Network/firewall issue?

We work with the network and the firewall administrator to trace the network and firewall. We finally determine that a network caused the performance issue.

12.3.4 LDAP/DNS server issue

If you integrated Content Manager with LDAP, the ICMLOGON would call a user exit to authenticate a user ID and password with the LDAP server. Sometimes, the slow logon can be caused by one of the following reasons:

- The LDAP server responded slowly.
- DNS resolved the host name of LDAP server slowly.
- The network between Content Manager and LDAP server had performance issues.

The Library Server log file can tell us:

- When the user exit sent the authentication request to the LDAP server.
- When the user exit got the reply from the LDAP server.
- Whether the LDAP server authenticated the user ID and password successfully.

Based on the two timestamps mentioned above, we calculate how long the user exit talks to the LDAP server.

In one scenario, a Content Manager system is integrated with eight LDAP servers using a virtual host name. This virtual host name was resolved to eight different LDAP servers in different cities by a DNS server. There is an intermittent logon performance issue. We set the trace level to 15. Some of the extracted information from the ICMSERVER.LOG file is shown in Example 12-11.

Example 12-11 ICMSERVER.LOG with LDAP/DNS server issue

```
ICMPLSLG ICMLOGON                         00764 12/16 17:43:12.286 GMT ;16174312286387
30739437893591544842 TA012005 Input
    UserID            <TA012005>
    DB2UserID         <CMCONCT>
    Language          <   >
    Application       <ICM Connector>
……
ICMPLSLG callUserExit                02024 12/16 17:43:12.287 GMT ;16174312286387
30739437893591544842 TA012005 ICMLogExit input
ICMPLSLG callUserExit                04026 12/16 17:44:29.153 GMT ;16174312286387
30739437893591544842 TA012005 Closing Library...
ICMPLSLG callUserExit                 02066 12/16 17:44:29.154 GMT ;16174312286387
30739437893591544842 TA012005 User Exit returned
    RC           0
    Reason       1
    ExtRC        0
    ExtReason    0
……
ICMPLSLG ICMLOGON                     01825 12/16 17:44:29.155 GMT ;16174312286387
30739437893591544842 TA012005 76871 msec
ICMPLSLG ICMLOGON                    01826 12/16 17:44:29.155 GMT ;16174312286387
30739437893591544842 TA012005 Exit rc=0 reason=0 extrc=0 extreason=0
```

We found that:

- TA012005 user ID used 76.8 seconds to login the system.
- The Content Manager server sent the LDAP authentication request out at 17:43:12 and got the reply at 17:44:29.This action used 76 seconds.

Thus, almost all of time is spent on talking to LDAP server to verify the user ID.

We thought either one of the following situations might happened:

- One of the LDAP servers delayed the entire authentication process.
- The DNS server/network between the Content Manager server and the LDAP server delayed the entire authentication process.

Working with the LDAP server administrator, the LDAP server looked fine. We did an IP trace on the network. Later on, we discovered that there was a configuration issue in the DNS server.

About Microsoft system

If you are using a Microsoft Windows system and you want to trace TCP/IP on Windows platform, Microsoft offers tuning guidance on their Web site. The link is:

http://www.microsoft.com/technet/itsolutions/network/deploy/depovg/tcpip2k.mspx

12.4 Slow search

When a user performs a basic search or an advanced search in Windows client, eClient, or a custom client (using Information Integrator for Content API), the Information Integrator for Content API (both Windows client and eClient are based on Information Integrator for Content API also) gets an XQuery from the clients (or application), and then the API side query engine (within the Information Integrator for Content API package) creates a partial SQL based on the Xquery, and then the Content Manager Library Server finishes the entire SQL with server-side knowledge that only the server can know, such as filling in ACL place-holders.

Thus, the SQL statement sent to the DB2 UDB is partially composed by Information Integrator for Content API, and is partially composed by the Library Server, based on the XQuery. XQuery is the Content Manager query language which is an XML-based query language that conforms to XQuery Path Expressions (XQPE), a subset of W3C XML Query working draft. The query language searches hierarchical item types and locates items quickly and easily.

An XQuery is translated to a SELECT SQL statement in Content Manager. Then, Content Manager runs this SELECT SQL statement in the backend database. The database server tries to pick the best access plan for any given SELECT SQL based on statistics and other intelligence. In some cases for this particular situation, some databases might choose the most optimal plan and others can choose a plan that might result in poor performance. Among reasons for the choice, a less-than optimal plan can result if the statistics are not current for the database or it lacks some index on the tables. Additionally, the same XQuery can be written several ways.

For most of slow search scenarios, it is caused by some SELECT SQL statements which run slowly on the DB2 UDB server. If we confirm from the ICMSERVER.LOG file that some SELECT SQL statements caused the performance issue, we can then tune these SELECT SQL statements and the DB2 UDB server.

There are multiple ways to write an XQuery to get the same result. Although the different ways are functionally equivalent, some of these ways might be better than others in terms of performance since Content Manager can generate a different SQL statement which might has better performance. So, rewriting XQuery can resolve the search performance issue in some cases.

For example, we set the Library Server trace level at 15 and perform a search on Resumes item type in the Windows client. In the log file, you can find the XQuery and the exact SQL statement. Example 12-12 shows the partial ICMSERVER.LOG file. In the example, the ICMSEARCH stored procedure exited with rc=0. The ICMSEARCH used 317 msec (1000 msec= 1 second).

Example 12-12 ICMSERVER.LOG: Slow search

```
ICMPLSQU ICMSEARCH                      00321 06/02 05:49:46.850 GMT ;02054733826423
15126134622165964457 ICMADMIN Entry
......
ICMPLSQU parseInitialParms              00509 06/02 05:49:46.850 GMT ;02054733826423
15126134622165964457 ICMADMIN Search Parms:
   StructID        0
   Max Results     101
   Timeout         0
   Order By        <>
   XQPE String     </'Resumes'[@SEMANTICTYPE BETWEEN 1 AND 2]>
   Num Query Parts 2
   Latest Ver Only 1
   Opt Z           500
   Opt Y           -1
   Opt X           -1
   Num ITypes Sent 0
ICMPLSQU parseInitialParms              00561 06/02 05:49:46.850 GMT ;02054733826423
15126134622165964457 ICMADMIN Return rc=0
ICMPLSQU openQueryCursor                00778 06/02 05:49:46.850 GMT ;02054733826423
15126134622165964457 ICMADMIN Entry
......
ICMPLSQU outputLongStringToLog          02423 06/02 05:49:47.121 GMT ;02054733826423
15126134622165964457 ICMADMIN Entry
ICMPLSQU outputLongStringToLog          02438 06/02 05:49:47.121 GMT ;02054733826423
15126134622165964457 ICMADMIN Query Stmt Length
   SQL Stmt Length: 791
ICMPLSQU outputLongStringToLog          02446 06/02 05:49:47.121 GMT ;02054733826423
15126134622165964457 ICMADMIN Query Stmt Text
   SQL Stmt: <SELECT T.* FROM ( SELECT DISTINCT Resumes_1.ITEMID, Resumes_1.COMPONENTID,
Resumes_1.VERSIONID, 1032 AS COMPONENTTYPEID, 1015 AS ITEMTYPEID FROM ICMUT01032001 Resumes_1
WHERE (Resumes_1.SEMANTICTYPE BETWEEN 1 AND 2) AND ((( ( (EXISTS (SELECT 1 FROM
ICMSTCOMPILEDACL C , ICMSTITVIEWDEFS V WHERE 1015=V.ITEMTYPEID AND V.ITEMTYPEVIEWID IN (1015)
AND (C.ACL = V.ACLCODE OR C.ACL = -1) AND C.UNUM=2 AND C.RPRIV='1'  )) OR (EXISTS (SELECT 1
FROM ICMSTCOMPILEDACL C, ICMSTCOMPILEDPERM P   , ICMSTITVIEWDEFS V WHERE 1015=V.ITEMTYPEID AND
V.ITEMTYPEVIEWID IN (1015) AND (C.ACL = V.ACLCODE OR C.ACL = -1) AND C.UNUM=1 AND C.RPRIV='1'
AND P.PRIVDEFCODE=121 AND P.UNUM=2  )) ))))) T , ICMSTITEMS001001 I WHERE T.ITEMID = I.ITEMID
AND T.VERSIONID = I.VERSIONID OPTIMIZE FOR 500 ROWS FOR READ ONLY WITH UR>
```

```
ICMPLSQU outputLongStringToLog          02453 06/02 05:49:47.121 GMT ;02054733826423
15126134622165964457 ICMADMIN Return rc=0
ICMPLSQU openQueryCursor                02233 06/02 05:49:47.171 GMT ;02054733826423
15126134622165964457 ICMADMIN Query cursor opened successfully for sql string.
ICMPLSQU openQueryCursor                02250 06/02 05:49:47.171 GMT ;02054733826423
15126134622165964457 ICMADMIN Return rc=0
......
ICMPLSQU ICMSEARCH                      00416 06/02 05:49:47.171 GMT ;02054733826423
15126134622165964457 ICMADMIN   317 msec
ICMPLSQU ICMSEARCH                      00417 06/02 05:49:47.171 GMT ;02054733826423
15126134622165964457 ICMADMIN Exit rc=0 reason=0 extrc=0 extreason=0
```

12.4.1 SQL tuning

We can tune the SQL that is generated based on the XQuery to resolve the performance issue.

For example, in one scenario, there was a search performance issue on Content Manager. Setting trace level to 15, from the ICMSERVER.LOG file, we found the performance information (46405 msec), as shown in Example 12-13.

Example 12-13 ICMSERVER.LOG: Slow search sample

```
ICMPLSGL ICMGETLINKEDITEMS                00428 10/05 21:04:53.776
?05210353323256
   Statement length 895
   Buffer size      5000
   # UserIDs        2
   Statement        <
 SELECT
   LINKS.SOURCEITEMID OTHERITEMID,
   ITEMS.ITEMTYPEID, ITEMS.CHANGED,
   LINKS.LINKTYPE, LINKS.LINKITEMID,
   ITEMS.COMPONENTTYPEID, ITEMS.VERSIONID, ITEMS.COMPONENTID
 FROM CML.ICMSTLINKS001001 LINKS,
   CML.ICMSTITEMS001001 ITEMS,
   CML.ICMSTITEMTYPEDEFS ITEMTYPES
 WHERE LINKS.TARGETITEMID= ? AND
   LINKS.SOURCEITEMID=ITEMS.ITEMID AND
   ITEMS.ITEMTYPEID = ITEMTYPES.ITEMTYPEID AND
(
 (ITEMTYPES.ITEMLEVELACLFLAG = 0 AND
  (EXISTS (SELECT 1 FROM CML.ICMSTCOMPILEDACL AS ICMC,
    CML.ICMSTITVIEWDEFS AS VIEW
    WHERE ITEMS.ITEMTYPEID = VIEW.ITEMTYPEID AND
    ICMC.ACLCODE = VIEW.ACLCODE AND
    ICMC.USERID IN ('ICMPUBLC',?) AND ICMC.PRIVDEFCODE=121))
 )
 OR
 (ITEMTYPES.ITEMLEVELACLFLAG = 1 AND
  (EXISTS (SELECT 1 FROM CML.ICMSTCOMPILEDACL AS ICMC
    WHERE ICMC.ACLCODE = ITEMS.ACLCODE AND
    ICMC.USERID IN ('ICMPUBLC',?) AND ICMC.PRIVDEFCODE=121))
 )
)>

……
ICMPLSGL ICMGETLINKEDITEMS                00510 10/05 21:05:40.177
?05210353323256 46405 msec
```

We made up the values for the three parameter markers (?) in the SQL query and re-ran this SQL from the DB2 command line. This SQL query ran more than 40 seconds and it took 50% CPU out of a four-way Sun system where the DB2 UDB server was running.

We ran **reorg**, **runstat**, and **rebind** on the Library Server database. After that, there was still a performance issue on the SELECT SQL.

We ran a DB2 UDB explain on the above SQL (refer to "DB2 Explain" on page 328 for more information about DB2 explain):

1. Run EXPLAIN.DDL from sqllib\misc.
2. Start the query with `explain plan for` and save the query as a file called `q1.sql`.
3. Run **db2 -tvf q1.sql**
4. Run **db2exfmt -d <DBNAME> -u <UserName> <PWD> -o <OUT FILE> -e <SCHEMA>**

We got the explain data and checked it. Based on this information and our conclusion, we recommended more indexes on Content Manager tables. Later on, after the indexes were added, the performance was improved.

We also found that changing AVG_APPL from 5 to 1 improved the performance on this SELECT SQL.

Some indexes we recommended in this case are implemented in the current fix pack and Content Manager Version 8.3.

12.4.2 Rewrite XQuery

If you write the XQuery yourself and it is causing performance problems, you can rewrite the XQuery. There are many technical notes about how to rewrite the XQuery. You can find them in the IBM Content Manager support Web site:

http://www.ibm.com/software/data/cm/cmgr/mp/support.html

In this section, we describe some ideas based on some of these technical notes.

The generated SQL can vary depending on your exact original XQuery string. Small differences in the original XQuery can yield differences in the SQL that can cause the database to choose a different access plan.

In one scenario, users experienced poor performance when they ran a query to return an item that included conditions about both the root component attributes and child component attribute. Aside from maintaining the database, such as making sure the database statistics were current for the database, we also tried to rewrite the XQuery.

For example, the original query was:

```
/Journal[@Title LIKE "XML%" AND ./Journal_Article/@Author = "Kevin"]
```

That query exhibited poor performance when there was a large amount of data in the system. We tried rewriting it as follows:

```
/Journal[@Title LIKE "XML%" AND(@ITEMID =
/Journal/Journal_Article/[@Author = "Kevin"]/@ITEMID)]
```

Although the results of the query will be the same, the SQL generated for this query differs from the SQL generated to perform the original query (assuming that versioning is turned off). This causes the database to choose a different plan for execution.

You can do the same thing when versioning is used by adding conditions to the query. For versioning, use this query:

```
/Journal[@Title LIKE "XML%" AND(@ITEMID =
/Journal/Journal_Article/[@Author = "Kevin"]/@ITEMID) AND (@VERSIONID =
/Journal/Journal_Article/[@Author =  "Kevin"]/@VERSIONID)]
```

In general, avoiding redundant traversals is one of the ways to improve query performance greatly.

For example, consider a component-type hierarchy within an item type, where the Person component type is the parent of another component type, Address. Assume that the query needs to find addresses of single or married persons in California whose annual income is more than $100,000.

One way to write this query would be:

```
//Address[(../@MaritalStatus = 'Married' or ../@MaritalStatus =
'Single')
and ../@Income > 100000 and @State = 'California']
```

However, this can be quite inefficient because of repeated traversals to the parent component (Person).

Because the query involves a search based on several attributes (@MaritalStatus, @Income) in the parent component type, Person, it is a good idea to search for the parent component first then use an additional step to get to the Address component.

A more efficient way to write the query would be:

```
//Person[(@MaritalStatus = 'Married' or @MaritalStatus = 'Single')
and @Income > 100000]/Address[@State = 'California']
```

Writing the query in the latter form helps the query processor avoid several redundant joins, which can result in improved performance.

Again, access the Content Manager support Web site for more tips about how to rewrite an XQuery.

This type of rewrite helps in some cases. However, given the complexity in the database itself and variation in data, it might not perform better in all cases. Always test it first before implementing the changes.

12.4.3 Maintain Content Manager database periodically

As we have mentioned many times in this book, it is important to maintain the Content Manager databases periodically.

For example, in one scenario, it took a Content Manager system more than 30 seconds to return a hit list of 16 documents. The system also became slower when there were more documents being imported into the Content Manager system.

In this scenario, the performance issue should be in the Library Server database because getting a hit list involves only the Library Server. We checked to see how often the Library Server database was maintained, using tools such as **`reorg`**, **`runstats`**, and **`rebind`**. We found that it was not maintained periodically. After running **`reorg`**, **`runstats`**, and **`rebind`** on the Library Server database, the retrieval performance issue was gone.

Unexpected performance issues can occur if you create the Content Manager Version 8.2 Library Server and Resource Manager databases using the icmcreatelsdb script, icmcreatermdb script, or batch files. Although the database is created successfully, the DB2 optimization and a **`rebind`** of the database packages does not take place automatically. Thus, after a successful re-creation of the database, you should perform a DB2 **`runstats`** and **`rebind`**.

12.5 Slow import

When a user imports one or more documents into Content Manager in a Windows client, eClient, or custom client (using Information Integrator for Content API), the client or the application calls several stored procedures to add the metadata (attribute values) into the Library Server database, and stores the real documents in Resource Manager via HTTP.

The import process might involve some search operations. For example, when you want to import a document into a folder, the system needs to search for the folder first. If the folder does not exist, the import program would create the folder before importing this document into Content Manager.

Based on our experience, most slow imports are caused by slow search (within the entire import process) or slow response from Resource manager. If the slow response is from Resource Manager, then tuning WebSphere Application Server can resolve most of the issues. If the slow response is from the search, you can follow the previous section to trace down the issue.

In one scenario, users were using a third-party application to import documents into the Content Manager server. The performance was not good and the db2sysc used most of the memory. We worked with the DB2 UDB team. We suspected that there were some table scans. We used the event monitor and examined a statement event. See Example 12-14.

Example 12-14 Slow import example

```
Appl Handle: 41
   Appl Id: 0A40C88D.8812.040630125126
   Appl Seq number: 0001
   Record is the result of a flush: FALSE
   -------------------------------------------
   Type     : Dynamic
   Operation: Describe
   Section  : 6
   Creator  : ICMADMIN
   Package  : ICMPLSQU
   Cursor   : QUERY_CURSOR1
```

```
  Cursor was blocking: FALSE
  Text     : SELECT T.* FROM (SELECT DISTINCT CptEntreprise_1.ITEMID,
CptEntreprise_1.COMPONENTID, CptEntreprise_1.VERSIONID, 1008 AS
COMPONENTTYPEID, 1004 AS ITEMTYPEID FROM ICMADMIN.ICMUT01008001
CptEntreprise_1 WHERE (((((CptEntreprise_1.SEMANTICTYPE = 2) AND (NOT
EXISTS (SELECT DISTINCT ItemVersions_10.ITEMID,
ItemVersions_10.COMPONENTID, ItemVersions_10.VERSIONID,
ItemVersions_10.COMPONENTTYPEID, ItemVersions_10.ITEMTYPEID FROM
ICMADMIN.ICMSTLINKS001001 Links_4, ICMADMIN.ICMSTNLSKEYWORDS
NLSKeywords_6, ICMADMIN.ICMSVALLITEM001001 ItemVersions_10 WHERE
(((CptEntreprise_1.ITEMID = Links_4.TARGETITEMID) AND
((((Links_4.LINKTYPE = NLSKeywords_6.KEYWORDCODE) AND (4 =
NLSKeywords_6.KEYWORDCLASS)) AND ('ENU' = NLSKeywords_6.LANGUAGECODE))
AND (NLSKeywords_6.KEYWORDNAME = 'DKFolder'))) AND (Links_4.SOURCEITEMID
= ItemVersions_10.ITEMID)) AND (( (ItemVersions_10.ITEMTYPEID IN
(300,301,302,303,304,400,1000,1001,1002,1004) AND  ((EXISTS (SELECT 1
FROM ICMADMIN.ICMSTCOMPILEDACL AS C3, ICMADMIN.ICMSTITVIEWDEFS AS V3
WHERE V3.ITEMTYPEID=ItemVersions_10.ITEMTYPEID AND V3.ITEMTYPEVIEWID IN
(300,301,302,303,304,400,1000,1001,1002,1004) AND C3.ACLCODE=V3.ACLCODE
AND C3.USERID='ICMADMIN' AND C3.PRIVDEFCODE=121)) OR (EXISTS (SELECT 1
FROM ICMADMIN.ICMSTCOMPILEDACL AS C4, ICMADMIN.ICMSTITVIEWDEFS AS V4,
ICMADMIN.ICMSTCOMPILEDPERM AS P4 WHERE
V4.ITEMTYPEID=ItemVersions_10.ITEMTYPEID AND V4.ITEMTYPEVIEWID IN
(300,301,302,303,304,400,1000,1001,1002,1004) AND C4.ACLCODE=V4.ACLCODE
AND C4.USERKIND=2 AND C4.PRIVDEFCODE=121 AND
C4.PRIVDEFCODE=P4.PRIVDEFCODE AND P4.USERID='ICMADMIN')))) OR
(ItemVersions_10.ITEMTYPEID IN (200,201,202) AND  ((EXISTS (SELECT 1
FROM ICMADMIN.ICMSTCOMPILEDACL AS C5 WHERE ItemVersions_10.ACLCODE =
C5.ACLCODE AND C5.USERID='ICMADMIN' AND C5.PRIVDEFCODE=121)) OR (EXISTS
(SELECT 1 FROM ICMADMIN.ICMSTCOMPILEDACL AS C6,
ICMADMIN.ICMSTCOMPILEDPERM AS P6 WHERE ItemVersions_10.ACLCODE =
C6.ACLCODE AND C6.USERKIND=2 AND C6.PRIVDEFCODE=121 AND
C6.PRIVDEFCODE=P6.PRIVDEFCODE AND P6.USERID='ICMADMIN'))))))))) AND
(UCASE(CptEntreprise_1.ATTR0000001004) = UCASE('03'))) AND
(UCASE(CptEntreprise_1.ATTR0000001005) = UCASE('021743001014'))) AND
(UCASE(CptEntreprise_1.ATTR0000001007) = UCASE('32976503600044'))) AND
((( ((EXISTS (SELECT 1 FROM ICMADMIN.ICMSTCOMPILEDACL AS C3,
ICMADMIN.ICMSTITVIEWDEFS AS V3 WHERE V3.ITEMTYPEID=1004 AND
V3.ITEMTYPEVIEWID IN (1004) AND C3.ACLCODE=V3.ACLCODE AND
C3.USERID='ICMADMIN' AND C3.PRIVDEFCODE=121)) OR (EXISTS (SELECT 1 FROM
ICMADMIN.ICMSTCOMPILEDACL AS C4, ICMADMIN.ICMSTITVIEWDEFS AS V4,
ICMADMIN.ICMSTCOMPILEDPERM AS P4 WHERE V4.ITEMTYPEID=1004 AND
V4.ITEMTYPEVIEWID IN (1004) AND C4.ACLCODE=V4.ACLCODE AND C4.USERKIND=2
AND C4.PRIVDEFCODE=121 AND C4.PRIVDEFCODE=P4.PRIVDEFCODE AND
P4.USERID='ICMADMIN'))))))) AS T  , ICMADMIN.ICMSTITEMS001001 I WHERE
T.ITEMID = I.ITEMID AND T.VERSIONID = I.VERSIONID OPTIMIZE FOR 500 ROWS
  -------------------------------------------
  Start Time: 30/06/2004 15:04:54.043829
  Stop Time:  30/06/2004 15:05:05.267976
```

```
   Exec Time:  11.224147 seconds
   Number of Agents created: 1
   User CPU:   11.234375 seconds
   System CPU: 0.000000 seconds
   Fetch Count: 1
   Sorts: 0
   Total sort time: 0
   Sort overflows: 0
   Rows read: 1241387
   Rows written: 0
   Internal rows deleted: 0
   Internal rows updated: 0
   Internal rows inserted: 0
   SQLCA:
    sqlcode: 0
    sqlstate: 00000
 ......
```

According to the information in the example, a single Content Manager operation read up to 1,241,387 rows in a table. Later on, we added some indexes and the performance issue was resolved. As to the case above, you can also set the trace level to 15 to find this problematic SQL statement and tune it. We showed another way to trace down the SQL statement here.

In another scenario, users used a third-party application to import documents to the Content Manager system and the import was very slow. We analyzed the ICMSERVER.LOG file (with trace level set to 15) and found that the problem was the result of a complex SQL query. (Refer 12.4, "Slow search" on page 367.) The SQL used the view ICMSVALLITEM001001, which was defined as a union of ICMSTITEMS001001 and ICMSTITEMVER001001. The use of the view in the SQL resulted in the predicate being sent to the DB2 optimizer for evaluation. The optimizer chose to run table scans against both the ICMSTITEMS001001 and ICMSTITEMVER001001 tables even with indexes present. This is an internal Content Manager issue. This performance issue was fixed in Content Manager Version 8.2 Fix Pack 8 and later. Although the problem we show here was caused by Content Manager itself, this case study should be useful in troubleshooting performance problems in your custom application or configuration.

12.6 Slow migrator

When the migrator starts periodically, it runs queries against the Resource Manager database to find a batch of documents that are based on obj_actiondate and volume information in the RMOBJECTS table. It then migrates these documents to the next point in the migration path. In most cases, they go from a physical disk (where they start when they first arrive) to Tivoli Storage Manager. When a document is migrated from hard disk to TSM, the RMOBJECTS table is updated so that the document's obj_actiondate is changed and its obj_volumeid is changed to the volumeid for the proper TSM Management class.

A slow migrator can be resolved by using new TSM API client software and tuning the Resource Manager tables. Also, Content Manager Version 8.3 improved the performance of the migrator.

12.6.1 Tivoli Storage Manager

In one scenario, the Content Manager system was running on the Windows platform, and the migrator was moving data from hard disk to TSM. However, only 30,000 documents were migrated every day, and this was less than the business requirement for migrating documents to TSM. We found that the Content Manager system was using TSM API 5.1.5 on the Resource Manager server. Based on our experience, TSM API 5.3 provided better performance, so we upgraded TSM API Client to V5.3. Now, the migrator can fulfill the business requirement and migrate about 300,000 documents per day to the TSM server.

In another scenario, there was a performance issue when backing up data from hard disk to TSM. We asked the TSM administrator to see whether the TSM server could be tuned. Configuration changes significantly improved performance.

In yet another scenario, Content Manager data was stored on TSM-managed optical platters. During peak usage hours, storage and retrieval of objects was taking too long. A simple way to improve performance of both store and retrieve operations was needed. The easiest way to resolve this issue was to add sufficient disk space to Tivoli Storage Manager, and use Tivoli Storage Manager commands to move the data from the optical platters to disk. No changes in Content Manager were necessary when doing this.

12.6.2 Tune the Resource Manager database

In one scenario, there were millions of rows in the RMOBJECTS table, the migrator used almost 100% of CPU, and the overall system performance was very bad. DB2 UDB snapshot showed that the system took about 700 seconds to run each following SQL in one case. ("Snapshot monitoring" on page 296 describes how to get a DB2 snapshot).

The three SQL SELECT statements below are all legitimate candidate selection queries generated by the Resource Manager migrator for each volume during the migration. The first one is to select candidates for migration, the second is to select candidates for the regular delete, and the third is for the threshold delete:

```
SELECT OBJ_ITEMID,OBJ_VERSION,OBJ_LIBRARYID,OBJ_COLLECTIONID,
OBJ_STGCLASSID,OBJ_MGTCLASSID
FROM RMADMIN.RMOBJECTS
WHERE OBJ_VOLUMEID = 1 AND
OBJ_ACTIONDATE <= CURRENT DATE AND
(OBJ_STATUS = 'A' OR OBJ_STATUS='B') AND
OBJ_ITEMID>' ' AND
OBJ_LIBRARYID=1
ORDER BY OBJ_ITEMID, OBJ_STGCLASSID;

SELECT OBJ_ITEMID,OBJ_VERSION,OBJ_LIBRARYID,OBJ_COLLECTIONID,OBJ_STATUS
FROM RMADMIN.RMOBJECTS
WHERE OBJ_VOLUMEID = 1 AND (OBJ_STATUS= 'D' OR OBJ_STATUS='G')
ORDER BY OBJ_ITEMID;

SELECT OBJ_ITEMID,OBJ_VERSION,OBJ_LIBRARYID,OBJ_COLLECTIONID,
OBJ_STGCLASSID,OBJ_MGTCLASSID
FROM RMADMIN.RMOBJECTS
WHERE OBJ_VOLUMEID = 1 AND
OBJ_ACTIONDATE <> '12/31/9999' AND
OBJ_STATUS = 'D' OR OBJ_STATUS='G')
ORDER BY OBJ_ITEMID DESC;
```

We performed a DB2 explain on these SQL statements. Based on the explain data, we created some additional indexes and ran runstats. Then the DB2 UDB optimizer picked a good access plan that completed the rmobjects SELECT statements in 0.5-0.6 seconds.

Here are the indexes we created and the runstats we ran:

```
CREATE  INDEX RMADMIN.IDX_MIGRATOR ON RMADMIN.RMOBJECTS
( OBJ_VOLUMEID ASC, OBJ_STATUS ASC, OBJ_ACTIONDATE ASC, OBJ_LIBRARYID
ASC )
PCTFREE 10 MINPCTUSED 10
COLLECT  DETAILED  STATISTICS ;

CREATE INDEX "RMADMIN "."IDX_MIGRATOR2" ON "RMADMIN "."RMOBJECTS"
                ("OBJ_VOLUMEID" ASC,
                 "OBJ_STATUS" ASC)
                 COLLECT DETAILED STATISTICS;

RUNSTATS ON TABLE RMADMIN.RMOBJECTS ON ALL COLUMNS WITH DISTRIBUTION ON
ALL
```

```
COLUMNS DEFAULT NUM_FREQVALUES 100 NUM_QUANTILES 200 AND DETAILED
INDEXES
ALL ALLOW WRITE ACCESS ;
```

12.6.3 Performance improvement in Content Manager V8.3

In Content Manager Version 8.2, we have one thread for one volume. Starting in Content Manager Version 8.3, we have a thread pool for volumes. That can improve migrator performance greatly.

If you need to migrate a huge number of objects from one device to another and you experience the performance issue, you might consider the V8.3 upgrade.

12.7 Slow administration in the sys admin client

The system administration client provides the tools that you need to set up and manage your Content Manager system. When you work with the system administration client (such as create, change, and remove an item type; create and remove a user ID), you might experience a performance issue.

12.7.1 Tune database

In one scenario, a system administrator spent more than 40 minutes removing an item type in the system administration client.

We suspected that some SELECT or DELETE SQL statement used most of this time. We got the item type ID (1023) and component type ID (1042) based on the name of the item type, and ran SELECT to test the performance:

```
SELECT * FROM ICMSTCOMPDEFS WHERE itemtypeid = 1023
SELECT COUNT(*) FROM ICMSTITEMS001001 WHERE COMPONENTTYPEID = 1042
```

These SQL statements ran for a long time on the machine. We asked the system administrator to maintain the Library Server database. After maintaining the database using **`reorg`**, **`runstat`**, and **`rebind`**, the system functioned much faster. Our recommendation also included performing this type of task (in this case, removing item type) during off-peak time.

If the maintenance on Library Server database and performing it during off-peak time were not working, we would analyze the ICMSERVER.LOG file, with trace level set to 15, to find out which SQL statements the system was running. From there, we would tune these statements. We might need to tune the entire DB2 UDB server. Some SQL statements are static SQL that might be not in an ICMSERVER.LOG file. In such cases, you should run the event monitor or ask the IBM support team to help you find these statements. Try to maintain your Library Server database first — that might resolve the issue.

12.7.2 Apply fix pack

There are a lot of performance improvements for the system administration client in the latest fix packs and Content Manager Version 8.3.

In one scenario, the opening properties of a single user took about 30 minutes on a system with hundreds of users and ACLs. Getting a list of users and a list of ACLs took about 10 minutes. We upgraded the Content Manager to Version 8.3, and the performance improved.

12.8 Common rules

There are some common rules to avoid performance issues.

12.8.1 Avoiding the next performance crisis

Actions you might take to avoid the next performance crisis include:

- Use a test system to check software configuration and tuning options.
- Use a better regression test after loading new versions, features, or bug fixes.
- For every change you institute, plan a way to quickly remove the change from the system.
- Use better change-control procedures.
- Establish a baseline measurement.
- Acquire more training in skills required to run the system.

12.8.2 Using the default trace level

Although you can set the API log's trace setting higher than ERROR (the default), doing so can greatly reduce the overall performance because the system is providing extensive information to the log file. Therefore, when you are not debugging errors, set the Information Integrator for Content API log (dklog.log) trace level to ERROR.

Although you can set the Library Server stored procedure log's trace setting higher than zero, doing so can also greatly reduce overall performance while the system is providing extensive information to the log file. Therefore, when you are not debugging errors, set the Library Server stored procedure log trace level to zero, which is the default setting.

It is the same for Resource Manager: Set the trace level to DEBUG (trace full) only if you want to debug the issue in the Resource Manager server.

Often there is a requirement to generate logs and traces on a production system at the failure point. IBM DB2 Content Manager Version 8.3 has a new feature that generates tracing for a single user. In most cases, single-user tracing is enough to get the details of the problem. Single-user tracing is enabled through the system administration user interface. When it is enabled, other users get default ERROR logging. It goes into effect at the next logon of the Content Manager user ID.

12.8.3 Applying fix packs

Performance improvements are often implemented in Content Manager software fix packs. Check the fix pack readme document to find out whether it resolves your performance issue.

For example, in one scenario, a Content Manager client application search on “Documents and Folders” took much longer to return the search results list than a search directly on either the “Documents Only” or “Folders Only” level. We later found that a SQL statement required additional DB2 indexes, so we added three additional database indexes on the Library Server tables in the new fix pack.

12.8.4 Information Integrator for Content API programming

Here is a technical note from the IBM Content Manager support Web site about the general rule of Information Integrator for Content API programming. This site has a lot of similar technical notes, so we encourage you to visit the site and view them for more information.

Retrieving items that have links

You can streamline the process of retrieving items that have links by remembering the following points:

- Do not retrieve links until necessary.
- Retrieve only the necessary direction of links.

Link retrieval spends time creating DKDDO objects for each retrieved link. A few things can be done to avoid all or some of this processing time.

Do not retrieve links until necessary

In some cases, a link might not be needed at all or not needed until later when it could be retrieved individually. Avoid retrieving links for a collection of items if only a few items will actually need the links to be retrieved. You can accomplish this by not using retrieval options that include links, such as CONTENT_ITEMTREE, CONTENT_LINKS_OUTBOUND, and CONTENT_LINKS_INBOUND.

Retrieve only the necessary direction of links

You can go back and retrieve links later if necessary. If you need only outbound links, retrieve them using the CONTENT_LINKS_OUTBOUND retrieval option and do not retrieve the inbound links; use the CONTENT_LINKS_INBOUND retrieval option if you need only inbound links. In either case, do not use the CONTENT_ITEMTREE option, which includes both inbound and outbound links. If you do not need any links, do not use any of the options that include links as stated in the section above.

12.9 Utilities to analyze log files

To help you troubleshoot, we present some utilities (originally created by the Content Manager development team) to analyze the Content Manager log files and help you get meaningful data for troubleshooting the system:

- ICMLogSplit
- ICMLSPerf
- ICMLogDeltaTime
- ICMEventTable

These tools are executable files that can be run from a Windows workstation. In addition, for some of the tools, we provide Java source codes and shell scripts that you can run or modify based on your system requirements. Refer to Appendix B, "Additional material" on page 443 for information about downloading these tools and source files.

ICMLogSplit

ICMLogSplit splits a Content Manager server log file (by default, it is icmserver.log) into separate files for each user session. It finds the user information field in the log entries and uses this information as the file name.

ICMLSPerf

ICMLSPerf analyzes the Content Manager server log file (by default, it is icmserver.log) and summarizes the stored procedures execution information, including stored procedure name, count, Library Server time elapsed within the stored procedure, and application time elapsed within the stored procedure. This information is useful in a single-user environment. Figure 12-2 on page 382 shows sample icmlsperf output.

ICMLogDeltaTime

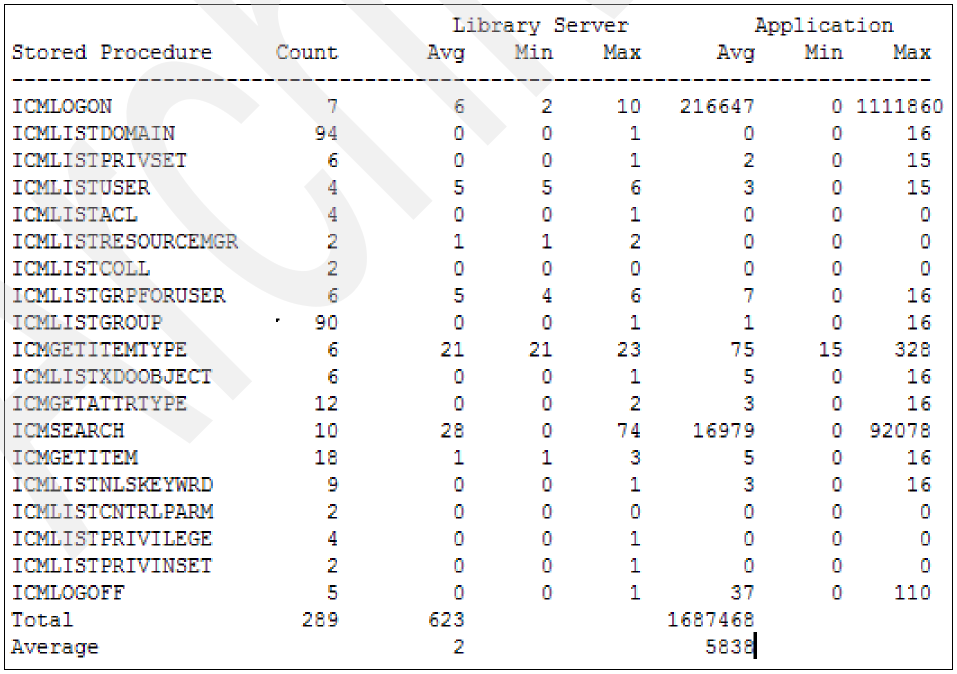
ICMLogDeltaTime reports the elapsed time between the trace records in a Content Manager server log file (by default, it is icmserver.log). The best way to use this is on a server log containing only performance data (tracelevel=-8) or use **grep** to find records containing msec.

ICMEventTable

ICMEventTable uses an event monitor log file to produce an output file and a summary file.

The output file has data related to the buffer pool read on temporary data, indexes, temporary indexes, type of read, operation done, return code, and statement being run.

The summary file provides information such as the package accessed, section, count, total time to run the SQL, average time to run the SQL, total Content Manager time, average Content Manager time, maximum reads, average number of rows read, written, and fetched, average buffer pool reads, and the statement that was run. It also provides summary details such as the elapsed time of the test, number of stored procedure calls, SQL statements run, prepare statements, SQL without CLOSE and PREP, transaction time, lock wait time, set statements, commits, and rollbacks.



```
                                Library Server          Application
Stored Procedure     Count      Avg    Min    Max      Avg    Min     Max
--------------------------------------------------------------------------
ICMLOGON                 7        6      2     10   216647      0 1111860
ICMLISTDOMAIN           94        0      0      1        0      0      16
ICMLISTPRIVSET           6        0      0      1        2      0      15
ICMLISTUSER              4        5      5      6        3      0      15
ICMLISTACL               4        0      0      1        0      0       0
ICMLISTRESOURCEMGR       2        1      1      2        0      0       0
ICMLISTCOLL              2        0      0      0        0      0       0
ICMLISTGRPFORUSER        6        5      4      6        7      0      16
ICMLISTGROUP            90        0      0      1        1      0      16
ICMGETITEMTYPE           6       21     21     23       75     15     328
ICMLISTXDOOBJECT         6        0      0      1        5      0      16
ICMGETATTRTYPE          12        0      0      2        3      0      16
ICMSEARCH               10       28      0     74    16979      0   92078
ICMGETITEM              18        1      1      3        5      0      16
ICMLISTNLSKEYWRD         9        0      0      1        3      0      16
ICMLISTCNTRLPARM         2        0      0      0        0      0       0
ICMLISTPRIVILEGE         4        0      0      1        0      0       0
ICMLISTPRIVINSET         2        0      0      1        0      0       0
ICMLOGOFF                5        0      0      1       37      0     110
Total                  289      623               1687468
Average                           2                  5838
```

Figure 12-2 ICMlsperf output

You can also look at the ICMSERVER.LOG file as is to identify problems. Chapter 13, “Case study” on page 385 presents a mini-performance tuning case study with a Document Manager system that uses Content Manager as the back-end content repository system.

13

Case study

This chapter extends the troubleshooting discussion from the previous chapter by using Document Manager performance tuning as a case study to further show how you can begin to understand what your system is doing and start troubleshooting Content Manager performance problems.

13.1 Document Manager overview

IBM DB2 Document Manager addresses the origination, approval, distribution, and revision of unstructured information. In the simplest terms, Document Manager enables electronic files to be reused or repurposed by multiple individuals.

Document Manager applies a discipline called *library sciences* to catalog corporate information, which could be in the form of hard copy (physical documents), electronic files generated by desktop applications, or just metadata or XML information created internally or externally. In any form, Document Manager is an application that enables users to index and file corporate information in a uniform and centralized way.

Document Manager is a *client* for Content Manager hosted by Microsoft Internet Explorer. A Document Manager system is built on a three-tiered, scalable computing model. You can deploy the system's components as needed and distribute them across multiple servers or clients.

Document Manager is built on a three-tier model:

- Document Manager clients tier
- Document Manager server and services tier
- Content repository tier

In this case study, the content repository for the Document Manager system is Content Manager.

13.2 A search scenario

This case study illustrates a way to identify potential problems in your custom application and show how you can use cache setting to improve the performance of a Document Manager - Content Manager system.

We start with a plain system that uses Document Manager as a client and Content Manager as the content repository. Document Manager cache is not configured initially. When users use the system, they complain that the logging time and system search response time seem to be slow.

To identify the problem, we turn on Content Manager Library Server performance trace. See 13.3, "Enabling Library Server trace" on page 389 for details.

We use `echo` to insert text into the icmserver.log, so that we will know at every step what actions we, as user, performed on the system, and what Content Manager operations have been triggered due to that action.

Our exercise consists of the following steps:

1. Open a command prompt window and go to the location where icmserver.log resides.
2. Launch the Document Manager client by typing:

   ```
   echo Launching the Document Manager client.>icmserver.log
   ```

3. Log in to Document Manager as John Doe by typing:

   ```
   echo The Document Manager client is now launched.>>icmserver.log
   echo Logging into Document Manager as John Doe.>>icmserver.log
   ```

4. Perform a simple search. Type:

   ```
   echo Log in completed.>>icmserver.log
   echo Perform a search.>>icmserver.log
   ```

5. Perform the same search:

   ```
   echo Search is done.>>icmserver.log
   echo Search again.>>icmserver.log
   ```

6. Type:

   ```
   echo 2nd search is done.>>icmserver.log
   ```

We extract part of the icmserver.log file when we perform a search in Example 13-1. For simplification, we reduced some of the white spaces, and use an ellipsis (...) to represent the repetitive numbers for the log entries.

Example 13-1 icmserver.log: log of searches in a Document Manager desktop with Content Manager

```
Perform a search.
ICMPLSLG ICMLOGON         00672 04/10 18:17:34.609 GMT ; ... ? JOHN.DOE
   Library server code built    <  Mar  1 2006 17:20:53  >
ICMPLSLG ICMLOGON         01907 04/10 18:17:34.609 GMT ; ... ? JOHN.DOE     9 msec
ICMPLSGA ICMGETATTRTYPE   00412 04/10 18:17:34.609 GMT ; ... JOHN.DOE     1 msec
ICMPLSGT ICMGETITEMTYPE   00919 04/10 18:17:34.640 GMT ; ... JOHN.DOE    21 msec
ICMPLSLX ICMLISTXDOOBJECT 00335 04/10 18:17:34.640 GMT ; ... JOHN.DOE     1 msec
ICMPLSGA ICMGETATTRTYPE   00412 04/10 18:17:34.640 GMT ; ... JOHN.DOE     1 msec
ICMPLSQU ICMSEARCH        00482 04/10 18:17:34.796 GMT ; ... JOHN.DOE    65 msec
ICMPLSGI ICMGETITEM       04346 04/10 18:17:34.796 GMT ; ... JOHN.DOE     2 msec
ICMPLSLK ICMLISTNLSKEYWRD 00860 04/10 18:17:34.796 GMT ; ... JOHN.DOE     0 msec
ICMPLSLF ICMLOGOFF        00200 04/10 18:17:34.906 GMT ; ...  ? JOHN.DOE     0 msec
Search is done.
Search again.
ICMPLSLG ICMLOGON         00672 04/10 18:19:19.390 GMT ; ... ? JOHN.DOE
   Library server code built    <  Mar  1 2006 17:20:53  >
ICMPLSLG ICMLOGON         01907 04/10 18:19:19.390 GMT ; ... ? JOHN.DOE    10 msec
ICMPLSGA ICMGETATTRTYPE   00412 04/10 18:19:19.406 GMT ; ... JOHN.DOE     2 msec
```

```
ICMPLSGT ICMGETITEMTYPE    00919 04/10 18:19:19.421 GMT ; ... JOHN.DOE    22 msec
ICMPLSLX ICMLISTXDOOBJECT  00335 04/10 18:19:19.437 GMT ; ... JOHN.DOE     0 msec
ICMPLSGA ICMGETATTRTYPE    00412 04/10 18:19:19.437 GMT ; ... JOHN.DOE     0 msec
ICMPLSQU ICMSEARCH         00482 04/10 18:19:19.578 GMT ; ... JOHN.DOE    65 msec
ICMPLSGI ICMGETITEM        04346 04/10 18:19:19.578 GMT ; ... JOHN.DOE     2 msec
ICMPLSLK ICMLISTNLSKEYWRD  00860 04/10 18:19:19.593 GMT ; ... JOHN.DOE     0 msec
ICMPLSLF ICMLOGOFF         00200 04/10 18:19:19.640 GMT ; ... ? JOHN.DOE     0 msec
2nd search is done.
```

We observed several interesting points in the log:

- When we perform a search, the ICMLOGON operation is called. It does not seem right for ICMLOGON to be called because we already logged into the system earlier. If you see this in your application, you should look into it and see why the operation is called. Is it a redundancy somewhere?
- At the end of the search, the ICMLOGOFF operation is called. Again, it does not seem right for ICMLOGOFF to be called because we are performing a search and not logging in and out of the system. If you see this in your application, you should look into it and see why this operation is called. Is it necessary? If not, you should remove this extra work from your application.
- Although we perform the exact same search twice, the system is making the same number of calls in both instances. For an efficient system, you might want to work on the application so that you save some calls to the system the second time a user performs the same search (or other action).
- There are also some ICMGETATTRTYPE and ICMGETITEMTYPE calls. Depending on your application, you might look into whether these calls are necessary.

Different applications make different calls to the system. The log can help you understand what calls are really being made to Content Manager Library Server. From it, you can identify any area that might cause a performance problem.

In this case study, we decided to turn on the mid-tier cache and perhaps save the extra login and logout calls. After we turn on the mid-tier cache and make some configuration changes, we perform the same search operation twice. For more information about Document Manager cache, refer to 13.4, “Configuring Document Manager cache” on page 391.

Example 13-2 on page 389 shows the portion of the output of the imcserver.log when we again made a simple search twice in a role.

Example 13-2 icmserver.log: log of searches in a Document Manager desktop with Content Manager 2

```
Perform a search.
ICMPLSQU ICMSEARCH        00482 04/10 18:39:24.328 GMT ; ... JOHN.DOE    74 msec
ICMPLSGI ICMGETITEM       04346 04/10 18:39:24.343 GMT ; ... JOHN.DOE     3 msec
ICMPLSLK ICMLISTNLSKEYWRD 00860 04/10 18:39:24.343 GMT ; ... JOHN.DOE     0 msec
Search is done.
Search again.
ICMPLSQU ICMSEARCH        00482 04/10 18:40:41.453 GMT ; ... JOHN.DOE    73 msec
ICMPLSGI ICMGETITEM       04346 04/10 18:40:41.468 GMT ; ... JOHN.DOE     3 msec
2nd search is done.
```

This time we see a significant improvement, including:

- The number of calls to Content Manager is significantly less than the previous scenario.
- There are no more ICMLOGON and ICMLOGOFF calls.
- There are no more ICMGETATTRTYPE or ICMGETITEMTYPE calls.
- The second search calls one less operation than the initial search call.

In summary, using **echo** (or some technique similar to this) and monitoring icmserver.log helps you identify problems and troubleshoot your application. You can use this technique to test your custom application before it goes to production and make sure it is only making calls as needed. When you reduce the number of unnecessary calls, you reduce the extra response time and thus improve the overall system performance.

Note: For guidelines to setting up a Document Manager system, refer to Appendix A, “Performance tuning guidelines for a Document Manager system” on page 435 for more details.

13.3 Enabling Library Server trace

As a background reference for the case study, this section shows how to enable Library Server trace.

The Content Manager Library Server performance trace can be controlled from the system administration client. Starting or stopping this trace takes effect immediately.

You can also start and stop the Library Server performance trace from a script. Example 13-3 on page 390 shows the script to turn on the performance trace.

Example 13-3 Turn Library Server performance trace on

```
db2 connect to icmnlsdb user icmadmin using password
db2 update icmstsyscontrol set tracelevel=8
db2 update icmstsyscontrol set keeptraceopen=1
db2 connect reset
```

The trace data goes into the trace file specified in the System Administration client. This method has the advantage of specifying `keeptraceopen=1`, which buffers the log and makes its impact low enough for use on a production system.

Example 13-4 shows the script to turn off the trace.

Example 13-4 Turn Library Server trace off

```
db2 connect to icmnlsdb user icmadmin using password
db2 update icmstsyscontrol set tracelevel=0
db2 update icmstsyscontrol set keeptraceopen=0
db2 connect reset
```

Example 13-5 shows sample output of the Library Server performance trace log.

Example 13-5 Sample output of the Library Server performance trace log

```
ICMPLSP3 ICMPREPCRTDOCPART 00735 11/06 18:21:28.744 ?06182118536221 4 msec
ICMPLSLM ICMLISTRESOURCEMGR 00788 11/06 18:21:28.754 ?06182118536221 2 msec
ICMPLSC2 ICMCREATEDOCPART 01252 11/06 18:21:29.516 ?06182118536221 11 msec
ICMPLSQU ICMSEARCH 00228 11/06 18:22:01.071 ?06182118536221 6 msec
ICMPLSGI ICMGETITEM 03832 11/06 18:22:01.091 ?06182118536221 13 msec
ICMPLSCO ICMCHECKOUTITEM 00407 11/06 18:22:04.015 ?06182118536221 2 msec
```

The second field of each entry specifies the name of the Library Server operation. On the same entry, you can also find the timestamp of when the operation is performed and its elapsed time in milliseconds. This information helps you monitor the overall workload of the Content Manager system. It can clearly show whether a performance issue is due to time spent in the Library Server or some other component of the Content Manager system. This often makes capturing a Library Server trace the first and most important step in troubleshooting a performance problem. It is especially useful for tuning custom applications to make most efficient use of Content Manager and minimize the application's impact on the Content Manager system.

Note: Remember to enable the trace only when you troubleshoot a problem because turning on the performance trace affects system performance. Turn off the trace as soon as you no longer need the information.

13.4 Configuring Document Manager cache

As background reference information for the case study, this section describes configuring Document Manager cache. We assume you have some basic knowledge of Document Manager components, its servers, and services.

13.4.1 Cache Manager overview

Cache objects are small files that contain configuration information about things such as menus, action dialogs, and folders. The cache files are stored on the Document Manager server. As changes are made to the Document Manager configurations through Designer, the Cache Manager updates those configurations in the library and to the cache files, which are stored on the Document Manager server.

The use of cache files enhances Document Manager Desktop performance by reducing the resource utilization that occurs in client/server applications.

Document Manager compares the time and date stamp of the cache objects stored on the user's desktop to the cache objects stored on the Document Manager server. If the Document Manager server has a cache object that is newer, it is delivered to the Desktop. If the cache object is the same time and date, it is not updated. The Document Manager Desktop reads the configuration information from these local cache objects instead of continually accessing the library for the information. This improves performance and reduces network traffic between the Document Manager Desktop and the repository.

13.4.2 Cache Manager considerations and strategies

The Cache Manager default is set to a login frequency of every 30 seconds. Every time this service runs, it consumes every available server resource to process the cache update. During the initial configuration setup, you can use this short frequency because of the ongoing changes made in the Document Manager Designer. However, when the system reaches a point where the changes are less frequent, you no longer need to have such a short frequency. Optimal server-desktop performance can be achieved by completely turning off the cache server and setting client cache to update after a large interval of time, such as every four weeks.

When planning the frequency of cache, determine how often you want the client desktop to check for more recent cache information. By default, the client desktop checks the cache at the start of every session. When Internet Explorer is launched and the Document Manager Desktop is started, Document Manager compares the local cache date and timestamps against those on the Document Manager server. If the server-side cache files are more recent, they are pulled to the local machine. Updates are performed automatically, so you never have to plan and implement rollouts of new configurations. The administrator has to weigh the benefit of having nearly real-time data (by forcing cache to update more frequently both on the Document Manager Desktop and server) against improved performance by having cache update less frequently.

13.4.3 Configuring Document Manager cache

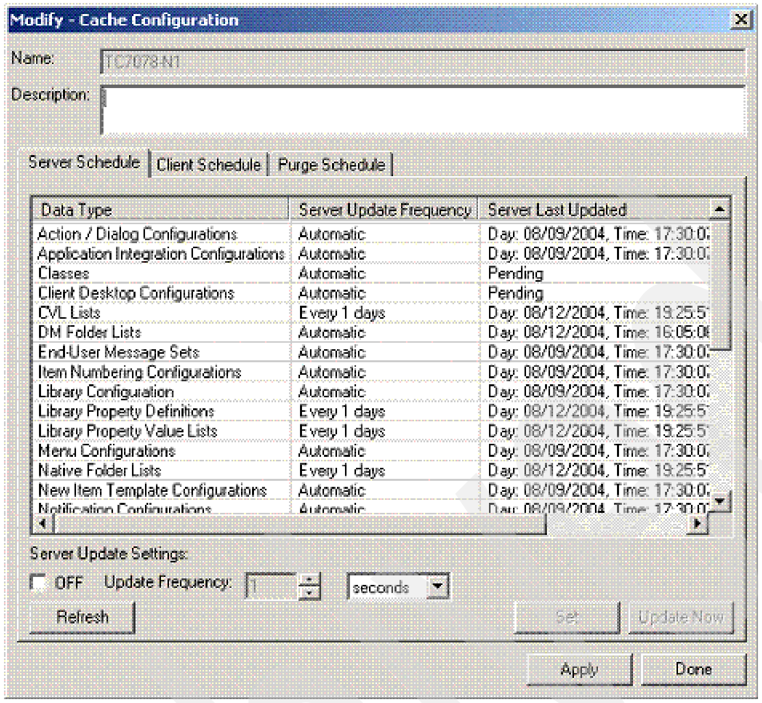
Both the server cache and client cache updates can be configured through the Designer or the `Configure` command in the Document Manager Desktop after the Desktop is opened.

Note: Cache must be run once before following the steps below. The first time the cache service runs, the cache object is created.

How to configure a cache object

Follow these steps to configure a new cache object:

1. Click **Start** → **Programs** → **IBM DB2 Document Manager** → **Designer**.
2. Expand the database, **Global - [Administrator]** component, **Desktop**, and **Cache Configuration**.
3. From the Designer toolbar, click **New**. The New Cache Configuration dialog appears.
4. To configure server schedule information for the new cache object, perform the following actions:
   a. Click the **Server Schedule** tab. A list of all cache objects written to the DB2 Document Manager server cache appears (Figure 13-1 on page 393).


Modify - Cache Configuration

Name: TC7078-N1

Description:

Server Schedule | Client Schedule | Purge Schedule

Data Type	Server Update Frequency	Server Last Updated
Action / Dialog Configurations	Automatic	Day: 08/09/2004, Time: 17:30:0
Application Integration Configurations	Automatic	Day: 08/09/2004, Time: 17:30:0
Classes	Automatic	Pending
Client Desktop Configurations	Automatic	Pending
CVL Lists	Every 1 days	Day: 08/12/2004, Time: 19:25:5
DM Folder Lists	Automatic	Day: 08/12/2004, Time: 16:05:0
End-User Message Sets	Automatic	Day: 08/09/2004, Time: 17:30:0
Item Numbering Configurations	Automatic	Day: 08/09/2004, Time: 17:30:0
Library Configuration	Automatic	Day: 08/09/2004, Time: 17:30:0
Library Property Definitions	Every 1 days	Day: 08/12/2004, Time: 19:25:5
Library Property Value Lists	Every 1 days	Day: 08/12/2004, Time: 19:25:5
Menu Configurations	Automatic	Day: 08/09/2004, Time: 17:30:0
Native Folder Lists	Every 1 days	Day: 08/12/2004, Time: 19:25:5
New Item Template Configurations	Automatic	Day: 08/09/2004, Time: 17:30:0
Notification Configurations	Automatic	Day: 08/09/2004, Time: 17:30:0

Server Update Settings:

☐ OFF Update Frequency: 1 seconds

Figure 13-1 Cache configuration: Server Schedule tab

b. Select one or more of these objects and alter the frequency of server-side updates, or force an immediate update by taking any of these actions:

   - Select the **OFF** check box to turn off the update setting. This cache object cannot be updated while the OFF check box is selected.
   - Increase or decrease the **Update Frequency** number.
   - To the right of the Update Frequency number, you can select an increment for server update frequency: seconds, minutes, hours, days, weeks, or months.
   - Click **Update Now** to force the update of selected objects, even though their update frequency is not expired.

c. Click **Refresh** to show the new time and date stamps after the cache service runs.

d. Click **Set** to post the changes made to the Server Update Settings. There is no need to click **Set** when using the **Update Now** button.

e. Click **Apply** to post the changes to the library environment.

5. To configure client schedule information for the new cache object:
   a. Click the **Client Schedule** tab. A list of all cache objects written to the Document Manager Client Cache appears.
   b. You can select one or more of these objects and alter the frequency of client-side updates by taking any of the following actions:
      - Select the **Do not store cache data locally**. **Remove immediately after use** check box to remove temporarily stored client cache on the user's computer when the user exits the DB2 Document Manager Desktop.

        Attention: This option is not recommended.

      - In the Update Frequency list, choose one of the following options:

        Each Access: Each time the user accesses a component of the DB2 Document Manager Desktop, such as a menu, the Desktop requests the DB2 Document Manager server to determine whether the menu cache object has been updated. If so, the DB2 Document Manager Desktop updates cache for the menu.

        Each Session: Each time the user starts a DB2 Document Manager Desktop session and logs into a library environment, the client cache is checked for update.

        Interval: The Desktop checks the DB2 Document Manager server for new cache files at the specified interval. Choose the frequency interval and increment to the right of Update Frequency.

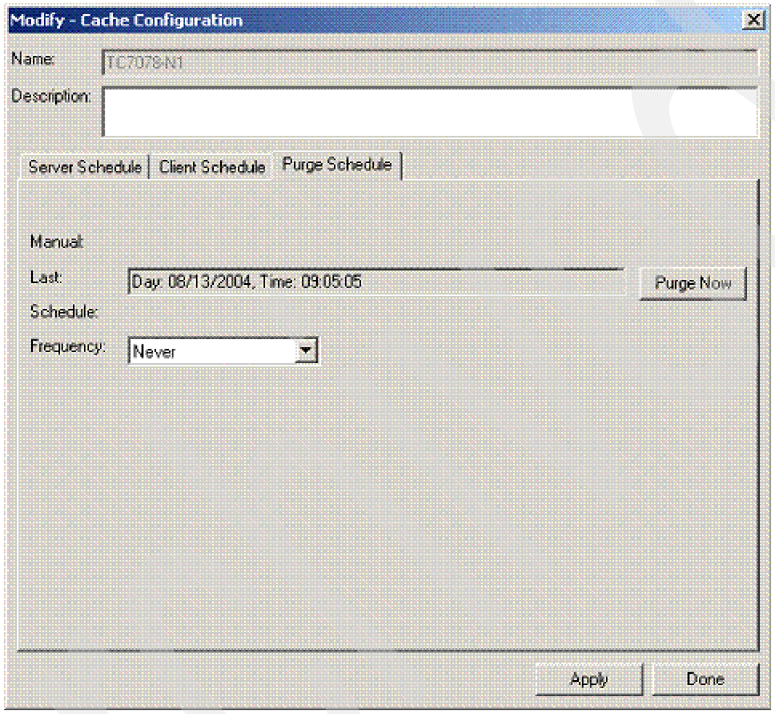
        Calendar: Choose a calendar interval of Daily, Weekly, or Monthly.
   c. Click **Set** to post the changes to the cache object list.
   d. Click **Apply** to post the changes to the library environment.
6. The Purge Schedule tab (Figure 13-2 on page 395) enables you to configure the purging of client cache objects. Purging removes the data files representing the client cache in the DB2 Document Manager client subdirectory. To configure the purge schedule for the new cache object:
   a. Click the **Purge Schedule** tab.
   b. Set the client cache to purge by taking any of the following actions:
      - Click **Purge Now** to force the removal of client cache data files even though the frequency does not call for the action.
      - Choose one of the following options from the Frequency list:

        Never: The client cache is never updated automatically.

Each Session: Each time the user starts a DB2 Document Manager Desktop session and logs into a repository environment, the client cache is deleted and new files are updated from the Cache server.

Interval: Choose the frequency interval and increment (seconds, minutes, hours, days, weeks, or months).

Calendar: Choose a calendar interval of Daily, Weekly, or Monthly.


Modify - Cache Configuration

Name: TC7078-N1

Description:

Manual:

Last: Day: 08/13/2004, Time: 09:05:05

Schedule:

Frequency: Never

Figure 13-2 Cache configuration: Purge Schedule tab

7. When you are finished, click **Apply** and **Done**.

14

Maintenance

In this chapter, we describe the maintenance tasks necessary to keep a Content Manager system up and running. This includes a discussion about which data to back up in order to restore an entire system, as well as information about which monitoring tasks should be performed on a regular basis.

14.1 Maintenance tasks overview

After a Content Manager system has been designed, installed, and configured, it still requires regular maintenance.

By maintaining a Content Manager system properly and in a timely manner, you get the best possible performance from it over time, as well as potentially avoid problems that can manifest themselves as system-endangering situations.

Much of this chapter is extracted from *IBM Content Manager System Administration Guide V8.2*, SC27-1335; the publication can be used in conjunction with this redbook to provide further details.

Regular maintenance tasks include the following activities:

- Optimizing server databases
- Monitoring LBOSDATA directory size
- Managing staging directory space
- Removing entries from the events table
- Removing log files
- Managing Resource Manager utilities and services
- Replacing or repartitioning a hard disk
- Backing up

14.2 Optimizing server databases

Database statistics should be updated periodically, daily, or weekly depending on the ingest load, and the Library Server and Resource Manager databases in order to maintain good performance. This should also be the first step whenever it appears that there are problems associated with Library Server performance, such as slower logons, searches, or indexing.

runstats and rebind database tables

Keeping DB2 statistics up to date on the tables and data helps the optimizer to choose the best execution access plans for SQL statements to maximize performance. We recommend that you run the **`runstats`** and **`rebind`** commands on a regular basis as part of ongoing database maintenance. Recalculating table statistics is *critical* to improving database performance and should be done regularly. If table statistics have not been recalculated recently, then this should be the first step in diagnosing DB2 performance issues.

Find instructions for running these commands in the DB2 Command Reference (**Start → Programs → IBM DB2 → Information → DB2 Information Center** and type `runstats` or `rebinds` in the search field). Use the *DB2 Command Reference*, SC09-4828, and the following instructions to run these commands:

1. Open a DB2 Command Window by clicking **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**. If you are using a UNIX machine to perform these tasks, after the db2profile script has been run (if it is not in root's .profile), the commands can be typed directly onto the UNIX command line. If you are not already connected to the database, connect to the database by entering:

   ```
   db2 connect to <db name> user <user ID> using <password>
   ```

   – `<db name>` is the name of the database.

   – `<user ID>` is user ID with administration rights on the database.

   – `<password>` is the valid password of the user ID.

2. Run **`runstats`** as follows:

   ```
   db2 runstats on table <table_name> with distribution and detailed
   indexes all
   ```

   This should be done for each table in the database, for both the Library Server and Resource Manager databases. For example, for SYSINDEXES table, use:

   ```
   db2 runstats on table sysibm.sysindexes
   ```

3. The DB2 system **`db2rbind`** command should be run after calculating table statistics to rebind all of the packages in the database. This is a system command and not a command line statement. This means that it does not have to be prefixed with `db2` as the **`runstats`** command does.

   Run **`db2rbind`** as follows:

   ```
   db2rbind <database_name> -l <logfile name> all /u <userid> /p
   <password>
   ```

   For example:

   ```
   db2rbind icmnlsdb –l bind.log all /u icmadmin /p password
   db2rbind rmdb –l bind.log all /u rmadmin /p password
   ```

4. Check your log file to see the results. Another way that you can check the success of a rebind is by using the DB2 Control Center:

   a. Open the Control Center by clicking **Start** → **Programs** → **IBM DB2** → **General Administration Tools** → **Control Center**.

   b. Go to the database against which you ran **`db2rbind`**.

   c. In the database, go to **Application objects** → **Packages**.

   Check the Last bind date and Last bind time columns. The date and time indicate when you last had DB2 rebind all of the packages.

reorgchk

A table can become fragmented after many updates, causing performance to deteriorate. Queries take longer because index entries in the Library Server and Resource Manager are no longer synchronized with the actual data in the database tables.

You can synchronize the data in the index with the database tables by running the **`reorgchk`** command in DB2. The **`reorgchk`** command gathers and compares both the index and the table statistics and recommends tables to reorganize. Following the recommendation, use the **`reorg`** command to reorganize the necessary tables.

When you reorganize tables, you remove empty spaces and arrange table data for efficient access. Reorganizing tables takes a lot more time than simply checking (**`reorgchk`**) which tables might require reorganization. Do not reorganize tables when you expect a lot of server activity, because performance might be affected. DB2 locks any data in a table that is currently being reorganized.

If you update tables often, you should reorganize periodically (for example, once a month). If you do not manage the DB2 database tables, work with the DB2 administrator for access or to coordinate when to run **`reorgchk`** and when to reorganize tables.

You can find instructions about how to update database tables in the DB2 Command Reference. (Click **Start** → **Programs** → **IBM DB2** → **Information** → **DB2 Information Center** and type `reorgchk` in the search field.) Use the *DB2 Command Reference*, SC09-4828 and the following instructions to check and update database tables:

1. Open a DB2 command window by clicking **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**. If you are using a UNIX machine to perform these tasks, after the db2profile script has been run (if it is not in root's .profile), the commands can be typed directly onto the UNIX command line. If you are not already connected to the database, connect to the database by entering:

   ```
   db2 connect to <db name> user <user ID> using <password>
   ```

   In this syntax:

   - `<db name>` is the name of the database.
   - `<user ID>` is user ID with administration rights on the database.
   - `<password>` is the valid password of the user ID.

2. Run **`reorgchk`** as follows:

   ```
   db2 reorgchk update statistics on table all > out.txt
   ```

`out.txt` is the log file into which the output is generated. When you run **reorgchk**, we recommend storing the results in a file due to the large amount of output generated. This file, known as a log file, contains the statistics you need to use to determine whether to reorganize a particular table. In our scenario, we pipe the results to out.txt. You can use any name for the log file.

If you have an idea about what tables usually need to be reorganized, you can perform **reorgchk** on only these tables.

3. Check the REORG column in your log file. DB2 displays one to three asterisks (***) in the REORG column when it detects a table to reorganize. The number of asterisks determine the urgency of reorganizing the table.

   The first two columns are the schema name and table name. You use these two names to reorganize tables. For example, a schema name can be icmadmin or sysibm, and a table name can be ICMSTNLSKEYWORDS or SYSINDEXES.

4. To reorganize a particular table, use:

   ```
   db2 reorg table <table name>
   ```

   `<table name>` is the name of the table you want to reorganize. For example, to reorganize the SYSINDEXES table, we use:

   ```
   db2 reorg table sysibm.sysindexes
   ```

5. If you use **reorg** on any table, it is also a good idea to use the **runstats** command to update the table statistics again:

   ```
   db2 runstats on table <table name>
   ```

   `<table name>` is the table you want to update statistics on. For example, for the SYSINDEXES table, we use:

   ```
   db2 runstats on table sysibm.sysindexes
   ```

6. When you finish reorganizing database tables, rebind all packages within the database using the **db2rbind** command to allow new access plans to be generated. You do not need to be connected to the database for this step. In the DB2 command window, enter:

   ```
   db2rbind <db name> /l report.txt
   ```

   In this syntax:

   - `<db name>` is the name of the database.
   - `report.txt` is the name of the log file that contains any errors that result from the package revalidation procedure.

Important: You need a user ID and password if you plan to update a schema that does not belong to you. This user ID and password must have DB2 administrative authority to complete this task.

This command uses the rebind API (`sqlarbnd`) to attempt the revalidation of all packages in a database. If the rebind of any of the packages encounters a deadlock or a lock timeout, the rebind of all packages will be rolled back.

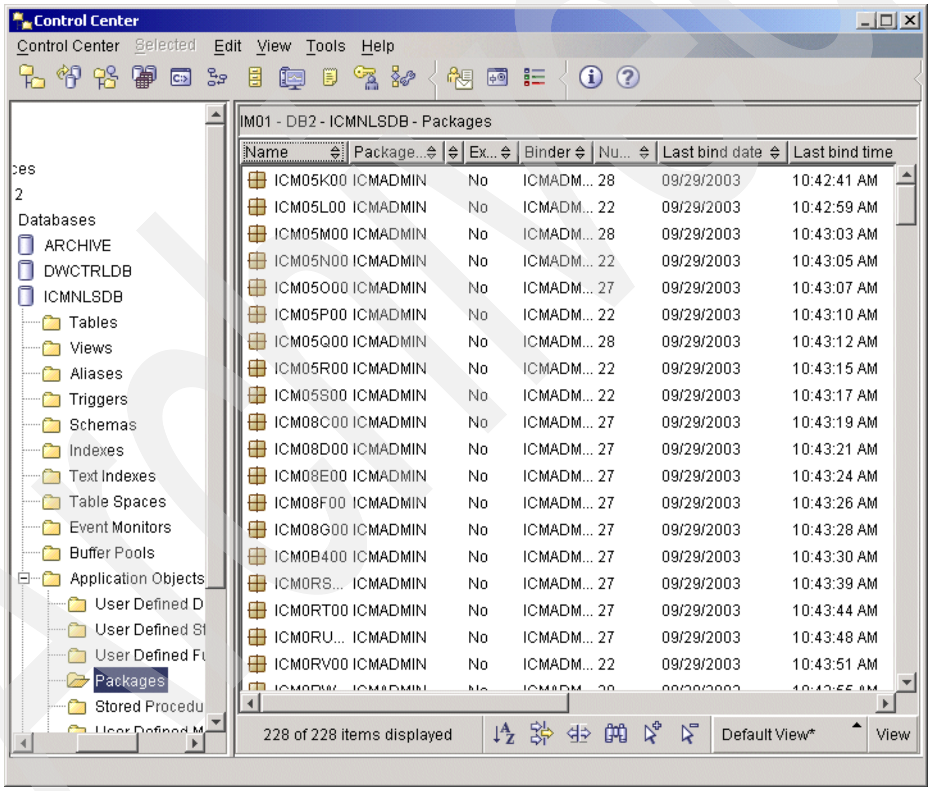
7. Check your log file to see the results. Another way to check the success of a rebind is by using the DB2 Control Center:

   a. Open the Control Center by clicking **Start** → **Programs** → **IBM DB2** → **General Administration Tools** → **Control Center**.

   b. Go to the database against which you ran **`db2rbind`**.

   c. In the database, go to **Application objects** → **Packages**.

   d. Check the Last bind date and Last bind time columns. The date and time indicate when you last had DB2 rebind all the packages (Figure 14-1).



Name	Package...	Ex...	Binder	Nu...	Last bind date	Last bind time
ICM05K00	ICMADMIN	No	ICMADM...	28	09/29/2003	10:42:41 AM
ICM05L00	ICMADMIN	No	ICMADM...	22	09/29/2003	10:42:59 AM
ICM05M00	ICMADMIN	No	ICMADM...	28	09/29/2003	10:43:03 AM
ICM05N00	ICMADMIN	No	ICMADM...	22	09/29/2003	10:43:05 AM
ICM05O00	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:07 AM
ICM05P00	ICMADMIN	No	ICMADM...	22	09/29/2003	10:43:10 AM
ICM05Q00	ICMADMIN	No	ICMADM...	28	09/29/2003	10:43:12 AM
ICM05R00	ICMADMIN	No	ICMADM...	22	09/29/2003	10:43:15 AM
ICM05S00	ICMADMIN	No	ICMADM...	22	09/29/2003	10:43:17 AM
ICM08C00	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:19 AM
ICM08D00	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:21 AM
ICM08E00	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:24 AM
ICM08F00	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:26 AM
ICM08G00	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:28 AM
ICM0B400	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:30 AM
ICM0RS...	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:39 AM
ICM0RT00	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:44 AM
ICM0RU...	ICMADMIN	No	ICMADM...	27	09/29/2003	10:43:48 AM
ICM0RV00	ICMADMIN	No	ICMADM...	22	09/29/2003	10:43:51 AM

Figure 14-1 Checking the success of a rebind using the DB2 Control Center

For more information about **`runstats`**, **`rebind`**, **`reorgchk`**, and other DB2 commands, see the *DB2 Command Reference*, SC09-4828. For a more detailed understanding of reorganizing and rebinding DB2 database tables, see the *DB2 System Administration Guide*, SC09-4820.

14.3 Monitoring LBOSDATA directory size

The LBOSDATA directory, an area of local disk that a Resource Manager controls, is used to store objects.

When using fixed disk attached to the Resource Manager for object storage, be sure that there is enough free space remaining for Content Manager to write objects to. If Content Manager runs out of space to write objects to, any new requests to store objects will fail.

Even when Tivoli Storage Manager (TSM) is used for the long-term storage of objects, Content Manager may be configured to keep objects locally, and only migrate to TSM after a period of time (for example, 30 days). The migration of objects to TSM is triggered by the length of time the objects have resided in the first storage class, assuming that there are only two storage classes and TSM is the second one.

In this instance. it is possible that the local fixed volume that the LBOSDATA directory resides on might become completely full if new objects are being added to it faster than they are being migrated to TSM by the Content Manager migrator process. This might occur during peak periods of object loading into Content Manager, such as around the end of a financial year for an accounting company that scans documents into Content Manager for reference purposes. In a worst-case scenario, the process of migrating from the LBOSDATA directory to TSM might not even be running.

Important: Remember to have the Content Manager migrator process running at all times, even if you do not migrate objects between storage classes, because it is used to physically delete objects from where the Resource Manager has stored them. When a user deletes an object from the standard client, only the row from the Library Server database is deleted immediately (for performance reasons), and the entry in the Resource Manager database and the object itself remain. The *migrator must be run* to reclaim the physical storage space.

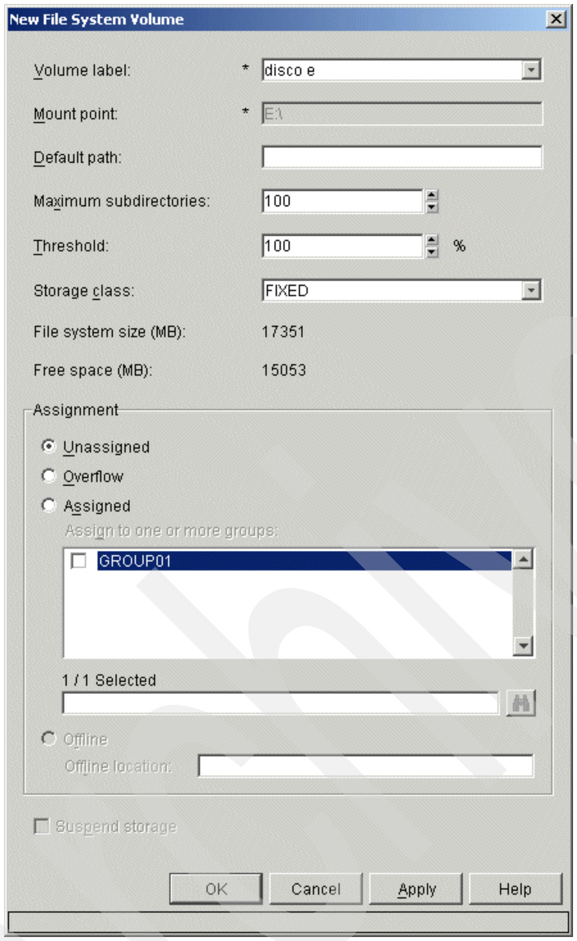
During peak activity times, it is even more important to monitor the amount of free space remaining in the local fixed disk that the LBOSDATA directory resides on. Of course, these peak periods of activity should have been taken into account when designing and sizing the system; nevertheless, monitoring the directories is good practice. Operating system tools should be used to monitor the current space occupied by the local objects and the amount of space remaining on a physical or logical volume.

It is possible to see how many megabytes of storage remain on a particular file system volume through the Content Manager system administration client, but this value is not updated dynamically. If space is being used on a file system volume while you are logged onto the system administration client, the number of megabytes free on the file system volume will not change. To see the change in volume free space, log off and log back onto the system administration client; however, this is not an entirely accurate way to monitor free space.

When you create a file system volume for Content Manager to use, such as the e: drive on a Windows machine, Content Manager takes the remaining free space on this physical or logical volume as the amount of free space currently available for objects. In other words, when you create a file system volume within Content Manager, you cannot assign only a certain percentage of it to be used.

In the same way, Content Manager does not reserve space for its objects on a file system volume. It is very important to make sure that no other applications use the same volume to store dynamic data. If this occurs, the amount of free space available to Content Manager to write objects can change unexpectedly.

Figure 14-2 shows the window that is used to define a new file system volume to a Resource Manager.



Volume label: *	disco e
Mount point: *	E:\
Default path:	
Maximum subdirectories:	100
Threshold:	100 %
Storage class:	FIXED
File system size (MB):	17351
Free space (MB):	15053

[✓ Unassigned] Overflow Assigned

Assign to one or more groups:

☐ GROUP01

1 / 1 Selected

Offline

Offline location:

☐ Suspend storage

Figure 14-2 Defining a new file system volume

If a file system volume becomes full, it is possible to define a new volume, assuming that you have the physical space available, and then add this new volume to the existing storage group, in order to provide further space for Content Manager to store objects to.

When you define a file system volume, a threshold percentage can be entered. This value is used as a limit at which point Content Manager will attempt to migrate objects to the next storage class, if one exists, and if the migrator process is running. The default value for the threshold is 95%. This threshold limit should never be reached in the normal course of events, and the threshold limit mechanism should not be relied on as the default way of monitoring and dealing with overly full volumes. In most circumstances, the default value of 95% will be fine to use in a production system.

A more effective way to prevent Content Manager from running out of space is to create overflow volumes, which are volumes that can be used by any storage groups, when all other storage systems (such as a file system volume) in a storage group are full. To create an overflow volume, select **Overflow** under the Assignment section of the New File System Volume window (Figure 14-2 on page 405). You can define as many overflow volumes as you desire.

It is also important to monitor the space remaining in the TSM volumes used by Content Manager, as well as the space remaining in your TSM database, and log volumes. Contact your TSM administrator to perform these functions.

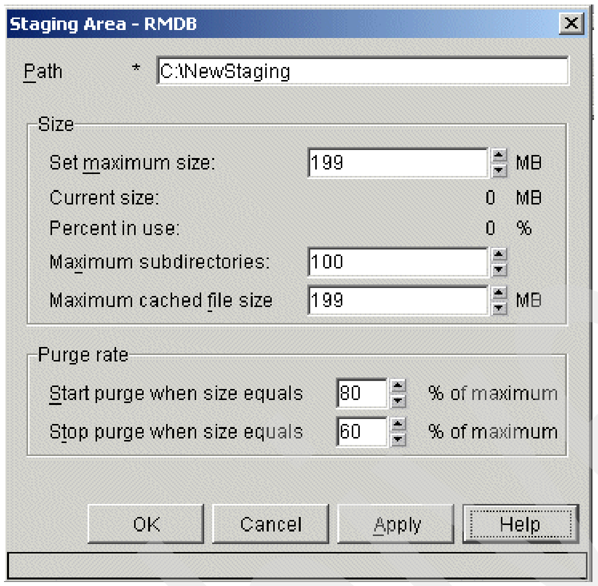
14.4 Managing staging directory space

The staging area is used as a temporary storage area for objects retrieved from TSM storage and as the location to store objects when the LAN cache is enabled for a Resource Manager. Using the staging area enables faster response time for subsequent retrievals of the same objects.

The system administration client enables users to manage the staging directory to get the most benefits from LAN caching and from TSM object retrieval caching. Staging directory management tasks include:

- Setting automatic cache purge specifications: A purge removes the oldest, least frequently used objects from the staging directory.
- Defining subdirectories to hold cached objects: Storing cached objects in subdirectories can improve system retrieval time because the system can target the search without looking through individual objects stored in the staging directory.
- Defining the size of the staging directory: Depending on the size and volume of cached objects, you might need to modify the original parameters defined for the staging directory.

Figure 14-3 shows the Staging Area properties window, which is accessed through the Content Manager system administration client. Right-click the Resource Manager database and select **Staging Area**.



Path: C:\NewStaging

Set maximum size: 199 MB
Current size: 0 MB
Percent in use: 0 %
Maximum subdirectories: 100
Maximum cached file size 199 MB

Start purge when size equals 80 % of maximum
Stop purge when size equals 60 % of maximum

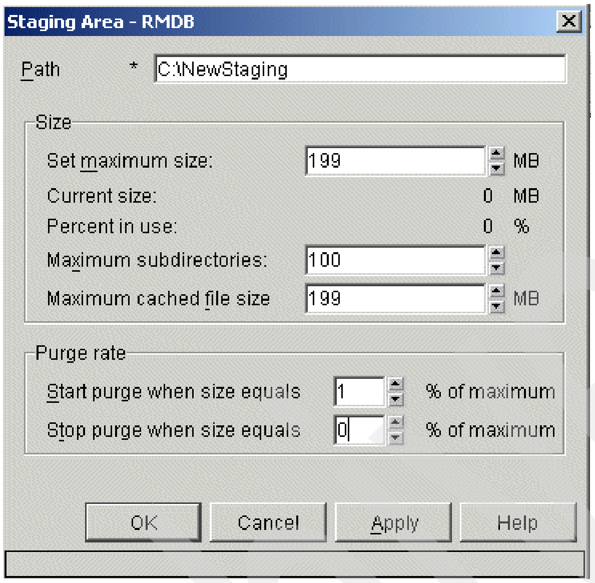
Figure 14-3 Staging Area properties window

Defining the maximum size of the cached object. The system will not cache objects that exceed the maximum size; however, if you decrease the maximum size and objects that were stored earlier exceed the new maximum size, the system will retain these existing objects.

14.4.1 Purger process

The *purger* process is used to maintain the size of the staging area. When the staging area size reaches a preset upper limit, the purger will begin to remove files until it reaches a preset lower limit. Using Figure 14-3 as our example, this means that our staging area is 199 MB in size, and purging will commence when this 199 MB area is 80% (159.2 MB) full if the purger process is started. When the staging area reaches 159.2 MB full in size, the purger will start randomly deleting files until the staging area reaches 60% of 199 MB (119.4 MB).

All of the staging area values are configurable. For example, if you want to completely clear the staging area, you can set the start purge size to 1% of the maximum staging area, and the stop purge size to 0% of the maximum staging area size. Figure 14-4 shows this configuration.


Staging Area - RMDB

Path * C:\NewStaging

Size

Set maximum size: 199 MB

Current size: 0 MB

Percent in use: 0 %

Maximum subdirectories: 100

Maximum cached file size 199 MB

Purge rate

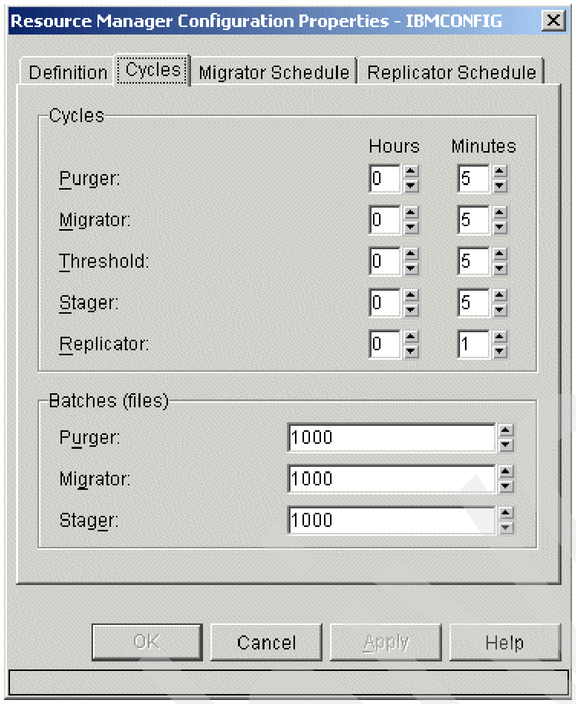
Start purge when size equals 1 % of maximum

Stop purge when size equals 0 % of maximum

Figure 14-4 Configuration to clear staging area completely

With the configuration set in Figure 14-4, you should be able to clear the entire staging area, assuming that the staging area is at least 1% of 199 MB full at the time. If the staging area is below 1% full at the time, you need to reduce the size of the staging area from 199 MB to a size where 1% of the staging area maximum size was smaller than the currently occupied space in the staging area.

The staging area maximum size and purge rates are monitored periodically, not constantly. For this reason, you might need to wait up to five minutes, the default setting, before changes you have made to the staging area come into effect. The cycle time for this checking is configured via the Resource Manager configuration window. To open this window, go to the Content Manager system administration client, open a Resource Manager, and select **Configurations**. Then select the Resource Manager configuration that you are currently using (the default is IBMCONFIG), and select the **Cycles** tab (Figure 14-5 on page 409).



Cycles

	Hours	Minutes
Purger:	0	5
Migrator:	0	5
Threshold:	0	5
Stager:	0	5
Replicator:	0	1

Batches (files)

Purger:	1000
Migrator:	1000
Stager:	1000

Figure 14-5 Resource Manager cycles

The threshold cycle sets the amount of time that elapses before the staging area size is updated. Figure 18-5 displays the defaults for a Resource Manager. The other cycles refer to the amount of time that elapses before the various Resource Manager utilities check to see whether they have any work to do.

The settings for the staging area and cycle times that are best suited to your environment might differ from the default settings. For example, if your system produces instances when the staging area is heavily used, you might need to adjust the cycle time so that the purger checks the staging area more regularly to see whether it has any work to do.

14.5 Removing entries from the events table

When you use the Content Management system administration client, the Library Server records activities related to item and document routing in one of the events tables, ICMSTSADMEVENTS or ICMSTITEMEVENTS.

The events table grows with each logged event. To reduce the size of the events table, you can remove the expired and unused events from the table. The

EventCode column in the events table indicates the classification of events as the values shown in Table 14-1.

Table 14-1 Library Server events table definitions

Value	Definition
1 - 200	System administration function event codes
200 - 900	Item, document routing, and resource management function event codes
1000 +	Application event codes

You can delete events from the events table by performing either of these tasks:

- To delete an event for a system administration function from a Library Server, connect to your database and use the following SQL command:

  ```
  delete from ICMSTSYSADMEVENTS where eventcode <=200 and Created
  < 2002-01-01-12.00.00.000000
  ```

- To delete an event for an item function from a Library Server, connect to your database and use the following SQL command:

  ```
  delete from ICMSTITEMEVENTS where eventcode <=600 and Created
  < 2002-05-01-12.00.00.000000
  ```

To reclaim the file system space after you delete the events, run the database reorganization utility on the Library Server database.

14.6 Removing log files

It is important to remember to remove log files on a regular basis when they are no longer needed for troubleshooting or audit purposes. This prevents the log files from become overly large, taking up unnecessarily large areas of disk, and becoming unwieldy due to their sheer size. When the log files are removed, they will be re-created by the particular application that created them.

Some log files cannot be deleted while the system is in use because they are being written to. You must stop the component that is writing to the log file in order to delete it. For a list of log files that you should regularly check the size of and if necessary delete, see Appendix F, “Configuration and log files” on page 687.

It is especially important to remember to check on log files and remove them when any form of tracing is enabled, because the log files will grow much more quickly than usual. See Chapter 21, “Troubleshooting” on page 561 for more about enabling tracing.

14.7 Managing Resource Manager utilities and services

This section describes utilities and processes that are installed on the Content Manager Resource Manager. The utilities are available on AIX, Linux, Solaris, and Windows. Some of the utilities exist as services on Windows. For all other utilities, you must log on to the server where the Resource Manager is installed, with a user ID that has DB2 administrative (DBADM) authority.

The utilities and processes include:

- The stand-alone application services: RMMigrator, RMPurger, RMReplicator, and RMStager.
- The Asynchronous Recovery utilities.
- Resource Manager/Library Server validation utility and the Resource Manager volume validation utility. These two utilities are installed with the Content Manager Resource Manager.

14.7.1 Configuration of Resource Manager utilities and services

This section provides general background information about configuring Resource Manager utilities and services.

In Content Manager, there is a central environment setup file, setprocenv.sh for UNIX or setprocenv.bat for Windows. This file stores a set of parameters for each deployed Resource Manager. These parameters are configured automatically when the Resource Manager is deployed and are used by the Resource Manager services and utilities.

Log configuration settings are specified using the logging and tracing utility in the system administration client.

Environment setup file on UNIX (including AIX, Linux, and Solaris)

The following services and utilities depend on one central file for environment setup, IBMCMROOT/config/setprocenv.sh:

- The stand-alone application services: RMMigrator, RMPurger, RMReplicator, and RMStager.
- The asynchronous recovery utilities: icmrmdel and icmrmtx
- The validation utilities: icmrmlsval and icmrmvolval

The setprocenv.sh file contains one set of environment variables for each Resource Manager.

Environment setup file on Windows

The following utilities depend on one central file for environment setup, IBMCMROOT\config\setprocenv.bat:

- The asynchronous recovery utilities: icmrmdel and icmrmtx
- The validation utilities: icmrmlsval and icmrmvolval
- The stand-alone application services: RMMigrator, RMPurger, RMReplicator, and RMStager, when started from the command line.

Important: These services are usually started as Windows services. If network-attached storage is in use, however, they must be started from the command line, and the environment setup file must be configured.

The setprocenv.bat file contains one set of environment variables for each Resource Manager.

Environment setup file variables

The following variables are used in the central environment setup file, setprocenv. There is one set of variables for each Resource Manager. Each variable is prefixed with the Resource Manager's identifier.

IBMCMROOT	DB2 Content Manager installation directory.
dbname	Resource Manager database name.
dbtype	Resource Manager database type: DB2 or Oracle.
rmappname	Resource Manager application name.
nodename	WebSphere Business Integration Server Foundation or WebSphere Application Server nodename.
was_home	WebSphere Business Integration Server Foundation or WebSphere Application Server home installation directory.
db2home	DB2 instance home directory where the Resource Manager database resides, if the Resource Manager database is a DB2 database. On Windows, enter the directory as a fully qualified path with drive letter (for example: C:\Program Files\IBM\SQLLIB). On UNIX, enter the directory as a fully qualified path (for example: /home/db2inst1/sqllib). Leave blank if the Resource Manager database is an Oracle database.
db2_jdbc_abspath	If the Resource Manager is using DB2 Type 4 connector, set this to the fully qualified path for the JDBC driver location. On Windows, enter the directory as a fully qualified path with drive letter (for example: C:\Program

	Files\IBM\SQLLIB\java\db2jcc.jar). On UNIX, enter the directory as a fully qualified path (for example: /home/db2inst1/sqllib/db2jcc.jar). Leave blank if DB2 Type 4 connector is not in use.
db2_jdbc_license_abspath	If the Resource Manager is using DB2 Type 4 connector, set this to the fully qualified path for the JDBC license file. On Windows, enter the directory as a fully qualified path with drive letter (for example: C:\Program Files\IBM\SQLLIB\java\db2jcc_license_cisuz.jar). On UNIX, enter the directory as a fully qualified path (for example: /home/db2inst1/sqllib/db2jcc_license_cisuz.jar). Leave blank if DB2 Type 4 connector is not in use.
orahome	Oracle home installation directory, if the Resource Manager database is an Oracle database. Leave blank if the Resource Manager database is a DB2 database.
ora_jdbc_abspath	Fully qualified path for the Oracle JDBC driver location, if the Resource Manager database is an Oracle database. Oracle JDBC 9.0.x is required. Leave blank if the Resource Manager database is a DB2 database.
waittime	Time that the application process main thread waits for the child threads to shut down before terminating itself.
sleeptime	Time that the client must wait for the process main thread to return an acknowledgement before re-polling its status.
CMRM_LOG_DIR	Directory where the log file for the stand-alone application services (RMMigrator, RMPurger, RMStager, and RMReplicator) is located.
CMRM_LOG_FILE	Name of the log file for the stand-alone application services (RMMigrator, RMPurger, RMStager, and RMReplicator).
initjavaheap	Initial heap size for the stand-alone application services.
maxjavaheap	Maximum heap size for stand-alone application services.

Resource Manager common utility parameters

In Content Manager, the central environment setup file, setprocenv, contains information about each Resource Manager. This information is used by the Resource Manager services and utilities. However, you can override information in the environment setup file by using the following parameters when starting services and utilities from the command line.

Tip: If any of the values specified in the tags contain blanks, use quotation marks to surround the values (for example, `-orajdbc "xxx yyy zzz"`).

-db dbname	Resource Manager database (RMDB) name. The parameter -dbname dbname is also valid.
-app rmappname	Resource Manager application name under WebSphere Business Integration Server Foundation or WebSphere Application Server name. The parameter -rmappname rmappname is also valid.
-dbtype databasetype	Resource Manager database type, where *databasetype* is either db2 or Oracle.
-was was_home	WebSphere Business Integration Server Foundation or WebSphere Application Server home installation directory. The parameter -was_home was_home is also valid.
-node nodename	WebSphere Business Integration Server Foundation or WebSphere Application Server nodename. The parameter -nodename nodename is also valid.
-db2home db2home	DB2 instance home directory where the Resource Manager database resides, if the Resource Manager database is a DB2 database. On Windows, enter the directory as a fully qualified path with drive letter (for example: C:\Program Files\IBM\SQLLIB). On UNIX, enter the directory as a fully qualified path (for example: /home/db2inst1/sqllib). -insthome insthome is also valid.
-db2_jdbc db2_jdbc_abspath	If Resource Manager is using DB2 Type 4 connector, set this to the fully qualified path for the JDBC driver location. On Windows, enter the directory as a fully qualified path with drive letter (for example: C:\Program Files\IBM\SQLLIB\java\db2jcc.jar). On UNIX, enter the directory as a fully qualified path (for example: /home/db2inst1/sqllib/db2jcc.jar).
-db2_jdbc_license db2_jdbc_license_abspath	If Resource Manager is using DB2 Type 4 connector, set this to the fully qualified path for the JDBC license file. On Windows, enter the directory as a fully qualified path with drive letter (for example: C:\Program Files\IBM\SQLLIB\java\db2jcc_license_cisuz.jar). On

UNIX, enter the directory as a fully qualified path (for example: /home/db2inst1/sqllib/db2jcc_license_cisuz.jar).

-orahome ORACLE_HOME
Oracle home installation directory, if the Resource Manager database is an Oracle database.

-orajdbc ora_jdbc_abspath
Fully qualified path for the Oracle JDBC location, if the Resource Manager database is an Oracle database. Oracle JDBC 9.0.x is required.

14.7.2 Configuring the Resource Manager services on UNIX

There are four services: RMMigrator, RMPurger, RMReplicator, and RMStager. In general, Resource Manager processes are configured using the setprocenv.sh file described in 18.7.1, "Configuration of Resource Manager utilities and services" on page 487. However, the values for dbname and rmappname can be changed if passed into the Process starting routine. These parameters override the ones that are set by the $IBMCMROOT/config/setprocenv.sh file.

Attention: On AIX, the parameters, dbname, rmappname, and application are all case sensitive. All of the process service names are registered in the /etc/services file.

Here is an example of how an entry for the services file appears:

```
RMMigrator_RMDB 7500/tcp #Resource Manager Migrator
```

In the example, RMMigrator is the stand-alone application process and RMDB is the database name. The dbname and application parameters passed to the /etc/rc.cmrmproc script should match the case in the service name registration in the /etc/services file.

14.7.3 Starting and stopping resource services on UNIX

You can start or stop a stand-alone application process as follows:

- To start all four applications:

  ```
  /etc/rc.cmrmprc -act start -db <dbname> -app <rmappname>
  ```

  In this syntax:

  – `<dbname>` is the database name on which these processes are running.

  – `<rmappname>` is the name of the Resource Manager Web application.

- To stop all four applications:

```
/etc/rc.cmrmproc -act stop -db <dbname> -app <rmappname>
```

- To start select applications:

```
/etc/rc.cmrmproc -act start -db <dbname> -app <rmappname> -proc
<application>
```

  `<application>` is the Resource Manager stand-alone process you want to start. For example, to start Resource Manager migrator, RMMigrator, on database rmdb with icmrm as the Resource Manager Web application, use:

```
/etc/rc.cmrmproc -act start -db rmdb -app icmrm -proc RMMigrator
```

- To stop select applications:

```
/etc/rc.cmrmproc -act stop -db <dbname> -app <rmappname> -proc
<application>
```

- To start all four applications using the default values for dbname and rmappname, specified in the $IBMCMROOT/config/setprocenv.sh file:

```
/etc/rc.cmrmproc start
```

14.7.4 Asynchronous recovery utility

Content Manager includes an automatic scheduled process called the asynchronous recovery utility. The asynchronous recovery runs at the start of each migrator cycle while the migrator is running and enabled and in its runtime window. The migrator normally should be started and enabled. It should be excluded from running only during peak load times.

The purpose is to periodically restore data consistency between a Library Server and its Resource Managers. This process is necessary for the following reasons:

- To provide a rollback function for failed transactions
- To complete scheduled deletion of items that are designated for deletion

The Library Server and Resource Manager can become inconsistent if the Resource Manager is down or communication between Information Integrator for Content and Resource Manager fails. The inconsistent state can be reconciled with the asynchronous transaction reconciliation utility.

Attention: Before performing any work, the migrator process will first run the Asynchronous Recovery utilities.

Another important result of running this utility is to clean up known successful transactions. As each create and update resource item transaction completes, a record is stored in the Library Server database. Over time, these records become more numerous and their database table increases in size. The table is cleaned up by the transaction reconciliation utility. It is important to run the utility on all Content Manager V8.1 or later Resource Managers.

Also, deleting Resource Manager resources is an asynchronous activity within Content Manager. When a user uses an application to delete an item, it is deleted, internally, from the Library Server. The asynchronous recovery deletion reconciliation utility is used to mark or physically delete the resource on the Resource Manager. Resource deletion is a multiple-step process. The Resource Manager migrator, running in the background, is responsible for taking all of the resources marked for deletion and physically deleting them. Resource deletion consists of three steps:

1. An Information Integrator for Content or Content Manager application deletes an item from the Library Server.
2. The Asynchronous Recovery Deletion Reconciliation utility marks the resource for deletion on the Resource Manager.
3. The Resource Manager migrator physically deletes the resource.

Although these processes are scheduled and automatic processes, you might want to run the programs themselves, for example, as part of a database backup procedure. To do so, you need to run two separate utility programs:

- The deletion reconciliation utility (ICMRMDEL)
- The transaction reconciliation utility (ICMRMTX)

Tip: In a production environment, synchronize the servers prior to any system backup. This not only ensures that your databases are in a consistent state, but also removes any database entries that represent deleted documents.

Configuring the asynchronous recovery utility

The asynchronous recovery standalone utilities, icmrmdel and icmrmtx, take the common utility parameters and use the default values specified in the environment setup file.

Asynchronous utility logging

By default, asynchronous utilities log to the console. You can modify the level of information logged and the output location in the icmrm_asyncr_logging.xml file. This XML file can be updated to output to FILE if desired. Make sure that the user ID that you use to run the utility has read permission to the XML file, and write permission to whatever log file you configure for use.

The icmrm_asyncr_logging.xml file is installed with the Resource Manager code in the WebSphere Application Server installedApps path.

- On UNIX, the default path to the file is:

  ```
  /usr/WebSphere/AppServer/installedApps/<nodename>/icmrm.ear
  /icmrm.war/icmrm_asyncr_logging.xml
  ```

- On Windows, the default path is:

  ```
  x :\WebSphere \AppServer
  \installedApps\<nodename>\icmrm.ear\icmrm.war
  \icmrm_asyncr_logging.xml
  ```

  `<nodename>` is the WebSphere Business Integration Server Foundation or WebSphere Application Server nodename.

Running the asynchronous recovery utilities on Windows

Run the asynchronous recovery utilities from a command prompt using two of the common utility parameters.

- To run the deletion reconciliation utility:
  a. Change to the IBMCMROOT\bin directory.
  b. Enter:

     ```
     icmrmdel.bat -db dbname -app rmappname
     ```

- To run the transaction reconciliation utility:
  a. Change to the IBMCMROOT\bin directory.
  b. Enter:

     ```
     icmrmtx.bat -db dbname -app rmappname
     ```

Running the asynchronous recovery utilities on UNIX

The asynchronous recovery utilities run when you start the migrator. You can also run the asynchronous recovery utilities from a command prompt using two of the common utility parameters. You must be logged in as the root user to run them manually.

- To run the deletion reconciliation utility:
  a. Change to the IBMCMROOT/bin directory.
  a. Enter:

     ```
     ./icmrmdel.sh -db dbname -app rmappname
     ```

- To run the transaction reconciliation utility:
  a. Change to the IBMCMROOT/bin directory.
  a. Enter:

     ```
     ./icmrmtx.sh -db dbname -app rmappname
     ```

Tip: After running the Asynchronous Recovery utilities, run the `runstats` function on your databases to ensure that they are operating efficiently. See 18.2, “Optimizing server databases” on page 474 for help using this command.

14.7.5 Validation utilities overview

The purpose of the validation utilities is to analyze discrepancies between three components: the Library Server, the Resource Manager, and storage systems used by the Resource Manager through its defined device managers. Any of these components can fail and require a restoration via a backup that might be out of synchronization with the other two components.

Because there is no direct link between the Library Server and the storage system (an example of a storage system is VideoCharger or Tivoli Storage Manager), differences must be reported between the Library Server and the Resource Manager, and the Resource Manager and the storage system using the following utilities:

- *The Resource Manager/Library Server validation utility* (icmrmlsval.sh or icmrmlsval.bat) generates reports that describe discrepancies between the Library Server and the Resource Manager.
- *The Resource Manager volume validation utility* (icmrmvolval.sh or icmrmvolval.bat) generates reports about discrepancies between the Resource Manager and the storage system.

The reports are in XML. You can use an XML tool or browser to view or manipulate the utility output files. Content Manager installs the XML document type definition (DTD) required by the validation utility output files.

You can modify the two utility files with information specific to your Content Manager system. The validation utilities are located in the bin directory in the Resource Manager installation directory.

The validation utility creates and drops a temporary DB2 table. The environment script requires the resource database name, user ID, password, schema, Web application path, and DB2 instance. To set the environment for both validation utilities, type **`setenvproc.bat`** or **`setenvproc.sh`**.

Logging

By default, the validation utilities log to a file named icmrm.validator.log in the WebSphere logs directory. You can modify the level of information logged and the location of the output in the icmrm_validator_logging.xml file. Be sure that the user ID that you use to run the utility has read permission to the XML file, and write permission to whatever log file that you configure for use.

The icmrm_validator_logging.xml file is installed with the Resource Manager code in the WebSphere Application Server installedApps path.

- On AIX, the default path to the file is:

  `/usr/WebSphere/AppServer/installedApps/<nodename>/icmrm.ear/icmrm.war/icmrm_validator_logging.xml`

- On Solaris, the default path is:

  `/opt/WebSphere/AppServer/installedApps/<nodename>/icmrm.ear/icmrm.war/icmrm_validator_logging.xml`

- On Windows, the default path is:

  `x:\WebSphere\AppServer\installedApps\<nodename>\icmrm.ear\icmrm.war\icmrm_validator_logging.xml`

  `<nodename>` is the WebSphere Business Integration Server Foundation or WebSphere Application Server nodename.

14.7.6 Resource Manager/Library Server validation utility

The Resource Manager/Library Server validation utility queries the Library Server for all of the objects that were created or updated in a specified time period. It then searches the Resource Manager database and detects any discrepancies. The utility runs on the Resource Manager server and requires connectivity to the Library Server database.

To start the utility, navigate to the Resource Manager bin directory and type **`icmrmlsval.sh`** or **`icmrmlsval.bat`**.

The utility requires input parameters that are described in Table 14-2. Both dashes (-) and forward slashes (/) are handled as the parameter separator. The parameter tags are supported in both lower and upper case.

Table 14-2 Resource Manager/Library Server validation utility parameters

Parameter	Description
-B *YYYY-MM-DD-HH.MM.SS*	The beginning time and date of the objects to examine. Use this parameter with the -E parameter to restrict the number of objects that the utility must examine. This parameter is optional. If it is not present, all of the objects prior to the -E date are returned, or all of the objects are returned if -E is also not defined.

Parameter	Description
-E *YYYY-MM-DD-HH.MM.SS*	The ending time and date of the objects to synchronize. Use this parameter with the -B parameter to restrict the number of objects that the utility must examine. This parameter is optional. If it is not present, all of the objects after the -B date are returned, or all of the objects are returned if -B is also not defined.
-F output-path	The absolute path to be used for the output files. The utility creates the UTF-8 XML files in this directory. This parameter is required.
-H	This parameter displays help information about how to invoke the utility. All other parameters are ignored and no processing occurs.

The utility creates a temporary table, RMLSITEMS, which is used to accumulate object statistics for the validation. At the end of the validation, this table is normally dropped. If the utility determines that the table is present, it presumes that another version of the utility is operating, and exits. If the table is left behind due to an aborted run, you need to drop this table. Connect to the Resource Manager database and drop the table with the following command:

```
db2 drop table RMLSITEMS
```

This is an example of how to invoke the Resource Manager/Library Server utility on an AIX server:

```
./icmrmlsval.sh -F /reportsdirectory -B 2002-08-30-00.00.00 -E
2002-09-01-00.00.00
```

Understanding the Resource Manager/Library Server reports

The base file names of the reports are `icmrmlsval`*`YYMMDDHHMMSS_Report Type string`*`.xml`. The Report Type string identifies the type of discrepancies a report contains. The description of the different report types are detailed in this section. The timestamp enables the administrator to run the utility multiple times without overwriting the output files. Examples of default names with the default report type are:

- cmrmlsval20020531123456_ORPHAN.xml
- cmrmlsval20020531123456_NOTINRM.xml
- cmrmlsval20020531123456_SIZEMISMATCH.xml
- cmrmlsval20020531123456_COLLECTIONMISMATCH.xml
- icmrmlsval20020531123456_DATEMISMATCH.xml

There are several types of Resource Manager/Library Server reports:

Orphan	Entries are added to the ORPHAN report if an object is on the Resource Manager but the Library Server does not have a reference to the object. The report contains information about the object from the Resource Manager database.
Not in RM	Entries are added to the NOTINRM report if the Library Server has a reference to an object but the object is not on the Resource Manager. The report contains information about the object from the Library Server database.
Size mismatch	Entries are added to the SIZEMISMATCH report if the size of an object on the Library Server does not match the size of an object on the Resource Manager. The report contains information about the object from the Resource Manager and Library Server databases.
Collection mismatch	Entries are added to the COLLECTION report if the collection of an object on the Library Server does not match the collection of an object on the Resource Manager. The report contains information about the object from the Resource Manager and Library Server databases.
Date mismatch	Entries are added to the DATEMISMATCH report if the object update date on the Library Server does not match the object update date on the Resource Manager. Under normal circumstances, if there is any synchronization problem between the Library Server and the Resource Manager, the object update date does not match. In order to reduce redundant entries in the different reports, entries are not added to the DATEMISMATCH report if they have been added to the collection mismatch or size mismatch reports. The report contains information about the object from the Resource Manager and Library Server databases.

14.7.7 Resource Manager volume validation utility

The Resource Manager volume validation utility checks each object in its database that was added or changed in a specified date range. It queries the device manager for the attributes of that object and generates reports for each object whose information in the database is different than reported by the device manager. You might want to use the utility if you have a restore data on a volume after a volume crash. The utility helps you to verify that the data is restored correctly. The Resource Manager must be running when you use the utility.

Tip: Use the Resource Manager volume validation utility during times of low traffic on the Resource Manager.

The validation utility does not search the storage system for orphaned objects (objects not referenced by the Resource Manager). Because of the wide variety of storage systems that are often used for storing files other than those managed by Content Manager, scanning for orphaned files can be extremely time consuming and might produce a large quantity of false positives.

The Resource Manager volume validation utility runs on the Resource Manager server and only requires access to its own database and the device managers responsible for the volumes that are being checked.

Starting the Resource Manager volume validation utility

The Resource Manager volume validation utility is icmrmvolval.sh or icmrmvolval.bat. To start the utility, navigate to the bin directory in the Resource Manager home directory.

The Resource Manager volume validation program uses specific input parameters (Table 14-3). Both dashes (-) and forward slashes (/) are handled as the parameter separator. The parameter tags are supported in both lower case and upper case.

Table 14-3 Resource Manager volume validation utility parameters

Parameter	Description
-B *YYYY-MM-DD-HH.MM.SS*	The beginning time and date of the objects to examine. Use this parameter with the -E parameter to restrict the number of objects that the utility must examine. This parameter is optional. If it is not present, all of the objects prior to the -E date are returned, or all of the objects are returned if -E is also not defined.
-E *YYYY-MM-DD-HH.MM.SS*	The ending time and date of the objects to synchronize. Use this parameter with the -B parameter to restrict the number of objects that the utility must examine. This parameter is optional. If it is not present, all of the objects after the -B date are returned, or all of the objects are returned if -B is also not defined.
-F output-path	The absolute path to be used for the output files. The utility creates the UTF-8 XML files in this directory. This parameter is required. If the files currently exist, they are overwritten.

Parameter	Description
-H	This parameter causes the program to display help information about how to invoke the utility. All of the other parameters are ignored and no processing occurs.
-V volume-name	The logical volume name on which you want to perform the validation. Use this parameter to limit the number of storage systems to one volume. This parameter is optional. If not used, all storage systems are searched.

Understanding the validation discrepancy reports

Base file names of the discrepancy reports are `icmrmvolval`*`YYMMDDHHMMSS_Report Type string`*`.xml`. The Report Type string identifies the type of discrepancies a report contains. The description of the different report types are detailed later in this section. The timestamp enables the administrator to run the utility multiple times without overwriting the output files. Examples of default names with the default report type are:

- cmrmvolval20020531123456_FILENOTFOUND.xml
- cmrmvolval20020531123456_SIZEMISMATCH.xml

There are two default discrepancy reports:

File not found	Entries are added to the FILENOTFOUND report if an object is in the Resource Manager database, but it is not found on the volume recorded in the database. A file is considered “not found” if the volume’s device manager reports either that the file does not exist or that the file has a zero file size when the size in the database is non-zero. The report contains the object information from the Resource Manager database.
Size Mismatch	Entries are added to the SIZEMISMATCH report if the size of an object in the Resource Manager database does not match the size reported by the device manager. The report contains the object information from the Resource Manager database and the size reported by the device manager.

14.8 Replacing or repartitioning a hard disk

If a volume or file system that is used by your Resource Manager becomes full, you can replace or repartition the physical disk on which it is located to make more space available.

Replacing or repartitioning the disk invalidates the information stored in the volumes table (RMVOLUMES) for that volume or file system. When updating the Resource Manager volumes, do not run the destager at any point of this process. Otherwise, the volumes will not be the same. Use the following procedures to update the information in the volumes table.

14.8.1 Replacing the staging volume for UNIX

The directory for the staging volume is in the Resource Manager database table, RMSTAGING. Follow these steps to replace the staging volume:

1. Change the permissions on the new staging directory to match those of your Resource Manager ID or what is currently in place for the existing staging directory.
2. If all files in the existing staging directory are currently read-writeable, you can skip this step because these files have been destaged already; otherwise, copy all existing files to the new staging volume:

   ```
   cp -rp current_staging_directory new_staging_directory
   ```

3. Update the location of your staging volume in the Resource Manager database. Open a DB2 command prompt and enter the following commands, each on a new line:

   ```
   db2 "connect to <RM db> user <user ID> using <password>"
   db2 "update rmstaging set sta_path=staging_path"
   ```

   – `<RM db>` is the Resource Manager database (in our scenario, it is rmdb).
   – `<user ID>` is the user ID (in our scenario, icmadmin) used to connect to the Resource Manager database.
   – `<password>` is the password for the user ID.
   – `<staging path>` is the location of the staging directory, as an absolute path with the trailing slash.

14.8.2 Replacing the storage volume for UNIX

The Resource Manager uses the vol_path + the string_table value of lbosdata + collection + num_bucket_value to develop the path. The logical_volume and mount_point are used in various calls to get file system information. Follow these steps to update the Resource Manager storage volume:

1. Stop the Resource Manager.
2. Change the permissions on the new staging directory to match those of your Resource Manager ID or what is currently in place for the existing staging directory.
3. Copy all existing files to the new storage volume:

   ```
   cp -rp current_storage_directory new_storage_directory
   ```

4. Update the location of your storage volume in the Resource Manager database. Use **`df -k`** to determine the FILESYSTEM and MOUNTED ON location for the new storage directory. To update the storage volume, enter the following commands, each on a new line:

   ```
   db2 "connect to <RM db> user <user ID> using <password>"
   db2 "select vol_volumeid,vol_logicalname,vol_mountpoint from
   rmvolumes"
   ```

   – `<RM db>` is the Resource Manager database (in our scenario, it is rmdb).
   – `<user ID>` is the user ID (in our scenario, icmadmin) used to connect to the Resource Manager database.
   – `<password>` is the password for the user ID.

5. Determine which VOLUMEID is the one you need to change. For example, to change VOLUMEID=1, the new logical volume is /dev/data1, and mount point is /rm/data1. Enter:

   ```
   db2 "update rmvolumes set vol_logicalname='/dev/data1 'where
   vol_volumeid=1"
   db2 "update rmvolumes set vol_mountpoint='/rm/data1 'where
   vol_volumeid=1"
   db2 "update rmvolumes set vol_size=0 where vol_volumeid=1"
   db2 "update rmvolumes set vol_path='/rm/data1 'where vol_volumeid=1"
   db2 "update rmvolumes set vol_freespace=0 where vol_volumeid=1"
   ```

6. Start the Resource Manager.

Notice that the latter two steps force the Resource Manager to recalculate the volume space and capacity during any new stores. These values are reflected in the RMVOLUMES tables when the Resource Manager shuts down.

14.8.3 Replacing the staging volume for Windows

The directory for the staging volume is in the Resource Manager database table (RMSTAGING). Follow these steps to replace the staging volumes:

1. Change the permissions on the new staging directory to match those of your Resource Manager ID or what is currently in place for the existing staging directory.
2. If all files in the existing staging directory are currently read-writeable, you can skip this step because these files have been destaged already; otherwise, copy all existing files to the new staging volume:

   ```
   copy -rp current_staging_directory new_staging_directory
   ```

3. Update the location of your staging volume in the Resource Manager database. Open a DB2 command prompt and enter the following commands, each on a new line:

   ```
   db2 "connect to <RM db> user <user ID> using <password>"
   db2 "update rmstaging set sta_path=<staging path>"
   ```

   – `<RM db>` is the Resource Manager database (in our scenario, it is rmdb).
   – `<user ID>` is the user ID (in our scenario, icmadmin) used to connect to the Resource Manager database.
   – `<password>` is the password for the user ID.
   – `<staging path>` is the location of the staging directory, as an absolute path with the trailing slash.

14.8.4 Replacing the storage volume for Windows

If you replace or repartition the hard disk that contains the LBOSDATA directory, you need to identify the new configuration to your system:

1. Stop the Resource Manager.
2. Restore the LBOSDATA directory to the new disk or partition.
3. Open a DB2 command prompt.
4. Manually edit the volumes table to change the following columns to zero for the volume that has been changed. Enter each command on a new line:

   ```
   update rmvolumes set vol_size=0 where vol_volumeid=<ID>
   update rmvolumes set vol_freespace=0 where vol_volumeid=<ID>
   ```

   `<ID>` is the volume ID.

5. The next time the Resource Manager writes or deletes an object, the information is read from the new disk or partition and placed in the volume's table.
6. If your volume is on a different partition, manually edit the RMVOLUMES table to update the VOL_LOGICALNAME and VOL_MOUNTPOINT.

   For example, assume that the volume that you wish to replace is defined in the RMVOLUMES table entry with VOL_VOLUMEID=1. If your new partition is F and this partition is labeled FDRIVE, enter:

   ```
   UPDATE RMVOLUMES set VOL_LOGICALNAME='FDRIVE' where vol_volumeid=1
   UPDATE RMVOLUMES set VOL_MOUNTPOINT='f:' where vol_volumeid=1
   ```
7. Start the Resource Manager.

14.9 Backup

This section covers backup of the Content Manager system.

14.9.1 Backup for Content Manager Multiplatforms

For Content Manager on Multiplatforms, it is important to back up four key components:

- The Library Server database: Use your database manager tools to facilitate this.
- The Resource Manager database: Use your database manager tools to facilitate this.
- The LBOSDATA directory on every Resource Manager.
- Tivoli Storage Manager (TSM) volumes: It is important to remember to back up any data that is migrated to TSM via Content Manager migration policies; otherwise, you will have a single point of failure and data loss within your system. This can be accomplished by using TSM copy storage pools, which might be made up of tape volumes that can be stored off site.

It is not necessary to back up objects in the staging area, because all of the objects in this directory exist elsewhere in the system, so have been backed up already if you use the guideline above.

With these four key components, you can rebuild your Content Manager system, even if the original server is completely destroyed.

If you choose to back up only the four components listed above (as opposed to a full system backup), you need to reinstall the Content Manager code and its software prerequisites onto another machine, in case the original machine is destroyed. The time taken to reinstall a system should be considered when forming a recovery plan, and you must also make sure you have easy access to the installation media, with the original CDs or a network drive.

If at all possible, perform full backups for each of your Content Manager servers. A hierarchical storage management product such as TSM is ideal for this. When choosing the type of media for backing up your system, consider the relative speed of the media. For example, restoring a DB2 database backup that spans multiple magnetic tapes takes much more time than restoring the same database backup from a fiber attached storage area network (SAN).

TSM can also be used to back up database backup images and database logs. These backups can be stored on any type of media that TSM supports; therefore, it is possible to back up DB2 archive logs to tape volumes on TSM automatically, reducing the amount of storage space needed on the server running DB2. DB2 can be integrated with TSM so that DB2 commands can be run as follows:

```
db2 backup database icmnlsdb use tsm
```

For information about integrating DB2 and TSM to provide this type of functionality, refer to *Backing Up DB2 Using Tivoli Storage Manager*, SG24-6247.

Be sure to you back up all components of the Content Manager system together. If you need to restore the system later, each component must be from the same point in time.

1. Identify the LBOSDATA areas. Run the appropriate query for your operating system: for UNIX, select **vol_mountpoint** from rmvolumes; for Windows, select **vol_logicalname** from rmvolumes.
2. Pause the system.
3. Perform the backups. Back up:
   – Library Server database
   – Resource Manager database
   – LBOSDATA areas
   – Data stored in Tivoli Storage Manager

   If possible, before backing up the database, perform **`db2stop`**/**`db2start`** to ensure that there are no clients or services connecting to the database in order to perform full backup. Or, for DB2 UDB V8.2 or higher, use the **`quiesce`** command. For more information about this command, refer to *IBM DB2 Universal Database Command Reference*, SC09-4828.
4. Resume the system.

Pausing DB2 Content Manager for backups

The Library Server PAUSESERVER utility enables you to stop all Content Manager transaction processing in preparation for Library Server and Resource Manager backup processes.

To pause Content Manager, run PAUSESERVER, specifying a future time (UTC). When the system time is equal to or greater than the time that you specify, the Library Server will block all new transactions.

If transactions are processing when the pause time is reached, those transactions will run to completion if they do not exceed the MAXTXDURATION. MAXTXDURATION is a column of the ICMSTSYSCONTROL table. It is a numeric value, indicating the maximum duration (time) allowed in seconds. If a transaction that is processing exceeds the maximum time allowed, it is cancelled and all work owned by the transaction is rolled back.

When all transactions have completed on the Library Server, there will be no client-initiated actions to any Resource Manager, thereby suspending Content Manager and leaving you free to create a consistent backup of all Content Manager servers.

To pause the Library Server, follow these steps:

1. Open a DB2 command prompt.
2. Change to the IBMCMROOT\bin directory.
3. Enter the version of the command for your operating system:
   - UNIX:

     ```
     ./pauseserver.sh dbname userid password SUSPENDSERVERTIME
     ```

   - Windows:

     ```
     pauseserver.bat dbname userid password SUSPENDSERVERTIME
     ```

This command updates the SUSPENDSERVERTIME field in the ICMSTSYSCONTROL table. When that time is less than or equal to the current time, all new transactions are rejected. If an application is storing an object in Resource Manager, those operations will complete if they can do so within the time specified in MAXTXDURATION in ICMSTSYSCONTROL. After that time, all requests to the Library Server are rejected.

For example, you want to pause Content Manager server on the Windows platform at 2005-12-14-16:42:00:000000, the local time. The Library Server database is icmnlsdb; userid is icmadmin; password is password. Run this command:

```
pauseserver icmnlsdb icmadmin password 2005-12-14-16:42:00:000000
```

Resuming DB2 Content Manager after backups

The RESUMESERVER utility enables you to resume transaction processing. To resume Content Manager, run RESUMESERVER, which updates SUSPENDSERVERTIME to null and resumes transaction processing.

To resume Library Server processing, follow these steps:

1. Open a DB2 command prompt.
2. Change to the IBMCMROOT\bin directory.
3. Enter the version of the command for your operating system:
   – UNIX:
     ```
     ./resumeserver.sh dbname userid password
     ```
   – Windows:
     ```
     resumeserver.bat dbname userid password
     ```

14.10 Maintenance review

This chapter has gone into detail about some maintenance-related subjects, and the key points to remember are as follows:

- Monitor object storage space. This includes local and remote storage devices.
- Make sure the Resource Manager and Library Server databases are optimized regularly.
- Make sure the Resource Manager utilities and services that you need in your environment are running and performing their jobs as expected.
- Regularly delete log files that you do not need any more, to prevent them taking up space unnecessarily. This is particularly important when you enable tracing, because the log files will grow in size extremely quickly.
- Back up critical components on a regular basis, and test that these backups work on a regular basis by restoring your system to another machine.

Part 5

Appendixes

A

Performance tuning guidelines for a Document Manager system

A Document Manager system can use Content Manager as its content repository. In this appendix, we provide the performance tuning guidelines for a Document Manager system.

Introduction

The performance tuning guidelines covered here include setting up and configuring the following Document Manager components and services:

- Desktop templates
- Views
- Standard search templates
- Item type configuration
- Class configuration
- Power searches
- Service manager
- Action and dialogs

We also provide general system consideration.

Note: The content provided here is extracted from a white paper written by the Document Manager and Content Manager development team. We include it in this book for your convenience.

Desktop templates

The Desktop template configuration contains a configuration setting that can contribute to a significant increase or decrease in performance and responsiveness by the Document Manager Desktop. This setting is located in the Data Lists tab of the Desktop Template configuration. The Data List Retrieval Mode identifies how the Document Manager Desktop updates three objects: Folders, Groups, and Users. In the Desktop template configuration, these objects can be set for retrieval from cache or directly from the library.

When the Data List Retrieval Mode is set to access from cache data, the Document Manager Desktop always pulls folder, user, or group information from cache. When the Data List Retrieval Mode is set to access from library data, the Document Manager Desktop performs a live lookup to the Library Server to pull folder, user, or group information.

Identify user information needs to determine how these settings should be applied in your environment. These settings can also be different between users and groups. Users and groups that regularly add and change their folders might require the folder data access setting to come from the library. In turn, this setting might mean that it takes a few seconds extra to log into the desktop or refresh a desktop view. Other groups and users whose folder structures are static might want to keep the folder access settings to cache data. This enables faster response time for desktop actions such as log in and desktop refresh.

Views

Document Manager view configurations can have an impact on Document Manager Desktop performance and its responsiveness. A number of elements can combine to affect how quickly the results of a search or the contents of a folder are presented to an end user.

The view configuration contains settings to limit the number of items returned at a time as a result of a search or of opening up a folder. Also included is a setting to identify the number of items to return to the user at a time: the return increment. The larger the number either of these is set to, the longer it takes for the desktop to display the results of a search or the contents of a folder. By using small, manageable numbers, the desktop responsiveness will improve.

When using smaller return increments and maximum return settings, the user base needs to perform well-defined searches to bring back the documents they are looking for.

Another view configuration setting that affects desktop performance is the number of attributed included in a single view. Within the view configuration, the number attributes are assigned to a view template can be adjusted to return fewer attributes to the desktop.

The desktop view configuration also allows configuration of advanced settings to display the organizational structure of a document to include compound documents and life cycle information. Enabling these settings within the view configuration increases the response time for the results of a search or the contents of a document to be displayed to the user.

Standard search templates

Settings in search templates can contribute to desktop performance enhancements. Providing users with key attributes that can help them quickly locate documents and data will enable them to perform well-defined searches.

When assigning attributes to be displayed in a search template, include those attributes that best define your data model and are optimized for performance. Additionally, using default operators such as *equals* provides the best possible response time when executing a search. Operators such as *contains* increases desktop response time for returning search results.

You can define several different search templates that are aligned with different document classes or business processes so that users can quickly identify the attribute information that helps them to locate their data. Additionally, any attributes included on a search dialog should be indexed for optimum database efficiency.

Document Manager systems working with Content Manager contain additional configuration settings within the search template definition that determines how the search is performed in the database. On the Library tab of the Search Configuration dialog, you can specify whether a search is performed across all item types in the library or within a specific item type. This setting affects how quickly the search results are returned to the user because searching across multiple item types requires a database search that is run across multiple tables. The same search run within a single item type returns information much faster because the search is run in a single database table.

Another configuration setting within search template configuration enables you to identify whether the search is case insensitive. Using this setting to perform case-insensitive searches increases the amount of time it takes to return results to the user because this type of search requires the database search to run a full table scan instead of utilizing database indexes.

Item type configuration

Content Manager item type settings can affect performance of the desktop and the overall system. Options such as attribute indexes and text searching can optimize the system's searching performance.

The item-type definition includes the ability to enable full text indexing for the item type. Although enabling text searching does not have a significant impact on desktop performance, it can inhibit users wanting to perform cross-item type searches. If you allow users to perform content searches across all item types, all item types must be set to text searchable.

Content Manager item types also enable the creation of attribute indexes. Indexing can help optimize database searching, which results in better desktop responsiveness and overall system optimization. We recommend creating single attribute indexes for all attributes included in an item type that are being used as search criteria. These include all attributes included on all user dialogs, attributes used to identify uniqueness, and any attributes that are used for locating documents by the Document Manager services such as document class, document state, and revision number.

Class configuration

The document class definition allows the ability to identify an attribute or combination of attributes that the system uses to check document uniqueness. Uniqueness can be checked at several different points in the life cycle: a document add, a document check-in, and the document revise commands can all check for uniqueness.

When uniqueness is checked, the system reads the information set in the Properties tab of the class configuration. Using three or more attributes as uniqueness criteria can increase the amount of time it takes to check uniqueness. For best performance, use the fewest attributes possible to meet your business and configuration needs; not checking for uniqueness at all will provide the best performance possible.

If you do have the system check for uniqueness, it is important to consider creating attribute indexes for all properties used within the properties tab.

You can achieve additional server performance improvements through the settings in the Triggers tab of the document class configuration dialog. These triggers are all processed by the Document Manager Lifecycle Manager whenever a document action is performed. Although this setting does not affect desktop performance, reducing the number of triggers that have to be processed by Lifecycle Manager helps to alleviate unnecessary resource use on the server.

Power searches

The power search command is an advanced level utility that can be made available to users to build complex search requests. The power search was not created to provide highly optimized searches, but it gives users the ability to choose any combination of attributes, operators, and library settings when they run a search.

We recommend making the Power Search command available to power users or more advanced users with the knowledge of how to build a complex search. Used correctly, refined complex searches can be created to return specific documents and data to the user quickly.

User training and education can help provide knowledge and skills around building complex searches. Using operators such as equals and choosing attributes that are indexed help to improve search results response time.

Gain additional performance through the library settings in the Power Search dialog. Similar to the standard search, a Power Search can be performed across multiple item types or within a single item types. The Library tab in the Search Configuration dialog enables the user to identify whether the search will be performed across all item types in the library or against a specific item type.

As with the standard search, searching across multiple item types requires the search to be performed against multiple tables in the database. In contrast, when the search is performed within a single item type, the database search is run against a single table, resulting in increased responsiveness to the end user.

Using the case-insensitive search setting within the power search also results in increased search results responsiveness. Performing case-insensitive searches requires the database query to complete a full table scan instead of using the database indexes. Use of database indexes improves response performance for the search being performed.

Service manager

Document Manager services can be set to minimize resource utilization on the Document Manager server.

Each Document Manager service has a login frequency setting. This setting determines how often the service wakes up and checks for work. The installation default for this setting is to check every 30 seconds.

The interval at which each service is set to process should be evaluated as part of the overall optimization of the Document Manager server. If a service such as Lifecycle or Rendition is not being utilized often or if there is no need to see immediate action, then the frequency interval can be set higher.

Blackout periods should be set for times when system backups are performed. Managing the services through the processing intervals and setting blackout times contributes to system stability and responsiveness.

Setting processing intervals to longer durations frees up resources on the Document Manager server for other operations and system needs. Configuring blackouts to be in effect during database outage times (such as when the database is being backed up) eliminate the effort and processing put forth on the part of the server during those times to connect to a system that is unavailable. Blackouts also contribute to system stability because the Document Manager server will not encounter errors in trying to connect to an unavailable database.

All Document Manager services should be managed (stopped and started) by Document Manager Service Manager. The service manager is designed to allow the Document Manager services to complete any in-process work when a service is stopped.

Additional optimization can be achieved through advanced configuration of the Document Manager services. Lifecycle, Rendition, Print, and Notification service can be configured to manage a specific subset of classes or configurations. The system can be configured to handle specific loads of documents by class to help balance workload and prioritize processing of designated document classes.

Action and dialogs

Options that are available in the actions and dialogs can increase performance time for actions such as add and checkin. Scanning for compound documents, incrementing revision number, checking for uniqueness, and applying unique document numbers can all add a significant amount of time to these processes.

Consider these options and their impact before including them in your system design. If these options are not required as part of your system, they should be disabled to eliminate additional processing time for document operations.

General system considerations

Additional system configurations should be evaluated to keep your Document Manager server running at optimum performance.

Sufficient drive space is required for the Document Manager server. If the Document Manager server's hard drive space is reaching its capacity, Windows virtual memory will be swapped more frequently. This can contribute to a sluggish response by the Document Manager server. Keeping system trace files to verbose logging settings can contribute to the utilization of free disk space. Although system trace files are useful in solving and identifying system problems, provisions should be made to prevent these files from growing too large.

A high degree of drive fragmentation impedes performance.

Virus scanning programs can impact performance. Every document moved either to or from the library repository is temporarily written to the Document Manager server. Every one of those documents may be scanned by virus software depending upon your scan interval settings.

Performance might be improved by the physical locations of the Document Manager server, Library Server, and Resource Manager. Close proximity of these systems can significantly improve performance. Co-locating these systems geographically, on the same Windows domain, or on the same subnet will improve system performance.

A DNS server can affect performance if it becomes unavailable or experiences problems. If the DNS is used to reference machine names, the Document Manager server will have problems locating systems if the DNS server is unavailable.

ODBC drivers on the Document Manager server should be up to date.

Document Manager services are often configured to run as either local or domain Windows accounts. Working with your IT group to ensure their understanding of these accounts and their security requirements helps prevent problems such as disabled or expired accounts.

B

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks™ Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG246949`

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com/redbooks`

Select **Additional materials** and open the directory that corresponds to the redbook form number, **SG246949**.

Using the Web material

Additional Web material that accompanies this book includes the following file:

File name	*Description*
SG246949.zip	Zipped tools to analyze log files

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	200 MB minimum
Operating System:	Windows
Processor:	Pentium IV or higher
Memory:	512 MB

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

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 on ordering these publications, see “How to get IBM Redbooks” on page 447. Note that some of the documents referenced here may be available in softcopy only:

- *AIX 5L Performance Tools Handbook,* SG24-6039
- *AIX 5L Practical Performance Tools and Tuning Guide*, SG24-6476
- *Content Manager Backup/Recovery and High Availability: Strategies, Options, and Procedures*, SG24-7063
- *Content Manager Implementation and Migration Cookbook*, SG24-7051
- *Database Performance Tuning on AIX*, SG24-5511
- *DB2 UDB/WebSphere Performance Tuning Guide*, SG24-6417
- *IBM Tivoli Storage Manager: A Technical Introduction*, REDP0044
- *IBM DB2 UDB V8 and WebSphere V5 Performance Tuning and Operation Guide, SG24-7068*
- *Image and Workflow Library: Capacity Planning and Performance Tuning for VisualInfo and Digital Library Servers*, SG24-4974
- *Tivoli Storage Management Concepts*, SG24-4877

Other publications

These publications are also relevant as further information sources:

- *IBM DB2 Content Manager for Multiplatforms - Planning and Installing Your Content Management System*, GC27-1332
- *IBM DB2 Content Manager for Multiplatforms - System Administration Guide, GC27-1335*
- *Tivoli Storage Manager for AIX - Administrator’s Guide*, GC32-0768

- *IBM DB2 Content Manager Enterprise Edition V8.3: Application Programming Guide*, SC18-9679
- *IBM DB2 Content Manager Enterprise Edition / IBM DB2 Content Manager for z/OS Client V8.3: Client for Windows Programming Reference*, SC27-1337
- *IBM DB2 Universal Database Version 8.2 Administration Guide: Performance*, SC09-4821
- *IBM DB2 Universal Database Version 8.2 Administration Guide: Planning*, SC09-4822
- *IBM DB2 Universal Database Version 8.2 Command Reference*, SC09-4828
- *IBM DB2 Universal Database Version 8.2 SQL Reference Volume 2*, SC09-4845
- *Tivoli Storage Manager Performance Tuning Guide*, SC32-9101

Online resources

These Web sites and URLs are also relevant as further information sources:

- *IBM Content Manager v8.2 Performance Tuning Guide*, by Content Manager Performance Team

  `http://www.ibm.com/support/docview.wss?uid=swg27003894`
- *Tivoli Storage Manager - Performance Tuning Guide*

  `http://www.tivoli.ibm.com/products/solution/storage/docs/tsm-tuning.html`
- *AIX 5L Version 5.1 Performance Management Guide*

  `http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/prftungdtfrm.htm`
- *IBM WebSphere Application Server, Version 5: Getting started*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v5r0/topic/com.ibm.wasee.doc/info/ee/ae/welcgetstarted.html`
- IBM WebSphere Application Server Information Center

  `http://www.ibm.com/software/webservers/appserv/was/library`
- Content Manager production information:

  `http://www.ibm.com/software/data/cm/`
- TechLine Sizing Support Portal (available to IBM representative only):

  `http://w3.ibm.com/support/americas/techline/sizewise.html`

- DB2 production information:

  http://www.ibm.com/software/data/db2/
- JDBC and CLI tracing information (part of the DB2 information Center):

  http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp?topic=/com.ibm.db2.udb.doc/ad/c0007959.htm
- WebSphere Application Server product information:

  http://www.ibm.com/software/webservers/appserv/was/
- WebSphere Application Server InfoCenter:

  http://www.ibm.com/software/webservers/appserv/infocenter.html
- Tivoli Storage Manager home page for TSM information

  http://www.ibm.com/software/tivoli/products/storage-mgr/

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Index

IBM

Redbooks

Performance Tuning for Content Manager

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Performance Tuning for Content Manager

Introduce performance tuning basics

Provide best-practice check lists and base product tuning details

Cover monitoring and tracing tools, troubleshooting techniques

This IBM Redbook deals with performance tuning for IBM DB2 Content Manager Version 8 for Multiplatforms. It is aimed at architects, designers, and system administrators of Content Manager systems.

The book starts with an introduction to performance tuning basics, and define performance and how it is measured. We describe performance methodology and the performance improvement process, along with a set of general guidelines you should use when planning a new system or maintaining and improving an existing system.

After a summary of best practices for Content Manager system performance in a check list, we go into detail about tuning the operating system, DB2, WebSphere Application Server, and Tivoli Storage Manager for Content Manager.

In addition, we include discussion about performance monitoring, analysis, and tracing. Our troubleshooting guidance uses real-life scenarios, and there is a case study to further show you how to troubleshoot. At the end, we include Content Manager maintenance information to ensure that your system is performing at its best.

Written for architects, developers, and system administrators, this book serves as a guide to help you from day one of the system planning and channel you through the right path for a successful running system.

SG24-6949-01 ISBN 0738495646