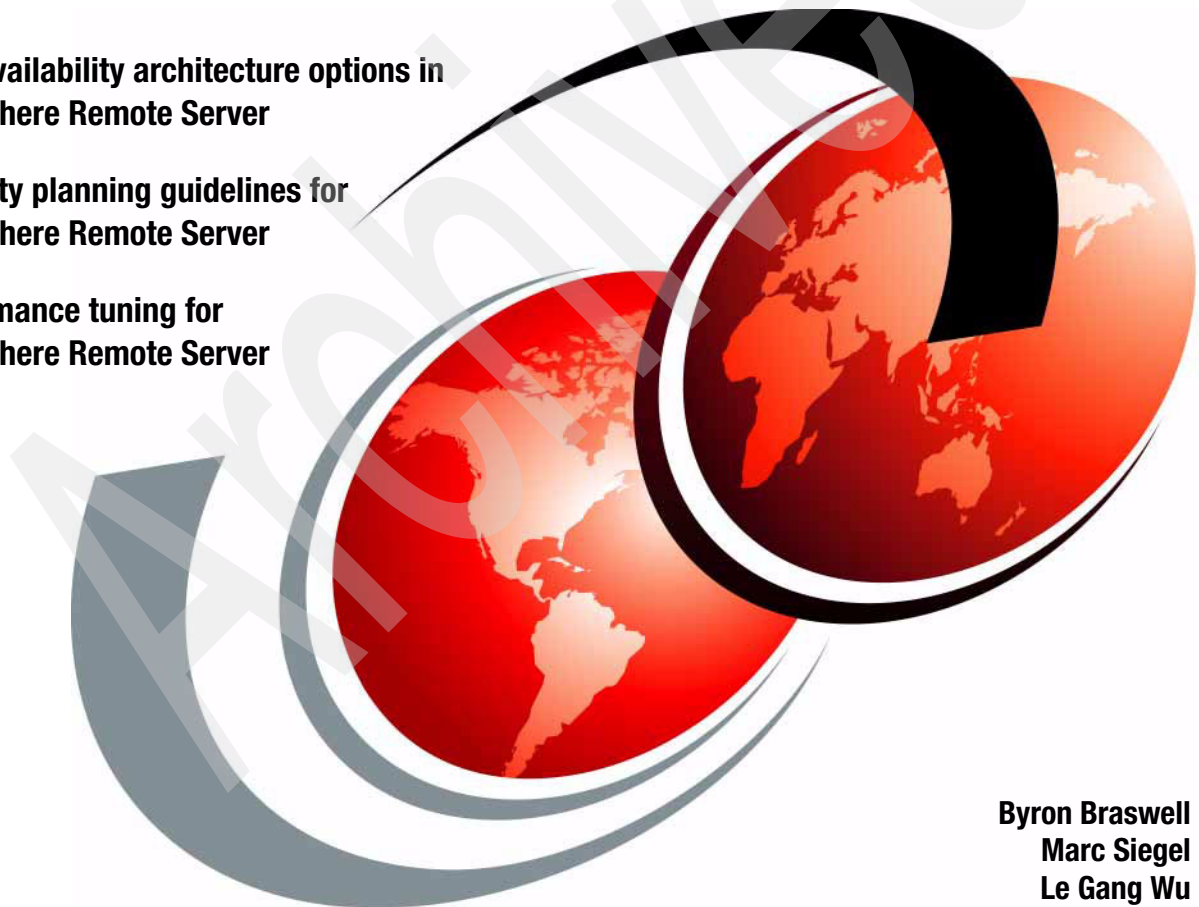


# Highly Available Architectures and Capacity Planning with WebSphere Remote Server V6

High availability architecture options in  
WebSphere Remote Server

Capacity planning guidelines for  
WebSphere Remote Server

Performance tuning for  
WebSphere Remote Server



Byron Braswell  
Marc Siegel  
Le Gang Wu





International Technical Support Organization

**HA Architectures & Capacity Planning with  
WebSphere Remote Server V6**

January 2007

Archived

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

## **Second Edition (January 2007)**

This edition applies to WebSphere Remote Server V6, WebSphere Application Server Network Deployment 6.0.2, WebSphere MQ 6.0.1, IBM DB2 UDB Workgroup Server V8.2.4, Tivoli Configuration Manager V4.2.3, Tivoli Enterprise Console 3.9.4, IBM Tivoli Monitoring 6.1, IBM Tivoli Composite Application Manager for WebSphere V6.0, IBM Tivoli Monitoring for Databases V6.1, and Tivoli Omegamon XE for Messaging V 6.0.

© Copyright International Business Machines Corporation 2005, 2007. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	xi
Trademarks .....	xii
<b>Preface</b> .....	xiii
The team that wrote this redbook .....	xiii
Become a published author .....	xv
Comments welcome .....	xvi
<b>Summary of changes</b> .....	xvii
January 2007, Second Edition .....	xvii
<b>Part 1. Introduction and technology overview</b> .....	1
<b>Chapter 1. Introduction to this redbook</b> .....	3
1.1 The objective .....	4
1.2 Conventions used in this book .....	4
1.3 Target audience of this book .....	5
1.3.1 Roles and skills .....	5
1.3.2 Matching topics in this book to roles and skills .....	7
<b>Chapter 2. WebSphere Remote Server V6 and its components</b> .....	9
2.1 Overview of IBM WebSphere Remote Server .....	10
2.1.1 WebSphere Remote Server V6.0 components .....	11
2.1.2 WebSphere Remote Server tiered offerings .....	12
2.2 WebSphere Application Server V6.0.2 .....	14
2.2.1 WebSphere Application Server Network Deployment .....	15
2.2.2 Support for open standards .....	16
2.3 WebSphere MQ V6.0.1 .....	17
2.4 DB2 UDB 8.2.4 Workgroup Server .....	18
2.5 Tivoli Configuration Manager V4.2.3 .....	19
2.6 IBM Tivoli Monitoring V6.1 .....	19
2.7 Tivoli Monitoring for Databases V6.1 .....	20
2.8 IBM Tivoli Composite Application Manager for WebSphere V6.0 .....	21
2.9 Tivoli Omegamon XE for Messaging V6.0 .....	22
2.10 Tivoli Enterprise Console 3.9.4 .....	23
2.11 WebSphere Remote Server software distribution packages .....	24
2.11.1 Sample environment .....	26
2.11.2 Software requirements .....	27
2.11.3 Hardware requirements .....	28

2.11.4 Components .....	29
2.12 IBM Remote Management Agent .....	30
2.12.1 Prerequisites .....	30
2.12.2 Architecture overview .....	31
2.12.3 Requirements .....	32
2.13 SNMP Trap Mapper .....	34
2.14 Log collection scripts .....	35
<b>Chapter 3. IBM Retail Environment for SUSE Linux.</b> .....	39
3.1 Product overview .....	40
3.2 Architecture .....	41
3.2.1 IBM Retail Environment for SUSE Linux software stack. ....	41
3.2.2 IBM Retail Environment for SUSE Linux software deployment. ....	43
3.3 Components .....	46
3.3.1 LDAP or role-based configuration .....	47
3.3.2 LDAP use and structure .....	47
3.3.3 Server services .....	49
3.4 Building and deploying a point-of-sale image .....	51
3.4.1 Building and deploying a point-of-sale image .....	52
3.4.2 Initial load .....	53
3.4.3 Subsequent loads .....	53
3.5 Requirements .....	54
3.5.1 Supported hardware .....	54
3.5.2 Supported software .....	58
3.6 Additional information .....	59
<b>Part 2. High availability architecture options in WebSphere Remote Server</b> .....	61
<b>Chapter 4. High availability solutions for WebSphere Remote Server.</b> ...	63
4.1 Overview of high availability architecture .....	64
4.2 IBM Tivoli System Automation .....	67
4.2.1 Components of IBM Tivoli System Automation. ....	70
4.3 High availability configuration for WebSphere Remote Server overview .	72
4.4 High availability configuration for WebSphere Remote Server with DRBD	73
4.5 Advanced high availability configuration for WebSphere Remote Server .	78
4.5.1 High availability configuration for IBM HTTP Server. ....	80
4.5.2 High availability configuration for WebSphere Application Server ..	82
4.5.3 High availability configuration for DB2 .....	83
4.5.4 High availability configuration for WebSphere MQ .....	84
4.5.5 High availability configuration for Remote Management Agent ....	85
<b>Chapter 5. Implementing high availability configuration for WebSphere Remote Server using DRBD.</b> .....	87
5.1 Overview of Distributed Replicated Block Device .....	88

5.1.1	How DRBD works .....	88
5.2	Setting up DRBD .....	88
5.2.1	Prerequisites for setting up DRBD .....	88
5.2.2	Installing DRBD .....	89
5.2.3	Configuring DRBD .....	89
5.3	Setting up Tivoli System Automation .....	93
5.3.1	Installing Tivoli System Automation .....	93
5.3.2	Creating a cluster with Tivoli System Automation .....	93
5.3.3	Setting up a tie-breaker .....	94
5.3.4	Installing Tivoli System Automation policies for WebSphere Remote Server .....	95
5.4	Configuring DB2 .....	96
5.5	Configuring WebSphere MQ .....	98
5.5.1	Configuring MQ Broker .....	101
5.6	Configuring WebSphere Application Server .....	102
5.7	Configuring DRBD .....	103
5.8	Hosts file update .....	104
5.9	Application monitoring .....	104
5.10	Failover testing .....	106
<b>Chapter 6. Advanced high availability configuration for WebSphere Remote Server .....</b>		<b>109</b>
6.1	Setting up Tivoli System Automation .....	110
6.2	Configuring high availability for IBM HTTP Server .....	110
6.3	Configuring high availability for WebSphere Application Server .....	112
6.4	Configuring high availability for DB2 .....	117
6.5	Configuring high availability for WebSphere MQ .....	128
6.6	Configuring high availability for Remote Management Agent .....	130
<b>Part 3.</b>	<b>Capacity planning .....</b>	<b>133</b>
<b>Chapter 7. WebSphere Remote Server capacity planning and testing .....</b>		<b>135</b>
7.1	Overview of capacity planning .....	136
7.2	Hardware platforms .....	136
7.3	Software platform .....	137
7.4	Testing tools .....	138
7.5	Testing procedures .....	139
7.5.1	CPU and virtual memory monitoring .....	139
7.5.2	Network monitoring .....	140
7.6	Test results .....	140
7.6.1	Small platform test results .....	141
7.6.2	Medium platform test results .....	142
7.6.3	Large platform test results .....	143
7.7	Test conclusions .....	143

<b>Part 4. Performance tuning</b>	145
<b>Chapter 8. Operating system performance tuning</b>	147
8.1 Tuning the operating system	148
8.1.1 Disabling daemons	148
8.1.2 Shutting down the GUI	151
8.1.3 Compiling the kernel	154
8.1.4 Changing kernel parameters	155
8.1.5 V2.6 Linux kernel parameters	158
8.1.6 Tuning the processor subsystem	161
8.1.7 Tuning the memory subsystem	163
8.1.8 Tuning the file system	164
8.1.9 Tuning the network subsystem	179
8.2 Tuning tools	183
8.2.1 The uptime command	184
8.2.2 The dmesg command	184
8.2.3 The top command	185
8.2.4 The iostat command	188
8.2.5 The vmstat command	190
8.2.6 The sar command	191
8.2.7 KDE System Guard	192
8.2.8 The free command	199
8.2.9 Traffic-vis	200
8.2.10 The pmap command	203
8.2.11 The strace command	203
8.2.12 The ulimit command	204
8.2.13 The mpstat command	206
8.3 Analyzing performance bottlenecks	207
8.3.1 Identifying bottlenecks	207
8.3.2 CPU bottlenecks	211
8.3.3 Memory bottlenecks	213
8.3.4 Disk bottlenecks	216
8.3.5 Network bottlenecks	220
<b>Chapter 9. Tuning the existing environment for better performance</b>	223
9.1 Testing the performance of an application	224
9.2 Tools of the trade	225
9.2.1 Web Performance Tools	226
9.2.2 ApacheBench	226
9.2.3 OpenSystem Testing Architecture	229
9.2.4 Other testing tools	237
9.3 Performance monitoring guidelines	238
9.3.1 Monitoring with Tivoli Performance Viewer and Advisors	238



9.3.2 Performance analysis . . . . .	241
9.4 Performance tuning guidelines . . . . .	246
9.4.1 Crucial tuning parameters . . . . .	247
9.4.2 Parameters to avoid failures . . . . .	247
9.4.3 Hardware and capacity settings . . . . .	248
9.4.4 Adjusting WebSphere Application Server system queues . . . . .	248
9.4.5 Application assembly performance checklist . . . . .	266
9.4.6 Java tuning . . . . .	274
9.4.7 Operating system tuning . . . . .	287
9.4.8 The Web server . . . . .	292
9.4.9 Dynamic Cache Service . . . . .	306
9.4.10 Security settings . . . . .	306
9.4.11 Tuning Secure Sockets Layer . . . . .	307
9.4.12 Object Request Broker . . . . .	309
9.4.13 Extensible Markup Language parser selection . . . . .	311
9.4.14 Transaction service settings: Transaction log . . . . .	311
9.4.15 Additional reference materials . . . . .	312
<b>Chapter 10. Performance tuning of WebSphere MQ . . . . .</b>	<b>315</b>
10.1 Performance factors . . . . .	316
10.1.1 Processing time . . . . .	316
10.1.2 I/O time . . . . .	316
10.1.3 Network transport time . . . . .	317
10.1.4 Message contention . . . . .	317
10.2 Types of WebSphere MQ applications . . . . .	317
10.3 Optimization techniques . . . . .	318
10.3.1 Additional system resources . . . . .	318
10.3.2 Application design . . . . .	318
10.3.3 Logging . . . . .	318
10.3.4 WebSphere MQ configuration . . . . .	319
<b>Chapter 11. Performance tuning of DB2 UDB Workgroup Server . . . . .</b>	<b>321</b>
11.1 Overview of performance tuning . . . . .	322
11.2 Performance related aspects of DB2 architecture . . . . .	322
11.2.1 Buffer pool . . . . .	322
11.2.2 Asynchronous read/write . . . . .	323
11.2.3 Tablespaces and containers . . . . .	325
11.2.4 Database agents . . . . .	326
11.2.5 Concurrency . . . . .	327
11.2.6 SQL . . . . .	330
11.2.7 Maintenance . . . . .	334
11.2.8 Application design . . . . .	334
11.3 Performance optimization tools . . . . .	335

11.3.1 IBM DB2 UDB Performance Expert for Multiplatforms . . . . .	335
11.3.2 IBM DB2 UDB Recovery Expert for Multiplatforms . . . . .	335
11.3.3 IBM DB2 High Performance Unload for Multiplatforms . . . . .	336
11.3.4 IBM DB2 UDB Table Editor for Multiplatforms . . . . .	336
11.3.5 IBM DB2 UDB Web Query Tool for Multiplatforms . . . . .	336
11.4 Monitoring and tuning tools . . . . .	337
11.4.1 Snapshot monitor . . . . .	337
11.4.2 Event monitor . . . . .	347
11.4.3 Explain utilities . . . . .	351
11.4.4 DB2 diagnostic log (DB2DIAG.LOG) . . . . .	353
11.4.5 Health Center and Memory Visualizer . . . . .	354
11.4.6 Design Advisor . . . . .	355
11.4.7 Configuration Advisor . . . . .	355
11.5 Application tuning . . . . .	356
11.5.1 Database design . . . . .	357
11.5.2 SQL tuning . . . . .	367
11.5.3 Stored procedures . . . . .	370
11.5.4 Declared global temporary tables . . . . .	372
11.5.5 Concurrency . . . . .	372
11.6 System tuning . . . . .	375
11.6.1 Tuning the buffer pools . . . . .	375
11.6.2 Table management . . . . .	379
11.6.3 Index management . . . . .	380
11.6.4 Prefetcher . . . . .	383
11.6.5 Cleaner . . . . .	383
11.6.6 Sort heap . . . . .	384
11.6.7 Locking . . . . .	385
11.6.8 Logging . . . . .	389
11.6.9 Tablespace . . . . .	389
<b>Part 5. Appendixes . . . . .</b>	<b>391</b>
<b>Appendix A. Additional material . . . . .</b>	<b>393</b>
Locating the Web material . . . . .	393
Using the Web material . . . . .	394
System requirements for downloading the Web material . . . . .	394
How to use the Web material . . . . .	394
<b>Glossary . . . . .</b>	<b>395</b>
<b>Abbreviations and acronyms . . . . .</b>	<b>405</b>
<b>Related publications . . . . .</b>	<b>409</b>
IBM Redbooks . . . . .	409

IBM product documentation ..... 410

Other publications ..... 411

Online resources ..... 412

How to get IBM Redbooks ..... 414

Help from IBM ..... 414

**Index** ..... 415

Archived

Archived

# Notices

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

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

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

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

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

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

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

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

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

## COPYRIGHT LICENSE:

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

# Trademarks

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

Redbooks (logo) ™  
developerWorks®  
ibm.com®  
iSeries™  
pSeries®  
xSeries®  
z/OS®  
zSeries®  
AIX®  
BladeCenter®  
CICS®  
Domino®  
DB2 Connect™  
DB2 Universal Database™

DB2®  
Informix®  
IntelliStation®  
IBM®  
IMS™  
Lotus®  
OMEGAMON®  
OS/2 WARP®  
OS/2®  
OS/390®  
OS/400®  
Rational Suite®  
Rational®  
Redbooks™

Resource Link™  
RS/6000®  
ServeRAID™  
SureOne®  
SurePOS™  
System x™  
TestStudio®  
Tivoli Enterprise™  
Tivoli Enterprise Console®  
Tivoli®  
TotalStorage®  
WebSphere®

The following terms are trademarks of other companies:

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

iPlanet, Enterprise JavaBeans, EJB, IPX, Java, JavaBeans, JavaPOS, JavaScript, JavaServer, JavaServer Pages, JDBC, JDK, JMX, JRE, JSP, JVM, J2EE, Solaris, Sun, Sun ONE, Ultra, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

i386, Intel, Pentium, Xeon, 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

IBM WebSphere® Remote Server V6 delivers a fully-integrated platform that helps manage remote environments such as retail stores and branch offices. WebSphere Remote Server is a key component of the Store Integration Framework and on demand operating environment. This infrastructure offering extends the IBM Enterprise Business Integration technology to distributed locations, enabling integration from the enterprise to the edge of the business.

WebSphere Remote Server helps retailers to manage their business more cost-effectively, increase employee productivity, and create a unique shopping experience for their customers. Employees have better access to customers, products, and sales information, thereby increasing productivity and providing better service to customers.

This IBM Redbook introduces highly available architectures using IBM WebSphere Remote Server, capacity planning for store environments, and performance tuning for operating systems and WebSphere Remote Server. This information can help IBM Clients and Business Partners integrate these tools into enterprise retail environments.

The high available architecture scenarios and the performance and tuning scenarios were developed and documented in a WebSphere Remote Server V5.1.2.1 environment.

This book also discusses the underlying and related technologies, including the installation and configuration processes. The topics covered include architecture, design, development, customization, deployment, and administration. In addition, this book will help you configure and optimize WebSphere Remote Server in an IBM Retail Environment for SuSE Linux® V2 environment.

The target audience for this book includes IBM Clients, Business Partners, and IBM personnel who implement Store Integration Framework projects. Key roles include project manager, IT architect, data base administrator, store application developer, store device developer, solution developer, and IT administrator.

## The team that wrote this redbook

This IBM Redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Raleigh Center.

**Byron Braswell** is a project leader and networking professional at the ITSO, Raleigh Center. He writes extensively in the areas of networking, application integration middleware, and personal computer software. Before joining the ITSO, he worked in IBM Learning Services Development in the field of networking education development. Byron has a Bachelor of Science degree in Physics and a Master of Science degree in computer sciences from Texas A&M University.

**Marc Siegel** is a Software Engineer for the IBM Software Group. After 10 years in various positions in the software industry, he joined IBM in March of 1998. His focus while with IBM has been with retail and eCommerce. His areas of expertise include system and functional testing of IBM retail solutions and middleware.

**Le Gang Wu** is a solution architect/consultant from IBM Development Lab in China. He has been involved in developing solutions for clients in the retail industry and business-to-business (B2B) area using IBM Middleware projects for three years. His areas of expertise include Retail In-Store Systems and B2B solutions. Le Gang holds a master degree in industry control science and engineering from ZheJiang University in China.



*The team: Byron, Marc, Le Gang*

The following authors contributed to the previous edition of this redbook:

Ryan Brown  
Ivor Castelino  
Kevin Collins



Maria Koshkina  
Anant Kumar  
Chris McCollum

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

Margaret Ticknor  
Linda Robinson  
Sarita Povaiah  
Carolyn Sneed  
Tamikia Barrow  
ITSO, Raleigh Center

Patricia Boerstler  
IBM Software Group, AIM Software, Project Manager - SWG Industry Solutions

Barry Foster  
IBM Software Group, AIM Software, Retail Industry Solutions Test

Andrew R Freed  
IBM Software Group, AIM Software, Staff Software Engineer, WebSphere  
Remote Server

Norman Korn  
IBM Software Group, AIM Software, Retail EBO Enablement Manager

Samuel J Robertson  
IBM Software Group, AIM Software, Applianceware Development

Albert T Wong  
IBM Software Group, AIM Software, IBM I/T Architect, Retail On Demand EBO

## **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:

## 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](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- ▶ Mail your comments to:

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

# Summary of changes

This section describes the technical changes 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 SG24-7184-01  
for HA Architectures & Capacity Planning with WebSphere Remote Server V6  
as created or updated on December 28, 2006.

## January 2007, Second Edition

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

### Updated information

Chapter 2, “WebSphere Remote Server V6 and its components” on page 9 and Chapter 3, “IBM Retail Environment for SUSE Linux” on page 39 have been updated to the WebSphere Remote Server V6 level. The content has also been enhanced with additional material on the updated components of WebSphere Remote Server V6 and the WebSphere Systems Management Accelerators.

The chapters in Part 2, “High availability architecture options in WebSphere Remote Server” on page 61 to Part 4, “Performance tuning” on page 145 of this book have been edited and corrected for format and grammatical clarity. However, the high availability scenarios, capacity planning scenario, and performance tuning recommendations are based on a WebSphere Remote Server V5.1.2.1 test environment.





# Part 1

## **Introduction and technology overview**

Part 1 describes the goals and target audience of this IBM Redbook. It also reviews the underlying software and technologies that contribute to high availability architectures and capacity planning.



# Introduction to this redbook

This chapter describes the objective and focus of this IBM Redbook and provides details about who can benefit from reading the book. It also includes references to sources of additional information.

This chapter includes the following sections:

- ▶ 1.1, “The objective” on page 4
- ▶ 1.2, “Conventions used in this book” on page 4
- ▶ 1.3, “Target audience of this book” on page 5

## 1.1 The objective

The objective of this book is to introduce the concepts of highly available architectures using the IBM WebSphere Remote Server, capacity planning for store environments, and performance tuning for operating systems and WebSphere Remote Server, in order to help IBM Business Partners and Clients integrate these tools into enterprise retail environments.

In its attempt to meet this objective, the book provides detailed information about the underlying and related technologies, as well as the installation and configuration processes.

## 1.2 Conventions used in this book

In writing this guide, we attempted to follow established standards and practices used in other documentation so that it is easy to read and contain familiar terminology.

This book uses the following highlighting conventions:

- ▶ **Bold** type indicates graphical user interface (GUI) controls such as buttons or menu choices.
- ▶ Monospaced type indicates examples of text that you enter exactly as shown, and directory paths.
- ▶ **Command** type indicates commands.
- ▶ *Italic* type is used for variables, for which you substitute your own values. These values should be used as listed unless the value *host\_name* is included. In such a scenario, a valid host name should be substituted in place of *host\_name*.
- ▶ [Blue](#) type indicates linkable Web addresses.

This book uses the following conventions for specific values that must be substituted with the actual values:

- ▶ *was\_home* indicates the installation directory for IBM WebSphere Application Server. The default installation directory for WebSphere Application Server on the Microsoft® Windows® 2003 operating system is C:\Program Files\WebSphere\AppServer.
- ▶ *wsad\_home* indicates the installation directory for IBM WebSphere Studio Application Developer. The default installation directory on the Microsoft Windows 2003 operating system is C:\Program Files\WebSphere\WSAD.



- ▶ *host\_name* indicates a fully qualified host name to be substituted in a Web address, for example, *myhost.com*.
- ▶ *node\_name* indicates the short name of the computer. This is the first part of the *host\_name*, without the domain, for example, *myhost*.
- ▶ *server\_name* indicates the name of the application server in the WebSphere Application Server.
- ▶ *dbname* indicates the name of the database.
- ▶ *dbuser* indicates the user ID that owns the schema in the database.
- ▶ *dbpassword* indicates the password for the associated *dbuser*.

**Note:** All references to operating system-specific functions refer to the Microsoft Windows platform unless otherwise indicated.

## 1.3 Target audience of this book

As is the nature of a handbook, this book is multipurpose in its intent. It includes details about varied topics, including architecture, design, development, customization, deployment, and administration. The target audience can be best matched by role to the topic of interest within the book.

### 1.3.1 Roles and skills

Several common roles are required for a team to execute a Store Integration Framework implementation project during the development life cycle. In this section, we define the key roles and skills to be used as a cross-reference with the topics of interest within the book. These roles include:

- ▶ Project manager
- ▶ IT architect
- ▶ Database administrator
- ▶ Store application developer
- ▶ Store device developer
- ▶ Solution deployer
- ▶ IT administrator

#### **Project manager**

The project manager is responsible for managing and leading the project team through all the phases of the project. The project manager also acts as a single point of contact to interact with the client. The designated person should have a

general understanding of the Store Integration Framework and be familiar with the product architecture.

### **IT architect**

The IT architect looks after the project's overall technical architecture and design, provides quality assurance, transfers knowledge to clients, and mentors the project's technical team. The architect should have WebSphere Application Server architecture and design skills. Based on the project's scope and complexity, one or more architects can work to create detailed technical designs. The technical designs are developed with the assistance of the lead developer.

### **Database administrator**

The database administrator (DBA) is responsible for the creation and administration of all databases and required storage requirements. This person is also in charge of the ongoing database operations.

### **Store application developer**

The store application developer is either an enterprise IT staff member or an independent software vendor (ISV) IT staff member. This person is responsible for developing and maintaining the application software that is intended for use in stores.

### **Store device developer**

The store device developer is either an enterprise IT staff member or an ISV IT staff member, who is responsible for building new hardware devices for use in stores.

### **Solution deployer**

The solution deployer is an enterprise IT staff member responsible for packaging and deploying applications on the in-store processor, and applying the application patches as necessary.

### **IT administrator**

The IT administrator is an enterprise IT staff member responsible for monitoring the health of the store IT infrastructure, that is, in-store processors (ISPs), devices, point-of-sale (POS), and applications. The IT administrator is also responsible for managing the same remotely.

## 1.3.2 Matching topics in this book to roles and skills

Table 1-1 provides a summary of the topics in this book by part and chapter, aligning them with the defined roles and skills.

Table 1-1 Matching book topics to roles and skills

Part	Chapter	Primary	Secondary
<b>Part 1, “Introduction and technology overview”</b>			
	Chapter 1, “Introduction to this redbook” on page 3	All roles	
	Chapter 2, “WebSphere Remote Server V6 and its components” on page 9	All roles	
	Chapter 3, “IBM Retail Environment for SUSE Linux” on page 39	All roles	
<b>Part 2, “High availability architecture options in WebSphere Remote Server”</b>			
	Chapter 4, “High availability solutions for WebSphere Remote Server” on page 63	IT architect Solution deployer	IT admin
	Chapter 5, “Implementing high availability configuration for WebSphere Remote Server using DRBD” on page 87	IT architect Solution deployer	IT admin
	Chapter 6, “Advanced high availability configuration for WebSphere Remote Server” on page 109	IT architect Solution deployer	IT admin
<b>Part 3, “Capacity planning”</b>			
	Chapter 7, “WebSphere Remote Server capacity planning and testing” on page 135	IT architect	IT admin
<b>Part 4, “Performance tuning”</b>			
	Chapter 8, “Operating system performance tuning” on page 147	IT admin	IT architect
	Chapter 9, “Tuning the existing environment for better performance” on page 223	IT admin	IT architect
	Chapter 10, “Performance tuning of WebSphere MQ” on page 315	IT admin	IT architect
	Chapter 11, “Performance tuning of DB2 UDB Workgroup Server” on page 321	IT admin DBA	IT architect
	Appendix A, “Additional material” on page 393	IT admin	IT architect



# WebSphere Remote Server V6 and its components

WebSphere Remote Server is a collection of IBM middleware products and is installed on a server in a store. This server is called the In-Store Processor (ISP).

WebSphere Remote Server is intended to be a base software stack for running Web applications within a store. The middleware products include WebSphere Application Server, DB2® and WebSphere MQ. In addition, WebSphere Remote Server includes IBM Tivoli® Monitoring products for monitoring the middleware previously mentioned, and IBM Remote Management Agent for monitoring store devices.

This chapter provides an overview of IBM WebSphere Remote Server V6 and its major components. It contains the following sections:

- 2.1, “Overview of IBM WebSphere Remote Server” on page 10
- 2.2, “WebSphere Application Server V6.0.2” on page 14
- 2.3, “WebSphere MQ V6.0.1” on page 17
- 2.4, “DB2 UDB 8.2.4 Workgroup Server” on page 18
- 2.5, “Tivoli Configuration Manager V4.2.3” on page 19
- 2.6, “IBM Tivoli Monitoring V6.1” on page 19
- 2.7, “Tivoli Monitoring for Databases V6.1” on page 20
- 2.8, “IBM Tivoli Composite Application Manager for WebSphere V6.0” on page 21

## 2.1 Overview of IBM WebSphere Remote Server

The In-Store Processor is a server in the store. This Linux or Windows based server has several devices connected to it, including point-of-sale (POS) controllers, kiosks, self-checkout systems, and other devices. The In-Store Processor runs all the software that the store operations require. WebSphere Remote Server is installed onto the In-Store Processor to form a base platform for other applications.

IBM WebSphere Remote Server delivers a fully-integrated platform to help manage remote environments such as retail stores and branch offices. This infrastructure offering extends the IBM Enterprise Business Integration technology to distributed locations, enabling integration from the enterprise to the edge of the business. Retailers can manage their business more cost effectively, help increase employee productivity, and create a unique shopping experience for consumers. Employees have better access to customer, product, and sales information, increasing productivity and providing better service to consumers.

WebSphere Remote Server is a key component of the Store Integration Framework and an on demand operating environment (ODOE). Key benefits include:

- ▶ Provides an ODOE by leveraging IBM WebSphere Application Server, IBM DB2 information management, IBM WebSphere MQ advanced messaging, and IBM Tivoli management and monitoring software.
- ▶ Streamlines store operations to cost-effectively integrate existing and new technologies, such as the following:
  - Mobile shopping devices with personalized information at the consumer's fingertips
  - Transaction log (TLOG) handling for non-IBM 4690 operating system (OS) applications to move POS data to headquarters
  - Radio Frequency Identification (RFID) technology to speed the checkout process and locate merchandise in the store
  - Digital merchandising solutions that can provide consumers with dynamic product information and comparisons on demand
  - Accurate inventory information to help prevent out of stock conditions and erroneous reordering of merchandise
- ▶ Manages store middleware by providing a step-by-step approach to store operations to efficiently transform stores
- ▶ Provides a platform for store innovation to differentiate customers' shopping experiences

- ▶ Provides a robust, standards-based platform for future expansion

WebSphere Remote Server provides both the store and enterprise aspects. The store aspect features a robust, standards-based, and highly available middleware platform for application solutions with business and operating monitoring. The enterprise aspect focuses on building, deploying, and managing the middleware to remote locations.

### **2.1.1 WebSphere Remote Server V6.0 components**

The WebSphere Remote Server V6.0 includes the following components:

- ▶ 2.2, “WebSphere Application Server V6.0.2” on page 14
- ▶ 2.3, “WebSphere MQ V6.0.1” on page 17
- ▶ 2.4, “DB2 UDB 8.2.4 Workgroup Server” on page 18
- ▶ 2.5, “Tivoli Configuration Manager V4.2.3” on page 19
- ▶ 2.6, “IBM Tivoli Monitoring V6.1” on page 19
- ▶ 2.7, “Tivoli Monitoring for Databases V6.1” on page 20
- ▶ 2.8, “IBM Tivoli Composite Application Manager for WebSphere V6.0” on page 21
- ▶ 2.9, “Tivoli Omegamon XE for Messaging V6.0” on page 22
- ▶ 2.10, “Tivoli Enterprise Console 3.9.4” on page 23

## 2.1.2 WebSphere Remote Server tiered offerings

Figure 2-1 illustrates the WebSphere Remote Server lineup of tiered offerings with three in-store configurations designed to provide a more flexible and affordable platform to help you to manage your business costs effectively.

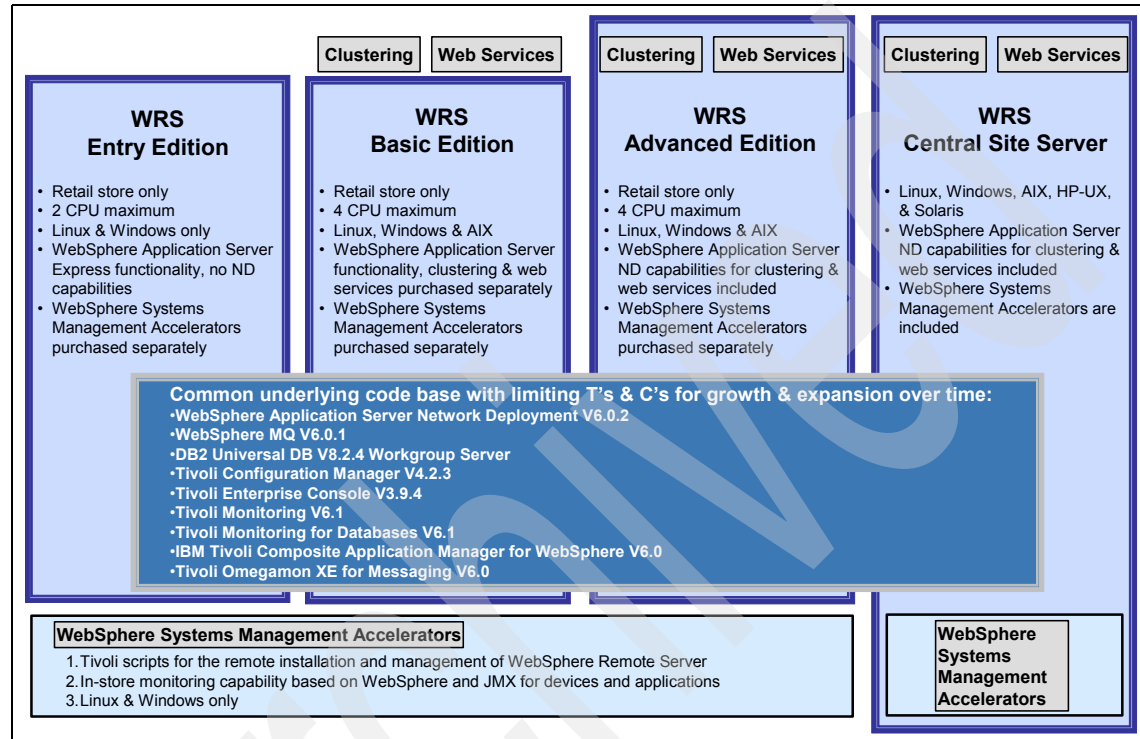


Figure 2-1 WebSphere Remote Server tiered offerings

### WebSphere Remote Server Entry V6.0

WebSphere Remote Server Entry V6.0 offers a cost-effective, preintegrated foundation platform for the store. This component delivers application server, messaging, and database function restricted to the same levels independently offered by WebSphere Application Server Express, WebSphere MQ, and DB2 Express through product-unique terms. Additionally, this component contains the necessary Tivoli management agents and enterprise tooling for remotely deploying and monitoring the store platform, independently offered as Tivoli Configuration Manager, Tivoli Enterprise™ Console, Tivoli Monitoring, Tivoli Monitoring for Databases, Tivoli Composite Application Manager for WebSphere, and Tivoli Omegamon XE for Messaging. The WebSphere Remote Server Entry option is restricted for use on machines with no more than two CPUs and there



are additional restrictions for WebSphere Application Server Network Deployment and DB2 UDB Workgroup Server Edition.

### **WebSphere Remote Server Basic V6.0**

WebSphere Remote Server Basic V6.0 provides an integrated foundation and enables premium Web services and high availability capabilities to be added on separately when the needs of a client expand. This component delivers application server function restricted to the level of WebSphere Application Server through product-unique terms, as well as functionality included in DB2 UDB Workgroup Server Edition and WebSphere MQ. Additionally, this component contains the necessary Tivoli management agents and enterprise tooling for remotely deploying and monitoring the store platform, independently offered as Tivoli Configuration Manager, Tivoli Enterprise Console®, Tivoli Monitoring, Tivoli Monitoring for Databases, Tivoli Composite Application Manager for WebSphere, and Tivoli Omegamon XE for Messaging. The WebSphere Remote Server Basic option is restricted to machines with no more than four CPUs and there are additional restrictions for WebSphere Application Server Network Deployment. The two features available as add-ons to WebSphere Remote Server Basic are:

- ▶ IBM Web services feature for WebSphere Remote Server V6.0  
This extends WebSphere Remote Server Basic by allowing incremental adoption of Web Services Gateway and Universal Description, Discovery and Integration (UDDI). These capabilities provide the foundation for a Service Oriented Architecture (SOA) for in-store applications.
- ▶ IBM clustering and high availability feature for WebSphere Remote Server V6.0  
This extends the WebSphere Remote Server Basic product by allowing incremental adoption of high availability features and configuration flexibility through workload balancing.

### **WebSphere Remote Server Advanced V6.0**

WebSphere Remote Server Advanced V6.0 is the IBM on demand software infrastructure optimized for store servers. It provides a scalable, flexible IT infrastructure for the store. This functionally-rich platform delivers the full function of DB2 UDB Workgroup Server Edition, WebSphere MQ, and WebSphere Application Server Network Deployment, and provides premium Web services and a high availability platform for in-store processors. Additionally, this component contains the necessary Tivoli management agents and enterprise tooling for remotely deploying and monitoring the store platform, independently offered as:

- ▶ Tivoli Configuration Manager
- ▶ Tivoli Enterprise Console

- ▶ Tivoli Monitoring
- ▶ Tivoli Monitoring for Databases
- ▶ Tivoli Composite Application Manager for WebSphere
- ▶ Tivoli Omegamon XE for Messaging

The WebSphere Remote Server Advanced option is restricted to machines with no more than four CPUs.

### **WebSphere Central Site Server V6.0**

WebSphere Central Site Server V6.0 provides a convenient package of DB2 UDB Workgroup Server Unlimited Edition, WebSphere MQ, and WebSphere Application Server Network Deployment with a copy of WebSphere Systems Management Accelerators to establish the enterprise environment, providing a centrally located server to communicate with remote locations.

### **WebSphere Systems Management Accelerators V6.0**

WebSphere Systems Management Accelerators V6.0 provides a quick and convenient way for clients to deploy and manage the Advanced, Basic, or Entry options of WebSphere Remote Server infrastructure including convenient packaging for WebSphere Application Server Network Deployment, DB2 UDB Workgroup Server Unlimited Edition, WebSphere MQ, and Remote Management Agent components.

## **2.2 WebSphere Application Server V6.0.2**

The WebSphere Application Server family of interoperable products provides a next-generation application server on an industry-standard foundation. The IBM WebSphere Application Server family is divided into the following packages for Version 6. Each edition addresses a distinct set of scenarios and needs.

WebSphere Application Server includes:

- ▶ WebSphere Application Server - Express

This edition is a lightweight server for static content, servlets, and JavaServer™ Pages™ (JSP™) files. This edition does not support enterprise beans.

- ▶ WebSphere Application Server

This edition addresses the basic programming and runtime needs of desktop developers and single-server production scenarios. The runtime environment for this edition addresses standards-based programming for the Web and component-based programming, as well as Web services.

The administration model for this edition presumes a single-server environment with no clustering for failover or workload balancing, and with no centralized administration of multiple server instances. However, you can add standalone nodes to a centrally-administered network (the cell) at any time after installing WebSphere Application Server Network Deployment, which controls the cell.

- ▶ **WebSphere Application Server Network Deployment**

This edition addresses application servers that run in multiple-server production scenarios. It provides centralized administration and basic clustering and caching support.

- ▶ **WebSphere Application Server for z/OS®**

This edition integrates the WebSphere Application Server product and the Network Deployment product into a single package that leverages the unique qualities of service inherent to the IBM z/OS platform. It addresses standards-based programming for the Web and component-based programming, as well as Web services, and provides centralized administration of multiple-server instances and basic clustering and caching support.

## **2.2.1 WebSphere Application Server Network Deployment**

WebSphere Application Server Network Deployment is bundled with WebSphere Remote Server. It is built on top of WebSphere Application Server and provides an operating environment with advanced performance and availability capabilities in support of dynamic application environments. In addition to all of the features and functions within the base WebSphere Application Server, this configuration delivers advanced deployment services that include clustering, edge-of-network services, Web services enhancements, and high availability for distributed configurations.

WebSphere Application Server Network Deployment provides the following benefits:

- ▶ Provides Universal Description, Discovery, and Integration (UDDI) V3, enabling you to describe and discover Web services in a more secure manner
- ▶ Delivers advanced Web services security to enhance the security of Web services' interactions
- ▶ Provides the Web services gateway, which enables Web services invocation by users from outside the firewall with the benefit of robust security protection
- ▶ Supports advanced failover and clustering capabilities, including:
  - Simplified administration using the development tool interface

- Browser-based administration for remote administration across firewalls
- Convenient administration through embedded administrative console
- Intelligent workload distribution across a cluster
- Failure bypass
- Clustering support
- Edge Components (formerly know as Edge Server), which delivers sophisticated load balancing, caching, and centralized security capabilities

## 2.2.2 Support for open standards

The WebSphere Application Server platform includes broad support for open, industry-supported standards. It offers a wide range of application programming interfaces (APIs) for XML, Java™, and Web services covering 35 different platforms, including Microsoft Windows, Linux, and other operating systems. Some of the supported standards are:

- ▶ Web services standards including Simple Object Access Protocol (SOAP), Universal Description, Discovery, and Integration (UDDI), Web Services Description Language (WSDL), Web Services Inspection Language (WSIL), and WS-I Basic Profile
- ▶ Java 2 Platform, Enterprise Edition (J2EE™), Enterprise JavaBeans™ (EJB™)
- ▶ HTML, cascading style sheets (CSS), and JavaScript™
- ▶ XML standards, including Extensible StyleSheet Language (XSL) and Extensible HTML (XHTML)
- ▶ Unified Modeling Language (UML)

For more information, refer to the WebSphere Application Server V6.0.x information center available on the Web at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

## 2.3 WebSphere MQ V6.0.1

IBM WebSphere MQ provides assured, once-only delivery of messages across more than 35 industry platforms using a variety of communications protocols. The transportation of message data through a network is made possible through the use of a network of WebSphere MQ queue managers. Each queue manager hosts local queues that are containers used to store messages. Through remote queue definitions and message channels, data can be transported to its destination queue manager.

To use the services of a WebSphere MQ transport layer, an application must make a connection to a WebSphere MQ queue manager, the services of which enable it to receive (*get*) messages from local queues or send (*put*) messages to any queue on any queue manager. The application's connection can be made directly (where the queue manager runs locally to the application) or as a client to a queue manager that is accessible over a network.

*Dynamic workload distribution* is another important feature of WebSphere MQ. This feature shares the workload among a group of queue managers that are part of the same cluster. This enables WebSphere MQ to balance the workload across available resources automatically and provide hot standby capabilities if a system component fails. This is a critical feature for companies that need to maintain around-the-clock availability.

WebSphere MQ supports a variety of application programming interfaces (including Message Queue Interface (MQI), Application Messaging Interface (AMI), and Java Message Service (JMS)), which provide support for several programming languages and point-to-point and publish or subscribe communication models. In addition to support for application programming, WebSphere MQ provides several connectors and gateways to a variety of other products, such as Microsoft Exchange, IBM Lotus® Domino®, SAP/R3, IBM CICS®, and IBM IMS™, to name just a few.

WebSphere MQ provides support for delivering XML documents and SOAP messages. It connects applications using Web services and provides support for the JMS interface standard. It offers security using the Internet standard Secure Sockets Layer (SSL).

You can find additional details about WebSphere MQ at the information center on the Web at:

<http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp>

For more information, refer to the IBM WebSphere MQ Web site:

<http://www.ibm.com/software/integration/wmq/>

## 2.4 DB2 UDB 8.2.4 Workgroup Server

IBM DB2 Universal Database™ 8.2.4 Workgroup Server is a multiuser version of DB2 Universal Database (UDB) that enables you to create and manage single partitioned or partitioned database environments. Partitioned database systems can manage high volumes of data and provide benefits such as high availability and increased performance. Other features include:

- ▶ A data warehouse server and related components
- ▶ DB2 Connect™ functionality for accessing data stored on midrange and mainframe database systems
- ▶ Satellite administration capabilities

DB2 UDB 8.2.4 Workgroup Server delivers new features to address the ever increasing demands and requirements on important data. These include:

- ▶ Administrators can reap immediate benefits from the broadened autonomic computing solutions offered in DB2 UDB 8.2.4 Workgroup Server. These solutions automate and simplify potentially time-consuming and complex database tasks.
- ▶ Application developers can reap the benefits from a significant amount of new capabilities and further integration of DB2 tooling into the Microsoft .NET and WebSphere Java environments. This simplifies the development and deployment of DB2 applications, enabling application developers to take advantage of the openness, performance, and scalability of DB2, without regard to the back-end database or the chosen application architecture.
- ▶ IT managers and administrators will benefit from the integration of industry-proven high-availability disaster recovery technology available in DB2 UDB 8.2.4 Workgroup Server. Line-of-business managers and the enterprise itself benefit the most, because applications face less risk of downtime.

Therefore, regardless of the role or size of business, DB2 UDB 8.2.4 Workgroup Server has exciting new features that can assist with daily challenges.

For more information, refer to the IBM DB2 Universal Database Workgroup Server information center available on the Web at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp>

## 2.5 Tivoli Configuration Manager V4.2.3

IBM Tivoli Configuration Manager provides the ability to capture your best practices for software distribution, automate these best practices, and enforce corporate standards. It helps you gain total control over your heterogeneous enterprise software and hardware. The software distribution module enables you to rapidly and efficiently deploy complex, mission-critical applications to multiple locations from a central point. The inventory module lets you automatically scan for and collect hardware and software configuration information from computer systems across your enterprise. This inventory configuration data can be used in the reference models to automatically remediate systems that are not compliant. Making sure patches are installed to reduce vulnerabilities is an example of where you can use this feature.

The software distribution capabilities of Tivoli Configuration Manager enable you to deploy software and patches across your enterprise, including pervasive devices. It is a key solution for customers who require rapid, centralized application deployment. The software and patch deployment life cycle has many steps, and Tivoli Configuration Manager helps you manage them through its desired state management model. From packaging, planning, and administration to delivery, installation, and reporting, Tivoli Configuration Manager gives you the tools you require.

You can find more details about the Tivoli Configuration Manager at the Tivoli information center available on the Web at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?toc=/com.ibm.itmwi.doc/toc.xml>

For more information, refer to the Tivoli Configuration Manager Web site at:

<http://www.ibm.com/software/tivoli/products/config-mgr/>

## 2.6 IBM Tivoli Monitoring V6.1

IBM Tivoli Monitoring provides monitoring for essential system resources to detect bottlenecks and potential problems and to automatically recover from critical situations. IBM Tivoli Monitoring saves system administrators from manually scanning through extensive performance data before problems can be solved. Using industry best practices, Tivoli Monitoring can provide immediate value to the enterprise. Combined with IBM Tivoli Enterprise Console, it can provide a true end-to-end solution.

You can use IBM Tivoli Monitoring to perform the following tasks:

- ▶ Visualize real-time monitoring data from your environment.
- ▶ Monitor resources in your environment for certain conditions, such as high CPU or an unavailable application.
- ▶ Establish performance thresholds and raise alerts when thresholds are exceeded or values are matched.
- ▶ Trace the causes leading up to an alert.
- ▶ Create and send commands to systems in your managed enterprise by means of the Take Action feature.
- ▶ Use integrating reporting to create comprehensive reports about system conditions.
- ▶ Monitor conditions of particular interest by defining custom queries using the attributes from an installed agent or from an ODBC-compliant data source.

This release of Tivoli Monitoring includes the Tivoli Enterprise Portal.

You can find additional details about Tivoli Monitoring at the Tivoli Monitoring information center on the Web at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?toc=/com.ibm.itm.doc/toc.xml>

For more information, refer to the IBM Tivoli Monitoring Web site at:

<http://www.ibm.com/software/tivoli/products/monitor/>

## 2.7 Tivoli Monitoring for Databases V6.1

IBM Tivoli Monitoring for Databases helps ensure the availability and optimal performance of DB2, Oracle, Informix®, Microsoft SQL Server, and Sybase database servers.

Through the use of best practices incorporated within IBM Tivoli Monitoring for Databases, the typical database administrator dilemma of determining what to monitor, when to monitor, and how to interpret and act upon the monitoring results is eliminated, leaving more time for the administrator to focus on more complex, business-critical tasks. IBM Tivoli Monitoring for Databases provides routine, consistent monitoring that anticipates and corrects problems before database performance and customer confidence is degraded.

All data captured by monitoring the databases is delivered through an intuitive user interface and made available through historical and real-time reports. IBM



Tivoli Monitoring for Databases provides an out-of-the-box set of monitors for quick deployment and activation leveraging IBM best practices. The database administrator can also define custom monitors, thresholds, and tasks.

Through the implementation of IBM Tivoli Monitoring for Databases, database administrators are alerted when key performance and resource allocation problems are detected. This solution helps customers maximize their return on investment through increased efficiency of their IT staff, improved compliance to service-level objectives, and reduced cost of database system administration and deployment.

## **2.8 IBM Tivoli Composite Application Manager for WebSphere V6.0**

IBM Tivoli Composite Application Manager for WebSphere provides immediate problem determination, availability monitoring and performance analysis for enterprise WebSphere applications running on Windows, UNIX®, OS/400®, and z/OS environments. IBM Tivoli Composite Application Manager for WebSphere monitors heterogeneous environments consisting of both mainframes and distributed systems. From development and testing to staging and productions, IBM Tivoli Composite Application Manager for WebSphere helps to identify problems and resolve them in real-time, as well as understand application performance, and assess resource consumption patterns for planning future growth.

IBM Tivoli Composite Application Manager for WebSphere, along with the optional Data Collectors for Customer Information Control System (CICS) and Information Management System (IMS), provide visibility into transactions that span across Application Server, CICS, and IMS platforms. Composite transactions between WebSphere servers and CICS and IMS can be traced and correlated, giving developers and analysts insight into the health and performance of mid-tier business logic.

IBM Tivoli Composite Application Manager for WebSphere runs as a service in IBM WebSphere Application Server. It is composed of two main parts:

- **Data Collector**

A Data Collector runs on each monitored WebSphere Application Server, and communicates essential operational data to the IBM Tivoli Composite Application Manager for WebSphere Managing Server. Unique sampling algorithms maintain low CPU and network overhead while providing application-specific performance information.

Many Data Collectors can work with a single Managing Server. The communication between Data Collectors and the Managing Server is independent of platform.

► **Managing Server**

The Managing Server is a Java 2 Platform, Enterprise Edition (J2EE) application that is configured within the IBM WebSphere Application Server. The Managing Server is shared by all of the monitoring software's components and serves as the control center of your installation. The Managing Server collects information from, and provides services to the Data Collectors in your environment.

The Managing Server relies on a WebSphere Application Server and a database, which must be installed at the same time as or prior to installing the Managing Server itself.

Additional data on IBM Tivoli Composite Application Manager for WebSphere is available at the IBM Tivoli Composite Application Manager information center on the Web at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?toc=/com.ibm.itmwi.doc/toc.xml>

You can find more information about IBM Tivoli Composite Application Manager on the IBM Tivoli Composite Application Manager Web site at:

<http://www.ibm.com/software/tivoli/products/composite-application-mgr-websphere/>

## 2.9 Tivoli Omegamon XE for Messaging V6.0

IBM Tivoli OMEGAMON® XE for Messaging is the new name for, and a follow-on version to IBM Tivoli OMEGAMON XE for WebSphere Business Integration 1.1. It is designed to:

- Provide stable, comprehensive, and proactive monitoring and management capabilities with robust security features for IBM WebSphere MQ, IBM WebSphere Message Broker, and IBM WebSphere InterChange Server on a variety of platforms
- Deliver rapid time-to-value and greater ease-of-use analyses and solutions through a powerful set of utility tools that leverage common IBM Tivoli Monitoring technology

IBM Tivoli OMEGAMON XE for Messaging helps:

- ▶ Monitor the critical performance data related to WebSphere MQ, WebSphere Message Broker and WebSphere InterChange Server
- ▶ Provide real-time status on the availability and performance of WebSphere MQ, WebSphere Message Broker, and WebSphere InterChange Server components
- ▶ Collect monitoring data for use in historical reporting, performance analysis, trend prediction, and enterprise-wide business impact analysis
- ▶ Configure all WebSphere MQ resources from a central repository that allows you to verify all configurations prior to deployment, schedule deployment of all WebSphere MQ objects, and provide for backup of WebSphere MQ resources
- ▶ Identify problems in real time and deliver quick problem resolution through local correlation, root cause analysis, and corrective actions

By leveraging the IBM Tivoli Monitoring platform, IBM Tivoli OMEGAMON XE for Messaging can take advantage of the features offered by IBM Tivoli Monitoring, including:

- ▶ Agent deployment and configuration
- ▶ Secure sockets layer (SSL) support
- ▶ Tivoli Enterprise Console integration
- ▶ Application launch
- ▶ Deleting managed systems
- ▶ Tivoli Data Warehouse with summarization and pruning capabilities
- ▶ Command line interface for scripting purposes

You can find additional data about Tivoli Omegamon XE for Messaging at the information center on the Web at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?toc=/com.ibm.omegamon.mes.doc/toc.xml>

## 2.10 Tivoli Enterprise Console 3.9.4

IBM Tivoli Enterprise Console provides sophisticated, automated problem diagnosis and resolution to improve system performance and reduce support costs. The new enhancements focus on time to value and ease of use with out-of-the-box best practices to simplify and accelerate deployment. The auto-discovery feature enables system administrators to understand the environment and process events appropriately. The Web console enhances visualization while providing remote access to events and console operations.

IBM Tivoli Enterprise Console highlights include:

- ▶ The real value in event management goes beyond simple filtering and provides root cause analysis and resolution. Tivoli Enterprise Console delivers this.
- ▶ The new Web console provides improved visualization and access from anywhere.
- ▶ Preconfigured rules provide best-practices event management out-of-the-box.
- ▶ Auto-discovery and problem diagnosis increase operator responsiveness and efficiency.
- ▶ Integrated network management extends the Tivoli Enterprise Console reach and diagnosis for end-to-end management of your IT environment.
- ▶ Tivoli Enterprise Console enables comprehensive management that even accepts events from non-Tivoli products and systems.

IBM Tivoli Enterprise Console also includes IBM Tivoli Risk Manager (limited license), providing monitoring and management of firewalls and intrusion detection systems, and IBM Tivoli Comprehensive Network Address Translator, enabling integrated management of overlapping IP domains.

For more information, refer to the IBM Tivoli Enterprise Console Web site at:

<http://www.ibm.com/software/tivoli/products/enterprise-console/>

## 2.11 WebSphere Remote Server software distribution packages

IBM WebSphere Systems Management Accelerators V6.0 provides a quick way to deploy and manage WebSphere Remote Server infrastructure that includes convenient packaging for IBM WebSphere Application Server Network Deployment, DB2 UDB Workgroup Server Unlimited Edition, WebSphere MQ, and Remote Management Agent components.

WebSphere Systems Management Accelerators is a collection of tools that provide multiple functions in setting up a store environment. WebSphere Systems Management Accelerators provide the following functions:

- ▶ Eases the installation of WebSphere Remote Server across numerous machines

WebSphere Systems Management Accelerators include installation scripts and response files for installing the WebSphere Remote Server components to the in-store processors (ISPs). For every WebSphere Remote Server

component, there is an installation and uninstallation script. By selecting reasonable defaults, these scripts make WebSphere Remote Server installation easy with a minimum configuration. Using the scripts, you can easily set some values that you want to change (for example, default passwords).

**Note:** These installation scripts run with or without the presence of Tivoli software on the ISPs.

- ▶ Extends the monitoring capability of the IBM Remote Management Agent  
WebSphere Systems Management Accelerators extend the monitoring capability of the IBM Remote Management Agent. By default, the Remote Management Agent does not broadcast event data outside of the ISP. The Accelerators include a plug-in with which you can retrieve event data from the Remote Management Agent, and another plug-in to broadcast that data to a Tivoli Enterprise Console. A framework is provided for you to write your own plug-in that broadcasts event data to another event consumer.
- ▶ Provides integration between WebSphere Remote Server components running at the ISP and the Tivoli products running at the enterprise central site:
  - Tivoli Configuration Manager
  - Tivoli Enterprise Console

**Note:** This integration requires the use of Tivoli Management Framework.

By using the software packages included in WebSphere Systems Management Accelerators, you can gain the benefits of using Tivoli Configuration Manager. WebSphere Systems Management Accelerators provide integration with Tivoli Configuration Manager. For every WebSphere Remote Server component, there are the following two groups of software packages for each ISP platform (Windows and Linux):

- Data packages

The data packages can be thought of as thin wrappers around a CD image. Installing a data package for a WebSphere Remote Server product is similar to copying files from the product CD to a hard drive.

- Installation packages

The installation packages can be thought of as thin wrappers around the WebSphere Systems Management Accelerators installation scripts. Installing an installation package for a WebSphere Remote Server product executes the installation script that installs that product.

The WebSphere Systems Management Accelerators include the following:

- ▶ CD images of the WebSphere Remote Server middleware products
- ▶ Installation or removal scripts for installing or removing the middleware products
- ▶ Installation response files

### 2.11.1 Sample environment

WebSphere Systems Management Accelerators can be used in environments where there is low bandwidth to the ISPs, provided that the WebSphere Systems Management Accelerators CDs are physically installed onto each ISP. In high bandwidth environments, the CDs can be installed directly from an enterprise central site location using the provided Tivoli Configuration Manager packages, or other transport mechanisms, such as FTP.

Figure 2-2 illustrates a sample store environment with a single enterprise central site and several remote ISPs.

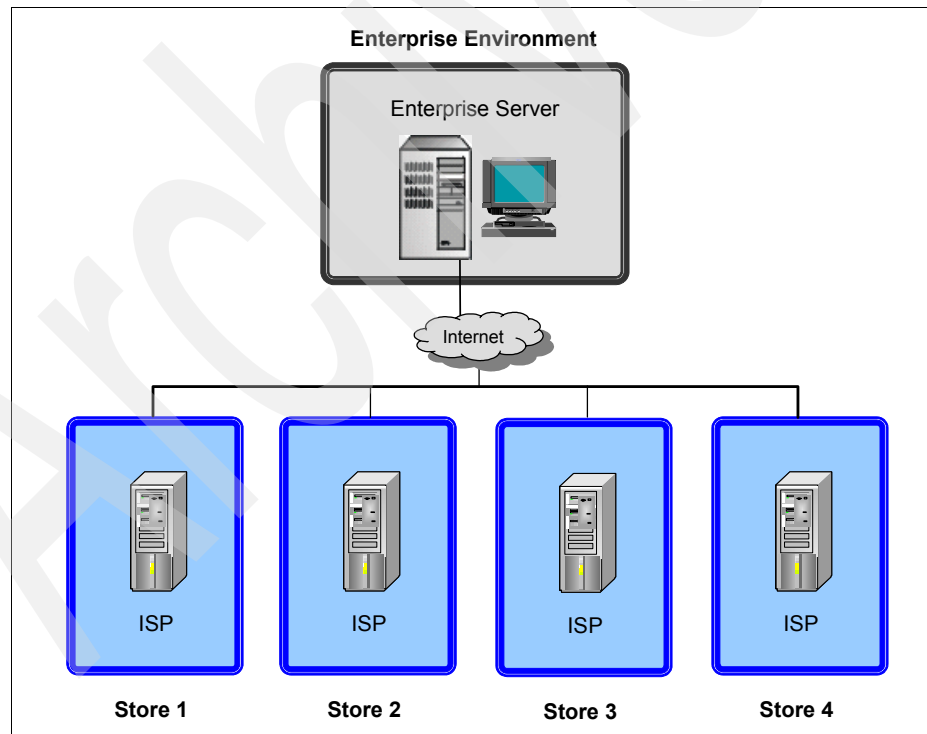


Figure 2-2 Sample store environment with several remote ISPs

## The Enterprise Server

The Enterprise Server is a Linux or AIX®-based server used as a central management point for all the In-Store Processors. Software installations are initiated and monitoring data is received from the Enterprise Server. After WebSphere Remote Server has been installed on all the ISPs using the WebSphere Systems Management Accelerators, it is possible to do most management of the ISPs without logging into them.

**Important:** The Enterprise Server can run with or without the Tivoli Framework, as follows:

- ▶ **Tivoli Framework environment**

The Enterprise Server can run at minimum, a Tivoli Management Region, Tivoli Configuration Manager, and Tivoli Enterprise Console. The Tivoli Enterprise Console can be installed on a separate machine.

- ▶ **Non-Tivoli Framework environment**

The Enterprise Server is only required to run enough software to initiate installations onto ISPs through the WebSphere Systems Management Accelerators. For example, FTP client and Telnet client.

## The In-Store Processor

The ISP is a server in the store. This Linux or Windows based server has several devices connected to it, including Point-Of-Sale Controllers, kiosks, self-checkout systems, and other devices. The ISP runs all the software that the store operations require. WebSphere Remote Server is installed onto the ISP to form a base platform for other applications.

### 2.11.2 Software requirements

WebSphere Systems Management Accelerators on the ISP and on the Enterprise Server have the following software requirements.

#### **Enterprise Server**

WebSphere Systems Management Accelerators on the Enterprise Server have the following software requirements.

##### ***For a Tivoli Framework environment***

- ▶ Tivoli Framework 4.1.1 must be installed.

- ▶ Tivoli Configuration Manager 4.2.3 FP1 must be installed and configured on the Tivoli Management Region server, including the following Java components:
  - MDIST GUI
  - Activity Planner GUI
  - Change Configuration Manager GUI
  - Inventory GUI
- ▶ The person performing the installation using the Accelerators must have root access or administrator access to the Tivoli Management Region server.

***For a non-Tivoli Framework environment***

- ▶ Some framework for distributing files must be present, such as FTP client.
- ▶ Some framework for connection to remote servers must be present, such as Telnet client or SSH client.

**In-Store Processor**

The ISP must be running SUSE Linux Enterprise Server (SLES) 9.3 (Kernel 2.6), Windows 2000 SP4, Windows 2003 SP1, or IBM Retail Environment for SUSE Linux V2.1.2.

***For a Tivoli Framework environment***

The Tivoli Light Client Framework 4.1.1 must be installed and communicating properly with a Tivoli Management Region.

***For a non-Tivoli Framework environment***

- ▶ Some framework for receiving files must be present, such as FTP server.
- ▶ Some framework for allowing remote script execution must be present, such as Telnet server or SSH server.

### 2.11.3 Hardware requirements

WebSphere Systems Management Accelerators on the ISP and on the Enterprise Server have the following hardware requirements.

**Enterprise Server**

The minimum hardware requirements for the Enterprise Server are:

- ▶ IBM PC server with Pentium® 4 or compatible processor
- ▶ 2 GHz or higher (Linux)
- ▶ RS/6000® server (AIX)
- ▶ 1 GB of RAM (more recommended)



### ***For a Tivoli Framework environment***

Each supported ISP platform requires a minimum of 5 GB free HD space. Each supported ISP platform requires 2.5 GB on the partition where the Accelerators are installed (the default is /opt), and the partition used by the software packages as the staging area (default is /var) requires 2.5 GB.

### ***For a non-Tivoli Framework environment***

Each supported ISP platform requires a minimum of 2.5 GB free HD space.

### **In-Store Processor**

The minimum hardware requirements for the ISP are:

- ▶ IBM PC server with Pentium 4 or compatible processor
- ▶ 2 GHz or higher
- ▶ 1 GB of RAM (recommended)
- ▶ 20 GB of free disk space on installation drive

## **2.11.4 Components**

The IBM WebSphere Systems Management Accelerators consists of a set of packages that you can use to install WebSphere Remote Server components on ISPs. The four versions of the packages provided are:

- ▶ Linux installation packages
- ▶ Linux data packages
- ▶ Microsoft Windows installation packages
- ▶ Microsoft Windows data packages

The Accelerators package consists of the following installable software components and fix packs:

- ▶ WebSphere MQ 6.0
- ▶ WebSphere MQ 6.0.1
- ▶ WebSphere MQ 6.0.1.1
- ▶ DB2 WebSphere Server Edition 8.2.4
- ▶ WebSphere Application Server Network Deployment 6.0
- ▶ WebSphere Application Server 6.0 Edge components
- ▶ WebSphere Application Server Refresh Pack 6.0.2
- ▶ WebSphere Application Server fix pack 6.0.2.7
- ▶ Remote Management Agent 1.0
- ▶ Simple Network Management Protocol (SNMP) Trap Mapper 6.0

- ▶ IBM Tivoli Composite Application Manager for WebSphere Data Collector
- ▶ IBM Tivoli Composite Application Manager for WebSphere Data Collector Fix Pack 1
- ▶ IBM Tivoli Composite Application Manager for WebSphere Data Collector Fix Pack 2
- ▶ Remote Management Agent Socket Data Provider 6.0
- ▶ Tivoli Configuration Manager Remote Management Agent Bridge 6.0

The following software is delivered on 12 CDs:

- ▶ Accelerators for Target Windows Server® (six CDs)
- ▶ Accelerators for Target Linux Server (six CDs)

## 2.12 IBM Remote Management Agent

IBM Remote Management Agent is a component of the Store Integration Framework and is distributed with the WebSphere Systems Management Accelerators for Retail offering. It facilitates the management of devices and applications through an open Java Management Extensions (JMX™) standard.

Remote Management Agent includes a Master Agent and a browser-based Viewer that run on a server in the store, and a General Agent that runs on the devices and with the applications being managed. The General Agent provides a template that you can customize to meet the management needs of specific devices and applications. The Master Agent consolidates information from the General Agents running in the store and provides a single point of access to the store for JMX-compatible enterprise systems management tools such as Tivoli.

### 2.12.1 Prerequisites

The prerequisites for installing the Remote Management Agent are as follows:

- ▶ The Remote Management Agent, Master Agent and General Agent require Java Runtime Environment (JRE™) V1.4.1.
- ▶ The General Agent should be supported on the following operating system environments:
  - IBM 4690 OS V3R3 corrective service disk (CSD) 04J0
  - Novell SUSE Linux Enterprise Server (SLES) V8 SP3 or V9
  - Microsoft Windows 2000 SP4
  - Microsoft Windows XP Professional SP2

- ▶ The Master Agent and Viewer should be supported on the following operating system environments:
  - Novell SUSE Linux Enterprise Server (SLES) V8 SP3 or V9
  - Microsoft Windows 2000 SP4
  - The Viewer is a Java application that requires WebSphere Application Server V5.0.2 or 5.1.1

## 2.12.2 Architecture overview

This section provides details about the Remote Management Agent and Viewer architecture.

The goals of the Remote Management Agent and Viewer architecture are to:

- ▶ Provide a single-store view for management
- ▶ Use an open, standards-based instrumentation model
- ▶ Maintain strict isolation between components and management tools so that all management applications behave in a consistent manner
- ▶ Expose device and application instrumentation

Accomplishing these goals requires adhering to the following disciplines:

- ▶ Configuration

The Remote Management Agent provides support for local and remote configuration of managed software and hardware components.

- ▶ Inventory

The Remote Management Agent provides support for local and remote inventory reporting of both software and hardware components. For software components, the inventory information typically encompasses a product description, information about the product manufacturer, and product version, including maintenance levels.

- ▶ Event notification and forwarding

Hardware and software components generate notification events that are monitored by Remote Management Agent and Viewer applications. The events include information pertaining to normal operations or errors. The architecture presents a multi-tiered framework for collecting and filtering events at the store-level for access by an enterprise system management application.

- ▶ Remote operations

The MBeans component within the Remote Management Agent provides various remote operations.

- Software distribution

The Remote Management Agent provides support for the automated installation or uninstallation of software packages.

- Monitoring

The Remote Management Agent provides support for monitoring the MBean attribute values, with notification events being generated when monitoring conditions occur.

The Remote Management Agent and Viewer architecture is based on JMX. The JMX specification describes a general management framework for any Java or non-Java program. It is a multi-tier architecture that includes MBeans, an instrumentation layer, MBeanServer, an agent layer, and a management layer.

MBeans are Java objects that expose a management interface for a managed resource. All MBeans within a Java virtual machine (JVM™) are registered with an object repository called MBeanServer. The MBeanServer provides access to each MBean's attributes and operations and handles the process of sending out notifications or events. MBeans are useful to the Remote Management Agent and Viewer when their interfaces are exported, so that they can be accessed by remote management applications.

For comprehensive information about the JMX framework, refer to the latest JMX specification that is available on the Web at:

<http://java.sun.com/products/JavaManagement/>

### 2.12.3 Requirements

This section describes the hardware and software requirements for running the Remote Management Agent and Remote Management Viewer.

#### Hardware requirements

The General Agent runs in a component JVM or as a system service on a Windows or Linux machine.

The Master Agent and Viewer run on a single machine within the store, the *in-store processor*. The in-store processor or Web server for use with the Store Integration Framework is the machine that runs the Remote Management Agent console service and provides a point of interface for enterprise applications or other in-store applications. The in-store processor should have a minimum 1 GHz processor with 1 GB of RAM.

**Note:** The Master Agent and General Agent services cannot be installed on the same computer.

## Software requirements

There are several software requirements for the Remote Management Agent and Viewer:

- ▶ Java 2: Remote Management Agent requires IBM JDK™ or JRE V1.4.1 or later

To get the JRE required to run the Remote Management Agent and Viewer, download and install the JavaPOS™ package from the following Web site. (Before you download the package, you need to register on the site.)

<http://www.clearlake.ibm.com/store/support/html/driverss.html>

- ▶ Supported operating systems

Install the IBM Remote Management Agent and Viewer on one of the following operating systems:

- SUSE Linux Enterprise Server (SLES) V8 Service Pack 3
- SUSE Linux Enterprise Server (SLES) V9
- IBM Retail Environment for SUSE Linux V1
- IBM Retail Environment for SUSE Linux V2
- Microsoft Windows 2000 Service Pack 4 with Microsoft .NET Framework V1.1 Service Pack 1
- Microsoft Windows XP (to be used with the General Agent service only) Service Pack 2 with Microsoft .NET Framework V1.1 Service Pack 1 Remote Management

- ▶ Viewer prerequisites

If the IBM Remote Management Viewer has to be used, install it on an in-store processor where the following software is already present:

- IBM WebSphere Application Server
- Supported Web browsers: Microsoft Internet Explorer® and Mozilla

- ▶ Installation roles

Install the Remote Management Agent either as a Master Agent or a General Agent. Since the Remote Management Viewer is designed to interact with only the Master Agent, it is not necessary to install the Remote Management Viewer with the General Agent.

## 2.13 SNMP Trap Mapper

The SNMP Trap Mapper is a component of the Store Integration Framework and is distributed with the WebSphere Systems Management Accelerators for Retail offering. It helps flow events from the Remote Management Agent to the Tivoli Enterprise Console. Figure 2-3 depicts the configuration of the SNMP Trap Mapper between the Remote Management Agent and the Tivoli Enterprise Console.

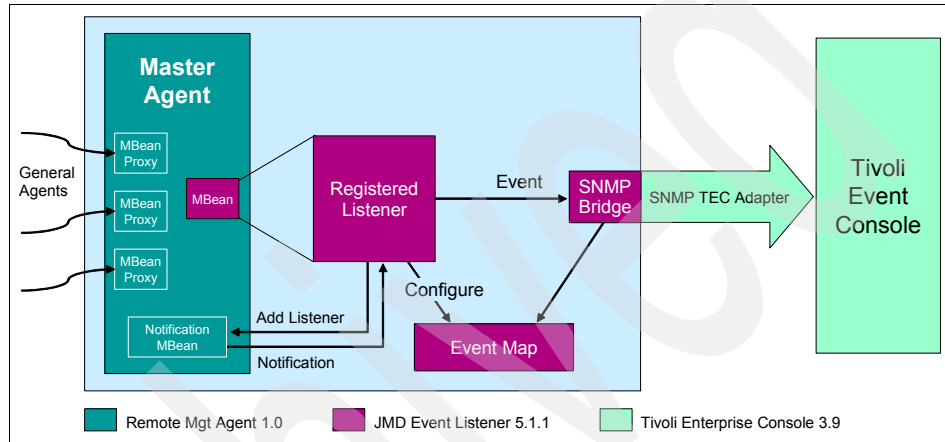


Figure 2-3 SNMP Trap Mapper

The SNMP Trap Mapper and its components follow this sequence:

1. The Master Agent:
  - a. Discovers the General Agents that exist within the environment
  - b. Creates proxy MBeans for MBeans that exist on those General Agents
  - c. Controls all interactions
2. The SNMP Trap Mapper:
  - a. Finds the Master Agent's MBean server
  - b. Registers itself with the server as an MBean
  - c. Adds itself as a listener for notifications
3. The MBean server:
  - a. Receives a notification from the Notification MBean
  - b. Invokes the "handleNotification" method
4. The "handleNotification" method propagates the event on to the SNMP bridge.
5. The SNMP bridge translates the JMX event into the appropriate SNMP event for the SNMP adapter.

6. The SNMP adapter:
  - a. Converts the SNMP event into an appropriate Tivoli Enterprise Console event
  - b. Propagates it to the Tivoli Enterprise Console

## 2.14 Log collection scripts

The software packages have a set of log collection scripts included in them. Use the following scripts to retrieve log files from the endpoints to the enterprise server:

- ▶ `setenv.sh`
- ▶ `runCollectLogs.sh`
- ▶ `collectLogs.sh`

The scripts are located in the *Software Packages for Linux* CD 1. They are:

- ▶ AIX 5L: `LogCollector/scripts/aix`
- ▶ Linux: `LogCollector/scripts/linux`

Perform the following steps to use these scripts:

1. Copy the `setenv.sh` script to your enterprise server under the following directory, depending on your operating system:
  - On AIX 5L: `/opt/IBM/`
  - On Linux: `/opt/IBM`
2. Enter the following command:
  - On AIX 5L: `./setenv.sh`
  - On Linux: `./setenv.sh`
3. Copy the `runCollectLogs.sh` and `collectLogs.sh` scripts to your enterprise server in the `/opt/IBM/SIF/Accelerators/collectlog/` directory.

4. Edit the runCollectLogs.sh file to set the appropriate values as shown in Example 2-1.

*Example 2-1 Editing the runCollectLogs.sh file*

---

```
#usage: ./collectLogs.sh <endpoint> <source path> <source file>
$DEST_PATH <destination server>
#repeat calling collectLogs.sh for collecting multiple log files
#
#example:
#./collectLogs.sh sifins1 /opt/WebSphere/AppServer/logs/server1
SystemOut.log $DEST_PATH ivtaix1
#./collectLogs.sh elake C:/WebSphere/AppServer/logs/server1
SystemErr.log $DEST_PATH ivtaix1
#./collectLogs.sh sifins2 /opt/IBM/SIF/logs SifRemove_SMA1.log
$DEST_PATH ivtaix1
```

---

5. To start to collect log files, enter the following command:
  - On AIX 5L: **./runCollectLogs.sh**
  - On Linux: **. runCollectLogs.sh**
6. Check the output log file collectLogsOut.log for the status. This file is located in the /opt/IBM/SIF/Accelerators/collectlog/ directory. Example 2-2 shows the log entries displayed on successful completion of the operation.

*Example 2-2 Log entries displayed on successful completion of operation*

---

```
[ Oct 14 11:08:27 EDT 2004 ] ++++++ Beginning
collection of the logs ++++++
[ Oct 14 11:08:30 EDT 2004 ] Successful retrieval of the log file,
SystemOut.log, from endpoint sifins1.
[ Oct 14 11:08:36 EDT 2004 ] ++++++ Completed
collection of the logs ++++++
[ Oct 14 11:08:36 EDT 2004 ] ++++++ Beginning
collection of the logs ++++++
[ Oct 14 11:08:39 EDT 2004 ] Successful retrieval of the log file,
SystemErr.log, from endpoint elake.
[ Oct 14 11:08:46 EDT 2004 ] ++++++ Completed
collection of the logs ++++++
[ Oct 14 11:08:46 EDT 2004 ] ++++++ Beginning
collection of the logs ++++++
[ Oct 14 11:08:49 EDT 2004 ] Successful retrieval of the log file,
SifRemove_SMA1.log, from endpoint sifins2. [ Oct 14 11:08:55 EDT
2004 ] ++++++ Completed collection of the logs
++++++
```

---



7. The retrieved log files are located in the  
/opt/IBM/SIF/Accelerators/collectlog/collectedlog/*endpoint* directory.

Archived



# IBM Retail Environment for SUSE Linux

IBM Retail Environment for SUSE Linux V2 consists of a SUSE Linux Enterprise Server (SLES) based distribution for IBM System x™ store servers and select IBM point of sale (POS) devices. It includes features tailored for the retail environment.

This chapter provides a description of the architecture and functionality provided in IBM Retail Environment for SUSE LINUX V2.0.

This chapter contains the following sections:

- ▶ 3.1, “Product overview” on page 40
- ▶ 3.2, “Architecture” on page 41
- ▶ 3.3, “Components” on page 46
- ▶ 3.4, “Building and deploying a point-of-sale image” on page 51
- ▶ 3.5, “Requirements” on page 54
- ▶ 3.6, “Additional information” on page 59

## 3.1 Product overview

The IBM Retail Environment for SUSE Linux is an open, cost-effective platform that is designed to enhance operations at the critical point of sale (POS). It combines a SUSE Linux operating system that has been optimized for retailers, with the IBM SurePOS™ and IBM System x systems and select IBM middleware and support services.

The IBM Retail Environment for SUSE Linux brings with it the following features:

- ▶ Provides a single point of contact for supporting the entire life cycle of the retail-optimized SUSE Linux operating system
- ▶ Helps run the same system on the mainframe, in-store servers, and at the POS
- ▶ Provides predefined, retail-tailored LINUX installation packages to help reduce complexity and facilitate ease of installation
- ▶ Enables leading-edge applications through support for open standards pervasive throughout the retail industry
- ▶ Exists as an integral part of the on demand store solutions from IBM

Figure 3-1 shows the on demand store solutions

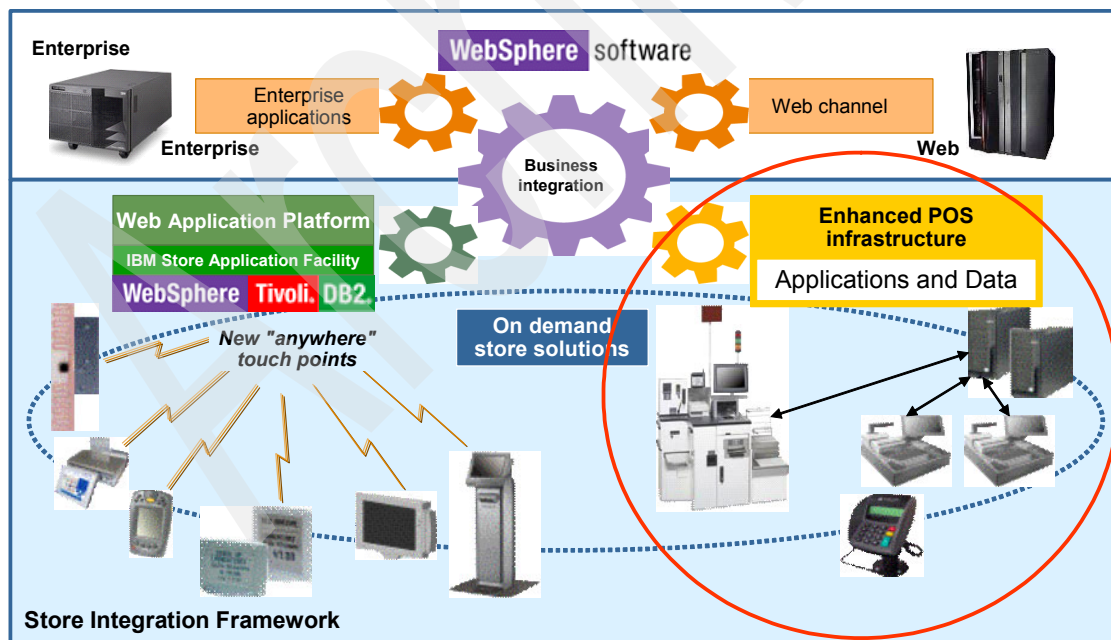


Figure 3-1 On demand store solutions

IBM Retail Environment for SUSE Linux is a tool to build POS client images. It is also a means to manage the distribution of these images to the POS terminal. IBM Retail Environment for SUSE Linux provides for the configuration of POS terminals and supports the remote loading of POS terminals.

It is a standard Linux distribution with POS-specific tools. It provides a pretested and preconfigured environment for all IBM terminals and POS peripherals. The benefits of IBM Retail Environment for SUSE Linux include:

- ▶ Reduces costs by streamlining store operations

This is possible due to the following reasons:

- The role of IBM as the single point of contact for Linux support
- Longer life cycle support and predictable upgrade release cycles
- Lower up-front and ongoing costs as compared to Microsoft Windows
- Supported Linux environment that is priced appropriately for POS clients

- ▶ Improves productivity by enabling IT employees

This is possible due to the following features:

- Simplified integration and installation
- Four template client images
- Infrastructure for centrally-managed maintenance and distribution

- ▶ Reduces the risk around a Linux solution for retailers

This is possible due to the existence of Open source leverage, with IBM as a stable solution partner.

## 3.2 Architecture

This section provides an overview of the IBM Retail Environment for SUSE Linux software stack and software deployment.

### 3.2.1 IBM Retail Environment for SUSE Linux software stack

Novell produces an enterprise class system, namely, the SUSE Linux Enterprise Server (SLES). Certain versions of this distribution, such as SLES 9, employ the technologies of the United Linux 1.0 standard system.

SLES 9 is the base for the Novell Linux Point of Service (NLPOS), a retail optimized Linux that is robust enough to support the configuration of fixed function clients in a distributed retail environment.

**Note:** Novell Linux Point of Service (NLPOS) was previously named SUSE Linux Retail Solution (SLRS).

The NLPOS comes with the following features:

- ▶ Secure, flexible, and cost-effective platform designed for the retail market
- ▶ Simplified installation and configuration
- ▶ SUSE Linux Enterprise Server 9 base
- ▶ Centralized management
- ▶ Ability to run on the client, including kiosks, POS, and self-service units, on the server and alongside other platforms
- ▶ Expert support and maintenance options available worldwide through Novell and Novell partners
- ▶ An indemnification plan giving enterprise customers additional reassurance about deploying Linux in their organizations
- ▶ A scalable, open, and reliable deployment infrastructure based on the SUSE Linux Enterprise Server five-year life cycle

The IBM Retail Environment for SUSE Linux platform configures and customizes NLPOS for use with IBM SurePOS clients in the retail environment. IBM Retail Environment for SUSE Linux provides vital IBM-specific hardware information such as client device drivers, and hardware-specific client device information to

sit on top of Novell's generic retail Linux distribution. Figure 3-2 depicts the IBM Retail Environment for SUSE Linux software stack.

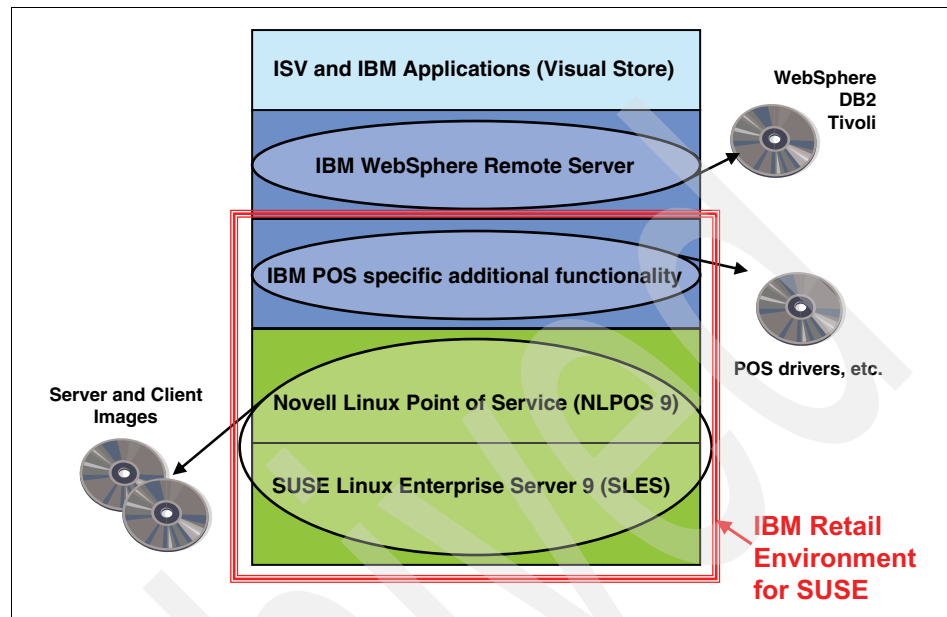


Figure 3-2 IBM Retail Environment for SUSE Linux software stack

The first three layers of the stack include the NLPOS, which is built on the SUSE Linux Enterprise Server 9. To this base, IBM adds additional tools and drivers for IBM POS terminals. IBM Software for Retail Outlets refers to the WebSphere Remote Server that provides a middleware platform for running applications within the store.

For more information about the components provided with the WebSphere Remote Server, refer to Chapter 2, “WebSphere Remote Server V6 and its components” on page 9.

### 3.2.2 IBM Retail Environment for SUSE Linux software deployment

Figure 3-3 depicts the standard tiers of retail computing.

- ▶ Tier 1 is a collection of enterprise servers that provide centralized management and processing for all data related to store servers.
- ▶ Tier 2 consists of the In-Store Processors and POS controllers that support user touchpoints and provide access to the store from the enterprise.

- Tier 3 includes all user touchpoints. These are places where a customer will access the infrastructure, for example, a kiosk, or places where store personnel will access information or process a transaction, such as a Webpad or POS terminal

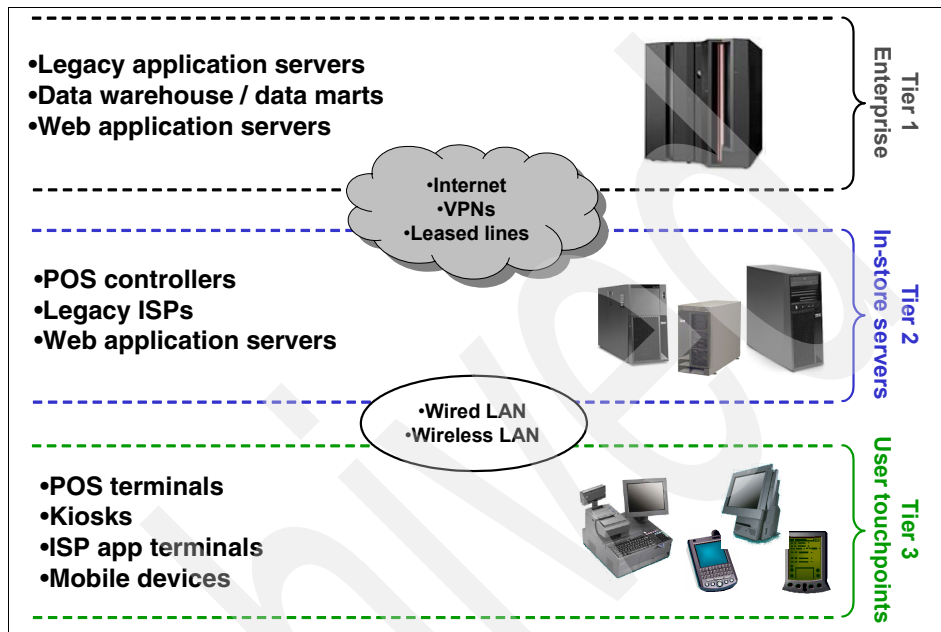


Figure 3-3 Three tiers of retailing

The IBM Retail Environment for SUSE Linux platform supports a flexible three-tier distributed retail environment, as depicted in Figure 3-4. In this environment:

- The first tier is the corporate enterprise or central office
- The second tier consists of one or more in-store servers
- The third tier includes one or more client point-of-sale (POS) systems used throughout the store



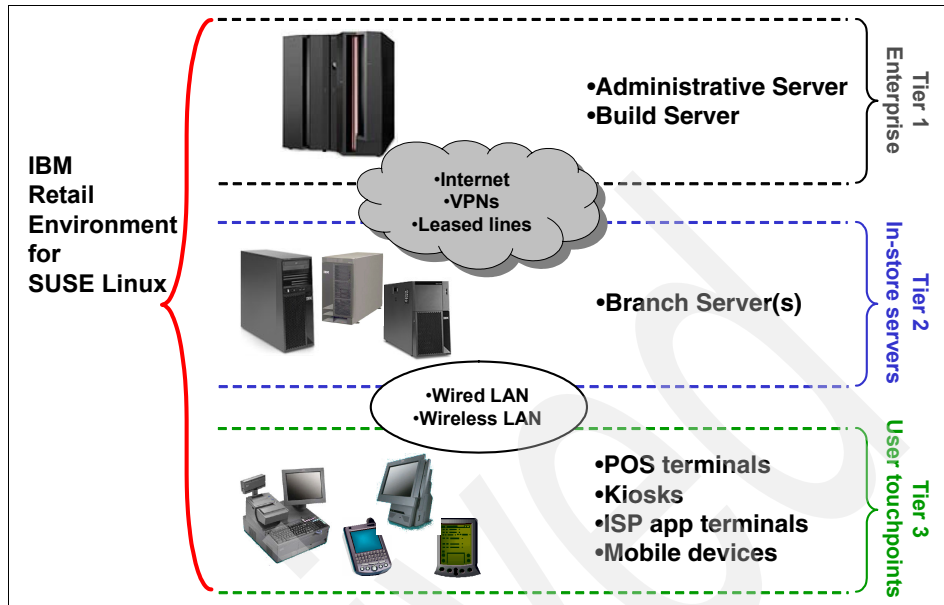


Figure 3-4 IBM Retail Environment for SUSE Linux solution

IBM Retail Environment for SUSE Linux provides a mechanism to create, deploy, and manage client POS disk images from a centralized location in the first tier to the other two tiers.

It includes an administrative server and build server in the first tier. The administrative server provides storage for a central Lightweight Directory Access Protocol (LDAP) directory, global and default parameters, application configurations, and a POS image repository. The build server provides a centralized method for creating, modifying, and updating POS images.

IBM Retail Environment for SUSE Linux also includes one or more distributed branch servers in the second tier. These servers provide a multicast boot and installation infrastructure for diskless and disk POS clients. The branch server operates as a software transport for operating system and application updates.

IBM Retail Environment for SUSE Linux includes support for automated remote installation of the branch server and configuration of two branch servers in an optional, two-node, high availability (HA) cluster with replicated data. The third tier consists of the client.

These machines boot over the network from the IBM Retail Environment for SUSE Linux branch server, which provides the POS images. No software is required on the POS clients, because they boot using the Preboot Execution

Environment (PXE) protocol from Basic Input/Output System-level (BIOS) configuration data. Figure 3-5 depicts the software stacks on the POS client and the in-store processor that houses the branch server.

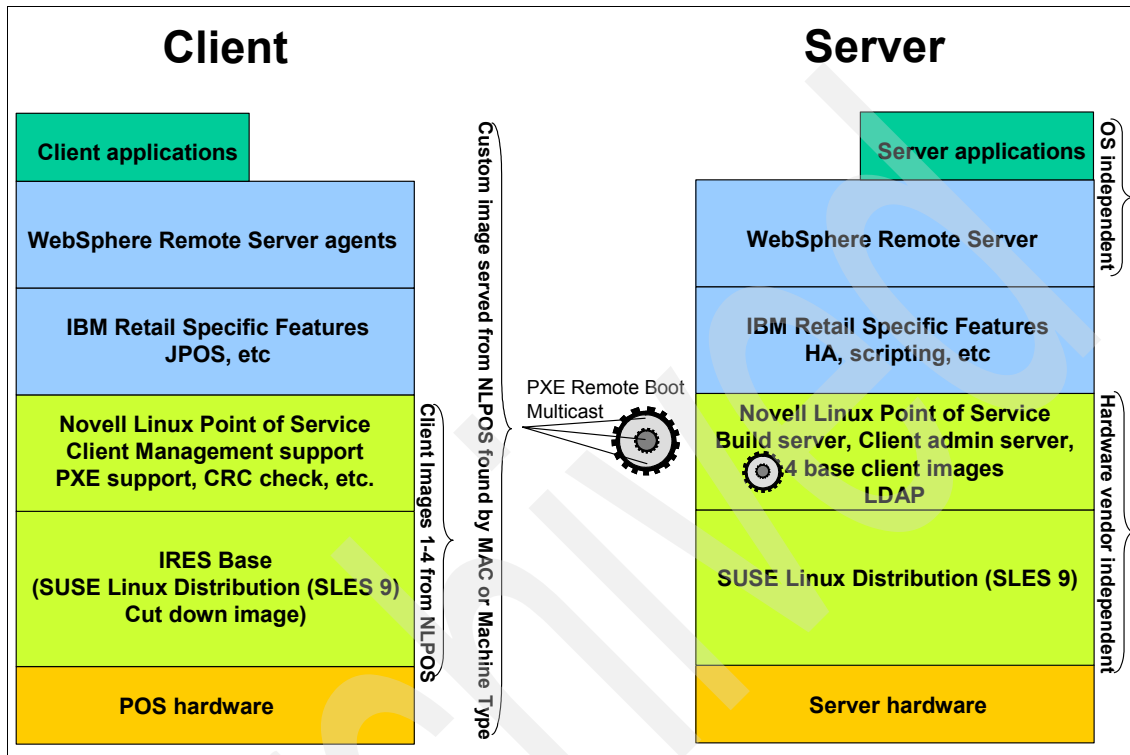


Figure 3-5 Software stacks

### 3.3 Components

The Novell Linux Point of Service, and therefore the IBM Retail Environment for SUSE Linux, which is a super-set of the NLPOS, architecture, comprises three servers:

- ▶ The administrative server - Tier 1 Enterprise level
- ▶ The build server - Tier 1 Enterprise level
- ▶ The branch server - Tier 2 Branch level - one in each branch or office

**Note:** Although it is possible to operate with all the three servers on a single machine in certain situations, such as a lab environment for example, the prescribed process is to install the administrative and build servers on an enterprise platform and a branch server on each in-store processor.

### 3.3.1 LDAP or role-based configuration

NLPOS provides a mechanism for controlling the configuration of the POS systems (Tier 3) which is based on an enterprise LDAP repository containing configuration records for POS clients and servers, as well as configuration records for such things as client load images. IBM Retail Environment for SUSE Linux V2 fully supports the use of configuration through the NLPOS LDAP configuration scheme. However, IBM Retail Environment for SUSE Linux V2 also provides an alternative configuration scheme referred to as *role-based configuration*, which helps simplify store system administration. Instead of configuring each client individually, you simply select a default configuration based on the system's role in your store – POS, help desk, service, gift registry, or many others. In some cases, adding a new POS system to your store may not require any configuration at all.

A choice of one or the other must be made at install time, and the choice affects how the software solution is installed, what software components are installed, and how the solution will be administered. For more information about IBM Retail Environment for SUSE Linux V2 role-based configuration, refer to *IBM Retail Environment for SUSE LINUX Version 2: Developer's Guide*, GC30-9723-01.

The following section assumes an LDAP configuration.

### 3.3.2 LDAP use and structure

All the administration data is kept in a central LDAP directory on the administrative server. This data is used to generate the necessary configuration files. In addition, the required operating system images for the POS devices are created and maintained here.

The LDAP directory is not replicated on the branch servers. The components on the branch servers access the administration directory directly. This requires that all the functionality for daily operation in the branches must be able to run without any connection to the directory service if necessary. During execution of administrative tasks, such as installation of new POS devices in a branch, steps must be taken to ensure that the WAN link to the central office is available. The services that are needed for the operation and management of the POS devices run on the branch servers. In addition, these services are configured over the central LDAP directory. The POS devices execute a network boot through PXE by default. The branch servers provide all the services required for this. The configuration of the POS devices from the initial network boot image is not done through directory queries, but rather through ASCII files that are loaded over Trivial File Transfer Protocol (TFTP). Additionally, the branch servers provide the Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) services.

The LDAP directory is designed to manage even the largest potential corporate structures. All the globally valid information for a chain or company, such as hardware types or OS images, is stored in the container global. The information pertaining to the branch servers is stored in the server container below the branch entry. This data can be used to generate an automatic install file, configure the services of a branch server, and carry out an inventory process.

The actual branches are further organized into regions. The branch containers store information about the deployed POS devices and branch servers. Store or reference this and all other information that can be modified by the branch server in the branch containers, to limit the need for granting write privileges to subtrees.

Example 3-1 displays example LDAP entries. It includes:

- ▶ Two client images: Java and browser
- ▶ A hardware image: IBM4800781
- ▶ Two stores: one in New York and one in Chicago

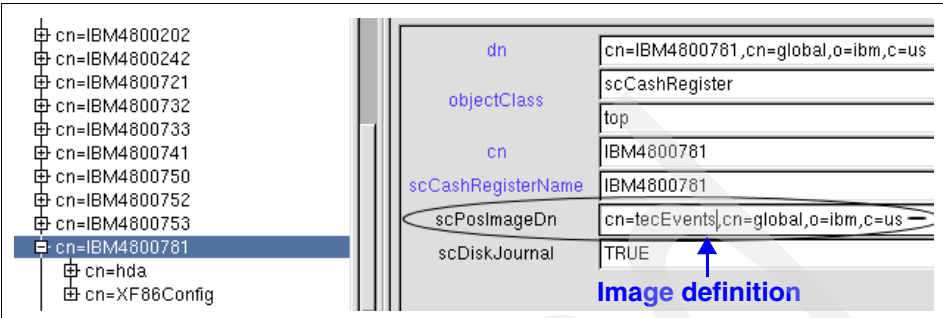
*Example 3-1 Sample LDAP entries*

---

```
0=valuetrend,c=ca
  cn=global
    cn=java (This is a load image name)
    cn=browser
    cn=IBM4840563 (This IBM Hardware)
  cn=standards
  cn=stores
    ou=New_York
      cn=store1
        cn=server
          cn=bs1 (This is a branch server in a given store)
        cn=Chicago
      cn=store23
        cn=server
          cn=bs23
```

---

Figure 3-6 depicts a hardware entry in the LDAP directory. Each unique piece of hardware has an entry in the LDAP directory's Global section.

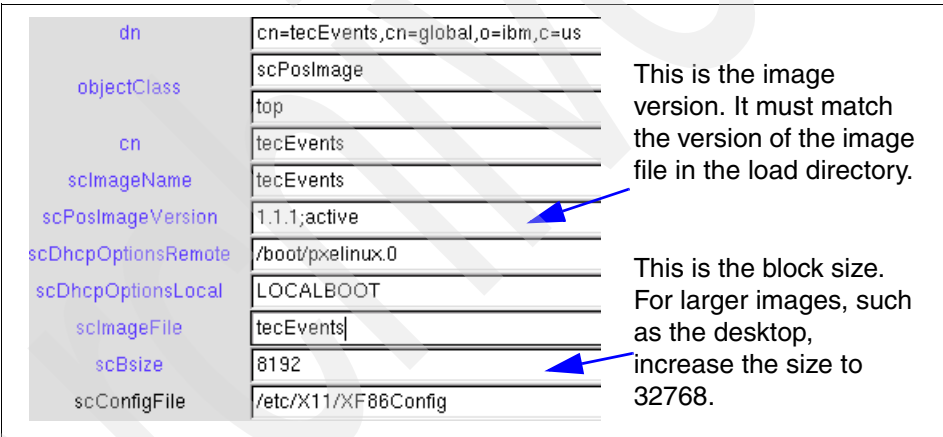


dn	cn=IBM4800781,cn=global,o=ibm,c=us
objectClass	scCashRegister
	top
cn	IBM4800781
scCashRegisterName	IBM4800781
scPosImageDn	cn=tecEvents,cn=global,o=ibm,c=us
scDiskJournal	TRUE

Image definition

Figure 3-6 LDAP hardware entry

Figure 3-7 depicts a unique POS image entry in the LDAP directory. Each unique POS image has an entry in the LDAP directory's Global section.



dn	cn=tecEvents,cn=global,o=ibm,c=us
objectClass	scPosImage
	top
cn	tecEvents
scImageName	tecEvents
scPosImageVersion	1.1.1;active
scDhcpOptionsRemote	/boot/pxelinux.0
scDhcpOptionsLocal	LOCALBOOT
scImageFile	tecEvents
scBsize	8192
scConfigFile	/etc/X11/XF86Config

This is the image version. It must match the version of the image file in the load directory.

This is the block size. For larger images, such as the desktop, increase the size to 32768.

Figure 3-7 LDAP POS image entry

### 3.3.3 Server services

The administrative server runs the directory service and the RSYNC Server. The branch servers run all the services required to start the POS devices, including Trivial File Transfer Protocol (TFTP), Dynamic Host Configuration Protocol (DHCP), and Domain Name System (DNS) for the respective branch, and a Network Time Protocol (NTP) server that synchronizes the system time with the administrative server.

## Domain Name System

Every branch server runs a DNS master for that branch. The zone files are generated on each branch server by building an image.

## Dynamic Host Configuration Protocol

A DHCP server is installed on the branch server. The `dhcpd.conf` file is generated by the `posldap2dhcp` script from the directory data for the branches.

## Trivial File Transfer Protocol

The TFTP service on the branch server is structured as described in the “TFTP server structure” on page 50, with `boot`, `image`, `POS`, and `upload` directories. There is a PXE default configuration with which all the POS devices first load the same initial `initrd` and same kernel.

### *TFTP server structure*

The TFTP server directory structure is divided into the following main areas under the `tftp_root1` directory:

- ▶ Image configurations  
The `/tftpboot/CR/` directory contains various `cong.MAC` Address image configuration files.
- ▶ Configuration files  
The `/tftpboot/CR/MAC Address/` directory contains various system configuration files, such as `XF86cong`.
- ▶ Boot files  
The `/tftpboot/boot/` directory contains `initrd.gz`, the kernel to boot, and PXE loader `pxelinux.0`.
- ▶ PXE configuration file  
The `/tftpboot/boot/pxelinux.cfg` directory contains the PXE configuration file.
- ▶ Image files and checksums  
The `/tftpboot/image/` directory contains all the image files and their checksums.
- ▶ Upload area  
The `/tftpboot/upload/` directory is the directory into which the `hwtype.MAC` Address files for registering new POS devices are uploaded.

## Network Time Protocol

The Network Time Protocol (NTP) service for branch servers synchronizes with the administrative server NTP, which in turn, must be configured to get the time

from a reliable source. The branch servers pull the images from the administrative server using the possyncimages script.

## **RSYNC**

RSYNC is used to release the area with operating system (OS) images on the administrative server. This service is used to transfer the images to the branch servers.

### **3.4 Building and deploying a point-of-sale image**

IBM Retail Environment for SUSE Linux V2.0 ships with four sample POS images. These images are not meant to be deployed in a production environment. They are designed to be starting points for creating a custom image. The images can be loaded to demonstrate the ease of building and deployment of a new POS image. The sample POS images are:

- ▶ **Linux without X Windows (Minimal) image**

This image is 37 MB, boots to a Linux command line interface, and is a good starting point for any client image that does not require Java, a browser, or X Windows. It provides a basic command line interface and is appropriate for adding a text-based application that does not require graphics. This is the smallest of the four images and is ideal for a diskless client with a 2 x 20 display or text-based full screen. To support a native application, add additional C/C++ libraries to this image.

- ▶ **Linux-tuned for a Java-based application (Java) image**

This image, which is 105 MB, is a good starting point for many Java-based applications, and includes both the X Windows system and a Java Runtime Environment (JRE). Using the image specification documents provided by IBM Retail Environment for SUSE Linux, the IBM JRE 1.4.2 is automatically included in the client image.

- ▶ **Linux-tuned for a browser-based application (browser) image**

This image is 155 MB, and is similar to the Java image, in that, it includes both the X Windows system and a JRE. However, it also includes a native browser. The browser image is the most common starting point for many application providers. It is slightly larger than the Java image, and a candidate for both diskless and disk POS terminals, depending on the size of the image and amount of available random access memory (RAM).

- ▶ **Linux-tuned for a full desktop environment (desktop) image**

This image, the largest of the four images, is 560 MB and includes a populated desktop, either GNU Network Object Model Environment

(GNOME) or KDE. It is used on POS devices with a hard disk. However, it is rarely used in building a client image, since most POS applications do not want to make a desktop available to the store associate. Use this image when creating a manager's POS terminal that will also be used for store mail and store reports.

### 3.4.1 Building and deploying a point-of-sale image

The processes involved in building and deploying a POS image is depicted in Figure 3-8 and described subsequently.

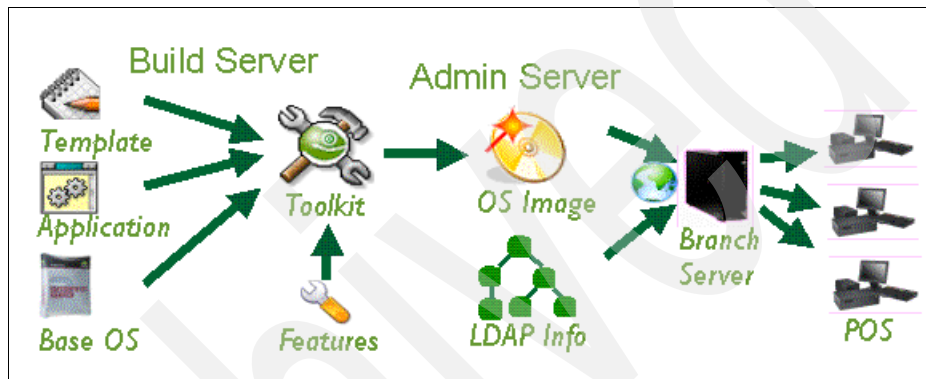


Figure 3-8 POS solution deployment and management

The following steps are involved in building and deploying a POS image:

1. Build the POS client image.
2. Send the image from the build server to the RSYNC directory on the administrative server.
3. On a request from the branch server, the image is sent from the RSYNC directory on the administrative server to the RSYNC directory on the branch server.
4. The branch server makes a request for all the images to be sent from the administrative server.
5. The branch server moves the image from the RSYNC directory to the TFTP directory.
6. The image is sent to the POS client on boot.
7. The initial program load (IPL) bootstrap checks to see if a new image is required.



### 3.4.2 Initial load

For the initial load, the following steps are involved:

1. A PXE load request is sent from the POS device to the branch server.
2. The bootstrap is returned to the POS device from the branch server.
3. The Linux boot image is then sent to the POS device.
4. The boot image sends a load request to the branch server.
5. The branch server determines if this is a new POS based on the MAC address.
6. The branch server sends a request to register the POS device, based on the Medium Access Control (MAC) address in the LDAP.
7. The terminal is registered in the LDAP, and the load data is returned to the branch server.
8. The branch server creates a flat load definition template for the POS device, based on the MAC address.
9. The branch server sends the client image to the POS device.

### 3.4.3 Subsequent loads

For subsequent loads, the following steps are involved:

1. The PXE load request is sent to the branch server.
2. The bootstrap is returned to the POS device.
3. The Linux boot image is sent to the POS device.
4. The boot image sends a load request to the branch server.
5. The branch server knows this terminal by the MAC address and returns the current image version and the message digest 5 (md5).
6. The POS device checks the installed image, the version, and the md5. If any of them differ, it requests a new image load. If all of them match, it loads from the hard drive.
7. The new image load is requested by the POS device, and the branch server sends the current client image.

## 3.5 Requirements

This section lists the hardware and software supported by IBM Retail Environment for SUSE LINUX V2.

### 3.5.1 Supported hardware

IBM Retail Environment for SUSE Linux supports a variety of POS and server hardware configurations. For additional information about IBM POS systems, refer to the following Web site:

<http://www.ibm.com/products/retail/products/>

Novell provides a YES CERTIFIED bulletin to identify which IBM hardware configurations are certified by Novell for which software products. For more information about the same, go to the following Web site:

<http://developer.novell.com/yessearch/Search.jsp>

When this Web site opens, select **IBM** from the Company list, and either **SUSE Linux Enterprise Server 9** or **Novell Linux Point of Service 9, Powered by SUSE Linux** from the Novell Product list.

#### Point-of-sale hardware

IBM Retail Environment for SUSE Linux supports many models of IBM POS hardware. Each of the IBM POS systems supported by IBM Retail Environment for SUSE Linux have been certified by Novell for running NLPOS and by IBM for running IBM Retail Environment for SUSE Linux. Some IBM POS systems have also been certified to act as branch servers.

Instead of every supported model being listed, an “x” in the Type field indicates the part of the type that may be substituted for any alphanumeric character. For example, 4694-2x5 indicates that both the 4694-205 and 4694-245 models are supported.

The following models are supported:

- ▶ 4694-2x5 SurePOS 4694
- ▶ 4694-2x6 SurePOS 4694
- ▶ 4694-2x7 SurePOS 4694
- ▶ 4694-3x7 SurePOS 4694
- ▶ 4810-x1x SurePOS 300
- ▶ 4810-x2x SurePOS 300
- ▶ 4810-x3x SurePOS 300
- ▶ 4614-A0x SureOne®
- ▶ 4614-Pxx SureOne

- ▶ 4615-Cxx SureOne
- ▶ 4615-Jxx SureOne
- ▶ 4800-1xx SurePOS 700
- ▶ 4800-2xx SurePOS 700
- ▶ 4800-73x SurePOS 700
- ▶ 4800-75x SurePOS 700
- ▶ 4800-7x1 SurePOS 700
- ▶ 4800-7x2 SurePOS 700
- ▶ 4835-xx0 IBM Kiosk
- ▶ 4835-xx1 IBM Kiosk
- ▶ 4835-xx2 IBM Kiosk
- ▶ 4835-xx3 IBM Kiosk
- ▶ 4836-x3x Anyplace Kiosk
- ▶ 4838-x3x Anyplace Kiosk
- ▶ 4840-xx1 SurePOS 500
- ▶ 4840-xx2 SurePOS 500
- ▶ 4840-xx3 SurePOS 500

## Server hardware

While servers may be dedicated in some cases, they may be combined with POS terminal functions in some others. Some combinations of servers may also be valid candidates for high availability configurations. The servers you select must have the appropriate certification for their intended purpose and adequate hardware capability to support the target load.

The following list provides details about the models to consider for your intended configuration:

- ▶ Dedicated servers may be selected from among certified xSeries® models, as well as from a select set of certified IBM POS models.
- ▶ Combined Branch Server/POS Terminal (POS-Branch) images may be run on the IBM POS models shown in Table 3-1.
- ▶ HA is supported only on servers with adequate hardware capability, as shown in Table 3-1.

Table 3-1 High availability hardware capability

Model	Dedicated server	POS-Branch combined	HA pair	Notes
a	Yes	No	Yes	<ul style="list-style-type: none"> <li>▶ 1 GB RAM required</li> <li>▶ xSeries models are not certified to run the POS-Branch image</li> </ul>
SP700 4800-C41  SP700 4800-741  SP700 4800-781  SP700 4800-C42  SP700 4800-742  SP700 4800-782	Yes	Yes	See Notes	<ul style="list-style-type: none"> <li>▶ 1 GB RAM required</li> <li>▶ Can run HA with dedicated server</li> <li>▶ Running HA with POS-Branch image is not recommended</li> </ul>
SP500 4840-5x3	Yes	Yes	No	<ul style="list-style-type: none"> <li>▶ 1 GB RAM required</li> <li>▶ SP500 models are not HA-capable</li> <li>▶ Use with POS-Branch only for lightly loaded terminal</li> </ul>

a. IBM Retail Environment for SUSE Linux supports xSeries servers that have been certified for SLES9 or NLPOS9. When selecting a branch server, consult the Novell SLES9 server requirements.

## IBM point-of-sale terminal as a server

IBM Retail Environment for SUSE Linux also supports selected POS terminals running as an SLES branch server. The POS terminal must have a minimum 1 GB of memory. There are two ways to install a POS terminal with SLES:

- ▶ Install NLPOS9 Admin/Branch the same way you would on an xSeries server. The method of installation could be FTP, HTTP, Samba, NFS, or CD. This method does not allow the running of a POS application.

**Note:** If two POS terminals are used in a high availability mode and installed with NLPOS9, both the POS terminals must run NLPOS9.

- ▶ Use the iresImageBuilder to build a POS-Branch image that is based on SLES distribution, and install this image with a CD. In this method, the POS system can also be running the POS application.

**Notes:** The POS-Branch image cannot be used on an xSeries server.

If the POS terminal is running a POS-Branch image and is being used in a high availability mode, both the POS terminals must run the POS-Branch image.

## Point-of-sale systems certified for running SUSE Linux Enterprise Server 9

The following systems have been certified to run SLES:

- ▶ SP700 4800-C41
- ▶ SP700 4800-741
- ▶ SP700 4800-781
- ▶ SP700 4800-C42
- ▶ SP700 4800-742
- ▶ SP700 4800-782
- ▶ SP500 4840-563
- ▶ SP500 4840-533
- ▶ SP500 4840-543
- ▶ SP500 4840-553
- ▶ SP500 4840-5x3

**Note:** Only the SurePOS 4800-700 models can be used as a high availability pair.

Any POS terminal that is used as a combined POS-Branch server should not be a high-volume terminal. Ideally, the combined POS-Branch server should be an

infrequently used terminal, where reduced performance is not a significant issue. Expect to see degraded POS performance in a combined system. To some extent, the degree of performance degradation depends on the other applications that run on the server side of the machine. If this machine has heavy usage as an ISP, it should not be used as a terminal.

**Note:** The minimum memory requirement for a POS terminal running as a branch server is 1 GB.

Servers in a high availability arrangement must run the same SUSE Linux Enterprise Server install. For example, when running the xSeries servers in high-availability mode, the server pair must be installed with NLPOS9. The POS-Branch image is not supported on the xSeries Servers. While running the IBM POS terminals in high availability mode, install the server pair with NLPOS9 or the POS-Branch server image.

If POS terminals are used in high availability mode, we recommend that you do *not* run a POS application.

### 3.5.2 Supported software

The IBM Retail Environment for SUSE Linux supports the following software:

- ▶ Novell Point of Service
  - Industry-standard Linux distribution (Novell Linux Point of Service) that is distributed and licensed to the retailer by Novell
  - Server images modeled on SUSE Linux Enterprise Server 9
  - Client images modeled on Novell Linux Desktop 9 (NLD)
- ▶ Java
  - IBM Java Development Kit (JDK) 1.4.2
- ▶ Drivers (included in the IBM Retail Environment for SUSE Linux V2 package)  
JavaPOS 1.7.5 and POSS/LIN 3.1.0
- ▶ Management tools

This includes SLES system management utilities, including centralized POS management with the offering's enterprise administrative server.

- ▶ **Middleware**  
Select IBM middleware supported on SUSE Linux Enterprise Server-based offerings and as part of Store Integration Framework
- ▶ **Applications**  
Multiple IBM Business Partner software solutions

## 3.6 Additional information

For additional information about IBM Retail Environment for SUSE Linux, refer to the following Web sites:

- ▶ IBM Retail Store Solutions page  
<http://www.ibm.com/products/retail/?re=industry>
- ▶ IBM Retail Environment for SUSE Linux V2 Web portal  
<http://www.ibm.com/products/retail/products/software/ires/>
- ▶ Novell Linux Point of Service (NLPOS)  
<http://www.novell.com/products/linuxpointofservice/>
- ▶ SUSE Linux Enterprise Server (SLES)  
<http://www.novell.com/products/server9/>
- ▶ IBM Retail Environment for SUSE Linux Announcement Letter  
<http://www.ibm.com/common/ssi/fcgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS205-150>







## Part 2

# High availability architecture options in WebSphere Remote Server

Part 2 discusses the high availability architecture options available in WebSphere Remote Server. It also provides information about the configuration requirements for high availability in a WebSphere Remote Server environment.



# High availability solutions for WebSphere Remote Server

This chapter provides an overview of high availability solutions for IBM WebSphere Remote Server and describes the basic high availability concepts. In the scenarios we created, we used IBM Tivoli System Automation software to implement the high availability solutions. This chapter also provides an overview of Tivoli System Automation. It also describes a high availability architecture for WebSphere Remote Server using Tivoli System Automation.

This chapter presents two solutions: a simple solution using Distributed Replicated Block Device (DRBD) and an advanced configuration solution.

This chapter contains the following sections:

- ▶ 4.1, “Overview of high availability architecture” on page 64
- ▶ 4.2, “IBM Tivoli System Automation” on page 67
- ▶ 4.3, “High availability configuration for WebSphere Remote Server overview” on page 72
- ▶ 4.4, “High availability configuration for WebSphere Remote Server with DRBD” on page 73
- ▶ 4.5, “Advanced high availability configuration for WebSphere Remote Server” on page 78

## 4.1 Overview of high availability architecture

High availability is the system management strategy of quickly restoring essential services in the event of a system, component, or application failure. The goal is to minimize the down time. The most common solution for the failure of a system performing critical business operations is to have another system waiting to assume the failed system's workload, and continue the business operations.

The term *cluster* is used here to describe a collection of nodes and resources such as disks and networks that work together to ensure that high availability of services is running within the cluster. If one of the machines in the cluster fails, the resources required to maintain the business operations are transferred to another available machine in the same cluster.

The two main cluster configurations are:

- ▶ Standby configuration

This is the most basic cluster configuration, in which one node works, while the other node acts as a standby. The standby node does not work and is referred to as an *idle node*. Such a configuration requires a high degree of hardware redundancy. This is the configuration we used in our scenario.

- ▶ Takeover configuration

This is a more advanced configuration, in which all the nodes perform some kind of work, and where critical work can be taken over in the event of a node failure. In a one-sided takeover configuration, a standby node performs additional, non-critical, non-movable work. In a mutual takeover configuration, all the nodes perform highly available or movable work. This approach has not been discussed in our scenario.

When planning high availability architecture, the following requirements should be in place:

- ▶ A method for automatic detection of failed resources must exist.
- ▶ Automatic transfer of resource ownership to one or more surviving cluster members should occur in the event of failure.

A number of software products are available to perform system monitoring and resource takeover functionalities. This book discusses the use of IBM Tivoli System Automation for Multiplatforms.

A basic high availability scenario is described to illustrate high availability architecture. Figure 4-1 shows the initial setup. The cluster contains two servers, node1 and node2. The two machines have the same hardware and software configuration. Both are running two applications, app1 and app2.

Node1 is a primary server. All the client requests are normally served by this cluster member. Node2 is a standby server. Both app1 and app2 on node2 are initially in the idle state. To provide failover support, Tivoli System Automation runs on both nodes.

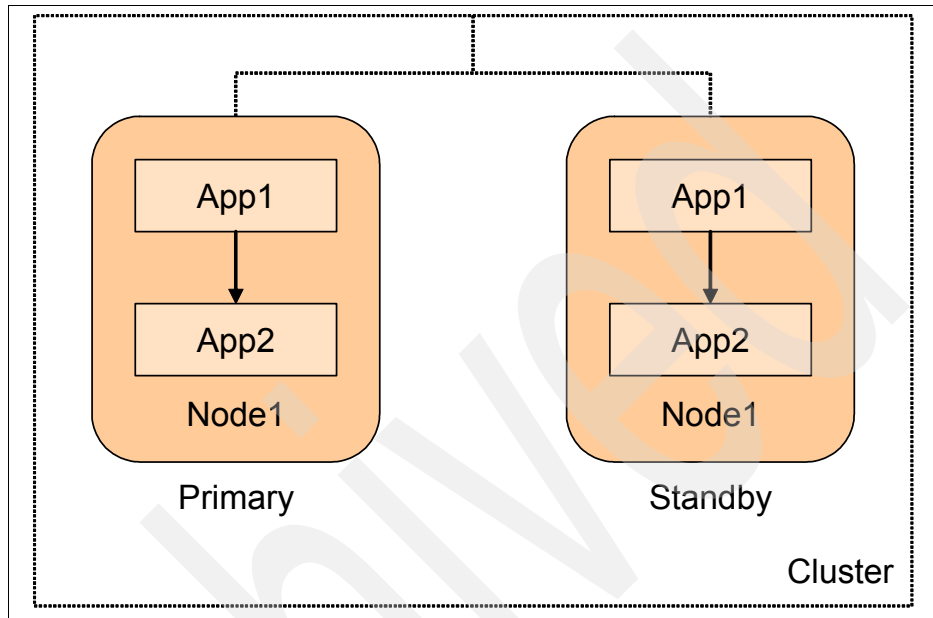


Figure 4-1 Initial configuration

In this configuration:

- Scenario 1: App1 has failed on node1, as illustrated in Figure 4-2.

In such as situation, Tivoli System Automation detects the failure and activates app1 on the standby server. A request from app1 on node1 is served by app1 on node2.

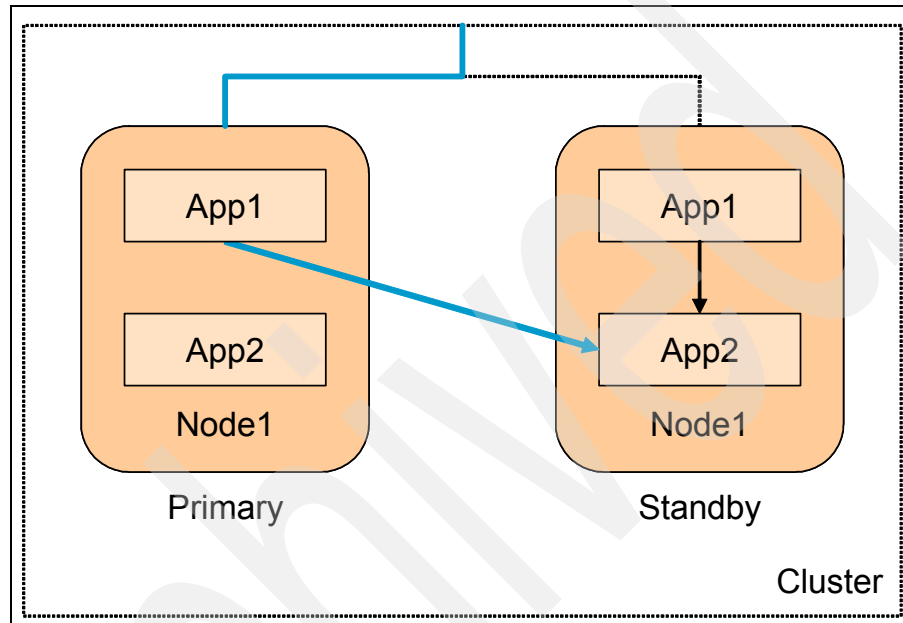


Figure 4-2 Scenario 1: Application failure on primary node

- Scenario 2: Hardware failure of node1 occurs, as illustrated in Figure 4-3. In such a situation, node2 takes over, and both the applications are activated on node2.

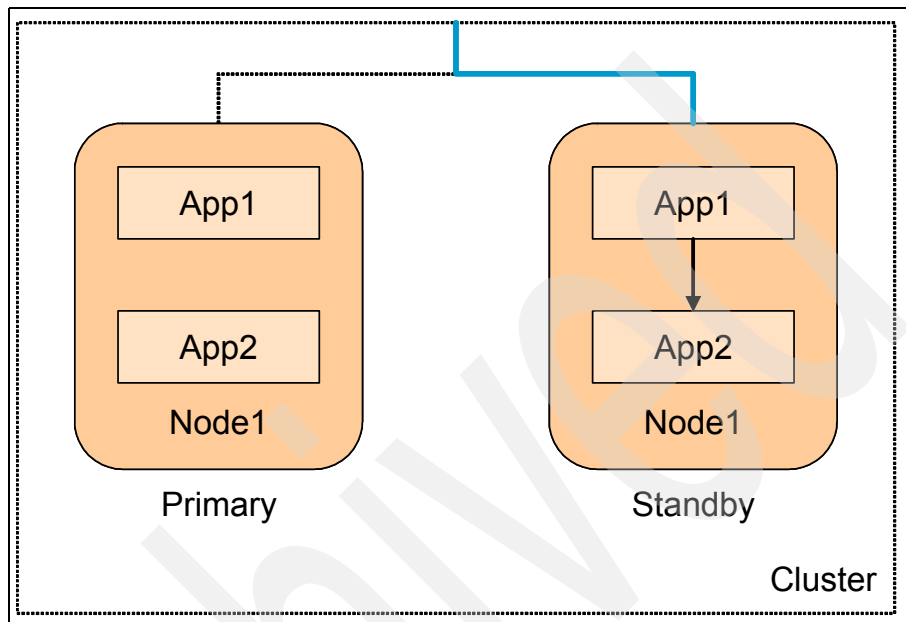


Figure 4-3 Scenario 2: Hardware failure of primary node

To accomplish a smooth failover, we recommend that the applications share or replicate some application data, with the configuration for data sharing being application-specific. The type of data to be shared or replicated and the mechanism for accomplishing this vary from application to application.

## 4.2 IBM Tivoli System Automation

Tivoli System Automation manages the availability of applications running on Linux systems or clusters on IBM @server xSeries, zSeries®, iSeries™, pSeries®, and Advanced Interactive eXecutive (AIX) systems or clusters.

Tivoli System Automation provides a high availability environment. It offers mainframe-like high availability through fast detection of outages and sophisticated knowledge about application components and their relationships. It enables quick and consistent recovery of failed resources and entire applications that are either in place or on another system of a Linux cluster or AIX cluster, without any operator intervention. It relieves operators from the tedium of manual

monitoring and remembering application components and relationships, thereby eliminating human errors.

Tivoli System Automation allows the configuration of high availability systems through the use of policies that define the relationships among the various components. These policies can be applied to existing applications with minor modifications. Once the relationships are established, Tivoli System Automation assumes responsibility for managing the applications on the specified nodes as configured. This reduces implementation time and the need for complex coding of applications. In addition, resources and systems can be added without modifying the scripts. Sample policies are available for Tivoli System Automation. To download them, refer to the following Web site:

<ftp://ftp.software.ibm.com/software/tivoli/products/sys-auto-linux/>

Tivoli System Automation automatically restarts the failed resources or applications that are either in place or on another system of a Linux or AIX cluster. This reduces system outages to a large extent.

Tivoli System Automation manages the cluster-wide relationships among the resources it is responsible for. If the applications need to be moved among nodes, the start and stop relationships, node requirements, and any preliminary or follow-up actions are automatically handled by Tivoli System Automation. This removes the need for manual command entry, thereby reducing human errors.

Resources can be grouped together in Tivoli System Automation. Once grouped, all the relationships among the members of the group can be established, including location relationships, start and stop relationships, and so on. After the entire configuration is complete, operations can be performed against the entire group as a single entity. Once again, this eliminates the need for operators to remember the application components and relationships, reducing the possibility



of errors. Figure 4-4 provides an overview of the Tivoli System Automation architecture.

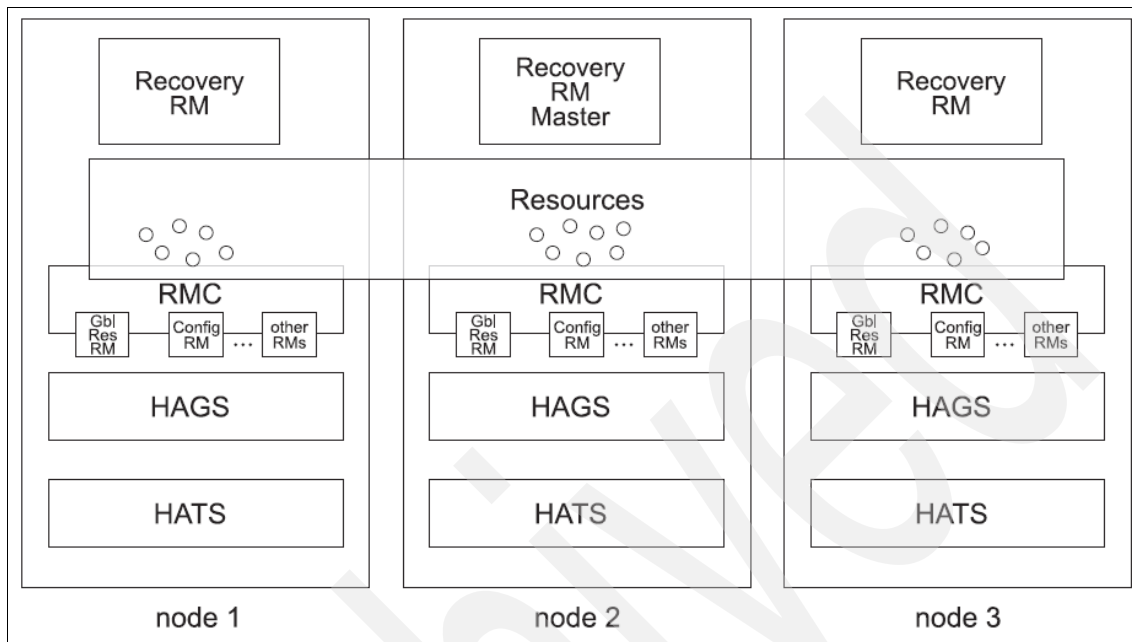


Figure 4-4 Overview of the Tivoli System Automation architecture

Table 4-1 summarizes the features, advantages, and benefits of Tivoli System Automation.

Table 4-1 Features, advantages, and benefits of Tivoli System Automation

Features	Advantages	Benefits
Customizable out-of-the-box resource monitoring	Out-of-the-box outage detection	Fast outage detection, reducing operator requirements for manual monitoring
End-to-end heterogeneous application recovery	Fast and consistent enterprise-wide restart in place or move capabilities for components or whole applications, reducing system outages	Reduces down time of critical business applications. Maintains system availability in business context. Improves reliability of the IT infrastructure.
Automatic starting, stopping, and moving of heterogeneous applications	Takes care of enterprise-wide relationships, start/stop order, and required pre and post start/stop actions.	Relieves operator from manual command entry. Improves efficiency by reducing human errors.

Features	Advantages	Benefits
Resources can be grouped into applications. Grouping can be enterprise-wide.	Reduces complexity of operations by automating at the application level, reducing manual intervention, and need to remember application components and dependencies.	Improves efficiency, reliability, and serviceability of the IT Infrastructure, enabling a better integration of IT resources with business goals.
Able to define interdependent resource relationships and associate conditions with resources.	Frees operators from remembering application components and relationships. Can use sophisticated knowledge about application components and their relationships to decide corrective actions within the right context.	Reduces operation errors. Improves service levels and optimal IT resource utilization by enabling lower-priority business applications to be shut down, while keeping higher-priority business applications running, based on business priorities.
Policy-based automation	New resources or systems can be added without re-writing scripts	Reduces automation implementation time, and coding and support effort. Leverages manpower through reduced education requirements. No programming skills are required for policy definition. Eases application growth and scaling.
Reliable, scalable	Cluster-wide heart beating and reliable messaging service	Helps faster cluster implementation to enable a more reliable, dynamic automation

### 4.2.1 Components of IBM Tivoli System Automation

Tivoli System Automation has numerous user-friendly components as explained in the following sections.

#### **Reliable Scalable Cluster Technology**

Reliable Scalable Cluster Technology (RSCT), a set of software products that together provide a comprehensive clustering environment for AIX and Linux, is fully integrated into Tivoli System Automation. It is the infrastructure that provides clusters with improved system availability, scalability, and ease of use.

RSCT provides three basic components or layers of functionality:

- **Resource Monitoring and Control (RMC)**

This provides global access for configuring, monitoring, and controlling resources in a peer domain.

- ▶ High Availability Group Services (HAGS)  
This is a distributed coordination, messaging, and synchronization service.
- ▶ High Availability Topology Services (HATS)  
This provides a scalable heartbeat for adapter and node failure detection, besides a reliable messaging service in a peer domain.

## Resource Manager

A Resource Manager (RM) is a software layer between a resource and RMC. Resource classes are managed by various resource managers, depending on the type of resource being managed.

The following resource managers are provided by Tivoli System Automation:

- ▶ Recovery RM (IBM.RecoveryRM)  
This resource manager serves as the decision-making engine for Tivoli System Automation. Once a policy is defined for defining resource availabilities and relationships, this information is supplied to the Recovery RM. This RM runs on every node in the cluster, with only one Recovery RM designated as the master. The master evaluates the monitoring information from the various resource managers. Once a situation that requires intervention develops, the Recovery RM drives the decisions that result in start or stop operations on the resources, as needed.
- ▶ Global Resource RM (IBM.GblResRM)  
The Global Resource RM supports two resource classes:
  - IBM.Application  
The IBM.Application resource class defines the behavior for general application resources. Use this class to start, stop, and monitor processes. As a generic class, it is very flexible and can be used to monitor and control various kind of resources. Most of the applications you will automate will be done using this class.
  - IBM.ServiceIP  
This application class defines the behavior of Internet Protocol (IP) address resources. It allows you to assign IP addresses to an adapter. In effect, it allows IP addresses to *float* among nodes.
- ▶ Configuration RM (IBM.ConfigRM)  
The Configuration RM is used in cluster definition. In addition, quorum support, which is a means of ensuring data integrity when portions of a cluster lose communication, is provided.

- ▶ Event response RM (IBM.ERRM)

The Event Response RM provides the ability to monitor conditions in the cluster for the RMC system to react in certain ways.

- ▶ Test RM (IBM.TestRM)

The Test RM manages test resources and provides functions to manipulate the operational state of these resources. The resource manager is operational only in a peer domain mode, and provides the resource class IBM.Test. The Test resource manager does not control real resources.

## Operations console

The operations console is a browser-based graphical user interface (GUI). It is used to monitor and manage the resources managed by Tivoli System Automation. This is done in a mode called *direct access mode*. For end-to-end automation management component of Tivoli System Automation for Multiplatforms, two more modes are provided:

- ▶ End-to-end automation mode
- ▶ First-level automation mode

These modes are described in *End-to-End Automation Management User's Guide and Reference*, SC33-8211.

## End-to-end automation adapter

In the base component of Tivoli System Automation, the end-to-end automation adapter is used to operate automated resources directly from the operations console. In the end-to-end automation management component of Tivoli System Automation, the end-to-end automation adapter is used to automate the operation of resources within first-level automation domains.

## 4.3 High availability configuration for WebSphere Remote Server overview

**Attention:** The following high available architecture scenarios were developed and documented in a WebSphere Remote Server V5.1.2.1 environment.

The WebSphere Remote Server stack contains components such as the WebSphere Application Server, DB2, WebSphere MQ, IBM HTTP Server, and Remote Management Agent.

The high availability solution for WebSphere Remote Server provides failover support for each of these components, as well as for the WebSphere Remote Server as a whole. Multiple solutions exist for providing failover support to the WebSphere Remote Server stack.

In this book, we present two different approaches to building highly available configurations for the WebSphere Remote Server, each of which are characterized by unique strengths and weaknesses. Depending on your requirements, choose one of these configurations or a combination of the two.

The first solution, which is easier to configure, uses DRBD to replicate data on the two servers. DRBD is a feature of SUSE Linux that provides data replication. In order to avoid complexities while providing failover support for individual components, the DRBD fails the whole machine when an application failure occurs. For details on about this configuration, refer to 4.4, “High availability configuration for WebSphere Remote Server with DRBD” on page 73.

The second solution, which is more complex, provides failover support for individual components and does not use DRBD. It uses a component-specific mechanism for data replication, for example, the High Availability Disaster Recovery (HADR) feature for DB2. For details about this configuration, refer to 4.5, “Advanced high availability configuration for WebSphere Remote Server” on page 78.

Both the solutions use Tivoli System Automation to detect failures and control the failover.

## **4.4 High availability configuration for WebSphere Remote Server with DRBD**

The configuration referred to in this book assumes the presence of a primary server and a standby server, with both having an identical setup. However, only the primary server is used to process the client's requests. If the primary server fails, the standby server takes over. Use the DRBD to replicate the data between the primary and the standby server.

DRBD is a block device designed to build high availability clusters. This is done by mirroring an entire block device through a dedicated network. DRBD takes over the data, writes it to the local disk, and sends it to the other host. On the other host, it takes it to the disk there. Thus, DRBD provides synchronous replication from the primary to the secondary server, preparing it for being up-to-date and for failover, if necessary.

Use the Tivoli System Automation to monitor the environment for failures. If a failure is detected, the Tivoli System Automation executes a failover workflow, whose logic can be customized, to bring up the secondary server as the primary server.

In this configuration, the chosen approach is to fail the entire server if a failure occurs in the application that is running on the server. An application fails due to the failure of any of the stack components. Once the failure is detected, start all the resources on the standby server and give control to that server.

To shield the client from control transfer, use a floating IP address. This address is created and controlled by Tivoli System Automation. Clients use this IP address to connect to the server. Initially, the IP address points to the primary server. In case of failure, it switches to the standby server. At this point, all the requests from the client automatically go to the standby server. Figure 4-5 illustrates the setup we worked on.

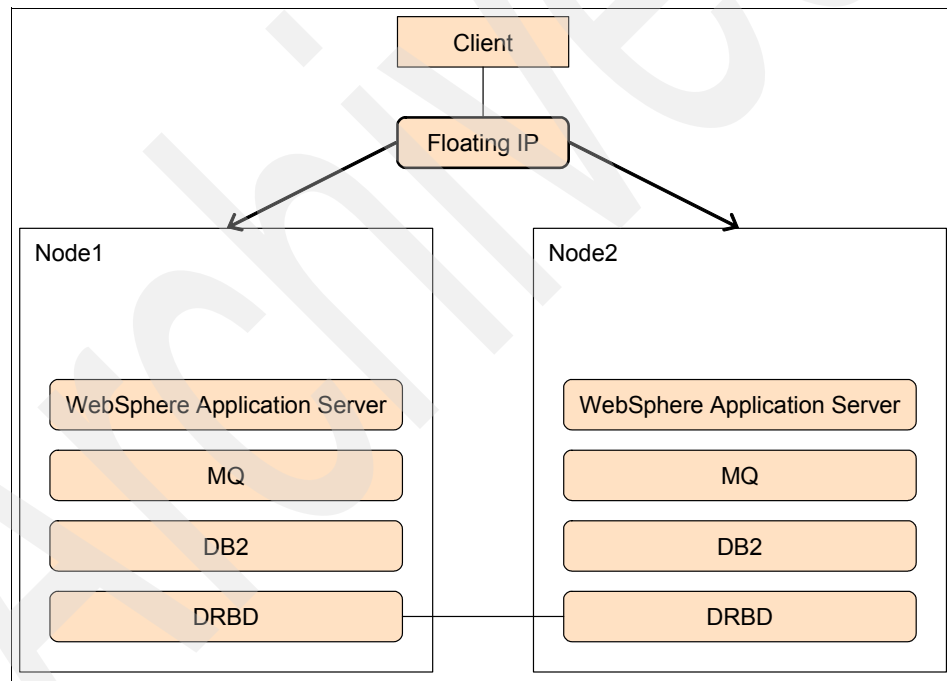


Figure 4-5 Floating IP address configuration

To detect the failure of the primary server, monitor the resources on the primary server using Tivoli System Automation. The main resource to monitor is the actual application running on the WebSphere Application Server. This application uses all the stack components. If failure occurs in one of the

components, the application will fail by itself eventually. In our scenario, we executed the failover to the standby server. All the resources were stopped on the primary server and started on the standby server. The floating IP was transferred to the standby server. To simultaneously stop all the software running on the primary server, we used the reboot function.

This implementation makes several assumptions. In this scenario, since our main resource to monitor was the application, the first assumption is that scripts are provided by the application for application monitoring. These scripts need to be application-specific and be available for this scenario to work.

In this scenario, we did not use WebSphere Network Deployment. This means that the WebSphere Application Server running on each node were independent servers. The two servers were set up identically and were running the same application. Since the servers were independent of each other, no session replication took place. The assumption we make is that the actual application running on the WebSphere Application Server takes care of session information loss in case of failure.

In the sample application used, there was no critical session information that had to be recovered. The only challenge was authentication information. To overcome this, we changed our sample application. If the client could not access the server with the current session connection, it would open a new session. In case of failover, this new session opened on another server.

To recover all the other information such as MQ and DB2 data during failover, store the MQ queues and the DB2 table spaces in the DRBD-managed partition.

Figure 4-6 illustrates the flow during the failover in setup. Initially, node1 is the primary node and serves all the requests. The primary node has all the resources running on it. Since the WebSphere Application Server and the DB2 take a while to start, start them on the standby server as well, initially. Although the WebSphere Application Server and the DB2 run on the standby server, they are in a passive mode and do not take any requests. The floating IP initially points to the primary server.

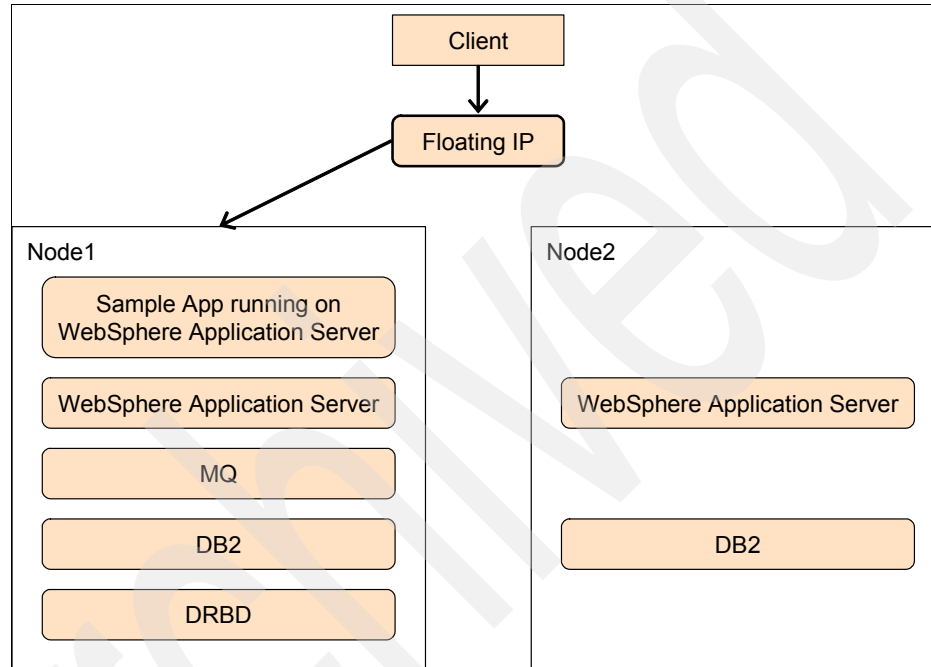


Figure 4-6 Initial floating IP address points to node1



Figure 4-7 displays a node1 failure. If either a hardware or software failure occurs on the primary server, Tivoli System Automation detects the failure, initiating the floating resources on node2. Since the WebSphere Application Server and the DB2 are already running on node2, this process is reasonably fast.

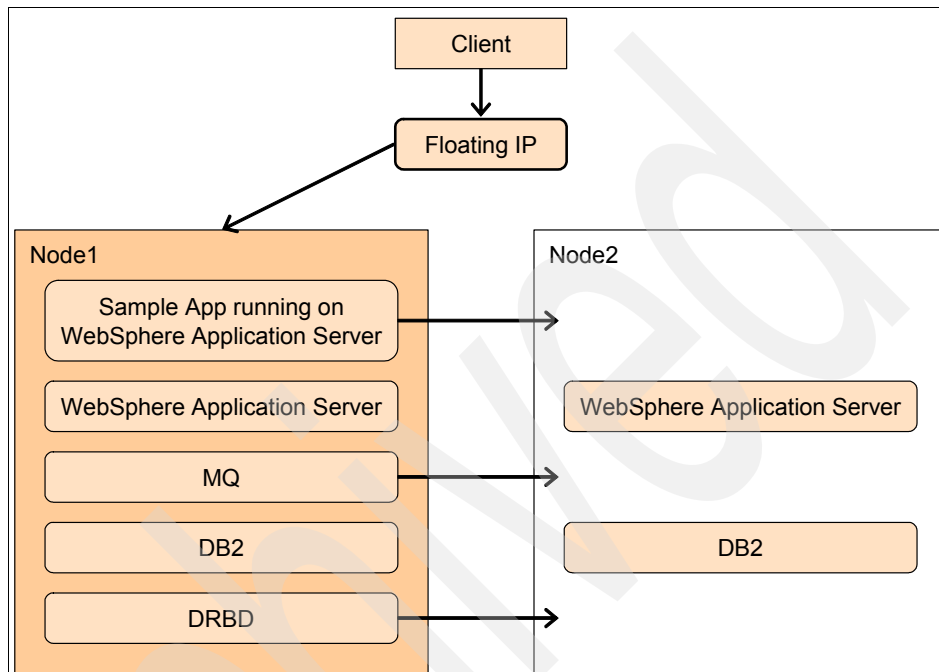


Figure 4-7 Node1 failure

Figure 4-8 displays failover to node2. Once the failover process is complete, all the components run on node2. Tivoli System Automation mounts the DRBD partition on the standby server, making all the data available for MQ and DB2 running on that server. Floating IP now points to the standby server. The standby server becomes primary and starts servicing all the requests. At the same time, node1 gets rebooted. Once node1 comes up after the reboot, it becomes the standby node.

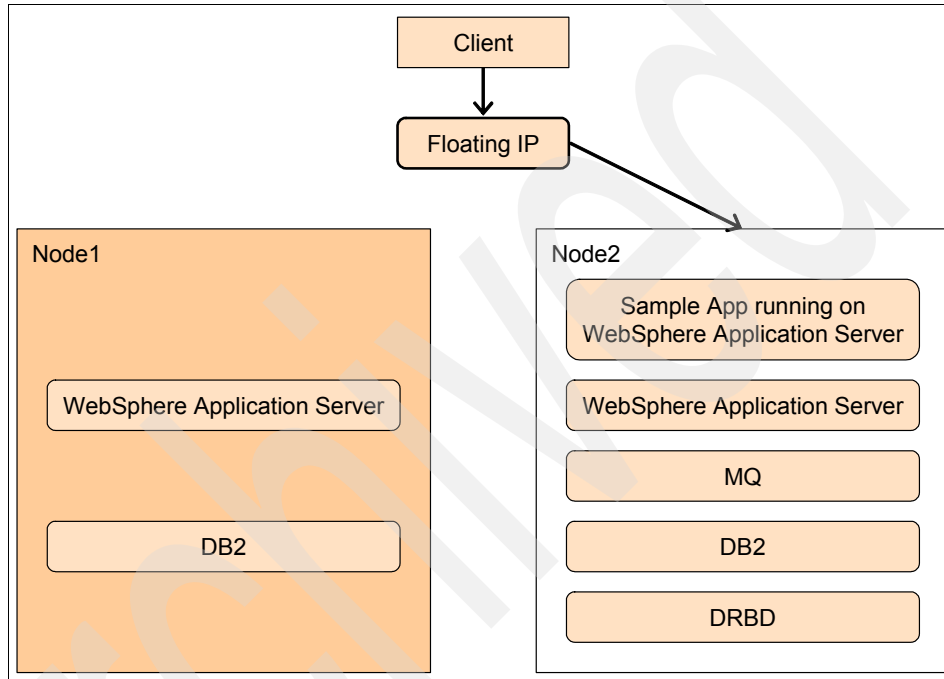


Figure 4-8 Failover to node2

The steps involved in setting up this configuration are described in Chapter 5, “Implementing high availability configuration for WebSphere Remote Server using DRBD” on page 87.

## 4.5 Advanced high availability configuration for WebSphere Remote Server

Using Tivoli System Automation allows you to fail the node and individual software components. The WebSphere Remote Server stack has IBM HTTP Server, WebSphere Application Server, DB2, WebSphere MQ, and Remote Management Agent (RMA). Default scripts are available for most of these

components. In addition to this, high availability solution is documented separately for each component in various IBM redbooks and white papers. For more details, refer to “Related publications” on page 409.

The goal is to implement and document a high availability solution for the entire WebSphere Remote Server stack that will provide failover support for individual components. Make sure that the sum total of all the resources and relationships is integrated and works for the entire software stack.

This section provides details about using the Tivoli System Automation to provide a high availability solution for a WebSphere Remote Server stack running in a store environment. Chapter 6, “Advanced high availability configuration for WebSphere Remote Server” on page 109 provides step-by-step instructions to set up an advanced high availability solution for WebSphere Remote Server.

Figure 4-9 illustrates the software stack on the two systems. In our setup, we used two identical machines, with both of them having the same software stack

running on them. We did not use a shared disk or DRBD. Depending on your requirements, use a shared disk or DRBD as part of the high availability solution.

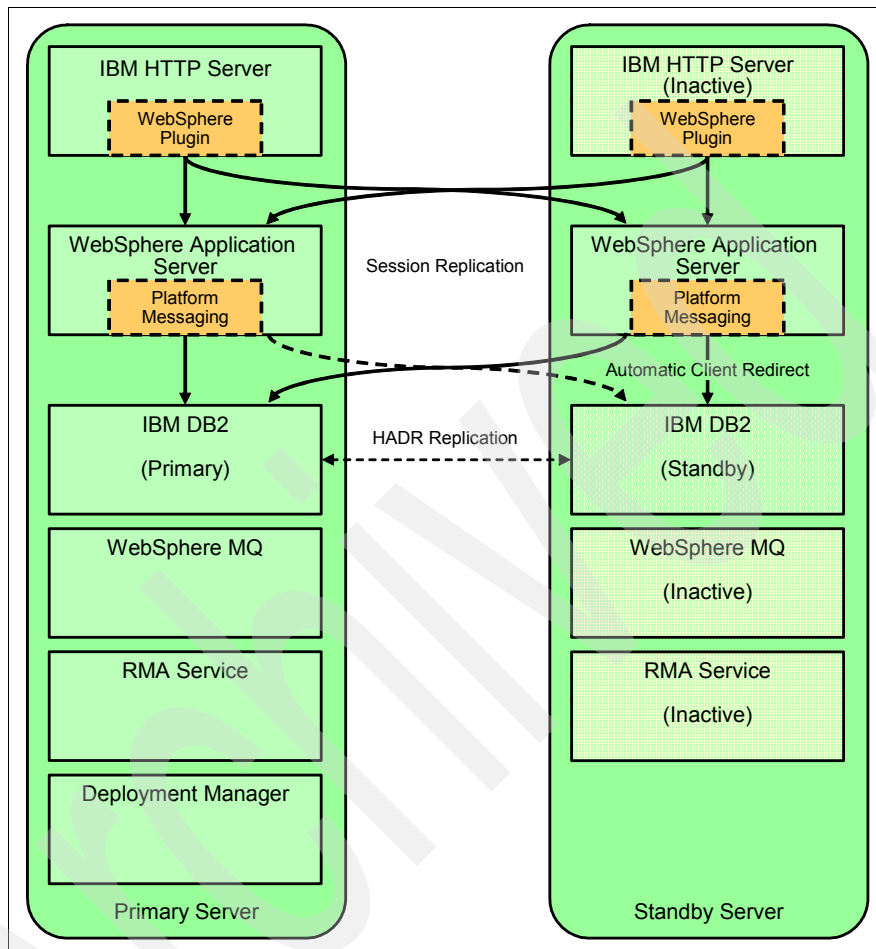


Figure 4-9 Object model

The following sections explain the high availability configuration of each of the stack components.

### 4.5.1 High availability configuration for IBM HTTP Server

In the high availability configuration, the IBM HTTP Server is a floating resource. This means that the server is not tied to a particular node. It can be started in both nodes, but can run on only one node at a time. On both servers, the IBM HTTP Server is configured to use one IP address, regardless of the node it

currently runs on. Thus, the location of the Web server is transparent outside the cluster where no changes have to be performed, when the Web server is moved from one node to another.

The Web server's IP address is a separate IP address in the cluster and does not match any IP address assigned to the network adapters on each cluster node that are made in system definitions outside of Tivoli System Automation. In contrast, the address for the IBM HTTP Server IP is created by the Tivoli System Automation and is an additional alias address on an appropriate network adapter on the node where the Web server resides. When the Web server moves to a new location, the alias address is removed from the former node and recreated on the new node, where the Web server is about to be restarted. Therefore, when the Tivoli System Automation detects failure of the IBM HTTP Server, it starts up the IBM HTTP Server server on the standby node and changes the alias to point to the standby node. The steps relevant to this are detailed in Chapter 5, "Implementing high availability configuration for WebSphere Remote Server using DRBD" on page 87.

The Tivoli System Automation provides scripts and a sample configuration file to configure the Tivoli System Automation resources for the IBM HTTP Server. Figure 4-10 illustrates the resources created for IBM HTTP Server.

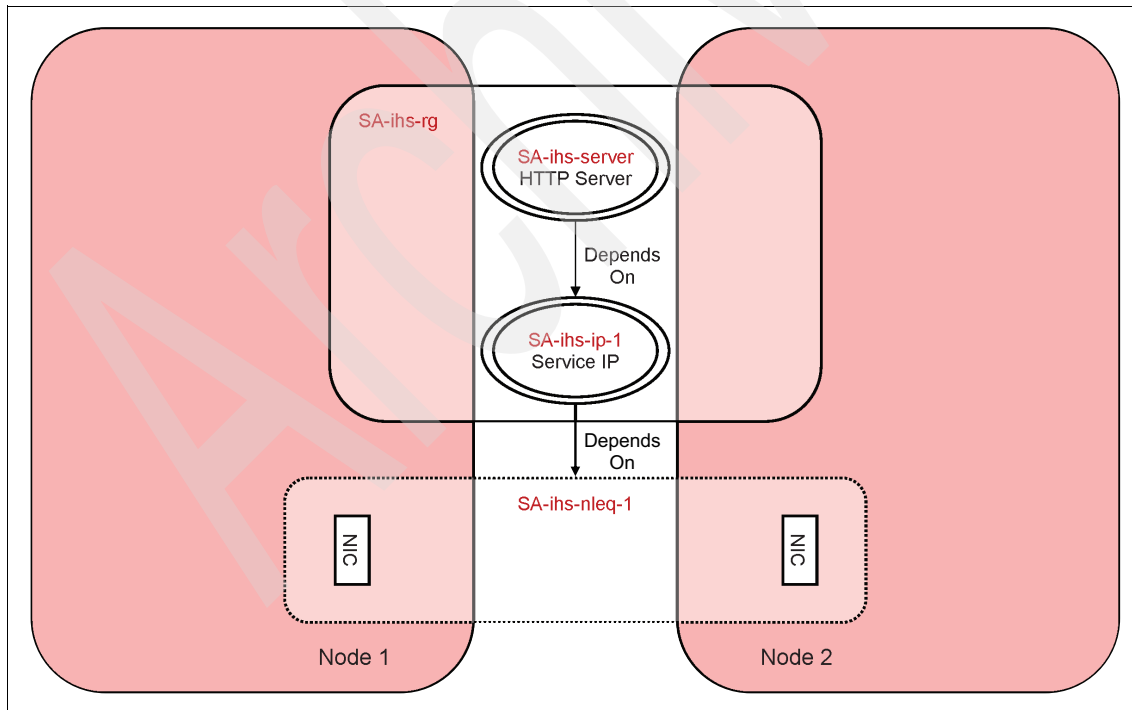


Figure 4-10 Tivoli resources for the IBM HTTP server

## 4.5.2 High availability configuration for WebSphere Application Server

To set up a high availability architecture for WebSphere Application Server, use WebSphere Network Deployment to set up a WebSphere Application Server cluster. The application servers running on primary and standby node should be a part of the cluster. To avoid a delay caused by server startup, run both the servers at the same time. However, only one server should service requests. To accomplish this, have a floating IP. In case the primary node fails, the IP switches to the standby mode, and the application server on the standby node takes over. To avoid any loss of session data, session replication needs to be enabled. Refer to WebSphere Application Server documentation in “Related publications” on page 409 for information about how to set this up.

Default policies for WebSphere Application Server are available and are used for monitoring and recovery. Figure 4-11 shows the Tivoli System Automation resources configuration.

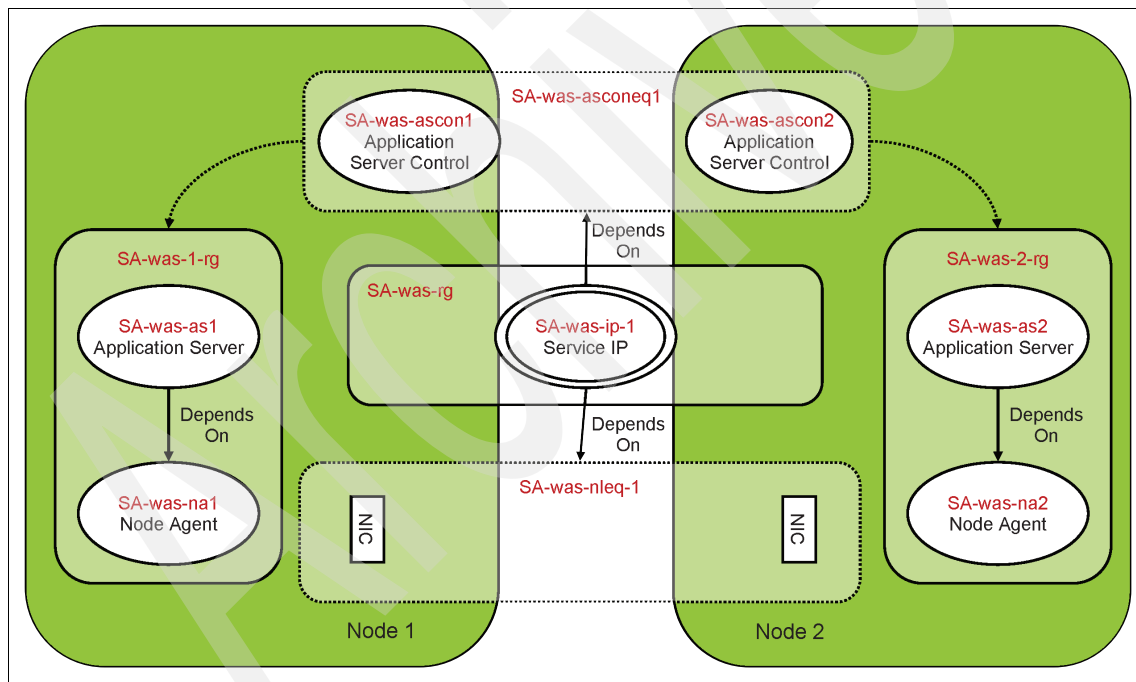


Figure 4-11 Tivoli System Automation resources configuration

### 4.5.3 High availability configuration for DB2

To implement a high availability configuration for DB2, ensure that you have a data replication mechanism between the two DB2 instances. The DB2 feature that allows this is called HADR. This protects against data loss by replicating data changes from a primary database to a standby database. Transactions that occur on the primary database are applied to the designated standby database. If the primary database becomes unavailable, the standby database can be used. Chapter 6, “Advanced high availability configuration for WebSphere Remote Server” on page 109 details the procedure involved in configuring the database to use HADR.

The Tivoli System Automation is used to restart the primary database, so that no operator is required to bring the system to its original state. The IBM DB2 8.2 product set includes default policies for the Tivoli System Automation. Figure 4-12 shows the setup for DB2.

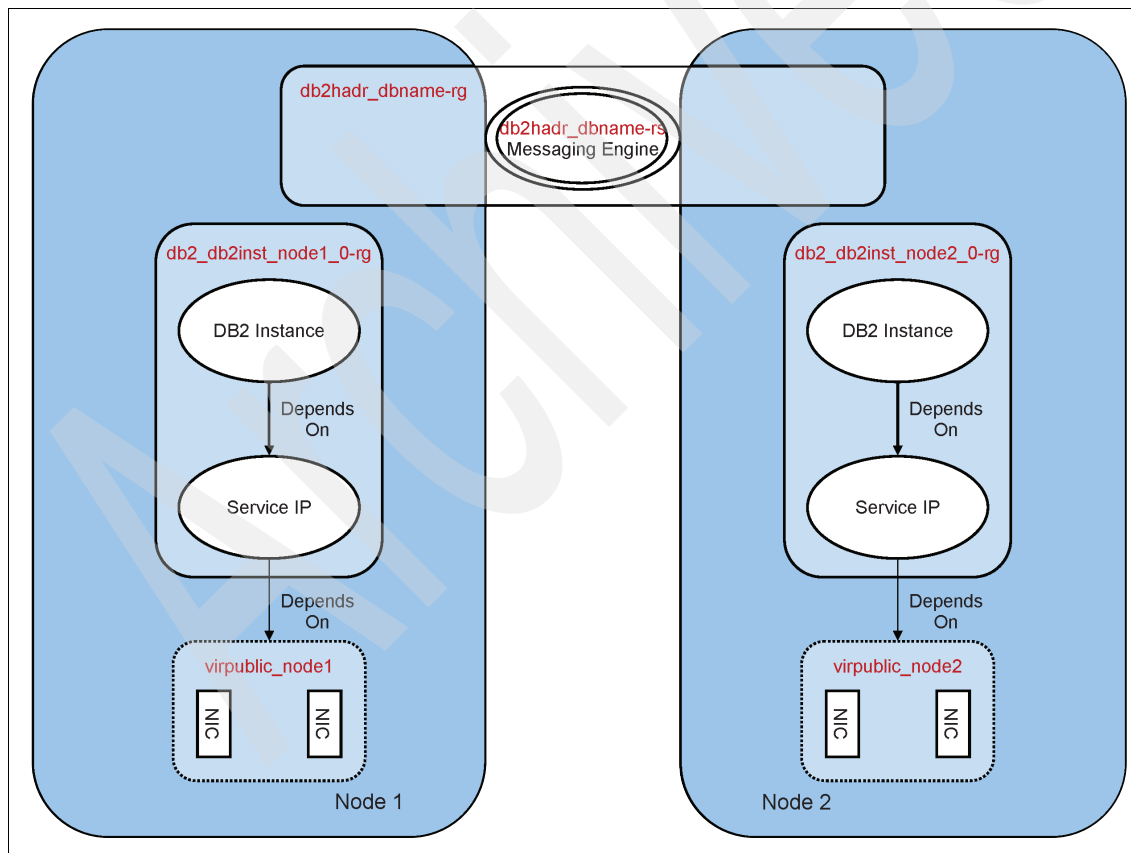


Figure 4-12 Tivoli support for DB2

#### 4.5.4 High availability configuration for WebSphere MQ

In this scenario, we limit ourselves to the simple setup without DRBD, as a result of which data is not replicated between the nodes. In this setup, WebSphere MQ messages that are present on the queues when system fail over occurs, will be lost. Depending on the application and the type of messages being sent, this can either be a negligible or considerable drawback. In this book, we assume that the loss of messages is negligible and that the shared disk will not be used.

The Tivoli System Automation provides scripts and a sample configuration file to configure Tivoli System Automation resources for WebSphere MQ. Figure 4-13 illustrates the resources created by the script to manage WebSphere MQ.

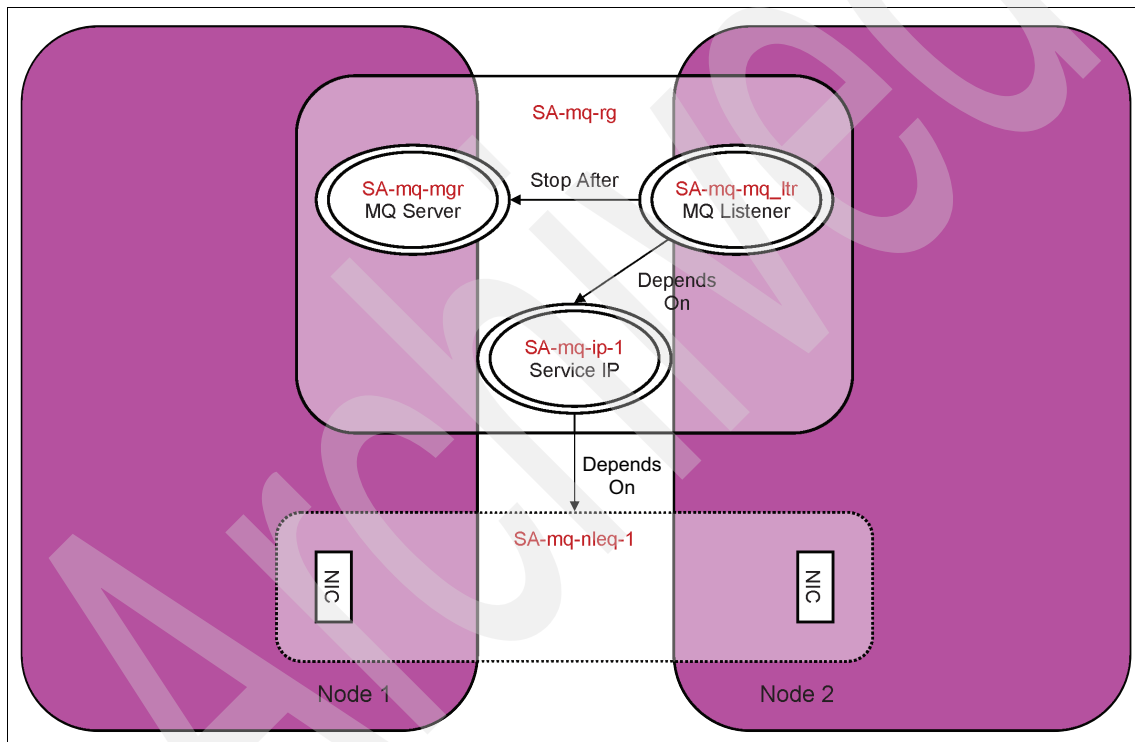


Figure 4-13 Tivoli resources to manager WebSphere MQ



## 4.5.5 High availability configuration for Remote Management Agent

RMA should run on one server at a time. To shield client applications from managing connections to two servers, float an IP address between the systems, float an IP address between the systems along with the RMA. In fact, this scenario is identical to that used for the IBM HTTP Server. The Tivoli System Automation does not provide a default policy to manage this product. You can make a similar configuration for the RMA by modifying the configuration files for the IBM HTTP Server policy. Figure 4-14 illustrates the modified Tivoli policy to manage RMA.

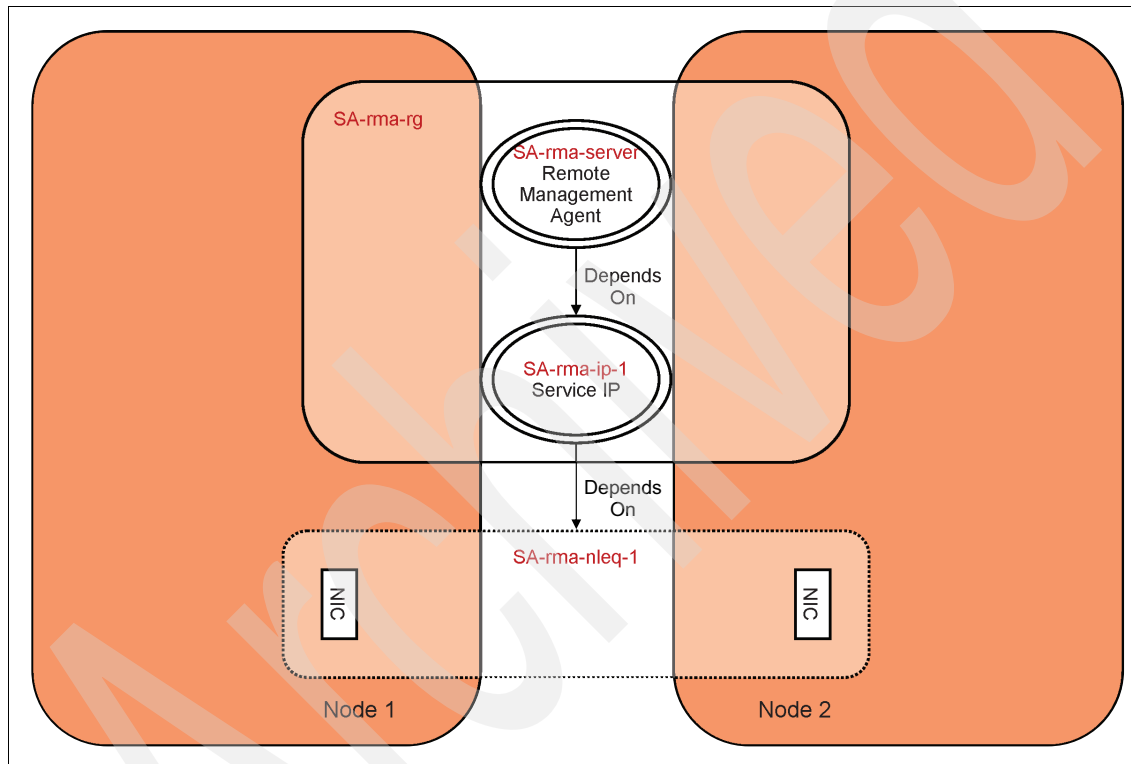


Figure 4-14 Modified Tivoli policy to manage RMA

Chapter 6, “Advanced high availability configuration for WebSphere Remote Server” on page 109, provides instructions on using the Tivoli System Automation scripts to set up a high availability solution for WebSphere Remote Server.



## Implementing high availability configuration for WebSphere Remote Server using DRBD

This chapter provides detailed instructions for implementing high availability solution for WebSphere Remote Server using Distributed Replicated Block Device (DRBD). It contains the following sections:

- ▶ 5.1, “Overview of Distributed Replicated Block Device” on page 88
- ▶ 5.2, “Setting up DRBD” on page 88
- ▶ 5.3, “Setting up Tivoli System Automation” on page 93
- ▶ 5.4, “Configuring DB2” on page 96
- ▶ 5.5, “Configuring WebSphere MQ” on page 98
- ▶ 5.5.1, “Configuring MQ Broker” on page 101
- ▶ 5.6, “Configuring WebSphere Application Server” on page 102
- ▶ 5.7, “Configuring DRBD” on page 103
- ▶ 5.8, “Hosts file update” on page 104
- ▶ 5.9, “Application monitoring” on page 104
- ▶ 5.10, “Failover testing” on page 106

## 5.1 Overview of Distributed Replicated Block Device

DRBD is a block device that is designed to build high availability clusters. This is done by mirroring an entire block device through a dedicated network. You can think of it as a network RAID-1 configuration. DRBD takes over the data, writes it to the local disk, and sends it to a disk in another host.

The other component that is required along with DRBD for a high availability solution is Tivoli System Automation, which detects the failure of one node and brings up the DRBD on another. Use the DRBD to replicate application data for applications such as WebSphere Application Server, WebSphere MQ, and DB2.

### 5.1.1 How DRBD works

Each of the many devices the DRBD provides has a state that can be *primary* or *secondary*. The application is supposed to run and access the device on the node with the primary device. Every write is sent to the local, lower-level block device and to the node with the device in secondary state. The secondary device simply writes the data to its lower-level block device. Reads are always carried out locally.

If the primary node fails, Tivoli System Automation switches the secondary device to primary state, and starts the application there. If the failed node comes up again, it becomes a new secondary node and synchronizes its content with the primary. This happens in the background, without interruption of service. DRBD carries out intelligent resynchronization when possible.

## 5.2 Setting up DRBD

Our setup included two identical machines with the same software stack containing WebSphere Application Server, DB2, WebSphere MQ, and DRBD, installed on both.

### 5.2.1 Prerequisites for setting up DRBD

There are certain prerequisites for network configuration:

- ▶ Presence of two network adapters on each machine
- ▶ Connection of the two servers by a crossover cable or private network

DRBD requires a separate disk partition. Create an unmounted primary partition. In our setup, we also used another partition for DRBD metadata. This partition

requires a minimum of 128 MB for metadata. Table 5-1 describes our partitions for DRBD.

Table 5-1 Partitions for DRBD

Device	Size	Type	Mount
/dev/hdb1	203 MB	Linux native	
/dev/hdb2	55.7 GB	Linux native	

## 5.2.2 Installing DRBD

The DRBD package comes with the IBM Retail Environment for SUSE Linux software CDs. If you do not have it on your system, install it by selecting **YaST2 → Install and Remove Software**.

## 5.2.3 Configuring DRBD

To configure the DRBD for the high availability solution, perform the following tasks:

1. Ensure that you have defined the device `/dev/nb0` by entering the following command:  

```
ls /dev/nb0
```
2. If this device does not exist, create it on both the servers by entering the following command:  

```
mknod /dev/nb0 b 147 0
```
3. Edit the `/etc/fstab` file on both the servers.
  - a. If the following line is present, comment it out:  

```
/dev/system/drbd /work ext3 defaults 1 2
```
  - b. Add the following line to the `/etc/fstab` file:  

```
/dev/nb0 /work ext3 noauto 0 0
```
  - c. Save your changes.
4. Edit the `/etc/drbd.conf` file on both the servers.
  - a. The installed DRBD package contains a sample `drbd.conf` file. This file is found in the DRBD installation directory. (In our setup, the sample file in question was `/usr/share/doc/packages/drbd`.) Copy this file into the `/etc/` directory by entering the following command:  

```
cp /usr/share/doc/packages/drbd/drbd.conf /etc/drbd.conf
```

- b. Edit the file to specify the resource definition. as shown in Example 5-1. Linux-ires3 and ires4 are server names for the two servers and the IP addresses used are the Internet Protocols (IPs) for the private network connecting the two servers. In Example 5-1, only one resource is defined since all the other resource definitions have been deleted.

*Example 5-1 drbd.conf*

---

```
resource drbd0 {
  protocol C;

  startup {
    deg-wfc-timeout 120;
  }

  disk {
    on-io-error detach;
  }

  net {
  }

  syncer {
    rate 10M;
    group 1;
    al extents 257;
  }

  on linux-ires3 {
    device      /dev/nb0;
    disk        /dev/hdb2;
    address      9.43.188.11:7789;
    meta-disk    /dev/hdb1[0];
  }
  on ires4 {
    device      /dev/nb0;
    disk        /dev/hdb2;
    address      9.43.188.10:7789;
    meta-disk    /dev/hdb1[0];
  }
}
```

---

**Note:** The Tivoli System Automation scripts assume that the device used is nb0 and the resource name is drbd0. To use another device or resource name, modify the drbd\_mount.ksh script provided with the book. For information about how to get the Tivoli System Automation scripts, refer to Appendix 'Additional material' .

c. Save the changes to the file. Ensure that both the servers have the same version of the /etc/drbd.conf file.

d. Run the following command on both the servers to add DRBD to the system startup:

```
chkconfig --add drbd
```

5. Run the following commands on both the machines:

```
modprobe drbd; drbdadm up all; dmesg | tail; cat /proc/drbd
```

6. After executing this command on both the servers, the contents of the /proc/drbd should look as follows:

```
# cat /proc/drbd
version: 0.7.0 svn $Rev: 1442 $ (api:74/proto:74)

0: cs:Connected st:Secondary/Secondary ld:Inconsistent
   ns:0 nr:0 dw:0 dr:0 al:0 bm:1 lo:0 pe:0 ua:0 ap:0
```

**Troubleshooting:** The command in step 5 may generate an error or the content of the /proc/drbd may be displayed as follows:

```
# cat /proc/drbd
version: 0.7.0 svn $Rev: 1442 $ (api:74/proto:74)

0: cs:Connected st:Secondary/unknown ld:Inconsistent
   ns:0 nr:0 dw:0 dr:0 al:0 bm:1 lo:0 pe:0 ua:0 ap:0
```

Run the following commands on both the servers:

```
drbdadm down all
drbdadm up all
cat /proc/drbd
```

The contents of the /proc/drbd will now appear as shown in step 6.

7. Decide which of the two servers will be the primary server, and enter the following command on that server:

```
drbdadm -- --do-what-I-say primary all
```

8. At this point, synchronization takes place. Enter the following command:

```
dmesg | tail ; cat /proc/drbd
```

Example 5-2 shows the output that you should see.

*Example 5-2 dmesg output*

---

```
>dmesg | tail ; cat /proc/drbd
drbd0: Resync started as SyncTarget (need to sync 5000 KB [1250 bits
set]).
```

```
version: 0.7.0 svn $Rev: 1442 $ (api:74/proto:74)
```

```
0: cs:SyncTarget st:Secondary/Primary ld:Inconsistent
   ns:0 nr:15000 dw:15000 dr:0 al:0 bm:6 lo:0 pe:0 ua:0 ap:0
   [=====>.....] sync'ed: 50.0% (5000/5000)K
   finish: 0:00:12 speed: 5 (5) K/sec
```

# or, to give an example from a larger device:

```
0: cs:SyncTarget st:Secondary/Primary ld:Inconsistent
   ns:0 nr:27311780 dw:27311780 dr:0 al:0 bm:3447 lo:134 pe:493
   ua:134 ap:0
   [=====>.....] sync'ed: 93.7% (1818/28487)M
   finish: 0:01:07 speed: 27,482 (25,008) K/sec
```

---

9. Create a directory on both the machines to be used as a mount point for DRBD partition, with the following command:

```
mkdir -p /work
```

10. Ensure that DRBD is working correctly by performing the following tasks:

- a. Mount the partition on the primary server by entering the following command:

```
mount /dev/nb0 /work
```

- b. Create a test file in the /work directory.

- c. Enter the following command on the primary server:

```
umount /work && drbdadm secondary all
```

- d. Enter the following command on the secondary server:

```
drbdadm primary all
```

- e. If the /work directory is not created on the secondary server, create it by entering the following command:

```
mkdir -p /work
```



- f. Mount the /work directory on the secondary server by entering the following command:  

```
mount /dev/nb0 /work
```
- g. Ensure that the test file is found in the /work directory.

The DRBD is now configured.

## 5.3 Setting up Tivoli System Automation

The process of setting up Tivoli System Automation involves multiple steps. The following sections explain each step.

### 5.3.1 Installing Tivoli System Automation

Install Tivoli System Automation on both the servers by performing the following tasks:

1. Obtain the Tivoli System Automation base components installation tar file named `Base_Component.tar`.
2. Untar the file by entering the following command:  

```
tar -xvf Base_Component.tar
```
3. Switch to the untarred `SAM2100Base` directory and run the following command:  

```
./installSAM
```
4. Read the license agreement and type `y` (for yes) to accept the agreement. The Tivoli System Automation installation starts.
5. Repeat these steps to install Tivoli System Automation on the secondary server.

For more information about installing Tivoli System Automation, refer to *IBM Tivoli System Automation for Multiplatforms: Base Component User's Guide, Version 2.1*, SC33-8210-04.

### 5.3.2 Creating a cluster with Tivoli System Automation

Before configuring individual Tivoli System Automation for the various software in the stack, follow these steps to set up Tivoli System Automation cluster:

1. Access a console on each node in the cluster and log in as root.

2. Ensure that the environment variable `CT_MANAGEMENT_SCOPE=2` is defined on each node. To do this, edit the `/etc/profile` file and add the following lines to the end of the file.

```
...
CT_MANAGEMENT_SCOPE=2
export CT_MANAGEMENT_SCOPE
```

**Note:** Create a `.bashrc` file under the `/root` directory for the environment variables to be set in `/etc/profile`.

3. Type the **preprnode** command on all nodes to allow communication between the cluster nodes. Node1 and node2 are the host names of the two nodes in the following command:

```
preprnode hostname node1 hostname node2
```

4. Create a cluster with the name `HA_Domain` running on node1 and node2 by entering the following command from any node:

```
mkrpdomain HA_Domain hostname node1 hostname node2
```

5. To look up the status of the `HA_Domain`, run the **lsrpdmain** command:

```
>lsrpdmain
Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
HA_Domain Offline 2.4.3.1          No           12347 12348
```

The cluster is defined offline.

6. Run the following **starttrpdmain** command to bring the cluster online:

```
starttrpdmain HA_Domain
```

7. Run the **lsrpdmain** command again, and the cluster is still in the process of starting. After a short time, the cluster starts. Thus, when you run the **lsrpdmain** command again, the cluster is seen online.

### 5.3.3 Setting up a tie-breaker

In a two-node cluster, should network communication between the nodes fail, it is difficult to determine which system should take control of the available resources. Tivoli System Automation grants control to the system that has Operational Quorum. Typically, Operational Quorum is granted to the subcluster that has more than half the nodes in the cluster. If there are only two nodes in the cluster, this is impossible to achieve if the two nodes are not communicating.

Tivoli System Automation provides a tie-breaker feature that grants Operational Quorum to the system that reserves the tie-breaker. Two types of automatic tie-breakers are available. The first relies on a shared disk between the systems

and is the preferred tie-breaker method. (In our setup, which had minimal configuration, a shared disk was not available.) The second type of tie-breaker is the network tie-breaker. This uses the IP address of an external system available in the network.

The first system that is able to access or ping the IP address wins the tie-breaker, and is granted Operational Quorum. A good choice for the external IP address is that of the network gateway. Record the IP address using the dotted quad notation, for example, 9.1.0.102, for the network tiebreaker.

To define the network tie-breaker resource, use the following Reliable Scalable Cluster Technology (RSCT) command:

```
mkrsrc IBM.TieBreaker Type="EXEC" Name="networktiebreaker"  
DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/samtb_net  
Address=[networkTiebreakerIP] Log=1';
```

Activate the network tie-breaker with the following command:

```
chrsrc -c IBM.PeerNode OpQuorumTieBreaker="networktiebreaker"
```

### 5.3.4 Installing Tivoli System Automation policies for WebSphere Remote Server

**Attention:** This high available architecture scenario was developed and documented in a WebSphere Remote Server V5.1.2.1 environment.

The rpm containing default policies should be installed on both the servers. (In our setup, we used some of the default policies provided with Tivoli System Automation.) To do this, obtain the sam.policies-1.2.2.0-05201.i386.rpm file from “Using the Web material” on page 394 and run the following command:

```
rpm -i sam.policies-1.2.2.0-05201.i386.rpm.
```

**Tip:** To check the installation, view the /usr/sbin/rsct/sapolicies directory. Numerous directories, including was, mqm, apache, ihs, and so on should be visible.

This book also provides sample policies for a high availability setup with DRBD. The policies are provided in Appendix ‘Additional material’.

To install Tivoli System Automation policies for WebSphere Remote Server, perform these tasks on both the servers:

1. Download the cluster.zip file.
2. Create a /usr/local/cluster directory.
3. Unzip the cluster.zip file into this directory. Ensure that file permissions are changed to make the files executable.

## 5.4 Configuring DB2

To replicate DB2 data between the two servers using DRBD, create the DB2 table spaces on the DRBD partition. Before beginning work on the DRBD partition, make sure that the DRBD partition is set to *primary* on your intended primary node. The /work directory should be mounted on the primary server.

You can perform the following tasks only if you have DB2 installed and a database has been created and exists on the primary server:

1. Log in to the primary server as root.
2. Create a /work/HA\_DB2 directory.
3. Move the table spaces to the DRBD partition using the following commands:

```
cp -R /home/db2inst1/db2inst1 /work/HA_DB2
mkdir /home/db2inst1/db2inst1/backup
mv /home/db2inst1/db2inst1/ /home/db2inst1/db2inst1/backup
```

4. Create a symbolic link by entering the following command:
5. Make sure that all the permissions are correct on the work directory. Both root and db2inst1 users should have access to that directory.
6. Start the DRBD partition on the standby server by performing these tasks:

- a. Enter the following command on the primary server:

```
umount /work && drbdadm secondary all
```

- b. Enter the following command on the secondary server:

```
drbdadm primary all
```

- c. To specify where the device is to be mounted, enter the following command:

```
mount /dev/nb0 /work
```

- d. Ensure that you see the directory /work/HA\_DB2/db2inst1 on the secondary server now.

7. Create a symbolic link from the DB2 directory to the DRBD on the secondary server by entering the following commands:

```
mkdir /home/db2inst1/db2inst1/backup
mv /home/db2inst1/db2inst1/ /home/db2inst1/db2inst1/backup
ln -s /work/HA_DB2/db2inst1 /home/db2inst1/db2inst1
```

8. Catalog the database on the secondary server by entering the following command:

```
db2 catalog database database name
```

You can now access the database from either server, provided the DRBD partition is mounted on that server.

The next step is to configure the DB2 resources with Tivoli System Automation. DB2 will run on both the servers to shorten the failover time. However, only the DB2 on the primary server will be active. Run the following commands from either node:

1. Change the directory to /usr/local/cluster using the following command:

```
cd /usr/local/cluster
```

2. Create two application resources for DB2 using the following commands:

```
mkrsrc -f db2_1.def IBM.Application
mkrsrc -f db2_2.def IBM.Application
```

3. Create the floating IP resource using the following command:

```
mkrsrc IBM.ServiceIP Name="db2_ip" IPAddress=<db2 cluster
ip-address> NetMask=<netmask> NodeNameList="{<node1>','<node2>'}"
```

A floating IP is a separate IP address in the cluster and does not match any of the IP addresses assigned to the network adapters on each cluster node. The floating IP is created by Tivoli System Automation and is an additional address on an appropriate network adapter on the node where the DB2 resides.

4. Create a resource group with the following commands:

```
mkrsg db2_1-rg
mkrsg db2_2-rg
mkrsg db2_ip-rg
```

5. Add resources to the resource group with the following commands:

```
addrsgmbr -g db2_1-rg IBM.Application:db2_1
addrsgmbr -g db2_2-rg IBM.Application:db2_2
addrsgmbr -g db2_ip-rg IBM.ServiceIP:db2_ip
```

6. Create an equivalency and a relationship. Substitute node1 and node2 with your node names in the first command as follows:

```
mkequ db2_equ IBM.NetworkInterface:eth0:node1,eth0:node2
mkrel -S IBM.ServiceIP:db2_ip -G IBM.Equivalency:db2_equ -p
DependsOn db2-ip-on-db2_equ
```

7. Start the resources, using the **startall** command in the /usr/local/cluster:  
./startall
8. View the status of the resources by entering the **getstatus** command:  
./getstatus

All the DB2 resources will now be online.

## 5.5 Configuring WebSphere MQ

To create MQ queues directly on the DRBD partition, perform the following tasks:

1. On the primary server, create MQ manager on the DRBD partition by entering the following commands:

```
export MQSPREFIX="/work/HA_QM/data"
crtmqm -ld /work/HA_QM/log MY_QUEUE_MANAGER
```

2. Modify the /var/mqm/mqs.ini file on both the servers to include the contents shown in Example 5-3.

*Example 5-3 Modifying the /var/mqm/mqs.ini file*

---

```
QueueManager:
  Name=MY_QUEUE_MANAGER
  Prefix=/work/HA_QM/data
  Directory=MY_QUEUE_MANAGER
DefaultQueueManager:
  Name=MY_QUEUE_MANAGER
```

---

3. Configure the Tivoli System Automation resources for MQ in the following steps:
  - a. Change the directory to /usr/sbin/rsct/sapolicies/mqm with the following command:  
cd /usr/sbin/rsct/sapolicies/mqm

- b. Edit the sa-mq.conf file and modify values such as MQ owner name, queue manager name, node list, MQ virtual IP, according to the environment, as shown in Example 5-4.

*Example 5-4 Modifying the sa-mq.conf file*

---

```
# set default values
MQ_OWNER=<mquser (can be root)>
MQ_MGR=MY_QUEUE_MANAGER
# --list of nodes in the cluster
nodes="linux-ires3 ires4"
# --IP address and netmask
ip_mq="9.42.188.32,255.255.0.0"
(9.42.188.32 is the IP Address defined for store1 in the host
file)
# --List of network interfaces ServiceIP ip_x depends on.
#   Entries are lists of the form
<network-interface-name>:<node-name>,...
nieq_1="eth0:linux-ires3,eth0:ires4"
# --common local mountpoint for shared data

data_var="/work"
```

---

- c. Save and close the file.
- d. Run the following command from the /usr/sbin/rsct/sapolicies/mqm directory:
- ```
./cfgmq
```
- e. This generates the SA-mq-ip-store.def, SA-mq-mgr.def, SA-mq-mq\_lsn.def, mqctrl-mgr, mqctrl-mq\_lsn files under the same directory.
- Edit the mqctrl-mq\_lsn file to add the WebSphere Application Server listener startup script by entering the following command:
- ```
vi mqctrl-mq_lsn
```
- Add these two lines in the # start WAS listener ports section:
- ```
/opt/WebSphere/AppServer/bin/wsadmin.sh -f
/usr/local/cluster/startListenerPort.jacl
```
- f. Save and close the file.
- g. Copy all the files in the /usr/sbin/rsct/sapolicies/mqm directory to node 2.

- h. Create resources on one of the nodes by running the following commands:

```
cd /usr/sbin/rsct/sapolicies/mqm
mkrsrc -f SA-mq-mgr.def IBM.Application
mkrsrc -f SA-mq-mq_lsn.def IBM.Application
mkrsrc -f SA-mq-ip-mq.def IBM.ServiceIP
mkrsrc -f SA-mq-data-var.def IBM.Application
```

- i. Create the resource groups:

```
mkrgr SA-mq-rg
```

- j. Add resources to the resource groups with the following commands:

```
addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-mgr
addrgmbr -m T -g SA-mq-rg IBM.Application:SA-mq-mq_lsn
addrgmbr -m T -g SA-mq-rg IBM.ServiceIP:SA-mq-ip-mq
addrgmbr -g SA-mq-rg IBM.Application:SA-mq-data-var
```

- k. Make relations by running the commands shown in Example 5-5.

*Example 5-5 Making relations*

---

```
mkrel -S IBM.Application:SA-mq-mq_lsn -G
IBM.ServiceIP:SA-mq-ip-mq -p DependsOn SA-mq-mq_lsn-on-ip-mq
mkequ SA-mq-nieq-mq
IBM.NetworkInterface:eth0:linux-ires3,eth0:ires4
mkrel -S IBM.ServiceIP:SA-mq-ip-mq -G
IBM.Equivalency:SA-mq-nieq-mq -p DependsOn SA-mq-ip-on-nieq-mq
mkrel -S IBM.Application:SA-mq-mq_lsn -G
IBM.Application:SA-mq-mgr -p StopAfter SA-mq-mq_lsn-on-mgr
mkrel -p DependsOn -S IBM.Application:SA-mq-mgr -G
IBM.Application:SA-mq-data-var SA-mq-mgr-on-data-var
mkrel -p DependsOn -S IBM.Application:SA-mq-mq_lsn -G
IBM.Application:SA-mq-data-var SA-mq-mq_lsn-on-data-var
```

---



- I. Tune the timeout values as shown in Example 5-6.

*Example 5-6 Tuning Timeout values*

---

```
chrsrc -s "Name='SA-mq-mgr'" IBM.Application
MonitorCommandTimeout=4
chrsrc -s "Name='SA-mq-mgr'" IBM.Application
StartCommandTimeout=30
chrsrc -s "Name='SA-mq-mgr'" IBM.Application
StopCommandTimeout=30
chrsrc -s "Name='SA-mq-mgr'" IBM.Application
MonitorCommandPeriod=5

chrsrc -s "Name='SA-mq-mq_lsn'" IBM.Application
MonitorCommandTimeout=4
chrsrc -s "Name='SA-mq-mq_lsn'" IBM.Application
StartCommandTimeout=30
chrsrc -s "Name='SA-mq-mq_lsn'" IBM.Application
StopCommandTimeout=30
chrsrc -s "Name='SA-mq-mq_lsn'" IBM.Application
MonitorCommandPeriod=5

chrsrc -s "Name='SA-mq-data-var'" IBM.Application
StartCommand="/usr/local/cluster/drbd_mount.ksh start"
chrsrc -s "Name='SA-mq-data-var'" IBM.Application
StopCommand="/usr/local/cluster/drbd_mount.ksh stop"
chrsrc -s "Name='SA-mq-data-var'" IBM.Application
MonitorCommand="/usr/local/cluster/drbd_mount.ksh status"
chrsrc -s "Name='SA-mq-data-var'" IBM.Application
MonitorCommandTimeout=4
chrsrc -s "Name='SA-mq-data-var'" IBM.Application
StartCommandTimeout=30
chrsrc -s "Name='SA-mq-data-var'" IBM.Application
StopCommandTimeout=30
chrsrc -s "Name='SA-mq-data-var'" IBM.Application
MonitorCommandPeriod=5
```

---

### 5.5.1 Configuring MQ Broker

To configure the MQ broker, perform the following steps:

1. Open and edit the mq-brk.def file to specify the node list, the queue manager name, and the MQ user.

2. Create Tivoli System Automation resources for MQ broker, by entering the following commands:

```
cd /usr/local/cluster
mkrsrc -f mq-brk.def IBM.Application
```

3. Add resources to the resource groups with the following command:

```
addrgmbbr -m T -g SA-mq-rg IBM.Application:mq-brk
```

4. Make relations by entering the following commands:

```
mkrel -S IBM.Application:mq-brk -G IBM.Application:SA-mq-mgr -p
DependsOn mq-brk-on-mq
mkrel -p Collocated -S IBM.ResourceGroup:db2_ip-rg -G
IBM.ResourceGroup:SA-mq-rg db2ip_coll_mq
```

5. Increase the priority of the MQ resource group:

```
chrg -p 20 SA-mq-rg
```

6. Run the **startall** and **getstatus** commands:

```
./startall
./getstatus
```

## 5.6 Configuring WebSphere Application Server

**Attention:** This high available architecture scenario was developed and documented in a WebSphere Application Server V5.1.1.6 environment.

To create Tivoli System Automation resources for a WebSphere Application Server running on node1 and node2, perform the following tasks:

1. Create resources by entering the following commands:

```
cd /usr/local/cluster
mkrsrc -f was_1.def IBM.Application
mkrsrc -f was_2.def IBM.Application
```

2. Create resource groups by entering the following commands:

```
mkrsg was_1-rg
mkrsg was_2-rg
```

3. Add resources to the resource groups by entering the following commands:

```
addrgmbbr -m T -g was_1-rg IBM.Application:was_1
addrgmbbr -m T -g was_2-rg IBM.Application:was_2
```

Although the application server runs on both the nodes, only one node is active at a time. To direct requests to the active node and shield the client from knowing which node is active, configure the WebSphere Application Server to use one of the floating IPs created. The WebSphere Application Server Administration console can also be used for this. Perform the following steps:

1. Create a virtual host and host aliases for one of the floating IPs created.
  - a. Add a virtual host by selecting **Environment** → **Virtual Hosts** → **New**.
  - b. Create host aliases. Select **Environment** → **Virtual Hosts** → ***your\_virtual\_host*** → **Host Aliases** → **New**.
  - c. Update the Web server plug-in by selecting **Environment** → **Update Web Server Plugin**.
2. Configure the application to use the virtual host by selecting **Applications** → **Enterprise Applications** → ***your\_application*** → **Map virtual hosts for Web modules**.

## 5.7 Configuring DRBD

The key part of this high availability configuration is the DRBD. With the configuration of the Tivoli System Automation resources for all WebSphere Remote Server stack components, configure the Tivoli System Automation resources for the DRBD. The Tivoli System Automation is then responsible for starting and mounting the DRBD partition.

To configure the DRDB, perform the following tasks:

1. Create Tivoli System Automation resources for DRBD with the following command:

```
mkrsrc -f drbd_mount.def IBM.Application
```
2. Create a resource group by entering this command:

```
mkrg drbd-rg
```
3. Add resources to resource group by entering the following command:

```
addrgmbr -g drbd-rg IBM.Application:drbd_mount
```

## 5.8 Hosts file update

In our setup, we introduced two floating IP addresses in Tivoli System Automation configuration. Generally, the `/etc/hosts` file on each machine should contain the following information:

- ▶ Private network IP of the other node
- ▶ Floating IP for MQ
- ▶ Floating IP for DB2
- ▶ Public IPs of both nodes

Example 5-7 shows sample `/etc/hosts` file entries from the setup for the primary, that is, the `linux-ires3` node.

*Example 5-7 Sample /etc/hosts file entries*

---

|              |                                 |             |
|--------------|---------------------------------|-------------|
| 9.42.188.23  | ires4.rtp.raleigh.ibm.com       | ires4       |
| 9.42.188.24  | HA_HOST                         |             |
| 9.42.188.32  | MQ_HOST                         |             |
| 192.162.1.20 | ires4.rtp.raleigh.ibm.com       | ires4       |
| 9.42.188.22  | linux-ires3.rtp.raleigh.ibm.com | linux-ires3 |

---

In Example 5-7, `HA_HOST` and `MQ_HOST` are the floating IPs for DB2 and MQ respectively. The IP `192.162.1.20` is a private network IP of the second node (`ires4`). The other two IPs are the public IPs of the nodes. Example 5-8 shows the hosts file in the other node, `ires4`.

*Example 5-8 Hosts file in ires4*

---

|              |                                 |             |
|--------------|---------------------------------|-------------|
| 9.42.188.22  | linux-ires3.rtp.raleigh.ibm.com | linux-ires3 |
| 9.42.188.24  | HA_HOST                         |             |
| 9.42.188.32  | MQ_HOST                         |             |
| 192.168.1.10 | linux-ires3.rtp.raleigh.ibm.com | linux-ires3 |
| 9.42.188.23  | ires4.rtp.raleigh.ibm.com       | ires4       |

---

## 5.9 Application monitoring

In this configuration, the main resource to be monitored is the actual application that runs on top of the WebSphere Remote Server stack. If the individual stack component fails, but the application is still running, failover does not occur. Failover occurs only when the application fails. This happens either because of stack component failure, such as DB2, MQ, or WebSphere Application Server, or application failure.

When the script detects application failure, it reboots the primary node and the entire stack fails over to the standby. For this solution to work, an application-specific script that will monitor the application should be developed. This script is not provided with this book. It can be modeled on the sample control scripts provided for other components and should provide actions for the start, stop, and status options as displayed in Example 5-9.

*Example 5-9 Application control script*

---

```
case ${Action} in
    start) start the application
    stop) stop the application
    status) check the status of the application
            if the status is failed, reboot the machine
    *)
```

---

Once the control script is in place, configure the monitor resource. Example 5-10 shows the sample monitor.def file.

*Example 5-10 Sample monitor.def file*

---

```
PersistentResourceAttributes::
Name="monitor"
StartCommand="/usr/local/cluster/app.ksh start"
StopCommand="/usr/local/cluster/app.ksh stop"
MonitorCommand="/usr/local/cluster/app.ksh status"
MonitorCommandPeriod=60
MonitorCommandTimeout=40
StartCommandTimeout=10
StopCommandTimeout=10
ResourceType=1
UserName=root
NodeNameList='{node1,node2}'
```

---

In Example 5-10, app.ksh is the application control script. To configure the application monitor with Tivoli System Automation, perform the following tasks:

1. Create a monitor resource by entering the following command:
2. Create a monitor resource group and add the monitor resource to that group by entering the following commands:

```
mkrsrc -f monitor.def IBM.Application

mkrg mon-rg
addrmgr -g mon-rg IBM.Application:monitor
```

3. Create a relationship between the application monitor and MQ by entering the following commands:
- ```
mkrel -p Collocated -S IBM.Application:monitor -G
IBM.Application:SA-mq-mgr monior_collocated_mq
```

## 5.10 Failover testing

After all the resources are configured, start all of them by issuing the **startall** command from the `/usr/local/cluster` directory:

```
./startall
```

Check the status using the **getstatus** command:

```
./getstatus
```

Example 5-11 illustrates the results of these commands.

*Example 5-11 getstatus output*

---

- Resource Groups and Resources --

Group Name	Resources
-----	-----
db2_1-rg	db2_1
-	-
db2_2-rg	db2_2
-	-
db2_ip-rg	db2_ip
-	-
drbd_mount	drbd_mount
-	-
mon-rg	monitor
-	-
SA-mq-rg	mq-brk
SA-mq-rg	SA-mq-data-var
SA-mq-rg	SA-mq-mq_1sn-2
SA-mq-rg	SA-mq-mgr-2
SA-mq-rg	SA-mq-ip-mq
-	-
was_1-rg	was_1
-	-
was_2-rg	was_2
-	-

- Resources --

State	Resource Name	Node Name
-----	-----	-----
Online	db2_1	linux-ires3
-	-	-
Online	db2_2	ires4
-	-	-
Offline	db2_ip	ires4
Online	db2_ip	linux-ires3
-	-	-
Offline	drbd_mount	ires4
Online	drbd_mount	linux-ires3
-	-	-
Offline	monitor	ires4
Online	monitor	linux-ires3
-	-	-
Offline	mq-brk	ires4
Online	mq-brk	linux-ires3
-	-	-
Offline	SA-mq-data-var	ires4
Online	SA-mq-data-var	linux-ires3
-	-	-
Offline	SA-mq-mq_lsn-2	ires4

	SA-mq-mq_lsn-2	linux-ires3
Online	-	-
-	SA-mq-mgr-2	ires4
Offline	SA-mq-mgr-2	linux-ires3
Online	-	-
-	SA-mq-ip-mq	ires4
Offline	SA-mq-ip-mq	linux-ires3
Online	-	-
-	was_1	linux-ires3
Online	-	-
-	was_2	ires4
Online	-	-
-		

---

At this point, all the resources are online on the primary server, that is, linux-ires3.

To test the failover in case of hardware, perform the following tasks:

1. Shut down the primary server.
2. Observe if all the resources are switched to the standby server, which now becomes the primary server.

To test the failover in case of software, perform the following tasks:

1. Manually shut down one of the stack components, for example, stop WebSphere Application Server.
2. Check to see if the primary server reboots and all the resources are transferred to the standby server.



# Advanced high availability configuration for WebSphere Remote Server

This chapter provides detailed instructions for implementing an advanced high availability solution that provides individual component failover for WebSphere Remote Server, including instructions for configuring each of the WebSphere Remote Server components.

**Attention:** This high available architecture scenario was developed and documented in a WebSphere Remote Server V5.1.2.1 environment.

This chapter contains the following sections:

- ▶ 6.1, “Setting up Tivoli System Automation” on page 110
- ▶ 6.2, “Configuring high availability for IBM HTTP Server” on page 110
- ▶ 6.3, “Configuring high availability for WebSphere Application Server” on page 112
- ▶ 6.4, “Configuring high availability for DB2” on page 117
- ▶ 6.5, “Configuring high availability for WebSphere MQ” on page 128
- ▶ 6.6, “Configuring high availability for Remote Management Agent” on page 130

## 6.1 Setting up Tivoli System Automation

To set up a cluster with Tivoli System Automation, follow the steps outlined in 5.3, “Setting up Tivoli System Automation” on page 93.

We recommend that you install the default policies provided by Tivoli System Automation. Appendix A, “Additional material” on page 393, of this book contains the rpm with default policies. To install the policies, enter the following command:

```
rpm -i sam.policies-1.2.2.0-05201.i386.rpm
```

The policies are installed in the `/usr/sbin/rsct/sapolicies/` directory.

## 6.2 Configuring high availability for IBM HTTP Server

Tivoli System Automation provides policies to automate and manage the IBM Hypertext Transfer Protocol (HTTP) Server. The latter could run on either the primary or the standby system. To present a single point of contact to client applications, an Internet Protocol (IP) address is configured to float between the two systems. Configure the clients to connect to this shared IP address.

Ideally, the IBM HTTP Server instances must have access to a shared disk location, where static files are stored. However, since we considered the minimal configuration in our setup, we assume that each IBM HTTP Server instance has a local copy of static files.

To configure the IBM HTTP Server for high availability, perform the following steps:

1. Log in to the primary server as root. Change the directory to `/usr/sbin/rsct/sapolicies/ih`s by entering the following command:  

```
cd /usr/sbin/rsct/sapolicies/ih
```

2. Open a file, like the one shown in Example 6-1, to edit the `ihs.def` file. Substitute `node01` and `node02` with the host names of your two nodes.

*Example 6-1 Opening file*

---

```
PersistentResourceAttributes::
Name="ihs-rs"
StartCommand="/usr/sbin/rsct/sapolicies/ihs/ihs start"
StopCommand="/usr/sbin/rsct/sapolicies/ihs/ihsstop"
MonitorCommand="/usr/sbin/rsct/sapolicies/ihs/ihs status"
MonitorCommandPeriod=5
MonitorCommandTimeout=5
NodeNameList={"node01","node02"}
StartCommandTimeout=10
StopCommandTimeout=10
UserName="root"
ResourceType=1
```

---

3. Create the resource definition now with the `mkrsrc` command using the definition file:

```
mkrsrc -f ihs.def IBM.Application
```

4. The Web server's IP address, `ihsIP`, is a separate IP address in the cluster, and does not match any IP address assigned to the network adapters on each cluster node. In contrast, the address for `ihsIP` is created by Tivoli System Automation, and is an additional alias address on an appropriate network adapter on the node where the Web server resides. In this example, `ihsIP` has the following attributes:

- IP 9.42.188.20
- Netmask 255.255.255.0
- The IP address may be created on any node in the cluster.

Use command line parameters to the `mkrsrc` command to create the `ihsIP` resource:

```
mkrsrc IBM.ServiceIP NodeNameList="{ 'node01', 'node02' }" Name="ihsIP"
NetMask=255.255.255.0 IPAddress=9.42.188.20
```

5. Define the equivalency of the network adapters to be used for the floating IP. The following command creates an equivalency named `netequ`, which contains a network adapter from each node of the cluster:

```
mkequ netequ IBM.NetworkInterface:eth0:node01,eth0:node02
```

6. Create a resource group by entering the following command:

```
mkrgrp ihsgrp
```

7. Add both `ih-s-rs` and `ih-sIP` resources to the resource group `ih-sgrp`. Adding the resources to the resource group turns them into managed resources. Use the **`addrgmbr`** command for this:

```
addrgmbr -g ih-sgrp IBM.Application:ih-s-rs
addrgmbr -g ih-sgrp IBM.ServiceIP:ih-sIP
```

8. Two conditions relate resources `ih-s-rs` and `ih-sIP` to one another. First, ensure that both resources are started and available on the same node in the cluster. Second, it is no use starting the Web server `ih-s-rs` on a node on which the IP address `ih-sIP` has not been established yet. `ih-sIP` must be available before `ih-s-rs` is started. Tivoli System Automation provides a relationship type called *DependsOn* that gathers both the required conditions. A managed relationship is defined with the **`mkrel`** command.

Use the following commands to create managed relationships:

```
mkrel -p DependsOn -S IBM.Application:ih-s-rs -G IBM.ServiceIP:ih-sIP
ih-s_dependson_ip
```

```
mkrel -p DependsOn -S IBM.ServiceIP:ih-sIP -G IBM.Equivalency:netequ
ih-sIP_dependson_netequ
```

9. To bring resources online, enter the following command:

```
chrg -o online ih-sgrp
```

10. Test the failover of the IBM HTTP Server server.

## 6.3 Configuring high availability for WebSphere Application Server

The default policies for WebSphere Application Server that are provided with Tivoli System Automation are designed for WebSphere Application Server 6.

For more information about WebSphere Application Server HA, refer to *IBM WebSphere Application Server V5.1 System Management and Configuration WebSphere Handbook Series*, SG24-6195, or *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688.

The default policies that are provided cover four different scenarios for high availability solution for WebSphere Application Server 6. The fifth scenario is described here.

WebSphere Application Server takes a while to start. In our setup, to avoid the delay needed in startup during failover, we ran application servers on each node. Only the application server on the primary node takes requests. The floating IP is

used with the IBM HTTP Server to direct all the requests to the standby node in case of primary node failure.

To implement this solution, the application servers running on each node should be part of the WebSphere Application Server cluster and should use session replication to provide a seamless failover. Deployment manager, a separate application server for WebSphere that is dedicated to managing applications and servers across multiple machines, is used to manage the cluster. Each machine with a WebSphere Application Server that is managed by a deployment manager, is called a *node*. The collection of nodes managed by a deployment manager is called a *cell*, with only one deployment manager per cell.

Thus, in a two-server configuration, a deployment manager must run on either the primary or the standby server. A useful strategy is to put the deployment manager on the standby server. Under normal conditions, when both machines are operating normally, the deployment manager is available on the standby server, reducing the process load on the primary server. In a failover situation where the primary server has failed or is offline, the deployment manager is still available on the standby server.

Perform the following steps to configure high availability support for WebSphere Application Server on the two nodes:

1. Configure the WebSphere Application Server cluster by performing the following tasks:
  - a. Start the Deployment Manager on the standby server by entering the following command:

```
cd /opt/WebSphere/DeploymentManager/bin
./startManager.sh
```
  - b. Ensure that the administration console can be accessed at `http://hostname:port/admin`, where the default port is 9090.
  - c. Federate both the nodes into the deployment manager cell by performing the following tasks:
    - i. Log in to the standby server as root and run the `addNode.sh` script as follows:

```
cd /opt/IBM/WebSphere/AppServer/bin
./addNode.sh your_standby_node_name 8879 -trace -includeapps
```

This federates the node on the standby server to the deployment manager cell and starts a node agent process for the new node. The first parameter is the host name of the system, where the deployment manager is located, in this case, the standby server. The second parameter is the default port for the deployment manager, which is 8879.

The `-trace` option generates additional information to the log file. The `-includeapps` option copies any applications already installed on the node to the cell so that they may be deployed to other nodes.

- ii. To federate the WebSphere Application Server node on the primary server into the deployment manager cell, log in to the primary server as root and run the `addNode.sh` script as follows:

```
cd /opt/IBM/WebSphere/AppServer/bin
./addNode.sh your_standby_node_name 8879 -trace
```

For the primary server, do not include the `-includeapps` option. The assumption is that the same default applications are present on the standby server as on the primary server. Thus, there is no need to copy these applications to the deployment manager a second time.

- d. Create the server cluster.

A cluster allows multiple application servers to be managed together. All the application servers in a cluster share the same configuration and are called *cluster members*. An existing application server can be used as a template for the other cluster members. After the first server is added to the cluster, creating additional cluster members sets up new and identically configured servers on other nodes.

The existing default application server, that is, `server1`, on the standby server is used as a template for the cluster. Adding a cluster member for the primary server creates a new application server on the primary server that is identical to the default application server on the standby server. The default application server, that is, `server1`, is not required on the primary server since this application server will be recreated.

To create a server cluster, follow these steps:

- i. Open the WebSphere administration console.
- ii. Click **Servers** → **Clusters**.
- iii. Click **New** to access the Create New Cluster page.
- iv. Fill the required information to create the cluster. For more details, refer to the WebSphere Application Server V5.1 information center on the Web at:  
<http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1/index.jsp>
- v. After the cluster is created, add a new server to the cluster for the primary node by selecting **Clusters** → **your\_cluster\_name** → **Cluster Members** → **New**. For details about how to add cluster members, refer to the WebSphere Application Server information center on the Web at:

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

- vi. Depending on the application, configure the session replication. Select **Servers** → **your\_application\_server** → **Web Container** → **Session Management** → **Distributed Environment Settings**. In our setup, we used memory-to-memory replication.
  - vii. Save your changes and regenerate the Web server plugin.
2. Configure the Tivoli System Automation resources for WebSphere Application Server by performing the following tasks:
- a. Log in to the primary server as root. Change the directory to /usr/sbin/rsct/sapolicies/was by entering the command:
 

```
cd /usr/sbin/rsct/sapolicies/was/
```
  - b. Copy sa-was.conf.sample to sa-was.conf by entering the following command:
 

```
cp sa-was.conf.sample sa-was.conf
```
  - c. Edit the sa-was.conf file and modify the various parameters as detailed.
    - i. For parameters related to the WebSphere Application Server cluster, change the values as shown in Example 6-2.

*Example 6-2 Parameters relating to WebSphere Application Server cluster*

---

```
cluster_name=[clusterName]
webc_tcp=9080
```

---

- ii. For parameters related to node1, change the values as shown in Example 6-3.

*Example 6-3 Node 1 parameter in sa-was.conf*

---

```
node_name1=[primaryServer]
USER_INSTALL_ROOT1=/opt/IBM/WebSphere/AppServer/
as_name1=server1
as_tcp=8879
na_tcp=8878
#m_criteria1=WSAF_SIB_BUS=myCluster,
WSAF_SIB_MESSAGING_ENGINE=myCluster.000-myCluster,IBM_hc=myCluster,
type=WSAF_SIB
#t_criteria1="GN_PS=flyCell101\tuxNode01\tuxserver,IBM_hc=myCluster,
type=WAS_TRANSACTIONS"
```

---

- iii. For parameters related to node2, change the values as shown in Example 6-4.

*Example 6-4 Node 2 parameter in sa-was.conf*

---

```
node_name2=[standbyServer]
USER_INSTALL_ROOT2=/opt/IBM/WebSphere/AppServer/
as_name2=server1
as_tcp2=8879
na_tcp2=8878
#m_criteria2=WSAF_SIB_BUS=myCluster,WSAF_SIB_MESSAGING_ENGINE=myC
luster.001-myCluster,IBM_hc=myCluster,type=WSAF_SIB
#t_criteria2="GN_PS=flyCell01\tux2Node01\tux2server,IBM_hc=myClus
ter,type=WAS_TRANSACTIONS"
For parameters related to floating ip, change the following
(where ip is the new ip to be used by TSA as a floating ip):
ip_1="9.42.188.31,255.255.255.0"
# --List of network interfaces ServiceIP ip_x depends on.
# Entries are lists of the form
<network-interface-name>:<node-name>,...
nieq_1="eth0:ires3,eth0:ires4"
```

---

- iv. Save the changes.

- d. To create Tivoli System Automation resources, run the configuration script for scenario 5 with the -p option:

```
./cfgwas_scen5 -p
```

The control script for WebSphere Application Server and this script are provided in “Using the Web material” on page 394, in addition to the existing WebSphere Application Server scripts.

While executing this script, you may encounter an error stating that the ServiceIP cannot be created. To work around this problem, open the `cfgwas_scen5` script and run the last three commands manually.

- 3. At this point, IBM WebSphere is configured for control by Tivoli System Automation. To verify if everything is functioning properly, perform the following tasks:
  - a. Bring the resource group online by entering this command:  

```
chrg -o Online SA-was-rg
```
  - b. Check the status by running the **getstatus** command from `/usr/sbin/rsct/sapolicies/bin`:  

```
./getstatus
```
  - c. Test the failover of WebSphere Application Server.



## 6.4 Configuring high availability for DB2

The IBM DB2 8.2 product set includes default policies for Tivoli System Automation. These policies include configuration scripts that create the appropriate Tivoli System Automation resources, groups, and relationships.

IBM DB2 provides automation at two levels. The first level of automation manages a DB2 instance and the processes associated with it. The second level of automation manages a database that is configured for HADR. In a highly available configuration, the WebSphere Remote Server uses both levels of automation for DB2. High availability disaster recovery (HADR) is an IBM DB2 8.2 feature that replicates databases between two systems. You must install a separate license before you configure this feature.

If a DB2 instance fails, Tivoli System Automation detects the failure and attempts to restart the process. Each DB2 instance in the domain is modeled as a Tivoli System Automation resource, with the resource name derived from the instance name.

When WebSphere Remote Server is installed, a database instance is created. The default value is db2inst1, although this can be changed prior to the installation. Every DB2 instance must have a tcp/ip port to support client connections. Use the same port number on both the primary and the standby servers for the DB2 instance db2inst1.

To configure high availability support in DB2, perform the following steps:

1. Determine the service name and port numbers used. You can look up this information in the response file used during WebSphere Remote Server installation.
2. Log in to the primary system as root and edit the `/etc/services` file. Add the following line at the end of the file:  
`your_db2_service_name your_db2_port_number/tcp`
3. Repeat step 2 on the standby system. Log in to the standby system as root, edit the `/etc/services` file, and add the line following line at the end of the file:  
`your_db2_service_name your_db2_port_number/tcp`
4. Update the database manager configuration parameters on the primary system. Log in to the primary system as db2inst1, and execute the following command:  
`db2 update dbm cfg using SVCENAME your_db2_service_name`
5. Log in to the standby system as db2inst1 and repeat the command:  
`db2 update dbm cfg using SVCENAME your_db2_service_name`

6. Fix the regdb2salin script.

DB2 provides a /opt/IBM/db2/V8.1/ha/salinux/regdb2salin script that configures the appropriate Tivoli System Automation resources to manage DB2 instances. The naming convention for Tivoli System Automation resources used by the regdb2salin script relies primarily on the DB2 instance name. As a result, if the same DB2 instance name is present on both the systems, it results in conflicting resource names in the Tivoli System Automation resource namespace.

By default, the regdb2salin script assumes that the DB2 instance names are unique across the systems controlled by Tivoli System Automation. In order to avoid creating new DB2 instances, carry out a minor change in the regdb2salin script. This allows the use of the default DB2 instances created by the WebSphere Remote Server installation.

To edit the regdb2salin script, perform the following tasks:

- a. Log in to the primary system as root. Change the directory to /opt/IBM/db2/V8.1/ha/salinux by entering the following command:  

```
cd /opt/IBM/db2/V8.1/ha/salinux
```
- b. To edit regdb2salin scripts, locate the following line:  

```
CLUSTER_APP_LABEL=${insttype?}_${INSTNAME?}
```
- c. Append an underscore and \$(host name) so that the line appears as follows:  

```
CLUSTER_APP_LABEL=${insttype?}_${INSTNAME?}_${hostname}
```
- d. Save the changes.
- e. Make the same changes to the script on the standby system.

7. Create a workaround for the rsh.

The regdb2salin script also relies on the remote shell rsh to operate correctly. This is necessary for the correct configuration of a partitioned database. However, in a single partition database environment, rsh is not necessary. Since rsh is generally considered a security risk, some customers do not want it on their systems.

Fortunately, there is a simple workaround that allows the regdb2salin script to work without installing and configuring rsh. The simplest workaround is to create a substitute script that evaluates the command that is passed to rsh in the regdb2salin script.

Placing this substitute script at the first location in the PATH used by regdb2salin executes it in place of the rsh executable, which is typically found at /usr/bin/rsh. The regdb2salin script explicitly defines the PATH to be:

```
PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/sbin/rsct/bin
```

Therefore, place the substitute script in the /bin directory by performing the following steps:

- a. Create the substitute script by running the following command:

```
echo "shift; eval \${@}" > /bin/rsh
```

- b. Make the script executable by entering the following command:

```
chmod +x /bin/rsh
```

- c. The shell keeps a cache of the locations of common commands. To ensure that the substitute script is found instead of the actual rsh executable, refresh the cache by entering the following command:

```
rehash
```

- d. Repeat steps a through c on the standby system.

8. Run the regdb2salin script.

- a. Log in to the primary system, that is, [primaryServer], as root. Change the directory as follows:

```
cd /opt/IBM/db2/V8.1/ha/salinux
```

- b. Run the regdb2salin script:

```
./regdb2salin -a db2inst1 -r
```

The -r flag indicates that the DB2 instance should be restarted in place if the process fails for some reason.

- c. Repeat steps a and b on the standby system.

Tivoli System Automation should be configured to automate the start, stop, and monitoring processes of the default DB2 instances on both the primary and the standby systems.

9. Configure the DB2 database for HADR.

You can create databases either manually, using the DB2 commands, or automatically, as part of an application. For each critical database that should be protected, HADR must be configured. For purposes of demonstration, we describe the processes involved in creating a database manually and configuring it for HADR.

- a. To create a sample database to configure for HADR, perform the following steps:

- i. Log in to the primary system using the DB2 instance name, that is, db2inst1. Enter the following command to create the database:

```
db2 create database your_database_name
```

- ii. Identify the databases that are to be dynamically replicated between the systems using HADR. Static databases do not need HADR.

b. Configure a database for HADR manually by performing the following tasks:

i. On the primary system, log in as db2inst1 and turn on LOGRETAIN for the database with the following command:

```
db2 update db cfg for your_database_name using LOGRETAIN ON
```

ii. Back up the database with the following command:

```
db2 backup database your_database_name
```

iii. Copy the backup image to the standby system, that is, [standbyServer]. Log in to the standby system as db2inst1 and restore the database image with the following command:

```
db2 restore database your_database_name
```

DB2 HADR replicates data between the systems using Transmission Control Protocol/Internet Protocol (TCP/IP). The HADR communication occurs on a TCP port, which is different from the port that DB2 uses for communication with clients. A unique port must be selected for each HADR-enabled database. The same port may be used on both the systems. To reserve the port for HADR communications and to designate a name for the service, edit the /etc/services file on each system.

iv. Log in to the primary system as root and edit the /etc/services file. Add the following line at the end of the file:

```
your_hadr_service_name your_hadr_port_number/tcp
```

v. Repeat step iv on the standby system. Log in to the standby system as root, edit the /etc/services file, and add the following line at the end of the file:

```
your_hadr_service_name your_hadr_port_number/tcp
```

- vi. Update the HADR configuration parameters for the database on the primary system. Log in to the primary system as db2inst1 and execute the following commands shown in Example 6-5.

*Example 6-5 Updating HADR configuration parameters on primary system*

---

```
db2 update db cfg for your_database_name using HADR_LOCAL_HOST
your_primary_server
db2 update db cfg for your_database_name using
HADR_REMOTE_HOST your_standby_server
db2 update db cfg for your_database_name using HADR_LOCAL_SVC
your_hadr_service_name
db2 update db cfg for your_database_name using HADR_REMOTE_SVC
your_hadr_service_name
db2 update db cfg for your_database_name using
HADR_REMOTE_INST db2inst1
```

---

- vii. Update the HADR configuration parameters for the database on the standby system. Log in to the standby system as db2inst1 and execute the following commands shown in Example 6-6.

*Example 6-6 Updating HADR configuration parameters on secondary system*

---

```
db2 update db cfg for your_database_name using HADR_LOCAL_HOST
your_standby_server
db2 update db cfg for your_database_name using
HADR_REMOTE_HOST your_primary_server
db2 update db cfg for your_database_name using HADR_LOCAL_SVC
your_hadr_service_name
db2 update db cfg for your_database_name using HADR_REMOTE_SVC
your_hadr_service_name
db2 update db cfg for your_database_name using
HADR_REMOTE_INST db2inst1
```

---

**Note:** The values for HADR\_LOCAL\_HOST and HADR\_REMOTE\_HOST are different from those used on the primary system, and are in fact, reversed. All the other values are the same.

To make the automatic client reroute functionality available with DB2, update the alternate server information for each database on both the servers. To do this, log in to the primary server as db2inst1 and enter the following command:

```
db2 update alternate server for database your_database_name
using hostname your_standby_server port your_db2_port_numbe
```

viii. Repeat step vii on the standby server using the address of the primary server for the host name parameter. Log in to the standby server as db2inst1 and enter the following command:

```
db2 update alternate server for database your_database_name
using hostname your_primary_server port your_db2_port_numbe
```

ix. Create network equivalencies using the following commands:

For the primary server, enter the following command:

```
mkequ netequiv_1 IBM.NetworkInterface:eth0:linux-ires3
```

For the secondary server, enter the following command:

```
mkequ netequiv_1 IBM.NetworkInterface:eth0:ires4
```

x. Create a dependency relationship between the two servers by entering the following command:

```
mkrel -p DependsOn -S
IBM.Application:db2_db2inst1_linux-ires3_0-rs -G
IBM.Equivalency:netequiv_1
db2_db2inst1_linux-ires3_on_netequiv_1
```

xi. Run this command on the standby server too by changing the node name.

The databases are now configured for HADR.

c. To start the HADR service, log in to the standby server as db2inst1 and execute the following command:

```
db2 start hadr on db your_database_name as standby
```

d. Log in to the primary server as db2inst1 and execute the following command:

```
db2 start hadr on db your_database_name as primary
```

The HADR service is now operational for the database.

e. To verify this, enter the following command as db2inst1:

```
db2 get snapshot for db on your_database_name
```

If everything is functioning correctly, you see a section in the output, similar to the one displayed in Example 6-7.

*Example 6-7 db2inst1 command output*

---

HADR Status

```
Role                = Primary
State               = Peer
Synchronization mode = Nearsync
Connection status   = Connected, 10/19/2005 09:27:25
Heartbeats missed   = 0
Local host          = [primaryServer]
Local service       = [hadrTcpServiceName]
Remote host         = [standbyServer]
Remote service      = [hadrTcpServiceName]
Remote instance     = db2inst1
timeout(seconds)    = 120
Primary log position(file, page, LSN) = S0000000.LOG, 0,
000000000007D0000
Standby log position(file, page, LSN) = S0000000.LOG, 0,
000000000007D0000
Log gap running average(bytes) = 0
```

---

10. Automate the HADR database for Tivoli System Automation.

To create the Tivoli System Automation resources to manage the HADR service for the database, run the `reghadr` script found in `/opt/IBM/db2/V8.1/ha/salinux`. Before running this script, carry out the following changes:

a. Log in to the primary server as root and change directory as follows:

```
cd /opt/IBM/db2/V8.1/ha/salinux
```

- b. To edit the reghadrsalin script, locate the section that looks like the one displayed in Example 6-8.

*Example 6-8 Locating the reghadrsalin script*

---

```
# Ensure instance resource groups are online
  rg_inst1_online=$(lsrg -g db2_{$DB2HADRINSTANCENAME1}_0-rg | grep OpState | grep
-c " = Online")
  if [[ $rg_inst1_online != 1 ]]; then
    echo "Resource group db2_{$DB2HADRINSTANCENAME1}_0-rg is not created or not
online"
    exit 1
  fi
  rg_inst2_online=$(lsrg -g db2_{$DB2HADRINSTANCENAME2}_0-rg | grep OpState | grep
-c " = Online")
  if [[ $rg_inst2_online != 1 ]]; then
    echo "Resource group db2_{$DB2HADRINSTANCENAME2}_0-rg is not created or not
online"
    exit 1
  fi
```

---

- c. Edit the reghadrsalin script by changing the parameters indicated in bold in Example 6-9.

*Example 6-9 Edit reghadrsalin script*

---

```
# Ensure instance resource groups are online
  rg_inst1_online=$(lsrg -g ${RGNAME1} | grep OpState | grep -c " =
Online")
  if [[ $rg_inst1_online != 1 ]]; then
    echo "Resource group ${RGNAME1} is not created or not online"
    exit 1
  fi

  rg_inst2_online=$(lsrg -g ${RGNAME2} | grep OpState | grep -c " =
Online")
  if [[ $rg_inst2_online != 1 ]]; then
    echo "Resource group ${RGNAME2} is not created or not online"
    exit 1
  fi
```

---

- d. Locate the section where the isClusterOnline subroutine is called as displayed in Example 6-10.

*Example 6-10 Subroutine isClusterOnline*

---

```
# Verify cluster is online
```



```
#  
isClusterOnline
```

---

- e. Delete or comment the following line:

```
#isClusterOnline
```

- f. Locate the section where the variables RENAME1, RENAME2, and RENAME2 are defined. Before you define these variables, define the variables NODE1 and NODE2, using the host names of the primary and standby servers, respectively. Include the NODE1 and NODE2 variables in the definitions of RENAME1, RENAME2, and so on. Finally, add isClusterOnline immediately after this section.

When complete, the variables appear as displayed in Example 6-11, with the changes indicated in bold.

*Example 6-11 Defining variables*

---

```
NODE1=[primaryServer]  
NODE2=[standbyServer]  
RENAME1=db2_${INSTNAME1?}_${NODE1}_0-rg  
RNAME1=db2_${INSTNAME1?}_${NODE1}_0-rs  
RENAME2=db2_${INSTNAME2?}_${NODE2}_0-rg  
RNAME2=db2_${INSTNAME2?}_${NODE2}_0-rs  
isClusterOnline
```

---

- g. Save the file.
- h. Make similar changes to the three additional scripts hadr\_start.ksh, hadr\_stop.ksh, and hadr\_monitor.ksh. For example, to edit the hadr\_start.ksh script, find the section where RENAME1, RENAME2, and so on are defined and change the parameters displayed in bold in Example 6-12.

*Example 6-12 Making changes to additional scripts*

---

```
NODE1=[primaryServer]  
NODE2=[standbyServer]  
RENAME1=db2_${DB2HADRINSTANCE1?}_${NODE1}_0-rg  
RNAME1=db2_${DB2HADRINSTANCE1?}_${NODE1}_0-rs  
RENAME2=db2_${DB2HADRINSTANCE2?}_${NODE1}_0-rg  
RNAME2=db2_${DB2HADRINSTANCE2?}_${NODE1}_0-rs
```

---

- i. Save the changes.
- j. Make the same changes to hadr\_stop.ksh and hadr\_monitor.ksh.

- k. The same changes must be made on the standby server. Copy the modified files to the same location on the standby server or log in to the standby server as root and make the same changes.
- l. To run the reghadrsalin script, log in to the primary server as root. The primary server must be active in the HADR Primary role for the database. Furthermore, the HADR pair must be in peer state. If these conditions exist, run the reghadrsalin script:  

```
./reghadrsalin -a db2inst1 -b db2inst1 -d your_database_name
```

**Note:** This script should be run only once from the primary server.

### 11.Perform failover for the testing.

At this point, configure the DB2 instances on both the primary and standby servers and the HADR service for database for control by Tivoli System Automation.

- a. Verify if the Tivoli System Automation resources are created, are operating correctly, and if the databases are in peer state. We recommend that you ensure that the HADR resource group is currently hosted on primary node. In our setup, this was linux-ires3.

To do this, run the `getstatus` script from the `/opt/IBM/db2/V8.1/ha/salinux` directory. The output should look as displayed in Example 6-13.

Example 6-13 *getstatus* command output

```
./getstatus

-- Resource Groups and Resources --

              Group Name              Resources
              -----              -
db2_db2inst1_ires4_0-rg db2_db2inst1_ires4_0-rs
              -                      -
db2_db2inst1_linux-ires3_0-rg db2_db2inst1_linux-ires3_0-rs
              -                      -
db2hadr_sample-rg        db2hadr_sample-rs
              -                      -

-- Resources --

              Resource Name              Node Name
              -----              -
State
-----
```

	db2_db2inst1_ires4_0-rs	ires4
Online	-	-
-	-	-
	db2_db2inst1_linux-ires3_0-rs	linux-ires3
Online	-	-
-	-	-
	db2hadr_sample-rs	ires4
Offline	-	-
	db2hadr_sample-rs	linux-ires3
Online	-	-
-	-	-

---

- b. Reboot the primary machine.
- c. Execute following commands on the primary server after it comes online again:

```
chrg -o Online db2_db2inst1_<primary node>_0-rg
su - db2inst1
db2 start hadr on db <database name> as standby
```

- d. Change the user back to root and run the **./getstatus** command.
- e. The DB2 instance should failover to the standby server. In our setup, it was ires4. You should see the following text:

```
db2hadr_sample-rs          ires4          Online
db2hadr_sample-rs          linux-ires3      Offline
```

- f. Verify if the primary node is now in standby mode and is offline. The previous standby node now acts as primary, and is in the online mode.

Here are some useful commands:

- You can start the db2 instances in both machines using Tivoli System Automation with the following command:
 

```
chrg -o online db2_db2inst1_linux-ires3_0-rg db2_db2inst1_ires4_0-rg
```

 This starts DB2 resources on both machines. Run this command from either one of the machines, and then start the HADR resource using this command:
 

```
chrg -o online db2hadr_sample-rg
```
- You can switch the Tivoli System Automation mode to manual with the following command:
 

```
samctrl -M T
```

- ▶ To turn on Tivoli System Automation again, use this command:  
`samctrl -M F`
- ▶ You can reset a resource by using this command:  
`resetrsrc -s "Name = 'db2hadr_sample-rs'" IBM.Application`
- ▶ You can activate the database by using this command:  
`db2 activate database sample`

## 6.5 Configuring high availability for WebSphere MQ

Tivoli System Automation provides policies to automate and manage WebSphere MQ. The WebSphere MQ Server and listener will be running on either the primary or the standby system. To present a single point of contact to client applications, an IP address is configured to float between the two systems. Clients must be configured to connect to this shared IP address.

Ideally, the WebSphere MQ should have access to a shared disk location to provide storage of messages. However, since we considered the minimal configuration in our setup, we will assume that the number of messages in the queue at any given moment is minimal and the loss of these messages negligible. Since this varies from application to application, consider including shared storage in your setup. Perform the following steps:

1. Log in to the primary server as root. Change the directory to `/usr/sbin/rsct/sapolicies/mqm/` by enter the following command:  
`cd /usr/sbin/rsct/sapolicies/mqm/`
2. Copy `mqm/sa-mq.conf.sample` to `sa-mq.conf` using the following command:  
`cp mqm/sa-mq.conf.sample sa-mq.conf`
3. Edit `sa-mq.conf` and modify the parameters `MQ_OWNER`, `MQ_MGR`, `nodes`, `ip_1`, and `nieq_1`, `data_var` as follows:
  - a. For `MQ_OWNER`, specify the Linux user created as part of the WebSphere MQ installation as follows:  
`MQ_OWNER=mqm`
  - b. For `MQ_MGR`, specify the name of the queue manager configured for WebSphere MQ as follows:  
`MQ_MGR=[mqQueueMgr]`
  - c. For `nodes`, specify the primary and standby servers. List the primary server first and place a space between the values as follows:  
`nodes="[primaryServer] [standbyServer]"`

- d. For ip\_1, specify the shared IP address and the netmask, separated by a comma as follows:

```
ip_1="[mqIPAddress],255.255.255.0"
```

- e. For nieq\_1, specify the network interface for each server, listing the primary server first as follows:

```
nieq_1="eth0:[primaryServer],eth0:[standbyServer]"
```

- f. For data\_var, place a hash mark (#) at the beginning of the line to disable the shared data feature as follows:

```
#data_var="/mqdata"
```

4. Save the changes to the file.

You may encounter the following error when trying to start a queue manager or MQ listener:

AMQ6090: WebSphere MQ was unable to display an error message 20006220.

If this happens, work around the problem by running the following command:

```
export LD_ASSUME_KERNEL=2.4.19
```

If you encounter this issue, modify control scripts for MQ on both the primary and standby servers. The files you should modify are mqctrl-mgr and mqctrl-mq\_lsn. To do this, perform the following steps:

- a. Locate the following line:

```
export  
PATH=$PATH:/bin:/usr/bin:/sbin:/usr/sbin:/usr/sbin/rsct/bin:
```

- b. Add the following line after it:

```
export LD_ASSUME_KERNEL=2.4.19
```

- c. Save your changes.

5. To create Tivoli System Automation resources, run the **cfgmq** command with the -p option as follows:

```
./cfgmq -p
```

6. At this point, IBM WebSphere MQ is configured for control by Tivoli System Automation. To verify if everything is working properly, complete these steps:

- a. Bring the resource group online by entering the following command:

```
chrg -o Online SA-mq-rg
```

- b. Verify that the WebSphere MQ is operating properly.

## 6.6 Configuring high availability for Remote Management Agent

The Remote Management Agent (RMA) is available as a component of the WebSphere Remote Server. Tivoli System Automation does not provide a default policy to manage this product.

To create the scripts required to automate RMA for high availability, perform the following tasks.

1. Modify the Remote Management Agent control script.

Tivoli System Automation starts, stops, and monitors application resources through a control script. This script should exist on both servers, in the same location. The Remote Management Agent script requires only minor changes to work as a control script for Tivoli System Automation.

- a. Log in to the primary server as root and change the directory to `/usr/sbin/rsct/sapolicies/`. Then create a subdirectory named `rma`.

```
cd /usr/sbin/rsct/sapolicies/  
mkdir rma
```

- b. Copy the `rmsvc-ma` script file, that is, `rmsvc-ma`, to the `rma` directory and rename it accordingly, as follows:

```
cp /etc/init.d/rmsvc-ma ./rma/rmactrl-server
```

- c. Edit the copy of the control script, that is, `rmactrl-server`. At the bottom of the script, find the *case* statement. Make the following additions to the *status* stanza, that are shown in bold in Example 6-14.

*Example 6-14 Additions to the status stanza*

---

```
status)  
    # For IBM Tivoli System Automation, a return value of 1 means  
    Online  
    # and a return value of 2 means Offline  
    RETVAL=2  
    if [ -f $PIDFILE ] ; then  
        pid=`cat $PIDFILE`  
        if kill -0 $pid 1>/dev/null 2>&1 ; then  
            echo Running  
            RETVAL=1  
        else  
            echo Not Running  
        fi  
    else  
        echo Not Running
```

```
fi
;;
```

---

- d. The control script for Remote Management Agent must exist on the standby server as well, and in the same location. Log in to the standby server, create the rma subdirectory, copy the rmsvc-ma script, and make the changes as outlined in step c.

2. Disable Remote Management Agent as a system service.

By default, Remote Management Agent is installed as a system init script in /etc/init.d. This means that starting and stopping the Remote Management Agent is under control of Linux during boot and shutdown.

Unfortunately, this conflicts with the ability of Tivoli System Automation to manage the Remote Management Agent on the two systems. Therefore, the Remote Management Agent should be disabled as a system init script. Perform the following tasks to do this:

- a. Log in to the primary system as root and stop the Remote Management Agent if it is running:

```
/etc/init.d/rmsvc-ma stop
```

- b. Disable it as a service by using the following command:

```
insserv -r /etc/init.d/rmsvc-ma
```

- c. Log in to the standby system as root and repeat steps a and b.

3. Build the configuration environment.

To create the Tivoli System Automation configuration for Remote Management Agent, start with the configuration for IBM HTTP Server and customize the files as explained in the following steps:

- a. Log in to the primary server as root and change the directory to /usr/sbin/rsct/sapolicies/ with this command:

```
cd /usr/sbin/rsct/sapolicies/
```

- b. Copy the configuration script, that is, cfgihs, and the configuration file, that is, sa-ihs.conf, from /usr/sbin/rsct/sapolicies/ihs+data+ip/ to /usr/sbin/rsct/sapolicies/rma/. Rename cfgihs to cfgrma and sa-ihs.conf to sa-rma.conf as follows:

```
cp ./ihs+data+ip/cfgihs ./rma/cfgrma
cp ./ihs+data+ip/sa-ihs.conf ./rma/sa-rma.conf
```

- c. Edit the configuration script, that is, cfgrma, and change the following line:

```
PROD="rma"
```

- d. Save the changes.

- e. Edit the configuration file, that is, `sa-rma.conf`, and make the following changes:
  - i. Change the `script_dir` variable to match the current working directory:  
`script_dir="/usr/sbin/rsct/sapolicies/rma"`
  - ii. Change the prefix variable to SA-rma- as follows:  
`prefix="SA-rma-"`
  - iii. Change the nodes variable:  
`nodes="[primaryServer] [standbyServer]"`
  - iv. Modify the `ip_1` parameter to match the IP address that clients will use to connect to the Remote Management Agent:  
`ip_1="[rmaIPAddress],255.255.255.0"`
  - v. Modify the `nieq_1` parameter to match the network interfaces on each server, listing the primary server first:  
`nieq_1="eth0:[primaryServer],eth0:[standbyServer]"`
  - vi. The `data_work` variable is commented out with a hash mark as follows:  
`#data_work="/MOUNTPPOINT"`
- f. Save the changes.
4. Create the Tivoli System Automation resources by running the configuration script, that is, `cfgrma`, with `-p` option:  
`./cfgrma -p`
5. At this point, Remote Management Agent is configured for control by Tivoli System Automation. To verify that everything is working properly, complete these steps:
  - a. Bring the resource group online with:  
`chrg -o Online SA-rma-rg`
  - b. Verify that the Remote Management Agent is operating properly on the primary server, and that it is not running on the standby server.

The high availability setup for WebSphere Remote Server is now complete. You can now test the failover of the whole stack. To do this, reboot the primary server and see if all the resources failover to the standby server.





## Part 3

# Capacity planning

This part describes the setup, testing, and results from a test lab configuration that is used to gather sample performance numbers in a basic high availability configuration.



# WebSphere Remote Server capacity planning and testing

This chapter provides details about the WebSphere Remote Server capacity planning test environment, including descriptions of the hardware and software used, test procedures, and test results. The recommendations for improving system performance are covered in Chapter 8, “Operating system performance tuning” on page 147, through Chapter 11, “Performance tuning of DB2 UDB Workgroup Server” on page 321.

**Attention:** The capacity planning information documented in this chapter was created in a WebSphere Remote Server V5.1.2.1 environment.

This chapter contains information about the following topics:

- ▶ 7.1, “Overview of capacity planning” on page 136
- ▶ 7.2, “Hardware platforms” on page 136
- ▶ 7.3, “Software platform” on page 137
- ▶ 7.4, “Testing tools” on page 138
- ▶ 7.5, “Testing procedures” on page 139
- ▶ 7.6, “Test results” on page 140
- ▶ 7.7, “Test conclusions” on page 143

## 7.1 Overview of capacity planning

*Capacity planning* is the process by which you determine which hardware configuration meets the needs of your store adequately. When a middleware or an application is released into production, you should be confident that the chosen hardware configuration will help it function effectively and meet the performance requirement goals.

Performance requirement goals are often referred to as *service goals* or *service-level agreements*. While these are usually defined by the management, they are most likely to reflect the users' expectations. Two simple, but key performance requirements are:

- ▶ No single request should take more than eight seconds to process.
- ▶ The application must support up to 30,000 requests per day.

Gathering accurate performance requirements is an important part of the capacity planning process. If you underestimate the production-level demands of the middleware or application, they may not perform to the optimum level, or in a worst case scenario, may not even function properly. This translates into unhappy users.

Overestimating the demands placed on an application is also undesirable. Most organizations do not want to purchase more hardware than is absolutely necessary.

Every store environment is different, and because of this, the capacity planning tests that were carried out for this book focused on a sample environment, with only the required applications and middleware installed on the in-store server.

In our setup, we used a sample point of sale (POS) application installed on a set of in-store processors having varied hardware configuration, but similar IBM middleware.

## 7.2 Hardware platforms

We used three hardware platforms or configurations for the in-store processor and documented performance results for them in a variety of store formats, including small (having five POS registers), medium (having 40 POS registers), and large (having 100 POS registers).

We used ten IBM IntelliStation® Z Pro machines to simulate the POS registers. These machines were connected to the in-store processor through a private network.

We used three different configurations of the in-store processors:

- ▶ Small platform  
This platform is a Single Intel® Xeon® Processor 2.8 GHz in-store server with 512 MB random access memory (RAM) and a Small Computer System Interface (SCSI) half-duplex (HDX).
- ▶ Medium platform  
This platform is a Dual Intel Xeon Processor 2.8G Hz in-store server with 2 GB RAM and a SCSI HDX.
- ▶ Large platform  
This platform is a Dual Intel Xeon Processor 2.8 GHz in-store server with 4 GB RAM and a SCSI HDX.

Figure 7-1 displays the hardware configurations for testing.

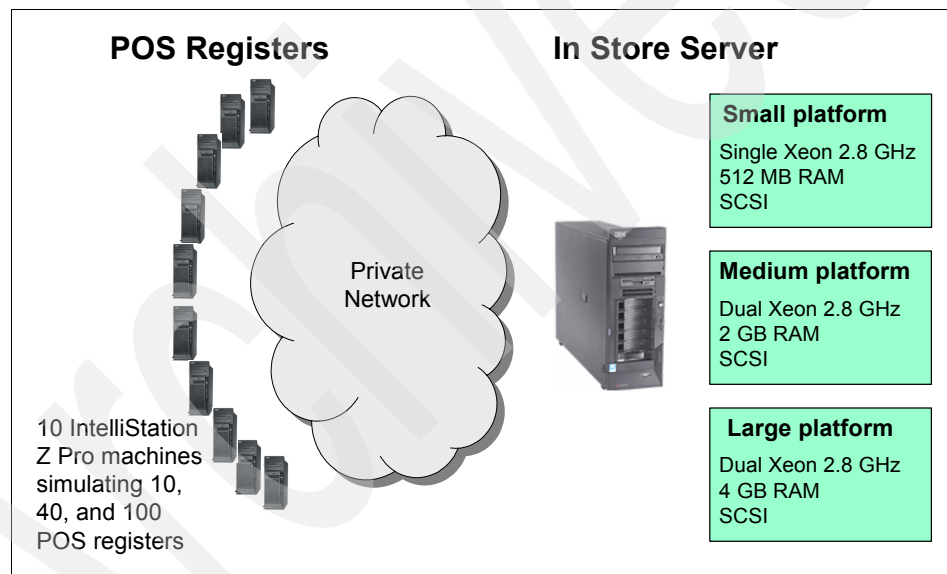


Figure 7-1 Hardware configurations for testing

## 7.3 Software platform

For our testing environment, the following IBM Middleware software was installed on the in-store processor:

- ▶ IBM Retail Environment for SuSE Linux 2.0
- ▶ WebSphere Application Server 5.1.2.1

- ▶ DB2 8.2 FP 2
- ▶ WebSphere MQ 5.3 CSD 9

We had a sample POS application, Oracle 360Commerce Back Office V7.0, installed on the IBM middleware at the in-store processor or in-store server. The ten IntelliStation Z Pro machines that were connected to the server functioned as the POS clients running the 360Commerce POS and load test tool.

For more information about Oracle 360Commerce applications, refer to the Oracle 360Commerce Web site at:

<http://www.oracle.com/360commerce/index.html>

## 7.4 Testing tools

The ten POS client machines had a sample POS Client simulator running on them. The simulator generated events to drive the in-store processor software being tested and to collect timing measurements.

The POS client simulator is capable of variably loading the in-store server by simulating any number of POS registers simultaneously. The load test tool allows users to specify the number of POS clients to simulate, and the total number of transactions to send to the server. The tool then sends the transactions through as quickly as it can and records the length of time taken to complete all the transactions.

## 7.5 Testing procedures

To keep the tests simple, easily reproducible, and comparable, we used a generic order comprising three items and the cash payment method. The total number of transactions varied between 250 and 1250 for each test. The memory and central processing unit (CPU) utilization were monitored using the **vmstat** command. Network usage was monitored using the **ntop** command. Section 7.5.1, “CPU and virtual memory monitoring” on page 139, and 7.5.2, “Network monitoring” on page 140 briefly discuss these tools.

For a more detailed description of these and other additional tools, refer to Chapter 8, “Operating system performance tuning” on page 147.

### 7.5.1 CPU and virtual memory monitoring

The tool used to measure CPU and virtual memory usage during the tests is the **vmstat** command. This command calculates statistic averages for memory, paging, and CPU activity on a sampling period of a length specified by the user. The **vmstat** command is typically run in a separate xterm window in the foreground with the output going to the terminal. It can also be run in the background while redirecting output to a file. The primary argument is the sample interval expressed in seconds.

To run **vmstat** with a sample interval of five seconds and redirect the output to a file, use this command:

```
vmstat 5 > vmstat.log
```

This command creates a file called **vmstat.log**. The important fields from the **vmstat** output are:

- ▶ Memory
  - swpd: The amount of virtual memory used
  - free: The amount of idle memory
  - buff: The amount of memory used as buffer
  - cached: The amount of memory used as cache
- ▶ CPU
  - us: Time spent running non-kernel code
  - sy: Time spent running kernel code
  - id: Time spent idle

## 7.5.2 Network monitoring

In our test environment, the following network monitoring tools were used to gather traffic and usage data:

- ▶ **netstat**

The **netstat** command provides numerous options and displays information about the network interfaces on the system. Relevant output includes network connections, routing tables, and interface statistics.

- ▶ **ntop**

The **ntop** command displays additional network information. To view this information, use a browser, such as Mozilla, to access the following URL:

`http://localhost:3000`

The **ntop** command shows a list of hosts that are currently using the network. It also provides information about the traffic generated and received by each host.

In our simplified test environment, network usage was not a factor. However, with additional applications installed and running in a real store environment, network monitoring may be useful in debugging system performance problems.

## 7.6 Test results

Each of the three platforms had separate capacity planning test results as explained in Table 7-1 through Table 7-4 on page 143 in the following sections. Each table contains the following headings:

- ▶ Transactions per minute: The average number of complete transactions, including logging into the POS, searching for three items, proceeding to checkout, and paying cash, per minute for the in-store server
- ▶ Number of simulated POS: The total number of simulated POS devices, ranging between five and 100, in the environment
- ▶ Physical machines: The number of real IntelliStation machines used to simulate the POS devices
- ▶ Transactions per simulated POS: The number of transactions that each POS will simulate during the test run
- ▶ Total transactions: The total number of transactions for the entire system
- ▶ CPU usage: The average CPU usage, that is, user and system, combined, for the in-store server running the sample POS application, that is, 360Commerce V7.0



- **Total time:** The average length of time required for each POS client simulator to complete its transactions

The transactions per minute can be better understood with this example. If your in-store server is capable of handling 30 transactions per minute, and you have ten POS devices, each POS can produce three sample transactions per minute. The data presented here can be used as a starting point for your capacity planning.

**Note:** These tests were performed without any performance tuning or configuration changes.

## 7.6.1 Small platform test results

This platform is a Single Intel Xeon Processor 2.8 GHz in-store server with 512 MB RAM. The initial plan was to use the same number of transactions and simulated POS devices as were tested in the medium and large platform tests, that is, 5, 10, 20, 40, and 100 simulated POS devices. However, the system could not handle the stress of more than 20 simulated POS clients.

A separate test was then run for a Single Intel Xeon Processor 2.8 GHz in-store server with 4 GB RAM to determine whether it was the low RAM or the single processor that was responsible for the slow performance. The test with the 4 GB RAM produced results that were nearly identical to the test with the 512 MB RAM. This indicates that the processor is the bottleneck in this environment. Table 7-1 shows the test results for 512 MB RAM.

*Table 7-1 Small platform test results with 512 MB RAM*

Transactions per minute	Number of simulated POS	Physical machines	Transactions per simulated POS	Total transactions	CPU usage	Total time
16.6	5	5	50	250	70 - 85%	15 min
21.7	10	5	50	500	95 - 100%	23 min
23.25	20	10	50	1000	100%	43 min

These test results show that a small store with ten or fewer concurrently running POS devices can handle 21.7 transactions per minute. This translates to roughly two transactions per minute per POS client. This is adequate for a small store. However, note that the CPU is utilized almost 100% at this transaction rate.

The test for 20 simulated POS devices was repeated on a machine with 4 GB RAM. The results are displayed in Table 7-2.

*Table 7-2 Small platform test results with 4 GB RAM*

Transactions per minute	Number of simulated POS	Physical machines	Transactions per simulated POS	Total transactions	CPU usage	Total time
23.8	20	10	50	1000	100%	42 min

## 7.6.2 Medium platform test results

This platform is a Dual Intel Xeon Processor 2.8 GHz in-store server with 2 GB RAM. In some cases, the transactions per minute limit is nearly double that of the single processor, small platform. This indicates that the CPU can be the bottleneck in this type of environment.

*Table 7-3 Medium platform test results*

Transactions per minute	Number of simulated POS	Physical machines	Transactions per simulated POS	Total transactions	CPU usage	Total time
20.8	5	5	50	250	35 - 55%	12 min
32.25	10	5	50	500	55 - 70%	15.5 min
31.7	20	10	50	1000	70 - 85%	31.5 min
39	25	5	50	1250	65 - 80%	32 min
33.3	40	10	25	1000	70 - 85%	30 min
35.7	100	10	10	1000	75 - 85%	28 min

### 7.6.3 Large platform test results

This platform is a Dual Intel Xeon Processor 2.8 GHz in-store server with 4 GB RAM. Additional RAM was added to create the large platform. As with the small platform, adding additional RAM did not seem to have much of an effect on the rate of transactions. These results may, however, be different in a real store environment, depending on the type of additional software or middleware present in the in-store server.

Table 7-4 Large platform test results

Transactions per minute	Number of simulated POS	Physical machines	Transactions per simulated POS	Total transactions	CPU usage	Total time
20.8	5	5	50	250	35 - 55%	12 min
32.25	10	5	50	500	60 - 75%	15.5 min
32.25	20	10	50	1000	70 - 85%	31 min
33.3	40	10	25	1000	70 - 85%	30 min
37	100	10	10	1000	75 - 85%	27 min

## 7.7 Test conclusions

The small platform test results indicate that such a platform would probably be appropriate for a small store with five to ten concurrently running POS devices. It is apparent from these test results that in going from a small platform to a medium platform by adding an additional CPU, the system performance improves to a large extent. The effect of adding an additional 2 GB RAM to create the large platform was, however, minimal.

In most tests, where the CPU usage was between 60% and 85%, approximately 20% of the CPU usage was consumed by the Java processes. The remaining 40% to 65% was used by DB2 processes. This indicates that performance gains can be realized by tuning the DB2 settings as described in Chapter 11, "Performance tuning of DB2 UDB Workgroup Server" on page 321.

These tests were conducted in a simplified store environment. The in-store server was set up with only the required middleware and a sample POS server application. From these tests, it is apparent that increasing the number of CPUs and the quality of CPUs, rather than increasing the system RAM, have a greater impact on improving store transaction processing rates. However, careful planning is essential to determine the effect that any additional required

applications will have on your server. If applications that are heavily dependent on RAM are necessary for your in-store server, additional RAM may be the answer to improving performance.

When attempting to calculate the required hardware, ask yourself the following questions:

- ▶ What is the realistic rate of transactions per POS client?
- ▶ What is the maximum number of transactions expected in a day?
- ▶ What is the maximum number of transactions during the busiest hour of the day?
- ▶ What is an acceptable response time for a transaction during normal store operation?
- ▶ Is there additional software or middleware, which will consume hardware resources, installed on the in-store server?

We ran these tests using default configurations and settings for the middleware and sample POS application. To obtain additional performance improvements, follow the guidelines detailed in Part 4, “Performance tuning” on page 145, for tuning the system.



## Part 4

# Performance tuning

This part provides assistance in tuning the operating system and components of the WebSphere Remote Server V6 running on the in-store processor.

Archived

# Operating system performance tuning

This chapter provides assistance in performance tuning the operating system that is installed on the in-store processor. While the WebSphere Remote Server software stack supports other operating systems, this chapter focuses on SUSE Linux Enterprise Server 9<sup>1</sup>, which is the foundation for IBM Retail Environment for SUSE Linux V2.0.

This chapter contains the following sections:

- ▶ 8.1, “Tuning the operating system” on page 148
- ▶ 8.2, “Tuning tools” on page 183
- ▶ 8.3, “Analyzing performance bottlenecks” on page 207

---

<sup>1</sup> SUSE Linux is a registered trademark of SUSE Linux AG, a Novell, Inc. company, and the information and screen captures were used with permission from Novell, Inc.

## 8.1 Tuning the operating system

This section describes the parameters that improve performance. It also provides a basic understanding about the techniques that are used in Linux, including:

- ▶ Linux memory management
- ▶ Page partitions in Linux
- ▶ File systems and their effect on performance
- ▶ Disabling unnecessary daemons
- ▶ Tuning parameters using `sysctl`

**Restriction:** Recompiling the kernel or adding the kernel modules other than those provided by SUSE, voids standard SUSE service and maintenance liabilities. Contact SUSE Linux for an individual service agreement on such a nonstandard setup.

### 8.1.1 Disabling daemons

Daemons or background services, that are probably not required, run on every server. Disabling these daemons frees the memory, decreases the startup time, and decreases the number of processes the CPU handles. A fringe benefit of this is increased security of the server because fewer daemons translates into fewer exploitable processes. On most servers, the daemons that are started by default can be stopped safely.

By default, SUSE Linux Enterprise Server 9 has started the daemons listed in Table 8-1. Consider disabling these daemons within your environment if possible. For a further explanation of these daemons, refer to the YaST tool window shown in Figure 8-2 on page 151.

Table 8-1 Tunable daemons started on a default installation

Daemon	Description
alsasound	Sound daemon
isdn	ISDN modem support
hwscan	Detects and configures hardware changes
portmap	Dynamic port assignment for Remote Procedure Call (RPC) services (such as Network Information Service (NIS) and Network File System (NFS))
postfix	Mail transport agent
splash	Splash screen setup



Daemon	Description
fbset	Framebuffer setup
splash_late	Starts before shutdown
splash_early	Kills animation after the network starts
xdm	X Display manager

To temporarily stop most daemons, use the stop parameter. For example, to stop the sendmail daemon immediately, enter the following command as root:

```
/etc/init.d/sendmail stop
```

In addition, SUSE Linux Enterprise Server 9 has three ways to work with daemons:

- ▶ A text-based user interface (UI): **/sbin/yast runlevel**
- ▶ A graphical user interface (GUI), YaST2, which can be started either with the **/sbin/yast2 runlevel** command or by selecting **Browse** → **YaST/** → **YaST modules** → **System** → **Runlevel editor**, as shown in Figure 8-1.
- ▶ The **/sbin/chkconfig** command

To prevent a daemon from starting when the machine boots, enter the following command as root:

```
/sbin/chkconfig -s sendmail off
```

The YaST2 GUI displayed in Figure 8-1 also lets you perform this action, that is, prevent a daemon from starting when the machine boots.

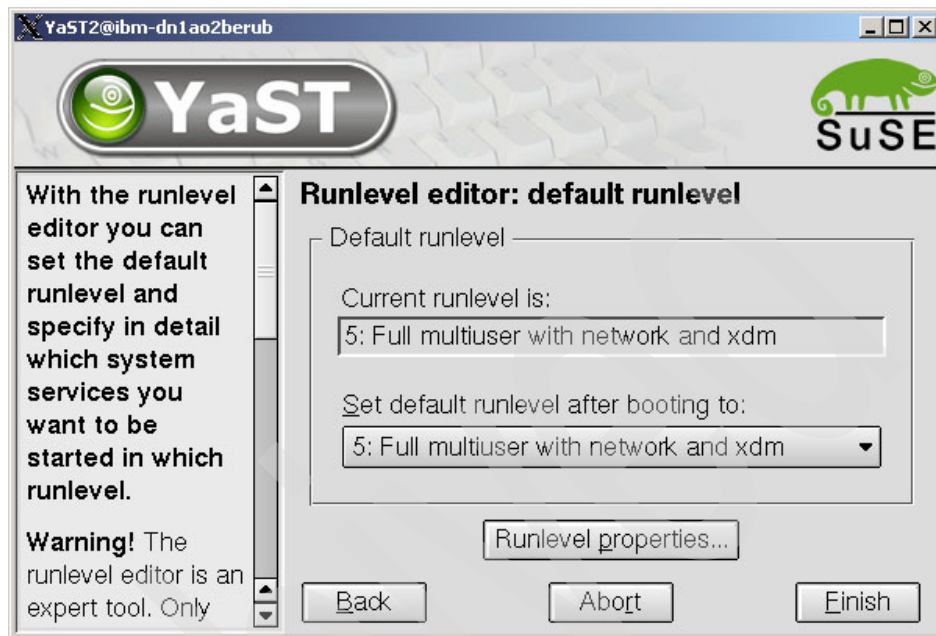


Figure 8-1 YaST Runlevel editor

To change the startup state of a daemon with the help of YaST, perform the following tasks:

1. In the main window displayed in Figure 8-1, click **Runlevel properties**.

2. In the Runlevel editor details window (shown in Figure 8-2), change the start state of a daemon by selecting or deselecting the appropriate runlevel box. Click **Finish** to save the runlevel changes.

**Tip:** Click the **Start/Stop/Refresh** button to stop a service immediately, since changing the runlevels do not take effect until the next startup.

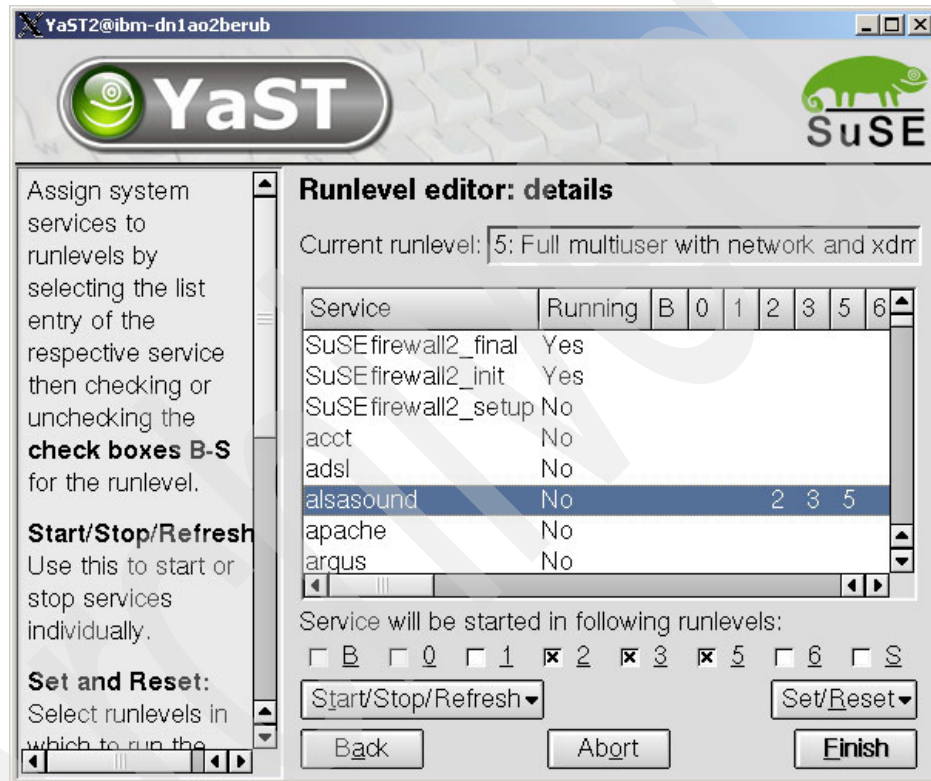


Figure 8-2 YaST Runlevel editor properties

### 8.1.2 Shutting down the GUI

Whenever possible, do not run the GUI on a Linux server. Normally, there is no need for a GUI on a Linux server. You can perform all administration tasks from the command line by redirecting the X display or through a Web browser interface. You can also use several Web-based tools such as webmin, Linuxconf, and Samba Web Administration Tool (SWAT).

If a GUI is needed, start it and stop it as the need arises, rather than continually running it. In most cases, the server should run at run level 3, which does not start the GUI when the machine boots. To restart the X server, enter the **startx** command in the command prompt.

To make the server run at level 3, which is the initial run level of a machine at boot, perform the following tasks:

1. Determine which run level the machine is running with the **runlevel** command.

This prints the previous and current run level. For example, `N 5` means that there was no previous run level (N), and that the current run level is 5.

2. To switch between run levels, use the **init** command. For example, to switch to run level 3, enter the **init** command as follows:

```
init3
```

The following different run levels are used in Linux:

- **0** Halt (Do not set **initdefault** to this, or the server will immediately shut down after finishing the boot process.)
- **1** Single user mode
- **2** Multi-user, without NFS (This is the same as 3, if you do not have networking.)
- **3** Full multi-user mode
- **4** Unused
- **5** X11
- **6** Reboot (Do not set **initdefault** to this, or the server will continuously reboot at startup.)

3. To set the initial run level of a machine at boot, modify the `/etc/inittab` file as shown in Figure 8-3 with the following command:

`id:3:initdefault:`

```
... (lines not displayed)

# The default runlevel is defined here
id:3:initdefault:

# First script to be executed, if not booting in emergency (-b)
mode
si::bootwait:/etc/init.d/boot

# /etc/init.d/rc takes care of runlevel handling
#
# runlevel 0 is System halt (Do not use this for
initdefault!)
# runlevel 1 is Single user mode
# runlevel 2 is Local multiuser without remote network (e.g.
NFS)
# runlevel 3 is Full multiuser with network
# runlevel 4 is Not used
# runlevel 5 is Full multiuser with network and xdm
# runlevel 6 is System reboot (Do not use this for
initdefault!)
#

... (lines not displayed)

# getty-programs for the normal runlevels
# <id>:<runlevels>:<action>:<process>
# The "id" field MUST be the same as the 1
# characters of the device (after "tty").
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
```

To start Linux without starting the GUI, set the runlevel to 3.

To only provide three consoles and thereby save memory, comment out the mingetty entries for 4, 5, and 6.

Figure 8-3 `/etc/inittab`, modified; only a part of the file displayed

With SUSE Linux Enterprise Server 9, you can accomplish the same steps by running the `YaST runlevel` command and changing the default run level as shown in Figure 8-1 on page 150.

By default, six consoles are saved. F1 to F6 are separate consoles. To regain some memory, limit the number of consoles to three from the original six. To do this, comment out each `mingetty ttyx` line you want to disable. In Figure 8-3, the consoles are limited to three.

**Tip:** Even if you have disabled the GUI locally on the server, you can connect remotely and use the GUI. To do this, use the `-X` parameter on the `ssh` command.

### 8.1.3 Compiling the kernel

To improve the performance and reduce the supportability of the server, it is not essential for you to compile the kernel. However, we recommend that you configure your Linux server to have the latest patches, kernels, and drivers provided by SUSE Linux and the hardware vendors. Features that fix bugs and improve the performance of the Linux machine are available regularly.

**Restriction:** Recompiling the kernel or adding kernel modules other than those provided by SUSE voids standard SUSE service and maintenance liabilities. Contact SUSE Linux to request an individual service agreement on such a nonstandard setup.

Before you begin, determine the hardware that is installed in the server. In doing so, you must know the following most important details:

- ▶ CPU type
- ▶ Amount of memory installed
- ▶ Small Computer System Interface (SCSI) adapter
- ▶ Redundant Array of Independent Disks (RAID) controller
- ▶ Fibre Channel adapter
- ▶ Network adapter
- ▶ Video adapter

The more information you have about the hardware being used, the easier it is to configure and compile a new kernel.

We recommend that you create an emergency boot disk. Before you create the diskette, determine the version of the kernel that is currently installed on the

server by using the **uname -a** command. Example 8-1 displays the output of this command.

*Example 8-1 Output of uname*

---

```
Linux ibm-dn1ao2berub 2.4.21-215-default #1 Tue Apr 27 16:17:49 UTC
2004 i686 unknown
```

---

In SUSE Linux Enterprise Server 9, the easiest way to create a bootdisk is through YaST. From the command line, follow the prompts after entering the following command:

```
yast bootfloppy
```

**Note:** To use the YaST bootdisk function, you need the SUSE Linux Enterprise Server 9 installation CD-ROMs.

We recommend that you test the boot disk before you start to compile the kernel.

A complete discussion of how to compile the kernel is covered in Chapter 9 of *SUSE Linux Enterprise Server 9, Administration and Installation Guide*, which you can find on the Web at:

<http://www.novell.com/documentation/sles9/index.html>

## 8.1.4 Changing kernel parameters

The Linux kernel is the core of the operating system and is common to all Linux distributions. To make changes to the kernel, modify the parameters that control the operating system. Make the changes on the command line using the **sysctl** command.

**Tip:** By default, the kernel includes the necessary module to enable you to make changes using the **sysctl** command without rebooting. However, if you remove this support, for example, during the operating system installation, reboot Linux before the change takes effect.

In addition, SUSE Linux offers a graphical method of modifying the `sysctl` command parameters as displayed in Figure 8-4.

To launch the `powertweak` tool, enter the following command:

```
/sbin/yast powertweak
```

To launch a text-based menu version, enter the following command:

```
/sbin/yast2 powertweak
```

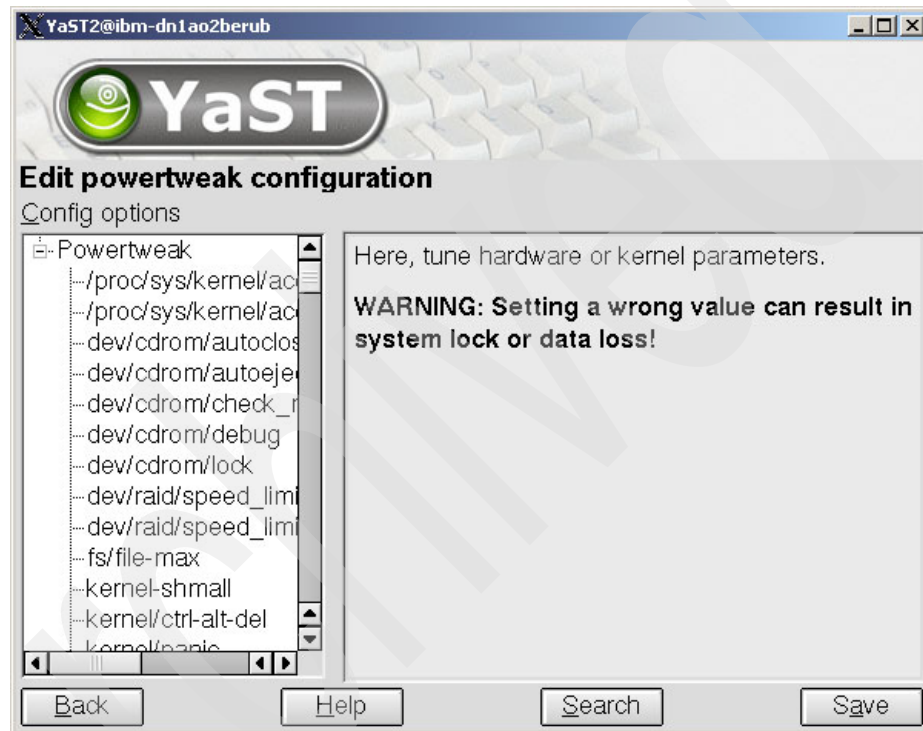


Figure 8-4 SUSE Linux powertweak

## Where the parameters are stored

The kernel parameters that control how the kernel behaves are stored in the `/proc` directory, in particular, `/proc/sys`. The files in the `/proc` directory tree provide a simple way of viewing configuration parameters that are related to the kernel, processes, memory, network, and other components. Each process running in



the system has a directory in `/proc`, with the process ID (PID) as its name. Table 8-2 lists some of the files that contain kernel information.

Table 8-2 Parameter files in `/proc`

File or directory	Purpose
<code>/proc/loadavg</code>	Contains information about the load of the server in one-minute, five-minute, and 15-minute intervals. The <b>uptime</b> command obtains information from this file.
<code>/proc/kcore</code>	Contains data to generate a core dump at run time for kernel debugging purposes: The <b>gdb</b> command creates the core dump as shown in the following example:  <pre>#gdb /usr/src/linux/vmlinux /proc/kcore</pre>
<code>/proc/stat</code>	Contains kernel statistics as process, swap, and disk input/output (I/O)
<code>/proc/cpuinfo</code>	Contains information about the installed CPUs
<code>/proc/meminfo</code>	Contains information about memory usage, which is used by the <b>free</b> command
<code>/proc/sys/abi/*</code>	Used to provide support for <i>foreign</i> binaries not native to Linux, that is, those compiled under other UNIX variants such as SCO UnixWare 7, SCO OpenServer, and SUN Solaris™ 2. By default, this support is installed, although it can be removed during installation.
<code>/proc/sys/fs/*</code>	Used to increase the number of open files the operating system allows, and to handle quota
<code>/proc/sys/kernel/*</code>	Used for tuning purposes. You can enable hotplug, manipulate shared memory, and specify the maximum number of pid files and level of debug in syslog.
<code>/proc/sys/net/*</code>	Used for tuning network in general, IPV4, and IPV6
<code>/proc/sys/vm/*</code>	Used to manage cache memory and buffer

## Using the **sysctl** command

The **sysctl** command uses the names of files in the `/proc/sys` directory tree as parameters. To modify the `shmmax` kernel parameter, display the `/proc/sys/kernel/shmmax` file using the **cat** command, and change the `/proc/sys/kernel/shmmax` file using the **echo** command, as shown in Example 8-2.

*Example 8-2 Changing the /proc/sys/kernel.shmmax file (part 1)*

---

```
#cat /proc/sys/kernel/shmmax
33554432
#echo 33554430 > /proc/sys/kernel/shmmax
#cat /proc/sys/kernel/shmmax
33554430
```

---

However, since using these commands can introduce errors, we recommend that you use the **sysctl** command because it checks the consistency of the data before making any change as shown in Example 8-3.

*Example 8-3 Changing the /proc/sys/kernel.shmmax file (part 2)*

---

```
#sysctl kernel.shmmax
kernel.shmmax = 33554432
#sysctl -w kernel.shmmax=33554430
kernel.shmmax = 33554430
#sysctl kernel.shmmax
kernel.shmmax = 33554430
```

---

This change to the kernel stays in effect only until the next reboot. To make a permanent change, edit the `/etc/sysctl.conf` file and add the appropriate command:

```
kernel.shmmax = 33554439
```

The next time you reboot, the parameter file is read. You can do the same without rebooting by entering the following command:

```
#sysctl -p
```

## 8.1.5 V2.6 Linux kernel parameters

Version 2.6 of the Linux kernel has many parameters that can improve the performance of the your installation. For more information about what is new with the V2.6 kernel, refer to the following Web site:

[http://www.infoworld.com/infoworld/article/04/01/30/05FElinux\\_1.html](http://www.infoworld.com/infoworld/article/04/01/30/05FElinux_1.html)

Some V2.4 features have been removed from V2.6, including:

- ▶ Support for Direct Rendering Manager (DRM)
- ▶ Logical Volume Manager 1
- ▶ Compress VFAT (CVF) support

In addition, while some features are still available in V2.6, they have been deprecated and may even be removed in future kernel releases, including the

devfs file system. Devfs has been replaced by the udev file system. This also means that to tune the Input/Output (I/O) scheduler, you should use the file that is exported in the sysfs directory `/sys/block/device/queue/iosched`, instead of `elvtune`, because `elvtune` uses Input/Output Controls (IOCTLs), which is also deprecated.

Table 8-3 lists the `sysctl` command parameters in the V2.6 kernel that are relevant to server performance.

**Important:** SUSE Linux Enterprise Server 9 is based on the V2.6 kernel.

Table 8-3 V2.6 kernel parameters that impact server performance

Parameter	Description and example of use
<code>kernel.mmap-hugepages-min-mapping</code>	Sets the minimum size for hugepages created with <code>mmap()</code> system call. The default is 256. <code>sysctl -w kernel.mmap-hugepages-min-mapping=2048</code>
<code>kernel.mmap-use-hugepages</code>	Enables the use for <code>mmap()</code> system call to create hugepages. The default is 0 (disabled). <code>sysctl -w kernel.mmap-use-hugepages=1</code>
<code>kernel.shm-use-hugepages</code>	Enables the use of hugepages. When enabled, pages are created with standard SYSv shared memory system call. The default is 0 (disable). <code>sysctl -w kernel.shm-use-hugepages=1</code>
<code>net.ipv4.conf.all.arp_announce</code>	Defines different restrictions between source IP address and from IP packages in Address Resolution Protocol (ARP) requests sent on interfaces. The default is 0, and valid values are 0, 1, and 2. <code>sysctl -w net.ipv4.conf.all.arp_announce=2</code>
<code>net.ipv4.conf.all.arp_ignore</code>	Defines different modes for replies in response to ARP requests. The default is 0. Valid values are 0, 1, 2, 3, and 8. <code>sysctl -w net.ipv4.conf.all.arp_ignore=8</code>
<code>net.ipv4.inet_peer_gc_maxtime</code>	Advises the garbage collector about how often to pass over the inet peer storage memory pool during low or absent memory pressure. The default is 120, measured in jiffies. <code>sysctl -w net.ipv4.inet_peer_gc_maxtime=240</code>
<code>net.ipv4.inet_peer_gc_mintime</code>	Sets the minimum time the garbage collector can pass cleaning memory. If your server has a heavy load, increase this time. The default is 10, measured in jiffies. <code>sysctl -w net.ipv4.inet_peer_gc_mintime=80</code>

Parameter	Description and example of use
net.ipv4.inet_peer_maxttl	The maximum time-to-live for the inet peer entries. New entries expire after this period if there is no memory pressure on the pool. The default is 600, measured in jiffies. sysctl -w net.ipv4.inet_peer_maxttl=500
net.ipv4.inet_peer_minttl	The minimum time-to-live for inet peer entries. Set to a high-enough value to cover fragment time to live in the reassembling side of fragmented packets. This minimum time must be smaller than net.ipv4.inet_peer_threshold. The default is 120, measured in jiffies. sysctl -w net.ipv4.inet_peer_minttl=80
net.ipv4.inet_peer_threshold	Sets the size of inet peer storage. When this limit is nearly reached, peer entries are thrown away, using the inet_peer_gc_mintime timeout. The default is 65644. sysctl -w net.ipv4.inet_peer_threshold=65644
vm.lower_zone_protection	Allows the increase of protected memory that lower zones receive against allocations that could use higher zones. The default is 0. sysctl -w vm.lower_zone_protection
vm.min_free_kbytes	Sets the value of each page's zone to minimum, low, or high, depending on the machine. The default is 724. For a small machine, use 128 KB. For large machines, use a maximum of 4096 KB. sysctl -w vm.min_free_kbytes=512
vm.nr_hugepages	Allocates or deallocates hugepages dynamically. The success of this operation depends on the amount of memory that is present in the system for this operation. The default is 0. sysctl -w vm.nr_hugepages=10

The V2.6 kernel has certain performance parameters that are usually not used. Table 8-4 displays the list.

Table 8-4 V2.6 kernel performance parameters that typically are not used

Parameter	Description and example of use
kernel.panic_on_oops	Enables kernel detection and handling of any process that causes a crash and calls the panic() function at the end. The kernel.panic parameter must also be set to 1. The default is 1 (enable). sysctl -w kernel.panic_on_oops=0

Parameter	Description and example of use
kernel.pid_max	Determines the maximum pid that a process can allocate. The default is 32768. sysctl -w kernel.pid_max=65536
net.ipv4.tcp_tw_recycle	The main states of Transmission Control Protocol (TCP) connection are ESTABLISHED, TIME_WAIT, and CLOSED, to enable the fast recycling function of TIME-WAIT sockets. The default is 0. sysctl -w net.ipv4.tcp_tw_recycle=10
net.ipv4.tcp_westwood	Enables TCP Westwood+ congestion control algorithm that optimizes the performance of TCP congestion control. The default is 0. sysctl -w net.ipv4.tcp_westwood=1
net.ipv6.conf.all.max_addresses	Sets maximum number of addresses per interface. The default is 0. sysctl -w net.ipv6.conf.all.max_addresses=10
vm.overcommit_ratio	Sets percentage of memory that is allowed for overcommit. The default is 50. sysctl -w vm.overcommit_ratio=17
vm.swappiness	Determines how likely the virtual machine (VM) subsystem is to swap to disk. The default is 60 and is typically sufficient. Valid values are 0 to 100. sysctl -w vm.swappiness=50

IBM Retail Environment for SUSE Linux V2 kernels have Linux Kernel Crash Dump (LKCD) installed. LKCD is a set of tools and kernel patches that support the creation and analysis of crash dumps. Refer to “Swap partition considerations with Linux Kernel Crash Dump” on page 179 for additional information.

### 8.1.6 Tuning the processor subsystem

The CPU is one of the most important hardware subsystems for servers whose primary role is that of an application or database server. However, in these systems, the CPU is often the source of performance bottlenecks.

To tweak processor tuning parameters, refer to 8.1.5, “V2.6 Linux kernel parameters” on page 158.

On high-end servers with Xeon processors, you can enable or disable *hyper-threading*. Hyper-threading is a way to virtualize each physical processor as two processors to the operating system. This is supported under SUSE Linux Enterprise Server 9.

By virtualizing the processor, you can execute two threads or processes at a time. This is also known as *thread-level parallelism*. By having your operating system and software designed to take advantage of this technology, you gain significant increases in performance, without needing an increase in clock speed. For example, if you enable hyper-threading on a four-way server, monitoring tools such as **top** will display eight processors, as shown in Figure 8-5.

```

10:22:45 up 23:40, 5 users, load average: 26.49, 12.03, 10.24
373 processes: 370 sleeping, 2 running, 1 zombie, 0 stopped
CPU states:  cpu    user    nice    system    irq    softirq    iowait    idle
              total   36.1%    0.1%    9.7%    0.3%     4.1%     1.6%    47.7%
              cpu00   17.0%    0.0%    5.9%    3.1%    20.8%     2.1%    50.7%
              cpu01   54.9%    0.0%   10.9%    0.0%     0.9%     1.3%    31.7%
              cpu02   33.4%    0.1%    8.5%    0.0%     2.5%     0.9%    54.2%
              cpu03   33.8%    0.7%   10.0%    0.0%     0.9%     2.1%    52.0%
              cpu04   31.4%    0.0%    9.3%    0.0%     2.9%     2.5%    53.6%
              cpu05   33.4%    0.0%    9.9%    0.0%     2.1%     0.7%    53.6%
              cpu06   30.5%    0.0%   11.1%    0.0%     1.7%     1.3%    55.1%
              cpu07   54.5%    0.0%   12.1%    0.0%     0.5%     1.9%    30.7%
Mem:  8244772k av, 3197880k used, 5046892k free,      0k shrd,  91940k buff
      2458344k active,          34604k inactive
Swap: 2040244k av,      0k used, 2040244k free          1868016k cached

```

Figure 8-5 Output of **top** on a four-way server, with hyper-threading enabled

While enabling hyper-threading, consider the following points:

- ▶ Symmetric multiprocessor (SMP)-based kernels are required to support hyper-threading.
- ▶ The higher the number of CPUs installed in a server, the less benefit hyper-threading has on performance. On servers that are CPU-bound, at most, expect the following performance gains:
  - Two physical processors: 15 - 25% performance gain
  - Four physical processors: 1 - 13% gain
  - Eight physical processors: 0 - 5% gain

For more information about hyper-threading, refer to the following Web site:

<http://www.intel.com/technology/hyperthread/>

EM64T is a 64-bit extension to Intel IA-32 processors. This means that the processors are capable of addressing more memory and can support new 64-bit applications while remaining fully compatible with all existing 32-bit applications. Support for this new processor is present in SUSE Linux Enterprise Server 9.

For more information about EM64T, refer to the following Web site:

<http://www.intel.com/technology/intel64/index.htm>

## Selecting the correct kernel

SUSE Linux Enterprise Server 9 includes several kernel packages, as listed in Table 8-5. To improve performance, select the most appropriate kernel for your system.

Table 8-5 Available kernels within the distribution

Kernel type	Description
SMP	Kernel has support for SMP and hyper-threaded machines
Standard	Kernel has support for single processor machines

### 8.1.7 Tuning the memory subsystem

Tuning the memory subsystem is a difficult task that requires constant monitoring to ensure that the changes do not negatively affect other subsystems in the server. If you choose to modify the virtual memory parameters in `/proc/sys/vm`, we recommend that you change only one parameter at a time and monitor how the server performs.

The tuning on VM parameters includes the following tasks:

- Configuring the way the Linux kernel flushes dirty buffers to disk. Disk buffers are used to cache data stored on disks that are very slow compared with random access memory (RAM). If the server uses this kind of memory, it can create serious problems with performance. Whenever a buffer becomes sufficiently dirty, use the following command:

```
sysctl -w vm.bdflush="30 500 0 0 500 3000 60 20 0"
```

`vm.bdflush` has nine parameters, but we recommend that you change only the following:

- Parameter 1 is `nfract`, the maximum percentage of buffer the `bdflush` daemon allows before queuing the buffer to be written to disk.
- Parameter 2 is `ndirty`, the maximum number of buffers `bdflush` flushes at once. If this value is very large, the `bdflush` daemon needs more time to finish the update to disk.
- Parameter 7 is `nfract_sync`, the maximum percentage of the buffer cache that is dirty before a synchronous flush occurs.

Leave the other parameters at their default values. For more details about `bdflush`, refer to “Setting `bdflush`” on page 172.

- Configuring the kswapd daemon to specify how many pages of memory are paged out by Linux. Use the following command to perform this task:

```
sysctl -w vm.kswapd="1024 32 64"
```

The kswapd daemon has three parameters:

- `tries_base` is four times the number of pages the kernel swaps in one pass. On a system with a lot of swapping, increasing the number may improve performance.
- `tries_min` is the minimum number of pages that kswapd swaps out each time the daemon is called.
- `swap_cluster` is the number of pages kswapd writes at once. A smaller number increases the number of disk I/Os performed, but a larger number could have a negative impact on the request queue.

**Note:** If you decide to make changes, check their impact using tools such as `vmstat`.

Other relevant VM parameters that may improve performance include:

- `buffermem`
- `freepages`
- `overcommit_memory`
- `page-cluster`
- `pagecache`
- `pagetable_cache`

### 8.1.8 Tuning the file system

Ultimately, all data must be retrieved from and stored to disk. Disk accesses are usually measured in milliseconds and are thousands of times slower than other components such as memory or Peripheral Component Interconnect (PCI) operations that are measured in nanoseconds or microseconds. The Linux file system is the method by which data is stored and managed on the disks.

Different file systems that differ in performance and scalability are available for Linux. Besides storing and managing data on the disks, the file systems are also responsible for guaranteeing data integrity. The newer Linux distributions include *journaling* file systems as part of the default installation. Journaling, or logging, prevents data inconsistency in case a system crashes. All modifications to the file system metadata are maintained in a separate journal or log and can be applied after a system crash to bring it back to its consistent state. Journaling also improves recovery time, because there is no need to perform file system checks at system reboot.



As with other aspects of computing, you will see a trade-off between performance and integrity. However, as Linux servers make their way into corporate data centers and enterprise environments, requirements such as high availability can be addressed.

In 8.1.8, “Tuning the file system” on page 164, we cover the default file system as well as additional file systems that are available on SUSE Linux Enterprise Server 9, and some simple ways to improve their performance.

## Hardware considerations for installing Linux

The minimum requirements for CPU speed and memory are well documented for current Linux distributions. These instructions also provide guidance about the minimum disk space required to complete the installation. However, they fall short on guidelines to initially set up the disk subsystem. As Linux servers cover a vast array of work environments because server consolidation makes its impact on data centers, one of the first questions to ask is about the function of the server being installed.

A server’s disk subsystems can be a major component of the overall system performance. Understanding the function of the server is the key to determining whether the I/O subsystem will have a direct impact on performance.

Here are examples of servers where disk I/O *is* most important:

- ▶ A *file and print server* must move data quickly between users and disk subsystems. Since the purpose of a file server is to deliver files to the client, the server should initially read all data from a disk.
- ▶ A *database server*’s goal is to search and retrieve data from a repository on the disk. Even with sufficient memory, most database servers perform large amounts of disk I/O to bring data records into memory and flush modified data to disk.

Following are examples of servers where disk I/O *is not* the most important subsystem:

- ▶ An *e-mail server* acts as a repository and router for electronic mail and tends to generate a heavy communication load. Networking is more important for this type of server.
- ▶ A *Web server* that is responsible for hosting Web pages, including static, dynamic, or both, benefits from a well-tuned network and memory subsystem.

### Selecting disk technology

Besides understanding the function of the server, you should also understand the size of the deployment the installation has to serve. Current disk subsystem technologies were designed with the size of deployment in mind. Table 8-6 displays a list of disk technologies that are currently available with the xSeries servers.

Table 8-6 Current disk technologies

Technology	Cost	Function	Limitations and capabilities
EIDE	Lowest cost	Direct-attached storage, for example, low-end servers, local storage (x305)	This extension of IDE is used for connecting internal storage. Maximum is two drives per EIDE controller.
SCSI	Low cost	Direct-attached storage, for example, mid-range to high-end server with local storage (x346, x365)	Although the standard for more than ten years, current I/O demands on high-end servers have stretched the capabilities of SCSI. Limitations include cable lengths, transfer speeds, maximum number of attached drives, and limits on number of systems that can actively access devices on one SCSI bus, affecting clustering capabilities.
Serial ATA (SATA)	Low cost	Midrange data storage applications	Generally available since late 2002, this new standard in HDD/system board interface is the follow-on technology to EIDE. With its point-to-point protocol, scalability improves since each drive has a dedicated channel. Sequential disk access is comparable to SCSI, but random access is less efficient. RAID functionality is also available.
iSCSI	Medium cost	Midend storage, for example, File/Web server	This became an RFC recently. Currently being targeted toward midrange storage and remote booting. Primary benefits are savings in infrastructure cost and diskless servers. It also provides the scalability and reliability associated with TCP/IP/Ethernet. High latency of TCP/IP limits performance.
Fibre Channel	High cost	Enterprise storage, for example, databases	Fibre Channel provides low latency and high throughput capabilities and removes the limitations of SCSI by providing cable distances of up to 10 km with fiber optic links, 2 Gbps transfer rate, redundant paths to storage to improve reliability. In theory, it can connect up to 16 million devices. In loop topologies, up to 127 storage devices or servers can share the same Fibre Channel connection allowing implementations of large clusters.

For additional information about available IBM storage solutions, refer to the following Web site:

<http://www.ibm.com/storage>

### ***Number of drives***

The number of disk drives significantly affects performance because each drive contributes to total system throughput. Capacity requirements are often the only consideration used to determine the number of disk drives configured in a server. Usually throughput requirements are not well-understood or are completely ignored. The key to a good performing disk subsystem depends on maximizing the number of read-write heads that can service I/O requests.

With RAID technology, you can spread the I/O over multiple spindles. There are two options for implementing RAID in a Linux environment, that is, software RAID and hardware RAID. Unless your server hardware comes standard with hardware RAID, start with the software RAID options that come with the Linux distributions. If the need arises, you can grow into the more efficient hardware RAID solutions.

Software RAID in the V2.6 Linux kernel distributions is implemented through the md device driver layer. This driver implementation is device-independent and, therefore, flexible in allowing many types of disk storage such as EIDE or SCSI to be configured as a RAID array. Supported software RAID levels are RAID-0 (striping), RAID-1 (mirroring), and RAID-5 (striping with parity), and can be completed as part of the initial installation or through the mdadm tool set.

If it is necessary to implement a hardware RAID array, you need a RAID controller for your system. In this case, the disk subsystem consists of the physical hard disks and the controller. IBM offers a complete product line of controllers as shown in Table 8-7.

*Table 8-7 Available IBM RAID controllers*

Storage controller	Product name	Features
ServeRAID™ Family	ServeRAID 7T	Entry-level, four-port SATA controller. Supports RAID level 0,1,10, and 5.
	ServeRAID 6M	Two-channel, Ultra320 SCSI with 14 disk drives per channel. Supports RAID levels 0,1,10,1E, 5, 50, and 5EE.
	ServeRAID 6I	Cost-effective “zero channel” using the onboard SCSI chipset. Supports standard RAID levels 0,00,1,10,5,50, and IBM exclusive 1E, 5EE, and 1E0.

Storage controller	Product name	Features
FAST	FAST100	Entry-level storage server with support for up to 56 SATA drives and dual active 2 GB RAID controllers.
	FAST 200	Compact 3U size with fully-integrated Fibre Channel technology supporting up to 10 internal FC disk drives and a max of 66, with additional external enclosures available in both single and dual (HA) controller models.
	FAST 600	Models include single and dual controller supporting from 14 FC drives up to 112 FC or SATA disks with Turbo model. Turbo model also provides a 2 GB cache.
	FAST 700	Dual active RAID controllers, transfer rates of 2 Gbps and support for up to 224 drives for a maximum physical capacity of 32 TB. 2 GB battery-backed controller cache.
	FAST 900	Dual active 2 GB RAID controllers, up to 795 MBps throughput and support for up to 32 TB of FC disk storage or 56 TB of SATA storage. 2 GB battery-packed controller cache can support high-performance applications such as online transaction processing (OLTP) and data mining.

**Tip:** In general, adding drives is one of the most effective changes that can be made to improve server performance.

For additional, in-depth coverage of the available IBM storage solutions, refer to *IBM TotalStorage Disk Solutions for xSeries*, SG24-6874.

### ReiserFS, the default SUSE Linux file system

The default file system on a SUSE installation since SUSE Linux 7.1 has been ReiserFS, developed by Hans Reiser. This system has the following key performance aspects:

- ▶ Improves reliability and recovery due to the journaling designed into the file system from the beginning
- ▶ Provides faster access through the use of balanced tree data structures that allows for storing both content data and security metadata
- ▶ Facilitates efficient use of disk space because this file system does not rely on block sizes

The current version of ReiserFS installed with SUSE LINUX Enterprise Server 9 is V3.6. Work is underway to deliver Reiser4, the next release. The new Reiser4 file system is expected to deliver an unbreakable file system by eliminating

corruption with the implementation of an “atomic” file system where I/O is guaranteed, to complete a 2x to 5x speed improvement by implementing new access algorithms, and to ease third-party upgrades without reformatting, through the use of plug-ins.

## Other journaling file systems

Here is a sample of current file systems and their features for the Linux Kernel 2.6:

- ▶ Ext3 is based on incremental modifications to the Linux standard ext2 file system.
- ▶ ReiserFS is designed and implemented as a journaling file system from the beginning by Hans Reiser.
- ▶ JFS was first designed by IBM for OS/2® WARP® Server. Then its code base was used for AIX JFS2. It has been available since Linux Kernel 2.4.20, and was added to 2.5.6.
- ▶ XFS was first introduced by SGI on its IRIX 64-bit systems. It has been available since Linux Kernel 2.4.x and was added to 2.5.3.

**Note:** During installation, SUSE Linux Enterprise Server 9 only offers the option of installing ext2, ext3, JFS, and XFS file systems.

## File system tuning in a Linux kernel

Out-of-the-box settings for the default file systems are adequate for most environments. However, here are a few pointers to help improve overall disk performance.

### ***Accessing time updates***

The Linux file system keeps records about when files are created, updated, and accessed. Default operations include updating the last-time-read attribute for files during reads and writes to files. Since writing is an expensive operation, eliminating unnecessary I/O leads to overall improved performance.

Mounting file systems with the noatime option eliminates the inode access times from being updated. If file update times are not critical to your implementation, as in a Web serving environment, mount file systems with the noatime flag in the /etc/fstab file as shown in Example 8-4.

*Example 8-4 Updating the /etc/fstab file with noatime set on mounted file systems*

---

```
/dev/sdb1 /mountlocation ext3 defaults,noatime 1 2
```

---

You must have a separate /var partition and mount it with the noatime option.

## ***Tuning the elevator algorithm***

The disk I/O elevator algorithm was introduced as a feature in the V2.4 kernel. It enables you to tune the algorithm that schedules block I/O by controlling the amount of time an I/O request remains on the queue before being serviced.

This is accomplished by adjusting the read and write values of the elevator algorithm. By increasing latency times, that is, larger values for read, write, or both, I/O requests stay on the queue for a longer period of time, giving the I/O scheduler the opportunity to coalesce these requests to perform more efficient I/O and increase throughput.

If your Linux server is in an environment with large amounts of disk I/O, finding the right balance between throughput and latency may be beneficial. Linux file systems are implemented as block devices. Improving the number of times those blocks are read and written can improve file system performance. As a guideline, heavy I/O servers benefit from smaller caches, prompt flushes, and a balanced high-latency read to write.

As with other system tuning, elevator algorithm tuning is an iterative process. Establish a baseline of the current performance, make any changes, and then measure the effect of the changes.

Example 8-5 shows how the **/sbin/elvtune** command is used to first show the current settings and then change the values for the read and write queues. If any changes are made, ensure that the **/sbin/elvtune** command is added to the **/etc/init.d/boot.local** file to make it a persistent change between system boots.

*Example 8-5 Finding current defaults for your installation and changing them*

---

```
[root@x232 root]# elvtune /dev/sda

/dev/sda elevator ID          2
      read_latency:          2048
      write_latency:         8192
      max_bomb_segments:      6

[root@x232 root]# elvtune -r 1024 -w 2048 /dev/sda

/dev/sda elevator ID          2
      read_latency:          1024
      write_latency:         2048
      max_bomb_segments:      6
```

---

**Note:** At the time this book was written, documentation for V2.6 of the Linux kernels states that the **e1vtune** command is obsolete and that tuning on newer kernels can be accomplished through the `/sys/block` structure.

## Selecting the journaling mode of an ext3 file system

You can set three different journaling options in the ext3 file system with the `data` option in the **mount** command:

- ▶ **data=journal**

This journaling option provides the highest form of data consistency by causing both file data and metadata to be journalled. It also has the higher performance overhead.

- ▶ **data=ordered (default)**

In this mode, only metadata is written. However, file data is guaranteed to be written first. This is the default setting.

- ▶ **data=writeback**

This journaling option provides the fastest access to the data at the expense of data consistency. The data is guaranteed to be consistent as the metadata is still being logged. However, no special handling of actual file data is done, and this may lead to old data appearing in files after a system crash.

There are three ways to change the journaling mode on a file system:

- ▶ Executing the **mount** command as follows:

```
mount -o data=writeback /dev/sdb1 /mnt/mountpoint
```

Here, `/dev/sdb1` is the file system being mounted.

- ▶ Including it in the options section of the `/etc/fstab` file as follows:

```
/dev/sdb1 /testfs ext3 defaults,journal=writeback 0 0
```

- ▶ Modifying the default `data=ordered` option on the root partition

To do this, make the change to the `/etc/fstab` file, and execute the **mkinitrd** command to scan the changes in the `/etc/fstab` file and create a new image. Update `grub` or `lilo` to point to the new image.

For more information about ext3, refer to the following Web site:

<http://www.redhat.com/support/wpapers/redhat/ext3/>

## Tuning ReiserFS

One of the strengths of ReiserFS is its support for a large number of small files. Instead of using the traditional block structure of other Linux file systems,

ReiserFS uses a tree structure that has the capability to store the actual contents of small files or the *tails* of those that are larger in the access tree. Since this file system does not use fixed block sizes, only the space that is needed to store a file is used, leading to less wasted space.

When mounting a ReiserFS file system, there is an option that improves performance, but at the expense of space. When mounting a ReiserFS, one can disable this tail packing option by specifying `notail`, so that the file system performs a little faster, but uses more disk space:

```
/dev/sdb1 /testfs reiserfs notail 0 0
```

## Setting bdflush

In the virtual memory subsystem, a tuning action helps improve the overall file system performance. The `bdflush` kernel daemon is responsible for making sure that dirty buffers, that is, any modified data that currently resides only in the volatile system memory, are committed to disk.

By modifying the `/proc/sys/vm/bdflush` parameters, you can modify the writing-to-disk rate, possibly avoiding disk contention problems. Changes in the `/proc` system take effect immediately, but are reset at boot time. To make the changes permanent, include the `echo` command in the `/etc/rc.d/rc.local` file:

```
echo 30 500 0 0 500 30000 60 20 0 > /proc/sys/vm/bdflush
```

There are nine parameters in `/proc/sys/vm/bdflush` of V2.6 Linux kernels:

<b>nfract</b>	The maximum percentage of dirty buffers in the buffer cache  The higher the value, the longer the write to the disk is postponed. When available memory is in short supply, large amounts of I/O have to be processed. To spread I/O out evenly, keep this at a low value.
<b>ndirty</b>	The maximum number of dirty buffers that the <code>bdflush</code> process can write to disk at one time  A large value results in I/O occurring in bursts, while a small value may lead to memory shortages if the <code>bdflush</code> daemon is not executed enough.
<b>dummy2</b>	Unused (formerly, <code>nrefill</code> )
<b>dummy3</b>	Unused
<b>interval</b>	The minimum rate at which <code>kupdate</code> will wake and flush  The default is five seconds, with a minimum value of zero seconds and a maximum of 600 seconds.



<b>age_buffer</b>	The maximum time the operating system will wait before writing buffer cache to disk  The default is 30 seconds, with minimum of one second and maximum of 6000 seconds.
<b>nfraction_sync</b>	The percentage of dirty buffers to activate bdflush synchronously. Default is 60%
<b>nfraction_stop</b>	Percentage of dirty buffers to stop bdflush  The default is 20%.
<b>dummy5</b>	Unused

## Tuning IDE drives

If your server is equipped with an integrated development environment (IDE) drive, use the **hdparm** command to set hard drive performance settings:

`/sbin/hdparm`

Some of the common options are:

- ▶ `/sbin/hdparm -c1 /dev/hda`  
This option turns on the 32-bit I/O on the PCI bus or the `/dev/hdb` or `/dev/hdc`, where each IDE block device is defined as `/dev/hdx`.
- ▶ `/sbin/hdparm -u1 /dev/hda`  
This option unmask other interrupts.
- ▶ `/sbin/hdparm -A /dev/hda`  
This option can be used to increase read-ahead when dealing with large sequential files.
- ▶ `/sbin/hdparm -d1 /dev/hda`  
This option enables direct memory access (DMA).
- ▶ `/sbin/hdparm -d1 -X66 /dev/hda`  
This option enables Ultra™ DMA transfers.

The IBM *@server*® BladeCenter® comes with an internal EIDE drive. By default, the drive has DMA access enabled. However, the write caching is

disabled. Example 8-6 shows how to set the write caching and its effect on the buffer cache reads.

*Example 8-6 Using hdparm on an IBM BladeCenter to turn on write cache*

---

```
[root@blade2 root]# hdparm -Tt /dev/hda

/dev/hda:
Timing buffer-cache reads: 1884 MB in 2.00 seconds = 942.00 MB/sec
Timing buffered disk reads: 76 MB in 3.03 seconds = 25.08 MB/sec
[root@blade2 root]# hdparm -W 1 /dev/hda

/dev/hda:
setting drive write-caching to 1 (on)
[root@blade2 root]# hdparm -Tt /dev/hda

/dev/hda:
Timing buffer-cache reads: 1932 MB in 2.00 seconds = 966.00 MB/sec
Timing buffered disk reads: 76 MB in 3.00 seconds = 25.33 MB/sec
[root@blade2 root]#
[root@blade2 root]# hdparm /dev/hda
```

---

We recommend that you take baseline performance measurements for your current settings before making any changes. Use the command, invoked as **hdparm -tT /dev/hda**, to see the effect on the speed of reading directly from the buffer cache with no disk accesses (-T) or displaying the speed of reading through the buffer cache without any prior caching of data (-t).

### Tagged Command Queuing for SCSI drives

Tagged Command Queuing (TCQ), first introduced in the SCSI-2 standard, is a method by which commands arriving at the SCSI drive are tagged and reordered while in the queue. This implementation increases I/O performance in server environments that have a heavy, random workload by reordering the requests to optimize the position of the drive head. Recently, this method of queuing and reordering, pending I/O requests, was extended to IDE drives. It is referred to as *Advanced Technology Attachment Tag Command Queuing* (ATA TCQ) or *legacy TCQ*, and *Native Command Queuing* (NCQ) in the Serial Advanced Technology Attachment (SATA) II specification.

Some xSeries servers include the integrated Adaptec AIC-7xxx SCSI controller. To check the current TCQ settings, run the following command:

```
cat /proc/scsi/aic7xxx/0
```

Refer to the explanation in `/usr/src/linux-2.6/drivers/scsi/README.aic7xxx` to learn how to change the default SCSI driver settings. It is not necessary to

recompile the kernel to try different settings. Specify a parameter, `aic7xxx=global_tag_depth:xx`, by adding a line in the `/etc/modules.conf` file:

Edit the `/etc/modules.conf` file to include  
`options aic7xxx aic7xxx=verbose.global_tag_depth:16`

**Note:** If you make a change in the `/etc/modules.conf` file involving a module in `initrd`, it requires a new image. Use the `mkinitrd` command to achieve this.

## Block sizes

The block size, the smallest amount of data that can be read or written to a drive, has a direct impact on a server's performance. If your server is handling many small files, a smaller block size makes it more efficient. If your server is dedicated to handling large files, a larger block size improves performance. Block sizes cannot be changed on-the-fly on existing file systems. Only a reformat can modify the current block size.

When a hardware RAID solution is being used, give careful consideration to the *stripe size* of the array or *segment* in the case of Fibre Channel. The *stripe unit size* is the granularity at which data is stored on one drive of the array before subsequent data is stored on the next drive of the array. Selecting the correct stripe size is a matter of understanding the predominant request size performed by a particular application.

As a general rule, streaming or sequential content benefits from large stripe sizes by reducing disk-head seek time and improving throughput. A more random type of activity, such as that found in databases, performs better with a stripe size that is equivalent to the record size.

## Guidelines for setting up partitions

A partition is a contiguous set of blocks on a drive that are treated as though they are independent disks. The default installation of SUSE Linux Enterprise Server 9 creates a monolithic installation with only three partitions:

- ▶ A swap partition: Automatically set to 2x RAM or 2 GB, whichever is larger
- ▶ A small boot partition, `/boot`, for example, 100 MB
- ▶ All the remaining space dedicated to `/`

There is a great deal of debate in Linux circles about the optimal disk partition. A single root partition method can cause problems in the future if you decide to redefine the partitions because of new or updated requirements. Too many partitions can lead to a file system management problem. During the installation process, Linux distributions permit you to create a multi-partition layout.

The benefits of running Linux on a multi-partitioned disk are:

- ▶ Improved security with finer granularity on file system attributes  
For example, the /var and /tmp partitions are created with attributes that permit easy access for all users and processes on the system, and are susceptible to malicious access. By isolating these partitions to separate disks, you can reduce the impact on system availability if these partitions have to be rebuilt or recovered.
- ▶ Improved data integrity, since loss of data resulting from a disk crash will be isolated to the affected partition  
For example, if there is no RAID implementation on the system (software or hardware) and the server suffers a disk crash, only those partitions on that disk have to be repaired or recovered.
- ▶ New installation and upgrades can be done without affecting other more static partitions  
For example, if the /home file system has not been separated to another partition, it will be overwritten during an operating system upgrade, losing all user files stored on it.
- ▶ More efficient backup process  
Partition layouts must be designed with backup tools in mind. It is important to understand whether backup tools operate on partition boundaries or on a more granular level like file systems.

Table 8-8 lists some partitions you can separate from root to provide more flexibility and better performance in your environment.

Table 8-8 Linux partitions and server environments

Partition	Contents and possible server environments
/home	A <i>file server environment</i> benefits from separating /home to its own partition. This is the home directory for all users on the system if there are no disk quotas implemented. Therefore, separating this directory should isolate a user's runaway consumption of disk space.
/tmp	If you are running a <i>high-performance computing environment</i> , large amounts of temporary space are needed during compute time, which is then released on completion.
/usr	This is where the <i>kernel source tree</i> and Linux <i>documentation</i> , as well as most executable binaries, are located. The /usr/local directory stores the executables that need to be accessed by all users on the system and is a good location to store custom scripts developed for your environment. If it is separated to its own partition, the files will not need to be reinstalled during an upgrade.

Partition	Contents and possible server environments
/var	The /var partition is important in <i>mail, Web, and print server environments</i> as it contains the log files for these environments, as well as the overall system log. Chronic messages can flood and fill this partition. If this occurs and the partition is not separate from the /, service interruptions are possible. Depending on the environment, further separation of this partition is possible by separating /var/spool/mail for a mail server or /var/log for system logs.
/opt	The installation of some third-party software products, such as Oracle's <i>database server</i> , default to this partition. If it is not separated, the installation will continue under /, and if enough space is not allocated, may fail.

For a more detailed and in-depth understanding of how Linux distributions handle file system standards, refer to the Filesystem Hierarchy Standard project home page on the Web at:

<http://www.pathname.com/fhs>

## The swap partition

The swap device is used when physical RAM is fully in use and the system needs additional memory. When free memory is not available on the system, it begins to page the least-used data from memory to the swap areas on the disks.

The initial swap partition is created during the Linux installation process, with current guidelines stating that the size of the swap partition should be two times the physical RAM. Linux kernels 2.4 and beyond support swap sizes of up to 24 GB per partition with an 8 TB theoretical maximum for 32-bit systems.

**Note:** Swap partitions should reside on separate disks.

If more memory is added to the server after the initial installation, additional swap space must be configured. There are two ways to configure additional swap after the initial install:

- ▶ Create a free partition on the disk as a swap partition. This is difficult if the disk subsystem has no free space available. In such a situation, create a swap file.
- ▶ If you have a choice, the preferred option is to create additional swap partitions. This improves performance since I/O to the swap partitions bypass the file system and all the overheads involved in writing to a file.

Another way to improve the performance of swap partitions or files is to create multiple swap areas. Linux takes advantage of multiple swap partitions or files and performs the reads and writes in parallel to disks. After creating additional

swap partitions or files, the `/etc/fstab` file contains entries similar to those shown in Example 8-7.

*Example 8-7 /etc/fstab file*

/dev/sda2	swap	swap	sw
0 0			
/dev/sdb2	swap	swap	sw
0 0			
/dev/sdc2	swap	swap	sw
0 0			
/dev/sdd2	swap	swap	sw
0 0			

Under normal circumstances, Linux first uses the `/dev/sda2` swap partition, then `/dev/sdb2`, and so on, until it allocates enough swapping space. This means that perhaps only the first partition, `/dev/sda2`, will be used if the need for a large swap space does not arise.

Spreading the data over all available swap partitions improves performance because all read/write requests are performed simultaneously to all selected partitions. If you change the file as shown in Example 8-8, you assign a higher priority level to the first three partitions.

*Example 8-8 Modified /etc/fstab to make parallel swap partitions*

/dev/sda2	swap	swap	sw,pri=3
0 0			
/dev/sdb2	swap	swap	sw,pri=3
0 0			
/dev/sdc2	swap	swap	sw,pri=3
0 0			
/dev/sdd2	swap	swap	sw,pri=1
0 0			

Swap partitions are used from the highest priority to the lowest (where 32767 is the highest and 0 is the lowest). Giving the same priority to the first three disks causes the data to be written to all three disks. The system does not wait until the first swap partition is full before beginning to write on the next partition. The system uses the first three partitions in parallel, and performance generally improves.

The fourth partition is used if additional space is needed for swapping after the first three are completely filled up. It is also possible to give all partitions the same priority to stripe the data over all partitions, but if one drive is slower than

the others, performance will decrease. A general rule is that the swap partitions should be on the fastest drives available.

**Note:** The swap space is not a replacement for RAM, since it is stored on physical drives that have a significantly slower access time than memory.

### Swap partition considerations with Linux Kernel Crash Dump

When a Linux system fails, it is useful to preserve an image - known as a *crash dump* - of system memory so that a post-analysis of the failure may be performed. To support this function, IBM Retail Environment for SUSE Linux V2 kernels have LKCD installed. LKCD is a set of tools and kernel patches that support the creation and analysis of crash dumps.

Part of the configuration for LKCD is the specification of the dump device where the dump file will be stored. By default, the first swap partition is used as the dump device. If the swap partition is smaller than the size of memory, the crash dump may not fit and the data will be truncated. In this situation, consider creating a dedicated swap partition to receive the crash dump. Use YaST to configure the partition as swap and make sure the partition will not be loaded by fstab. The dedicated swap partition can then be listed as the dump device. For example DUMPDEV=/dev/sdc1. You can find additional information on the Web at:

- ▶ <http://www.novell.com/coolsolutions/feature/15284.html>
- ▶ [http://lkcd.sourceforge.net/doc/lkcd\\_tutorial.pdf](http://lkcd.sourceforge.net/doc/lkcd_tutorial.pdf)

## 8.1.9 Tuning the network subsystem

Tune the network subsystem when the operating system is first installed, as well as when there is a perceived bottleneck in the network subsystem. A problem here can affect other subsystems, for example, CPU utilization can be affected significantly, especially when block sizes are too small, and memory use can increase if there is an excessive number of TCP connections.

### Preventing a decrease in performance

The following `sysctl` commands are used primarily to change security settings, but they also have the side effect of preventing a decrease in network

performance. These commands are changes to the default values in SUSE Linux.

- ▶ Disabling the following parameters with these commands prevents a malicious hacker from using a spoofing attack against the IP address of the server:

```
sysctl -w net.ipv4.conf.eth0.accept_source_route=0
sysctl -w net.ipv4.conf.lo.accept_source_route=0
sysctl -w net.ipv4.conf.default.accept_source_route=0
sysctl -w net.ipv4.conf.all.accept_source_route=0
```

- ▶ These commands configure the server to ignore redirects from machines that are listed as gateways. Since redirects can be used for malicious attacks, allow them only from trusted sources:

```
sysctl -w net.ipv4.conf.eth0.secure_redirects=1
sysctl -w net.ipv4.conf.lo.secure_redirects=1
sysctl -w net.ipv4.conf.default.secure_redirects=1
sysctl -w net.ipv4.conf.all.secure_redirects=1
```

In addition, you can allow the interface to accept or not accept any Internet Message Control Protocol (ICMP) redirects. The ICMP redirect is a mechanism for routers to convey routing information to hosts. For example, the gateway sends a redirect message to a host when the gateway receives an Internet datagram from a host on a network to which the gateway is attached. The gateway checks the routing table to get the address of the next gateway, and the second gateway routes the datagram's Internet to the destination in the network. Disable these redirects using the following commands:

```
sysctl -w net.ipv4.conf.eth0.accept_redirects=0
sysctl -w net.ipv4.conf.lo.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.all.accept_redirects=0
```

- ▶ If this server does not act as a router, it does not have to send redirects. Therefore, you can disable them with the help of these commands:

```
sysctl -w net.ipv4.conf.eth0.send_redirects=0
sysctl -w net.ipv4.conf.lo.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0
sysctl -w net.ipv4.conf.all.send_redirects=0
```

- ▶ Configure the server to ignore broadcast pings or smurf attacks by using the following command:

```
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

- ▶ Ignore all icmp packets or pings with the help of this command:

```
sysctl -w net.ipv4.icmp_echo_ignore_all=1
```



- Some routers send invalid responses to broadcast frames, and each one generates a warning that is logged by the kernel. These responses can be ignored by using the following command:

```
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
```

## Tuning TCP and UDP

To tune servers that support a large number of multiple connections, use the following commands:

- For servers that receive many connections at the same time, reuse the TIME-WAIT sockets for new connections with the following command:

```
sysctl -w net.ipv4.tcp_tw_reuse=1
```

If you enable this command, you should also enable the fast recycling of TIME-WAIT sockets status with this command:

```
sysctl -w net.ipv4.tcp_tw_recycle=1
```

- The parameter `tcp_fin_timeout` is the time to hold a socket in state FIN-WAIT-2 when the socket is closed at the server.

A TCP connection begins with a three-segment synchronization SYN sequence and ends with a three-segment FIN sequence, neither of which holds data. By changing the `tcp_fin_timeout` value, the time from the FIN sequence to when the memory can be freed for new connections can be reduced, thereby improving performance. This value, however, should be changed only after careful monitoring, because there is a risk of overflowing memory because of the number of dead sockets. To change the value, use this command:

```
sysctl -w net.ipv4.tcp_fin_timeout=30
```

- One of the problems found in servers with many simultaneous TCP connections is the large number of connections that are open, but unused. TCP has a keepalive function that probes these connections, and by default, drops them after 7200 seconds or two hours. This length of time may be too large for your server and may result in excess memory usage and a decrease in server performance.

Setting it to 1800 seconds or 30 minutes, for example, may be more appropriate and requires you to use this command:

```
sysctl -w net.ipv4.tcp_keepalive_time=1800
```

- Set the maximum operating system send buffer size (`wmem`) and receive buffer size (`rmem`) to 8 MB for queues on all protocols using the following commands:

```
sysctl -w net.core.wmem_max=8388608
```

```
sysctl -w net.core.rmem_max=8388608
```

These specify the amount of memory that is allocated to each TCP socket when it is created.

In addition, use the following commands for send and receive buffers. They specify three values, that is, minimum size, initial size, and maximum size:

```
sysctl -w net.ipv4.tcp_rmem="4096 87380 8388608"
sysctl -w net.ipv4.tcp_wmem="4096 87380 8388608"
```

The third value must be the same as or less than the value of `wmem_max` and `rmem_max`.

- Validate the source packets by reserved path. By default, routers route everything, even packets that are obviously not meant for this network. These packets can be dropped by enabling the appropriate filter with the help of these commands:

```
sysctl -w net.ipv4.conf.eth0.rp_filter=1
sysctl -w net.ipv4.conf.lo.rp_filter=1
sysctl -w net.ipv4.conf.default.rp_filter=1
sysctl -w net.ipv4.conf.all.rp_filter=1
```

- When the server is heavily loaded or has many clients with bad connections with high latency, it leads to an increase in half-open connections. This is common for Web servers, especially when there are many dial-up users. These half-open connections are stored in the *backlog connections* queue. Set this value to at least 4096. The default is 1024.

Setting this value is useful even if your server does not receive this kind of connection, as it can still be protected from a denial-of-service (syn-flood) attack. To set this value, use the following command:

```
sysctl -w net.ipv4.tcp_max_syn_backlog=4096
```

- Set the `ipfrag` parameters, particularly for NFS and Samba servers. Here, set the maximum and minimum memory used to reassemble IP fragments. When the value of `ipfrag_high_thresh` in bytes of memory is allocated for this purpose, the fragment handler drops packets until `ipfrag_low_thres` is reached.

Fragmentation occurs when there is an error during the transmission of TCP packets. Valid packets are stored in memory, as defined with these parameters, while corrupted packets are retransmitted.

For example, to set the range of available memory to between 256 MB and 384 MB, use the following commands:

```
sysctl -w net.ipv4.ipfrag_low_thresh=262144
sysctl -w net.ipv4.ipfrag_high_thresh=393216
```

## 8.2 Tuning tools

Various tools are available for monitoring and analyzing the performance of the server. Most of these tools use existing information stored in the `/proc` directory, to present it in a readable format.

In this section, a list of useful command line and graphical tools that are available as packages in SUSE Linux Enterprise Server or which you can download from the Internet, are listed. The three utilities **sar**, **iostat**, and **mpstat** are part of the Sysstat package that is provided on SUSE Linux CD1. They are also available at the Sysstat home page on the Web at:

<http://perso.wanadoo.fr/sebastien.godard/>

Table 8-9 lists the functions that these tools generally provide.

*Table 8-9 Linux performance monitoring tools*

Tool	Most useful tool function
uptime	Average system load
dmesg	Hardware and system information
top	Processor activity
iostat	Average CPU load, disk activity
vmstat	System activity
sar	Collect and report system activity
KDE System Guard	Real time systems reporting and graphing
free	Memory usage
traffic-vis	Network monitoring
pmap	Process memory usage
strace	Programs
ulimit	System limits
mpstat	Multiprocessor usage

These tools are in addition to the Capacity Manager tool, which is part of IBM Director. Capacity Manager monitors system performance over a period of time. Since IBM Director can be used on different operating system platforms, it is easier to collect and analyze data in a heterogeneous environment. Capacity Manager is discussed in detail in *Tuning IBM System x Servers for Performance*, SG24-5287.

## 8.2.1 The uptime command

Use the **uptime** command to see how long the server has been running and how many users are logged on, besides a quick overview of the average load of the server. Example 8-9 displays a sample output of the **uptime** command.

*Example 8-9 Sample output of uptime*

---

```
1:57am up 4 days 17:05, 2 users, load average: 0.00, 0.00, 0.00
```

---

The system load average is displayed for the last one-minute, five-minute, and 15-minute intervals. The load average is not a percentage but the number of processes in queue waiting to be processed. If processes that request CPU time are blocked, which means the CPU has no time to process them, the load average increases. If each process gets immediate access to CPU time and no CPU cycles are lost, the load decreases.

The optimal value of the load is one, which means that each process has immediate access to the CPU and no CPU cycles are lost. The typical load varies from system to system. For a uniprocessor workstation, for example, one or two is acceptable, where values of eight to ten are seen on multiprocessor servers.

Use the **uptime** command to pinpoint a problem with your server or the network. For example, if a network application is running poorly, run the **uptime** command, and you can see if the system load is high or not. If it is not high, the problem is more likely to be related to your network than to your server.

**Tip:** Use **w** instead of **uptime**. **w** also provides information about who is currently logged on to the machine and what the user is doing.

## 8.2.2 The dmesg command

The main purpose of the **dmesg** command is to display kernel messages. The **dmesg** command provides helpful information about problems occur that relate to hardware or while loading a module into the kernel.

In addition, with the **dmesg** command, you can determine what hardware is installed in the server. During every boot, Linux checks the hardware and logs information about it. To view these logs, use the **/bin/dmesg** command. Example 8-10 shows partial output from the **dmesg** command.

*Example 8-10 Partial output from dmesg*

---

```
EXT3 FS 2.4-0.9.19, 19 August 2002 on sd(8,1), internal journal
EXT3-fs: mounted filesystem with ordered data mode.
IA-32 Microcode Update Driver: v1.11 <tigran@veritas.com>
ip_tables: (C) 2000-2002 Netfilter core team
3c59x: Donald Becker and others. www.scyld.com/network/vortex.html
See Documentation/networking/vortex.txt
01:02:0: 3Com PCI 3c980C Python-T at 0x2080. Vers LK1.1.18-ac
00:01:02:75:99:60, IRQ 15
    product code 4550 rev 00.14 date 07-23-00
    Internal config register is 3800000, transceivers 0xa.
    8K byte-wide RAM 5:3 Rx:Tx split, autoselect/Autonegotiate interface.
    MII transceiver found at address 24, status 782d.
    Enabling bus-master transmits and whole-frame receives.
01:02:0: scatter/gather enabled. h/w checksums enabled
divert: allocating divert_blk for eth0
ip_tables: (C) 2000-2002 Netfilter core team
Intel(R) PRO/100 Network Driver - version 2.3.30-k1
Copyright (c) 2003 Intel Corporation

divert: allocating divert_blk for eth1
e100: selftest OK.
e100: eth1: Intel(R) PRO/100 Network Connection
    Hardware receive checksums enabled
    cpu cycle saver enabled

ide-floppy driver 0.99.newide
hda: attached ide-cdrom driver.
hda: ATAPI 48X CD-ROM drive, 120kB Cache, (U)DMA
Uniform CD-ROM driver Revision: 3.12
Attached scsi generic sg4 at scsil, channel 0, id 8, lun 0, type 3
```

---

### 8.2.3 The **top** command

The **top** command shows the actual processor activity. By default, it displays the most CPU-intensive tasks running on the server and updates the list every five seconds. You can sort the processes as follows:

- Process identification number (PID), that is, numerically

- Age, that is, newest first
- Time, that is, cumulative time
- Resident memory usage and time, that is, the time the process has occupied the CPU since startup.

Example 8-11 shows example output of the **top** command.

*Example 8-11 Example output from top command*

---

```
top - 02:06:59 up 4 days, 17:14,  2 users,  load average: 0.00, 0.00,
0.00
Tasks:  62 total,   1 running, 61 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.2% us,  0.3% sy,  0.0% ni, 97.8% id,  1.7% wa,  0.0% hi,
0.0% si
Mem:   515144k total,  317624k used,  197520k free,   66068k
buffers
Swap: 1048120k total,    12k used, 1048108k free,  179632k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
13737	root	17	0	1760	896	1540	R	0.7	0.2	0:00.05	top
238	root	5	-10	0	0	0	S	0.3	0.0	0:01.56	reiserfs/0
1	root	16	0	588	240	444	S	0.0	0.0	0:05.70	init
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	
migration/0											
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	
ksoftirqd/0											
4	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	
migration/1											
5	root	34	19	0	0	0	S	0.0	0.0	0:00.00	
ksoftirqd/1											
6	root	5	-10	0	0	0	S	0.0	0.0	0:00.02	events/0
7	root	5	-10	0	0	0	S	0.0	0.0	0:00.00	events/1
8	root	5	-10	0	0	0	S	0.0	0.0	0:00.09	kblockd/0
9	root	5	-10	0	0	0	S	0.0	0.0	0:00.01	kblockd/1
10	root	15	0	0	0	0	S	0.0	0.0	0:00.00	kirqd
13	root	5	-10	0	0	0	S	0.0	0.0	0:00.02	khelper/0
14	root	16	0	0	0	0	S	0.0	0.0	0:00.45	pdflush
16	root	15	0	0	0	0	S	0.0	0.0	0:00.61	kswapd0
17	root	13	-10	0	0	0	S	0.0	0.0	0:00.00	aio/0
18	root	13	-10	0	0	0	S	0.0	0.0	0:00.00	aio/1

---

To further modify the processes, use the **renice** command to give new priority to each process. If a process hangs or occupies too much CPU, terminate the process by using the **kill** command.

The columns in the output are as follows:

- ▶ PID: The process identification number
- ▶ USER: The name of the user who owns, and perhaps started the process
- ▶ PRI: The priority of the process  
For more details, refer to, “Process priority and nice levels” on page 187.
- ▶ NI: The niceness level, that is, whether the process tries to be nice by adjusting the priority by the number given  
For more details, refer to “Process priority and nice levels” on page 187.
- ▶ SIZE: The amount of memory (code+data+stack), in KB, which is being used by the process
- ▶ RSS: The amount of physical RAM used, in KB
- ▶ SHARE: The amount of memory shared with other processes, in KB
- ▶ STAT: The state of the process, with S=sleeping, R=running, T=stopped or traced, D=interruptible sleep, Z=zombie  
For more details on the zombie processes, refer to, “Zombie processes” on page 188.
- ▶ %CPU: The share of the CPU usage since the last screen update
- ▶ %MEM: The share of physical memory
- ▶ TIME: The total CPU time used by the process since it was started
- ▶ COMMAND: The command line used to start the task, including parameters

**Tip:** The `/bin/ps` command gives a snapshot view of the current processes.

## Process priority and nice levels

Process priority is a number that determines the order in which the process is handled by the CPU. The kernel adjusts this number up and down as needed. The *nice* value is a limit on the priority. The priority number is not allowed to go below the nice value. A lower nice value is a more favored priority.

It is not possible to change the priority of a process. It is possible to do so only indirectly, with the use of the nice level of the process. It may not always be possible to change the priority of a process using the nice level. If a process is running too slowly, assign more CPU to it by giving it a lower nice level. This means that all the other programs will have fewer processor cycles and run more slowly.

Linux supports nice levels from 19, the lowest priority, to -20, the highest priority. The default value is 0. To change the nice level of a program to a negative number, which makes it a high priority process, log in or **su** to root.

To start the program xyz with a nice level of -5, enter the following command:

```
nice -n -5 xyz
```

To change the nice level of a program that is already running, enter the following command:

```
renice level pid
```

To change the priority of the xyz program that has a PID of 2500, to a nice level of 10, enter the following command:

```
renice 10 2500
```

## Zombie processes

When a process has terminated after receiving a signal to do so, it normally takes time to finish all the tasks, such as closing open files, before ending. In that short time frame, the process is called a *zombie*.

After the process completes these shutdown tasks, it reports to the parent process that it is about to terminate. Sometimes a zombie process is unable to terminate itself, in which case, you will see that it has a status of Z, to mean (zombie).

It is not possible to terminate such a process with the **kill** command, because it is already considered *dead*. If you cannot get rid of a zombie, terminate the parent process. With this, the zombie disappears as well. However, if the parent process is the init process, do not end it since it is an important process. In such a situation, reboot to get rid of the zombie process.

## 8.2.4 The iostat command

The **iostat** command is a part of the Sysstat set of utilities. These are available on the Web at:

<http://perso.wanadoo.fr/sebastien.godard/>

The **iostat** command allows you to view average CPU times since the system was started. This is similar to the **uptime** command. However, the **iostat** command also creates a report on the activities server's disk subsystem. The report comprises two parts, CPU utilization and device or disk utilization. Example 8-12 shows sample output of the **iostat** command.



Example 8-12 Sample output of *iostat*

---

Linux 2.4.21-9.0.3.EL (x232)		05/11/2004			
avg-cpu:	%user	%nice	%sys	%idle	
	0.03	0.00	0.02	99.95	
Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
dev2-0	0.00	0.00	0.04	203	2880
dev8-0	0.45	2.18	2.21	166464	168268
dev8-1	0.00	0.00	0.00	16	0
dev8-2	0.00	0.00	0.00	8	0
dev8-3	0.00	0.00	0.00	344	0

---

To use the **iostat** command for performing detailed I/O bottleneck and performance tuning, refer to “Finding disk bottlenecks” on page 216.

The CPU utilization report has four sections:

- ▶ **%user**: This shows the percentage of CPU utilization taken up while executing at the user level, that is, applications.
- ▶ **%nice**: This shows the percentage of CPU utilization taken up while executing at the user level with a nice priority.
- ▶ **%sys**: This shows the percentage of CPU utilization taken up while executing at the system level, that is, kernel.
- ▶ **%idle**: This shows the percentage of time the CPU was idle.

The device utilization report is split into the following sections:

- ▶ **Device**: The name of the block device.
- ▶ **tps**: The number of transfers per second or I/O requests per second to the device. Many single I/O requests can be combined in a transfer request because a transfer request can have different sizes.
- ▶ **Blk\_read/s, Blk\_wrtn/s**: Blocks read and written per second indicate data read and written from and to the device in seconds. Blocks may also have different sizes. Typical sizes are 1024, 2048, or 4048 bytes, depending on the partition size. The block size of `/dev/sda1` can, for example, be located with the following command:

```
dumpe2fs -h /dev/sda1 |grep -F "Block size"
```

This gives an output similar to the one displayed in Example 8-13.

*Example 8-13 Blocks read and written per second*

---

```
dumpe2fs 1.34 (25-Jul-2003)
Block size:          1024
```

---

- ▶ **Blk\_read, Blk\_wrtn:** This indicates the total number of blocks read and written since the boot.

## 8.2.5 The vmstat command

The **vmstat** command provides information about processes, memory, paging, block I/O, traps, and CPU activity. Example 8-14 shows example output from the **vmstat** command.

*Example 8-14 Output from vmstat*

---

```
procs -----memory----- --swap-- -----io----- --system--
----cpu----
 r  b  swpd   free   buff  cache   si   so    bi    bo    in    cs us
sy id wa
 2  0      0 154804  77328 910900    0    0     4     6   103   19  0
0 100  0
```

---

The output shows the information in the following columns:

- ▶ **Process**
  - **r:** The number of processes waiting for run time
  - **b:** The number of processes in uninterruptable sleep
  - **w:** The number of processes swapped out but otherwise runnable (field is calculated)
- ▶ **Memory**
  - **swpd:** The amount of virtual memory used in KB
  - **free:** The amount of idle memory in KB
  - **buff:** The amount of memory used as buffers in KB
- ▶ **Swap**
  - **si:** Amount of memory swapped from the disk in KBps
  - **so:** Amount of memory swapped to the disk in KBps
- ▶ **IO**
  - **bi:** Blocks sent to a block device (block/s)
  - **bo:** Blocks received from a block device (block/s)

- ▶ System
  - in: The number of interrupts per second, including the clock
  - cs: The number of context switches per second
- ▶ CPU: These are percentages of total CPU time.
  - us: Time spent running non-kernel code (user time, including nice time)
  - sy: Time spent running kernel code (system time)
  - id: Time spent idle (prior to Linux 2.5.41, included IO-wait time)
  - wa: Time spent waiting for IO (prior to Linux 2.5.41, appeared as zero)

## 8.2.6 The sar command

The **sar** command is part of the Sysstat set of utilities. It is available from the following Web site:

<http://perso.wanadoo.fr/sebastien.godard/>

The **sar** command is used to collect, report, or save system activity information. It consists of three applications, including **sar**, which displays the data, and **sa1** and **sa2**, which are used to collect and store the data.

With **sa1** and **sa2**, configure the system to get the information and log it for later analysis. To do this, configure a cron job by adding the lines shown in Example 8-15 to `/etc/crontab`.

*Example 8-15 Starting automatic log reporting with cron*

---

```
# 8am-7pm activity reports every 10 minutes during weekdays.
*/10 8-18 * * 1-5 /usr/lib/sa/sa1 600 6 &
# 7pm-8am activity reports every an hour during weekdays.
0 19-7 * * 1-5 /usr/lib/sa/sa1 &
# Activity reports every an hour on Saturday and Sunday.
0 * * * 0,6 /usr/lib/sa/sa1 &
# Daily summary prepared at 19:05
5 19 * * * /usr/lib/sa/sa2 -A &
```

---

Alternately, use the **sar** command to run almost real-time reporting from the command line, as shown in Example 8-16.

*Example 8-16 Ad hoc CPU monitoring*

---

```
[root@x232 root]# sar -u 3 10
Linux 2.4.21-9.0.3.EL (x232)    05/22/2004
```

	CPU	%user	%nice	%system	%idle
02:10:40 PM	all	0.00	0.00	0.00	100.00
02:10:43 PM	all	0.33	0.00	0.00	99.67
02:10:46 PM	all	0.00	0.00	0.00	100.00
02:10:49 PM	all	7.14	0.00	18.57	74.29
02:10:52 PM	all	71.43	0.00	28.57	0.00
02:10:55 PM	all	0.00	0.00	100.00	0.00
02:10:58 PM	all	0.00	0.00	0.00	0.00
02:11:01 PM	all	0.00	0.00	100.00	0.00
02:11:04 PM	all	50.00	0.00	50.00	0.00
02:11:07 PM	all	0.00	0.00	100.00	0.00
02:11:10 PM	all	1.62	0.00	3.33	95.06
Average:	all				

---

The collected data displays a detailed overview of CPU utilization, including %user, %nice, %system, %idle, memory paging, network I/O and transfer statistics, process creation activity, activity for block devices, and interrupts/second over time.

## 8.2.7 KDE System Guard

KDE System Guard (KSysguard) is the KDE task manager and performance monitor. It features a client/server architecture that enables monitoring of local as well as remote hosts.

The graphical front end, which is displayed in Figure 8-6, uses *sensors* to retrieve the information it displays.

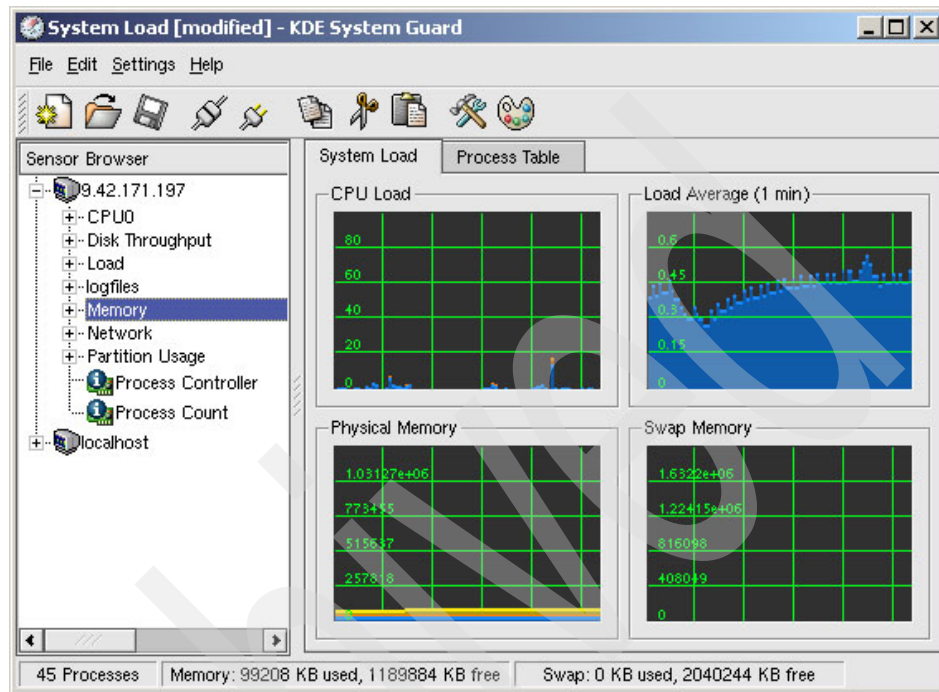


Figure 8-6 Default KDE System Guard window

A sensor returns both simple values and complex information such as tables. For each type of information, one or more displays are provided. These displays are organized in worksheets that can be saved and loaded independently of each other.

The KSysguard main window consists of a menu bar, an optional tool bar, status bar, sensor browser, and work space. When started for the first time, your local machine is listed as localhost in the sensor browser (as shown in Figure 8-7), with two tabs in the work space area. This is the default setup.

Each sensor monitors a certain system value. All the displayed sensors can be dragged and dropped in the work space. You can perform three tasks in the KSysguard window:

- ▶ Delete and replace sensors in the actual work space.
- ▶ Edit worksheet properties and increase the number of rows or columns or both.

- Create a new worksheet and drag and drop new sensors that meet your needs.

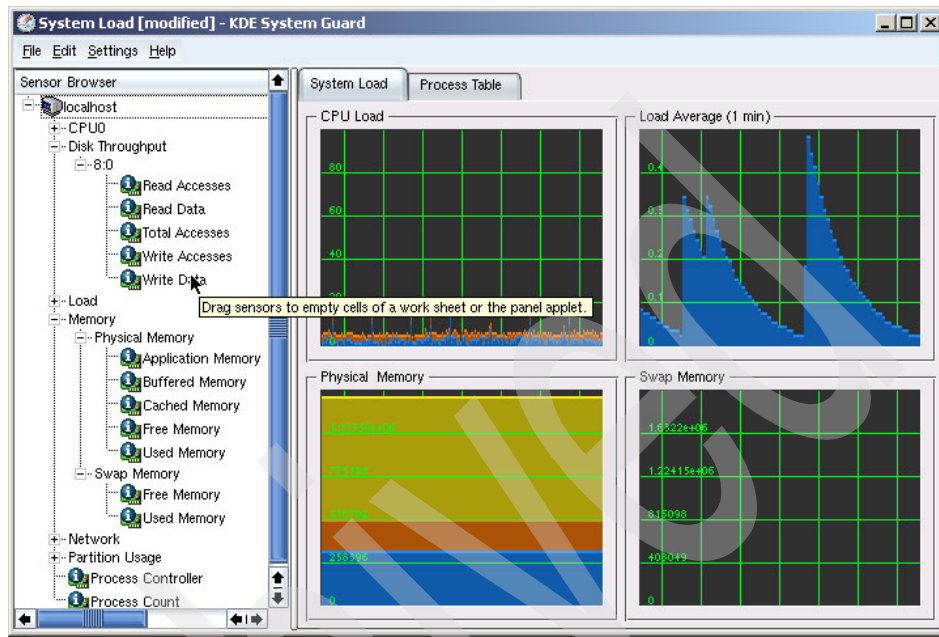


Figure 8-7 KDE System Guard Sensor Browser

## Work space

The work space displayed in Figure 8-7 shows two tabs:

- System Load (default view when starting KSysguard for the first time)
- Process Table

## System Load

The System Load worksheet consists of four sensor windows:

- CPU Load
- Load Average (one minute)
- Physical Memory
- Swap Memory

You can note from the Physical Memory window that it is possible to have multiple sensors displayed within one window. To determine which sensors are being monitored in a given window, move your mouse over the graph. Some descriptive text appears. Another way to do this is to right-click the graph and

click **Properties** → **Sensors**, as shown in Figure 8-8. This window also shows a key for what each color represents in the graph.

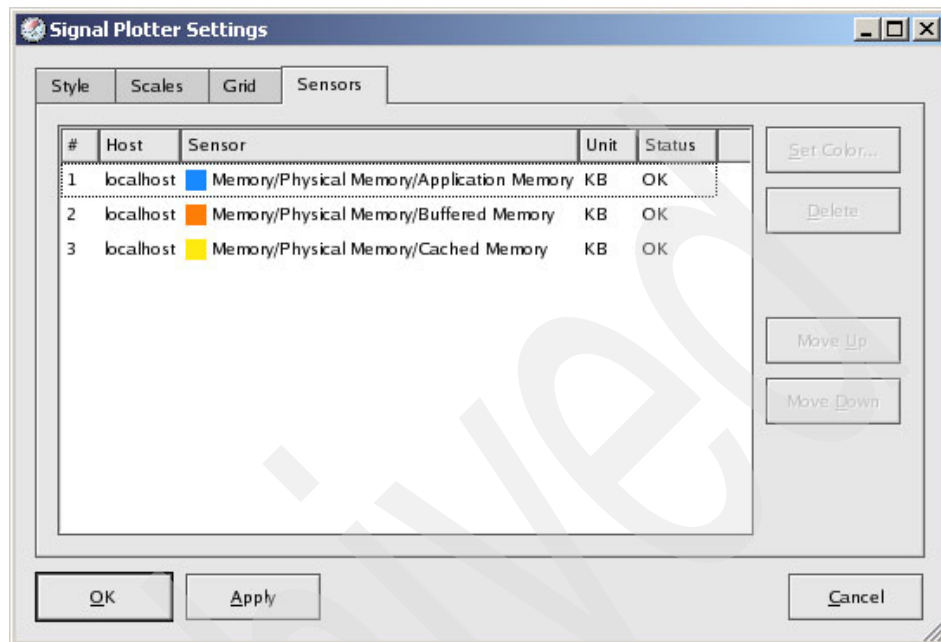


Figure 8-8 Signal Plotter Settings window

## Process Table

Click the Process Table tab to display information about all the running processes on the server, as shown in Figure 8-9. By default, the table is sorted by System CPU utilization. However, you can change this by clicking the heading by which you want to sort.

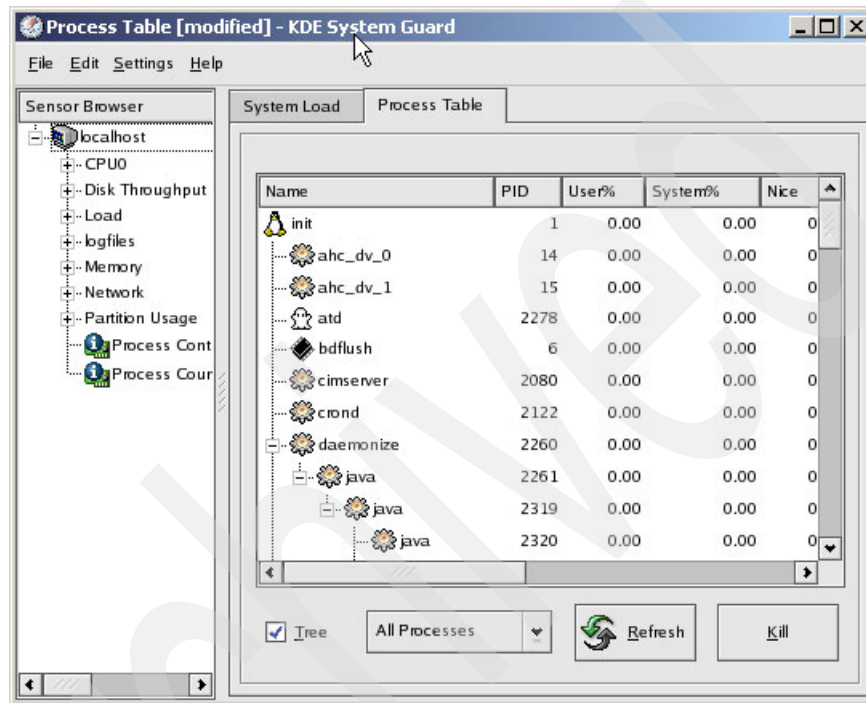


Figure 8-9 Process Table view

## Configuring a worksheet

In your environment or in the particular area you want to monitor, it may be necessary to use different sensors for monitoring. The best way to do this is to create a custom worksheet as shown in Figure 8-12 on page 199.

To create a worksheet, perform the following tasks:

1. Create a blank worksheet by clicking **File** → **New**.
2. In the Work Sheet Properties window (shown in Figure 8-10), enter a title and select the number of rows and columns. The corresponding dropdowns give you the maximum number of monitor windows, which in our case, is four. After setting the information is complete, click **OK**.



**Note:** The fastest update interval that can be defined is two seconds.

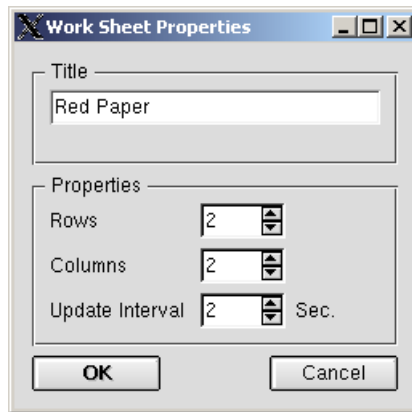


Figure 8-10 Worksheet properties

3. In the blank worksheet (shown in Figure 8-11), fill in the sensor boxes by simply dragging the sensors from the left side of the window to the desired box on the right. The display choices are:

- **Signal Plotter:** This sensor style displays samples of one or more sensors over time. If several sensors are displayed, the values are layered in different colors. If the display is large enough, a grid is displayed to show the range of the plotted samples.

By default, the automatic range mode is active. Therefore, the minimum and maximum values are set automatically. If you want fixed minimum and maximum values, deactivate the automatic range mode and set the values in the Scales tab from the Properties window, which you can access by right-clicking the graph.

- **Multimeter:** The Multimeter displays the sensor values as a digital meter. In the Properties windows, you can specify a lower and upper limit. If the range is exceeded, the display is colored in the alarm color.
- **BarGraph:** The BarGraph displays the sensor value as dancing bars. In the Properties window, you can also specify the minimum and maximum values of the range and a lower and upper limit. If the range is exceeded, the display is colored in the alarm color.
- **Sensor Logger:** The Sensor Logger does not display any values, but logs them in a file with additional date and time information.

For each sensor, define a target log file, the time interval the sensor will be logged and whether alarms are enabled.

Select **File** → **Save** to save the changes to the worksheet.

**Note:** When you save a worksheet, it is saved in your home directory, which may prevent other administrators from using your custom worksheets.

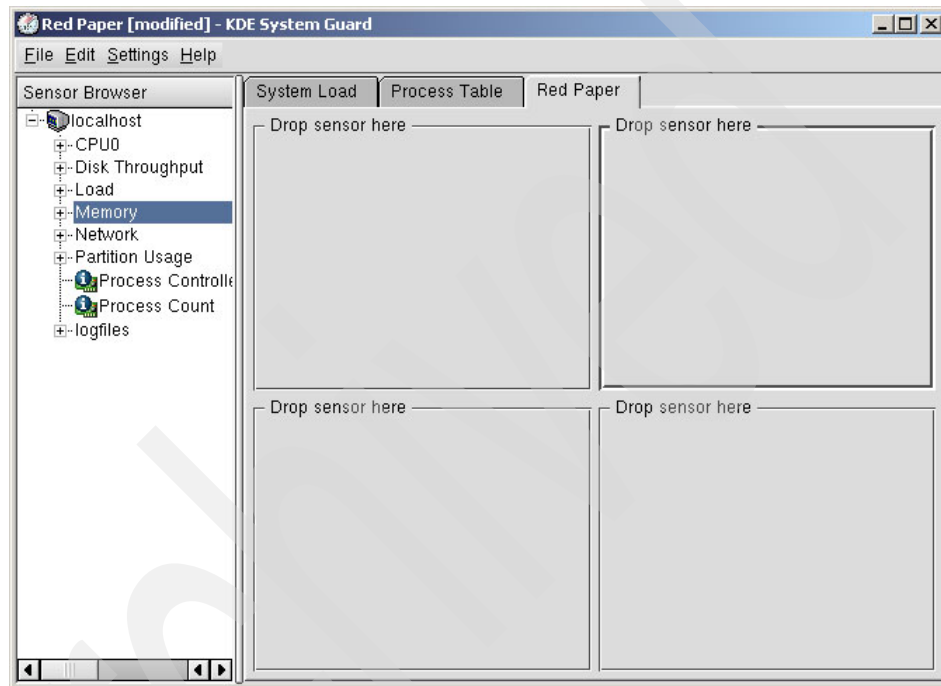


Figure 8-11 Empty worksheet

Figure 8-12 shows a sample saved worksheet.

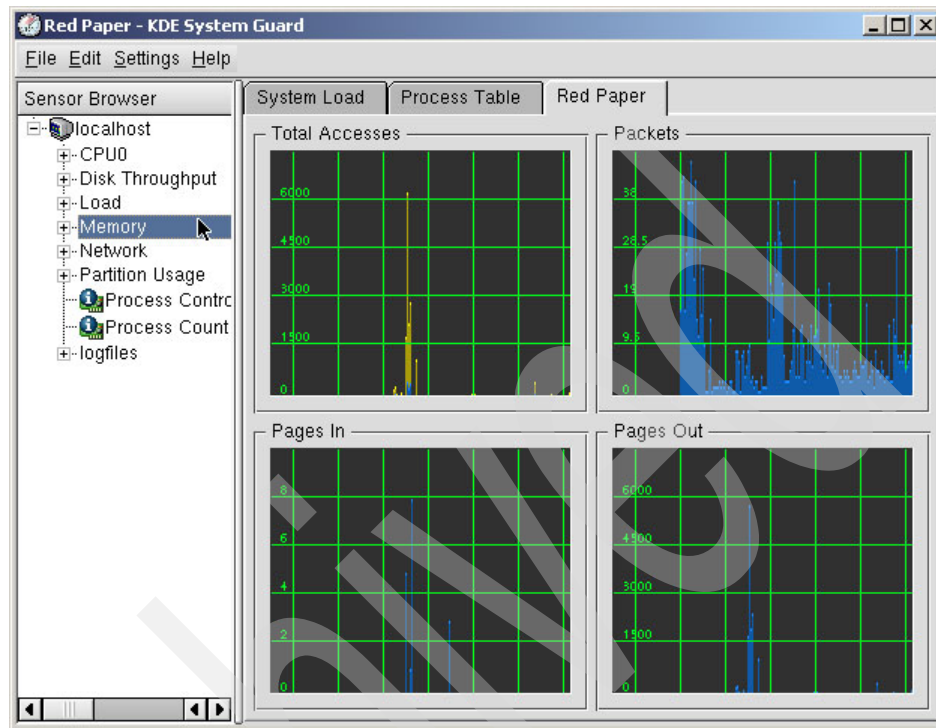


Figure 8-12 Saved worksheet

You can find more details about KDE System Guard on the Web at:

<http://docs.kde.org/en/3.2/kdebase/ksysguard>

### 8.2.8 The free command

The `/bin/free` command displays information about the total amount of free and used memory, including swap, on the system. It also includes information about the buffers and cache used by the kernel, as displayed in Example 8-17.

Example 8-17 Example output from the free command

	total	used	free	shared	buffers
cached					
Mem:	1291980	998940	293040	0	89356
772016					
-/+ buffers/cache:		137568	1154412		
Swap:	2040244	0	2040244		

## 8.2.9 Traffic-vis

*Traffic-vis* is a suite of tools that determines which hosts have been communicating on an IP network, with whom they have been communicating, and the volume of communication that has taken place. The final report can be generated in plain text, Hypertext Markup Language (HTML), or GIF formats.

To start the program for collecting data on interface eth0, for example, use the following command:

```
traffic-collector -i eth0 -s /root/output_traffic-collector
```

After the program starts, it is detached from the terminal and begins collecting data. To control the program, use the **killall** command to send a signal to the process. For example, to write the report to disk, enter the command:

```
killall -SIGUSR1 traffic-collector
```

To stop the collection of data, use the following command:

```
killall -SIGTERM traffic-collector
```

**Tip:** Do not forget to run the command to stop the collection of data. Otherwise, your system's performance will degrade due to a lack of memory.

You can sort the output by packets, bytes, TCP connections, the total of each one, or the number of sent or received or both of each one. For example, to sort the total packets sent and received on hosts, enter the following command:

```
traffic-sort -i output_traffic-collector -o output_traffic-sort -Hp
```

To generate a report in HTML format that displays the total bytes transferred, the total packets recorded, the total TCP connections requests, and other information about each server in the network, use the following command:

```
traffic-tohtml -i output_traffic-sort -o output_traffic-tohtml.html
```

This output file can be displayed in a browser, as shown in Figure 8-13.

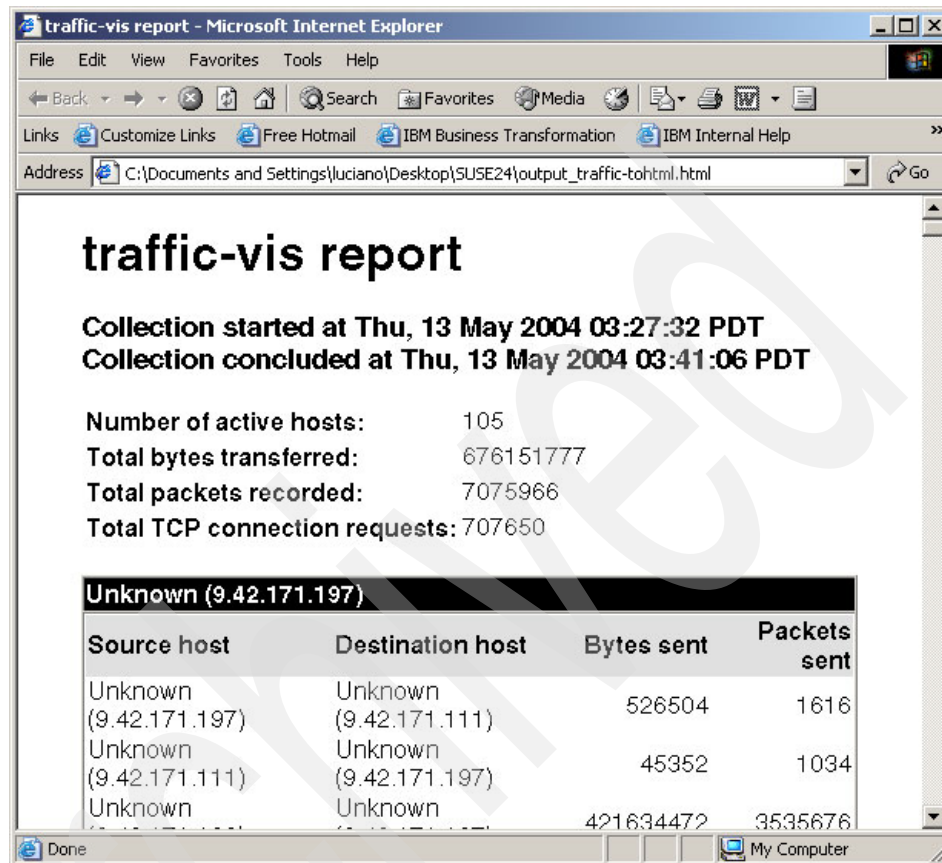


Figure 8-13 Report generated by traffic-vis in HTML format

To generate a report in GIF format with a width of 600 pixels and a height of 600 pixels, use the following command:

```
traffic-togif -i output_traffic-sort -o output_traffic-togif.gif -x 600 -y 600
```

Figure 8-14 shows the communication between systems in the network. You can also see that some hosts talk to others, but there are servers that never talk to each other. This output is typically used to find broadcasts in the network. To see which servers are using Internet Packet Exchange (IPX™)/Sequenced Packet Exchange (SPX) protocol in a TCP network, and to separate the two networks, remember that IPX is based on broadcast packets. To pinpoint others problems, such as damaged network cards or duplicated IPs on networks, use more

specific tools such as Ethereal, which is installed by default on the SUSE Linux Enterprise Server.

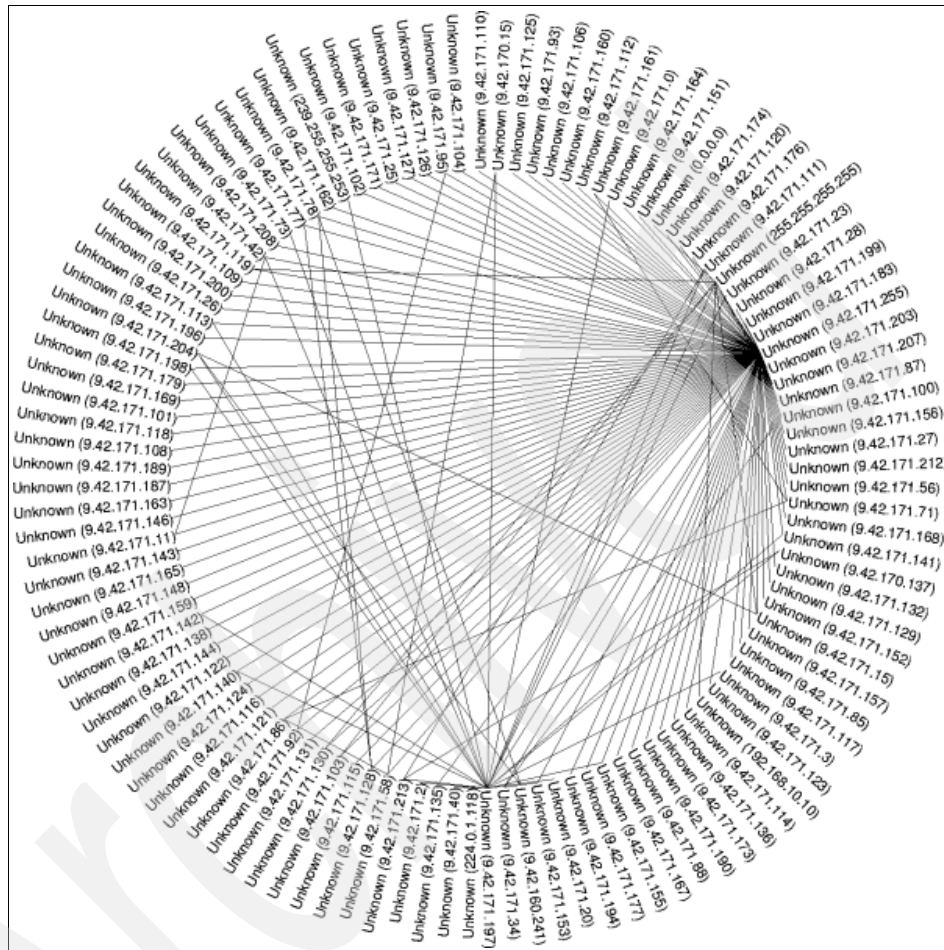


Figure 8-14 Communication between systems in the network

**Tip:** To produce output in one command, use pipes. For example, to generate a report in HTML, run:

```
cat output_traffic-collector | traffic-sort -Hp | traffic-tohtml -o  
output_traffic-tohtml.html
```

To generate a report as a GIF file, run:

```
cat output_traffic-collector | traffic-sort -Hp | traffic-togif -o  
output_traffic-togif.gif -x 600 -y 600
```

## 8.2.10 The pmap command

The **pmap** command reports the amount of memory that one or more processes are using. For example, use this command to determine which processes on the server are being allocated memory and whether this amount of memory is a cause of memory bottlenecks:

```
pmap -x <pid>
```

Example 8-18 shows the total amount of memory the cupsd process is using.

*Example 8-18 Total amount of memory the cupsd process is using*

---

```
linux:~ # pmap -x 1796
1796:  /usr/sbin/cupsd
```

Address	Kbytes	RSS	Anon	Locked	Mode	Mapping
08048000	244	-	-	-	r-x--	cupsd
ffffe000	4	-	-	-	-----	[ anon ]
-----						
total kB	6364	-	-	-		

---

For the complete syntax of the **pmap** command, issue the following command:

```
pmap -?
```

## 8.2.11 The strace command

The **strace** command intercepts and records the system calls that are called by a process, and the signals that are received by a process. This is a useful diagnostic, instructional, and debugging tool. System administrators use it to solve problems with programs. For more information, refer to Chapter 4, “Tuning Apache” in *Tuning SUSE LINUX Enterprise Server on IBM® server xSeries Servers*, REDP-3862.

To use the **strace** command, specify the process ID (PID) to be monitored, by entering the following command:

```
strace -p <pid>
```

Example 8-19 shows an output of the **strace** command.

*Example 8-19 Output of strace command*

---

```
[root@x232 html]# strace -p 815
Process 815 attached - interrupt to quit
semop(360449, 0xb73146b8, 1) = 0
poll([{fd=4, events=POLLIN}, {fd=3, events=POLLIN, revents=POLLIN}], 2, -1) = 1
accept(3, {sa_family=AF_INET, sin_port=htons(52534),
sin_addr=inet_addr("9.42.171.197")}, [16]) = 13
semop(360449, 0xb73146be, 1) = 0
getsockname(13, {sa_family=AF_INET, sin_port=htons(80),
sin_addr=inet_addr("9.42.171.198")}, [16]) = 0
fcntl64(13, F_GETFL) = 0x2 (flags O_RDWR)
fcntl64(13, F_SETFL, O_RDWR|O_NONBLOCK) = 0
read(13, 0x8259bc8, 8000) = -1 EAGAIN (Resource temporarily
unavailable)
poll([{fd=13, events=POLLIN, revents=POLLIN}], 1, 300000) = 1
read(13, "GET /index.html HTTP/1.0\r\nUser-A"... , 8000) = 91
gettimeofday({1084564126, 750439}, NULL) = 0
stat64("/var/www/html/index.html", {st_mode=S_IFREG|0644, st_size=152, ...}) = 0
open("/var/www/html/index.html", O_RDONLY) = 14
mmap2(NULL, 152, PROT_READ, MAP_SHARED, 14, 0) = 0xb7052000
writev(13, [{"HTTP/1.1 200 OK\r\nDate: Fri, 14 M"... , 264}, {"<html>\n<title>\n
RedPaper Per"... , 152}], 2) = 416
munmap(0xb7052000, 152) = 0
socket(PF_UNIX, SOCK_STREAM, 0) = 15
connect(15, {sa_family=AF_UNIX, path="/var/run/.nscd_socket"}, 110) = -1 ENOENT (No
such file or directory)
close(15) = 0
```

---

For the complete syntax of the **strace** command, use the following command:

```
strace -?
```

## 8.2.12 The ulimit command

The **ulimit** command is built into the bash shell and is used to provide control over the resources available to the shell and to the processes started by it on systems that allow such control. Use the **-a** option with the **ulimit** command to list all the parameters that can be set:

```
ulimit -a
```



Example 8-20 shows output of the **ulimit** command with the **-a** option.

*Example 8-20 Output of ulimit with -a option*

---

```
linux:~ # ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
stack size              (kbytes, -s) unlimited
cpu time                (seconds, -t) unlimited
max user processes      (-u) 4095
virtual memory          (kbytes, -v) unlimited
```

---

The **-H** and **-S** options specify the hard and soft limits that can be set for a given resource. If the soft limit is passed, the system administrator receives a warning. The hard limit is the maximum value that can be reached before the user gets the following error message:

Out of file handles.

You can, for example, set a hard limit for the number of file handles and open files (**-n**) with this command:

```
ulimit -Hn 4096
```

To set a soft limit for the number of file handles and open files, use the following command:

```
ulimit -Sn 1024
```

To see the hard and soft values, issue the following commands with a new value:

```
ulimit -Hn
ulimit -Sn
```

This command can be used to, for example, limit Oracle users on-the-fly. To set it on startup, enter the following lines in the **/etc/security/limits.conf** file, for example:

```
soft nofile 4096
hard nofile 10240
```

In addition, ensure that the `/etc/pam.d/login` and `/etc/pam.d/sshd` files have the following entry to enable the system to enforce these limits:

```
session required      pam_limits.so
```

For a complete syntax of the `ulimit` command, use the following command:

```
ulimit -?
```

### 8.2.13 The `mpstat` command

The `mpstat` command is a part of the Sysstat set of utilities that are available on the Web at:

<http://perso.wanadoo.fr/sebastien.godard/>

The `mpstat` command reports the activities of each of the CPUs available on a multiprocessor server. Global average activities among all CPUs are also reported. For example, to display three entries of global statistics among all processors at two second intervals, use the following command:

```
mpstat 2 3
```

Example 8-21 shows an output of the `mpstat` command on a uni-processor machine (xSeries 342).

**Tip:** You can also use this command in non-SMP machines.

*Example 8-21 Output of `mpstat` command on a uni-processor machine (xSeries 342)*

```
x342rsa:~ # mpstat 2 3
Linux 2.4.21-215-default (x342rsa)      05/20/04
```

07:12:16	CPU	%user	%nice	%system	%idle	intr/s
07:12:34	all	1.00	0.00	1.50	97.50	104.00
07:12:36	all	1.00	0.00	1.50	97.50	104.50
07:12:38	all	1.00	0.00	1.50	97.50	104.00
Average:	all	1.10	0.00	1.55	97.35	103.80

To display three entries of statistics for all processors of a multi-processor server at one second intervals, use the following command:

```
mpstat -P ALL 1 3
```

Example 8-22 shows output from running the **mpstat** command on a four-way machine (xSeries 232).

*Example 8-22 Output of mpstat command on a four-way machine (xSeries 232)*

```
[root@x232 root]# mpstat -P ALL 1 10
Linux 2.4.21-9.0.3.EL (x232)    05/20/2004
```

	CPU	%user	%nice	%system	%idle	intr/s
02:10:49 PM						
02:10:50 PM	all	0.00	0.00	0.00	100.00	102.00
02:10:51 PM	all	0.00	0.00	0.00	100.00	102.00
02:10:52 PM	0	0.00	0.00	0.00	100.00	102.00
Average:	all	0.00	0.00	0.00	100.00	103.70
Average:	0	0.00	0.00	0.00	100.00	103.70

For a complete syntax of the **mpstat** command, use the following command:

```
mpstat -?
```

## 8.3 Analyzing performance bottlenecks

This section is useful if you have a performance problem affecting a server. This is a reactive situation in which a series of steps lead you to a concrete solution that will restore the server to an acceptable performance level.

### 8.3.1 Identifying bottlenecks

Use the following actions to identify one or more bottlenecks:

1. Know your system.
2. Back up the system.
3. Monitor and analyze system performance.
4. Narrow down the bottleneck and find its cause.
5. Fix the cause of the bottleneck by trying a single change at a time.
6. Repeat steps 3 through 5 until you are satisfied with the performance of the system.

**Tip:** Document each step, especially the changes you make and their effect on performance.

## Gathering information

The only first-hand information you are likely to have access to are statements such as “There is a problem with the server”. It is important that you use probing questions to clarify and document the problem. Here is a list of questions you should ask to help you get a better picture of the system:

- ▶ Can you give me a complete description of the server in question?
  - Model
  - Age
  - Configuration
  - Peripheral equipment
  - Operating system version and update level
- ▶ Can you tell me the *exact* problem?
  - Symptoms
  - Description of error messages, if any

Any extra information the customer provides helps you locate the problem. For example, if a customer might say, “It is really slow when I copy large files to the server.” This may indicate a network problem or a disk subsystem problem.

- ▶ Who is experiencing the problem?

Is one person, a particular group of people, or the entire organization experiencing the problem? This helps you determine whether the problem exists in a particular part of the network, whether it is application-dependent, and so on. If only one user is experiencing the problem, it could be something to do with that user’s PC.

The perception that clients have about the server is usually a key factor. From this point of view, performance problems may not be directly related to the server. The network path between the server and the clients could be the cause of the problem. This path includes network devices and services provided by other servers, such as domain controllers.

- ▶ Can the problem be reproduced?

All the problems that can be reproduced can be solved. If you have sufficient knowledge of the system, you should be able to narrow down the problem to its root and decide on the actions to be taken.

The fact that the problem can be reproduced will help you see and understand it better. Document the sequence of actions necessary to reproduce the problem at any time:

- What steps must be followed to reproduce the problem?

Knowing the steps involved helps you reproduce the same problem on a different machine under the same conditions. If this works, it gives you the opportunity to use a machine in a test environment and eliminates the possibility of crashing the production server.

- Is it an intermittent problem?

If the problem is intermittent, gather the relevant information and find a path to move the problem in the reproducible category. The aim here is to have a scenario that reproduces the problem on command.

- Does it occur at certain times of the day or certain days of the week?

This helps you determine the cause of the problem. It may occur when everyone arrives for work or returns from lunch. Look for ways to change the timing, that is, make it happen less or more often. If you can do this, the problem becomes a reproducible one.

- Is it unusual?

If the problem falls in the non-reproducible category, you may conclude that it is the result of extraordinary conditions and classify it as fixed. However, in real life, there is a high probability that it will happen again.

A good way of troubleshooting a hard-to-reproduce problem is performing general maintenance on the server, that is, reboot, or bring the machine up-to-date on drivers and patches.

- ▶ When did the problem start? Was it gradual or did it occur very quickly?

If the performance issue appeared gradually, it is likely to be a sizing issue. If it appeared overnight, it could be because of a change made to the server or peripherals.

- ▶ Have any minor or major changes been made to the server, or are there any changes in the way the clients are using the server?

Did the customer alter something on the server or peripherals? Is a log of all the network changes available?

Demands could change, based on business requirements, in turn affecting the demands on servers and network systems. If a bottleneck arises due to this, ask the following questions:

- ▶ Are any other servers or hardware components involved?
- ▶ Are any logs available?

- ▶ What is the priority of the problem? When does it need to be fixed?
  - Does it need to be fixed in the next few minutes or in days?
 

You may have some time to fix it, or it may already be time to operate in panic mode.
  - How large is the problem?
  - What is the related cost of the problem?

## Analyzing the server's performance

At this point, start monitoring the server. The simplest way is to run the monitoring tools from the server that is being analyzed. For more information about monitoring tools, refer to 8.2, "Tuning tools" on page 183,.

**Important:** Before you undertake troubleshooting actions, back up all data and the configuration information to prevent a partial or complete loss.

Create a performance log of the server during its peak time of operation, for example, 9:00 a.m. to 5:00 p.m. This depends on the services being provided and people using these services. When creating the log, the following objects should be included, if available:

- ▶ Processor
- ▶ System
- ▶ Server work queues
- ▶ Memory
- ▶ Page file
- ▶ Physical disk
- ▶ Redirector
- ▶ Network interface

Before you begin, remember that a methodical approach to performance tuning is important. We recommend that you use the following process for the xSeries server performance tuning process:

1. Understand the factors affecting server performance. This book and *Tuning IBM System x Servers for Performance*, SG24-5287, provide further details.
2. Measure the current performance to create a performance baseline to compare with future measurements and to identify system bottlenecks.
3. Use monitoring tools to identify a performance bottleneck. You should be able to narrow down the bottleneck to the subsystem level.
4. Improve the component that is causing the bottleneck by performing the necessary actions for improving server performance in response to demands.

**Note:** It is important to understand that the greatest gains are obtained by upgrading the component that has a bottleneck when the other components in the server have ample *power* left to sustain an elevated level of performance.

5. Measure new performance. This helps compare the performance before and after the tuning steps.

When attempting to fix a performance problem, remember these points:

- ▶ Take measurements, that is, baseline measurements, before you upgrade or modify anything so that you can tell whether the change had any effect.
- ▶ Examine the options that involve reconfiguring existing hardware, and not just those that involve adding new hardware.

### 8.3.2 CPU bottlenecks

For servers whose primary role is that of an application or database server, the CPU is a critical resource, and can often be a source of performance bottlenecks. High CPU utilization does not always mean that a CPU is busy doing work. It may, in fact, be waiting on another subsystem. When performing an analysis, it is important to look at the system as a whole, besides all the subsystems, because there may be a cascade effect within the subsystems.

**Note:** There is a common misconception that the CPU is the most important part of the server. Unfortunately, this is often not the case, and as such, servers are often overconfigured with CPU and underconfigured with disks, memory, and network subsystems. Only specific applications that are truly CPU-intensive can take advantage of today's high-end processors.

#### Finding CPU bottlenecks

Determining bottlenecks with the CPU can be accomplished in several ways. As discussed in 8.2, “Tuning tools” on page 183, Linux has a variety of tools to help determine this. You can use the following tools:

- ▶ One useful tool is the **uptime** command. By analyzing the output from the **uptime** command, you can get a rough idea of what has been happening on the system for the past 15 minutes. For a more detailed explanation of this tool, refer to 8.2.1, “The uptime command” on page 184.

Example 8-23 shows sample output from running the **uptime** command from a CPU-strapped system.

*Example 8-23 uptime output from a CPU-strapped system*

---

```
18:03:16 up 1 day, 2:46, 6 users, load average: 182.53, 92.02, 37.95
```

---

- ▶ The KDE System Guard and the CPU sensors let you view the current CPU workload.

**Tip:** Do not add to the CPU problems by running too many tools at one time. Using many different monitoring tools at one time contribute to the high CPU load.

- ▶ With the **top** command, you can see details about CPU utilization and the processes that are the biggest contributor to the problem, as shown in Example 8-11 on page 186.
- ▶ If you have set up the **sar** command, you collect a lot of information, some of which is about CPU utilization over a period of time. Since analyzing this information is difficult, use **isag**, which takes **sar** output and plots a graph. Else, parse the information through a script and use a spreadsheet to plot it to see trends in CPU utilization. You can also use **sar** from the command line by issuing the **sar -u** command or the **sar -U processornumber** command.
- ▶ To gain a broader perspective of the system and current utilization of more than just the CPU subsystem, a good tool is the **vmstat** command, which is described in detail in 8.2.5, “The vmstat command” on page 190.

## Symmetric multiprocessor

Symmetric multiprocessor-based systems present their own set of problems that can be difficult to detect. In an SMP environment, there is the concept of *CPU affinity*. CPU affinity implies that you bind a process to a CPU.

This is useful because of the CPU cache optimization achieved by keeping the same process on one CPU rather than moving between processors. When a process moves between CPUs, the cache of the new CPU must be flushed. Therefore, a process that moves between processors causes many cache flushes to occur, which means that an individual process will take longer to finish. This scenario is difficult to detect because, while monitoring, it appears that the CPU load is very balanced and not necessarily peaking on any CPU.

Affinity is also useful in nonuniform memory access-based (NUMA) systems such as the xSeries 445 and xSeries 455, where it is important to keep memory, cache, and CPU access local to one another.



## Performance tuning options

The first step is to ensure that the system performance problem is because of the CPU and not one of the other subsystems. If it is the CPU that is causing the server bottleneck, perform these tasks to improve performance:

- ▶ Ensure that no unnecessary programs are running in the background, by using the **ps -ef** command. If such programs are found, stop them and use the **cron** command to schedule them to run on off-peak hours.
- ▶ Identify non-critical, CPU-intensive processes by using the **top** command and modify their priority using the **renice** command.
- ▶ In an SMP-based machine, try using the **taskset** command to bind the processes to CPUs to ensure that processes are not hopping between processors, causing cache flushes.
- ▶ Based on the application that is running, it is a good idea to scale up to bigger CPUs, rather than scale out with more CPUs. This is a function of whether your application was designed to effectively take advantage of more processors. For example, a single-threaded application will scale better with a faster CPU, than with more CPUs.
- ▶ General options include using the latest drivers and firmware, because this can affect the load they have on the CPU.

### 8.3.3 Memory bottlenecks

On a Linux system, many programs run at the same time. These programs support multiple users and some processes are used more often than others. Some of these programs use a portion of the memory, while the rest are *sleeping*. When an application accesses cache, the performance increases because an in-memory access retrieves data, thereby eliminating the need to access slower disks.

The operating system uses an algorithm to control which programs use physical memory, and which are paged out. This is transparent to user programs. Page space is a file created by the operating system on a disk partition to store user programs that are not currently being used. Typically, page sizes are 4 KB or 8 KB. In Linux, the page size is defined in the kernel header file `include/asm-architecture/param.h` using the variable `EXEC_PAGESIZE`. The process used to page a process out to disk is called *pageout*.

## Finding memory bottlenecks

Start your analysis by listing the applications running on the server. Determine how much physical memory and swap each of the applications needs to run. Figure 8-15 shows KDE System Guard monitoring memory usage.

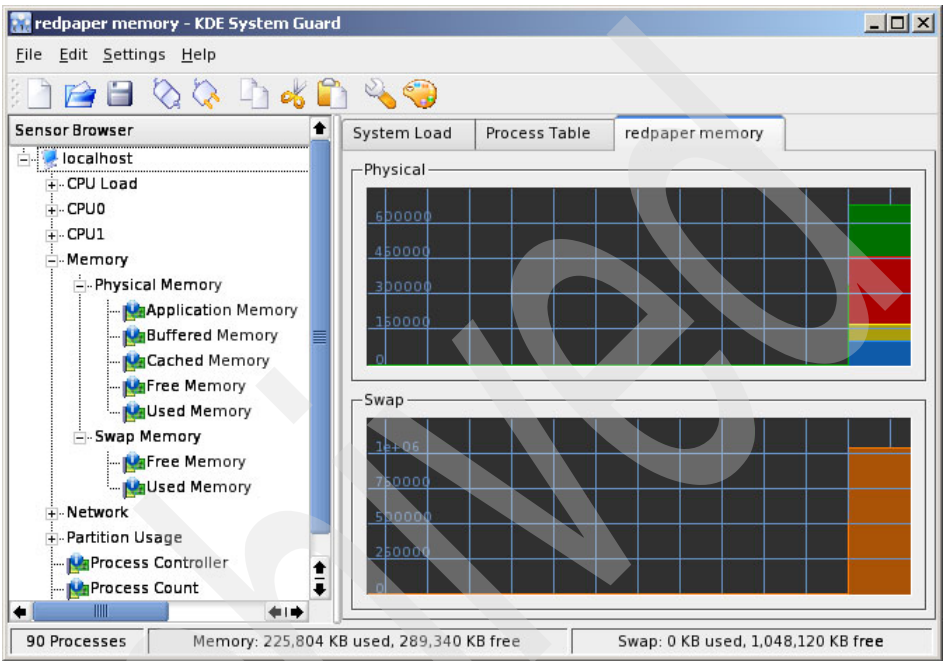


Figure 8-15 KDE System Guard memory monitoring

The indicators shown in Table 8-10 help define a memory problem.

Table 8-10 Indicators for memory analysis

Memory indicator	Analysis
Memory available	This indicates the physical memory available for use. If this value decreases significantly after starting the application, it could be a memory leak. Check the application causing it and make the necessary adjustments. Use the <code>free -l -t -o</code> command for additional information.
Page faults	There are two types of page faults: soft page faults, when the page is found in memory, and hard page faults, when the page is not found in memory and must be fetched from disk. Accessing the disk slows down the application considerably. The <code>sar -B</code> command provides useful information for analyzing page faults, specifically columns ppggin/s and ppggout/s.
File system cache	This is the common memory space.

Memory indicator	Analysis
Private memory for processes	This represents the memory used by each process running on the server. To see the amount of memory allocated to specific process, use the <b>psmap</b> command.

### ***Paging and swapping indicators***

As with all UNIX-based operating system, in Linux too differences exist between paging and swapping. While paging moves individual pages to the swap space on the disk, swapping is a bigger operation that moves the entire address space of a process to the swap space in one operation.

Swapping is a result of one of two situations:

- ▶ A process enters sleep mode.  
This normally happens because the process depends on interactive action, and editors, shells, and data entry applications spend most of their time waiting for user input. During this time, they are inactive.
- ▶ A process behaves poorly.  
Paging can be a serious performance problem when the amount of free memory pages falls below the minimum amount specified, because the paging mechanism is not able to handle the requests for physical memory pages, and the swap mechanism is called to free more pages. This significantly increases I/O to disk and quickly degrades a server's performance.

If your server is always paging to disk, that is, a high page-out rate, add more memory.

### **Performance tuning options**

If you suspect a memory bottleneck, perform one or more of the following actions:

- ▶ Tune the swap space using `bigpages`, `hugetlb`, and shared memory.
- ▶ Increase or decrease the size of pages.
- ▶ Improve the handling of active and inactive memory.
- ▶ Adjust the page-out rate.
- ▶ Limit the resources used for each user on the server.
- ▶ Stop the services that are not needed, as discussed in 8.1.1, "Disabling daemons" on page 148.
- ▶ Add memory.

### 8.3.4 Disk bottlenecks

The disk subsystem is often the most important aspect of server performance, besides being the most common bottleneck. However, problems can be hidden by other factors, such as lack of memory. Applications are considered to be *I/O bound* when CPU cycles are wasted while waiting for I/O tasks to finish.

The most common disk bottleneck is having too few disks. Most disk configurations are based on capacity requirements, and not performance. The least expensive solution is to purchase the smallest number of the largest-capacity disks possible. However, this places more user data on each disk, causing greater I/O rates to the physical disk and allowing disk bottlenecks to occur.

The second most common disk bottleneck is having too many logical disks on the same array. This increases the seek time and lowers performance to a great extent.

For more details about the disk subsystem, refer to 8.1.8, “Tuning the file system” on page 164.

#### Finding disk bottlenecks

A server exhibiting the following symptoms may be suffering from a disk bottleneck or a hidden memory problem:

- ▶ Slow disks result in memory buffers filling with write data or waiting for read data, which in turn delays all requests. This is because free memory buffers are unavailable for write requests, or the response is waiting for read data in the disk queue, or there is insufficient memory, as in the case of not having enough memory buffers for network requests causing synchronous disk I/O.
- ▶ Disk or controller utilization or utilization of both is very high.
- ▶ Most local area network (LAN) transfers happen only after disk I/O is completed, causing long response times and low network utilization.
- ▶ If disk I/O takes a relatively long time and disk queues become full, the CPUs are idle or have low utilization since they wait for a long period before processing the next request.

The disk subsystem is perhaps the most challenging one to configure properly. Besides looking at raw disk interface speed and disk capacity, it is important to also understand the workload. For example, is the disk access random or sequential? Is there large I/O or small I/O? Answering these questions provide the necessary information to ensure that the disk subsystem is adequately tuned.

Disk manufacturers tend to showcase the upper limits of their drive technology's throughput. However, we recommend that you understand the throughput of your workload to have a realistic expectation of your underlying disk subsystem.

Table 8-11 shows the true throughput for 8 KB I/Os for different drive speeds.

*Table 8-11 Exercise showing true throughput for 8 KB I/Os for different drive speeds*

Disk speed	Latency	Seek time	Total random access time <sup>a</sup>	I/Os per second per disk <sup>b</sup>	Throughput given for 8 KB I/O
15,000 RPM	2.0 ms	3.8 ms	6.8 ms	147	1.15 MBps
10,000 RPM	3.0 ms	4.9 ms	8.9 ms	112	900 KBps
7,200 RPM	4.2 ms	9 ms	13.2 ms	75	600 KBps

a. Assuming that the handling of the command + data transfer < 1 ms, total random access time = latency + seek time + 1 ms.

b. Calculated as 1/total random access time.

Random read/write workloads usually require several disks to scale. The bus bandwidths of SCSI or Fibre Channel are of lesser concern. Larger databases with random access workload benefit from having more disks. Larger SMP servers scale better with more disks. Given the I/O profile of 70% reads and 30% writes of the average commercial workload, a RAID-10 implementation performs 50 - 60% better than a RAID-5.

Sequential workloads tend to stress the bus bandwidth of disk subsystems. We recommend that you pay special attention to the number of SCSI buses and Fibre Channel controllers when maximum throughput is desired. Given the same number of drives in an array, RAID-10, RAID-0, and RAID-5 all have similar streaming read and write throughput.

There are two ways to approach disk bottleneck analysis:

► Real-time monitoring

Perform real-time monitoring while the problem is occurring. The drawback is that it may not be practical in cases where system workload is dynamic and the problem is not repeatable. However, if the problem is repeatable, this method is flexible because of the ability to add objects and counters as the problem becomes well-understood.

► Tracing

Tracing involves collecting performance data over time to diagnose a problem. It is a good way to perform remote performance analysis. Some of the drawbacks include having to analyze large files when performance problems are not repeatable, and not having all the key objects or parameters

or both in the trace and having to wait for the problem to occur again to get the additional data.

## The vmstat command

To track disk usage on a Linux system use the **vmstat** command. The columns of interest in the **vmstat** command with respect to I/O are the *bi* and *bo* fields. These fields monitor the movement of blocks in and out of the disk subsystem. Having a baseline is the key to being able to identify any changes over time.

Example 8-24 shows output from the **vmstat** command.

Example 8-24 *vmstat* output

---

```
[root@x232 root]# vmstat 2
```

r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy
id	wa												
2	1	0	9004	47196	1141672	0	0	0	950	149	74	87	
13	0	0											
0	2	0	9672	47224	1140924	0	0	12	42392	189	65	88	
10	0	1											
0	2	0	9276	47224	1141308	0	0	448	0	144	28	0	
0	0	100											
0	2	0	9160	47224	1141424	0	0	448	1764	149	66	0	
1	0	99											
0	2	0	9272	47224	1141280	0	0	448	60	155	46	0	
1	0	99											
0	2	0	9180	47228	1141360	0	0	6208	10730	425	413	0	
3	0	97											
1	0	0	9200	47228	1141340	0	0	11200	6	631	737	0	
6	0	94											
1	0	0	9756	47228	1140784	0	0	12224	3632	684	763	0	
11	0	89											
0	2	0	9448	47228	1141092	0	0	5824	25328	403	373	0	
3	0	97											
0	2	0	9740	47228	1140832	0	0	640	0	159	31	0	
0	0	100											

---

## The iostat command

When many files are opened and read and written to, and closed repeatedly, performance problems could occur. This becomes apparent as the seek times, that is, the time taken to move to the exact track where the data is stored, begin to increase. Use the **iostat** command to monitor the I/O device loading in real time. Different options allow you to drill down further to gather the necessary data.

Example 8-25 shows a potential I/O bottleneck on the device `/dev/sdb1`. This output shows the average wait times (`await`) of around 2.7 seconds and service times (`svctm`) of 270 ms.

*Example 8-25 Sample of an I/O bottleneck as shown with `iostat 2 -x /dev/sdb1`*

---

```
[root@x232 root]# iostat 2 -x /dev/sdb1
```

avg-cpu:	%user	%nice	%sys	%idle					
	11.50	0.00	2.00	86.50					

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s
avgqr-sz	avgqu-sz	await	svctm	%util				
/dev/sdb1	441.00	3030.00	7.00	30.50	3584.00	24480.00	1792.00	
12240.00	748.37	101.70	2717.33	266.67	100.00			

avg-cpu:	%user	%nice	%sys	%idle				
	10.50	0.00	1.00	88.50				

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s
avgqr-sz	avgqu-sz	await	svctm	%util				
/dev/sdb1	441.00	3030.00	7.00	30.00	3584.00	24480.00	1792.00	
12240.00	758.49	101.65	2739.19	270.27	100.00			

avg-cpu:	%user	%nice	%sys	%idle				
	10.95	0.00	1.00	88.06				

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s
avgqr-sz	avgqu-sz	await	svctm	%util				
/dev/sdb1	438.81	3165.67	6.97	30.35	3566.17	25576.12	1783.08	
12788.06	781.01	101.69	2728.00	268.00	100.00			

---

The **`iostat -x`** command for extended statistics provides low-level detail of the disk subsystem. Some of the important fields are:

- ▶ `%util`: The percentage of CPU consumed by I/O requests
- ▶ `svctm`: The average time required to complete a request, in milliseconds
- ▶ `await`: The average amount of time an I/O waited to be served, in milliseconds
- ▶ `avgqu-sz`: The average queue length
- ▶ `avgqr-sz`: The average size of request
- ▶ `rrqm/s`: The number of read requests merged per second issued to a device
- ▶ `wrqms`: The number of write requests merged per second issued to a device

For a more information about the fields, refer to 8.2.4, “The `iostat` command” on page 188.

Changes made to the elevator algorithm as described in “Tuning the elevator algorithm” on page 170, are seen in `avgrq-sz`, the average size of request, and `avgqu-sz`, the average queue length. As the latencies are lowered by manipulating the elevator settings, the `avgrq-sz` goes down.

To see the effect on the number of merged reads and writes the disk can manage, monitor the `rrqm/s` and `wrqm/s`.

### Performance tuning options

Once you are sure that the disk subsystem is causing the system bottleneck, you can perform various actions to solve the problem:

- ▶ Add a faster disk controller if the workload is of a sequential nature and is stressing the controller bandwidth. However, if the workload is more random in nature, the bottleneck is likely to involve the disk drives. In such a situation, add more drives to improve performance.
- ▶ Add more disk drives in a RAID environment. This spreads the data across multiple physical disks and improves performance for both reads and writes. It also increases the number of I/Os per second. Besides this, use hardware RAID instead of the software implementation provided by Linux. If hardware RAID is used, the RAID level is hidden from the operating system.
- ▶ Offload processing to another system in the network, including users, applications, or services.
- ▶ Add more RAM. Adding memory increases system memory disk cache, which in turn improves disk response times.

## 8.3.5 Network bottlenecks

A performance problem in the network subsystem can cause many problems such as *kernel panic*. To analyze these anomalies and detect network bottlenecks, each Linux distribution includes traffic analyzers.

### Finding network bottlenecks

We recommend KDE System Guard because of its GUI and ease of use. This tool is also available on the distribution CDs. Section 8.2.7, “KDE System Guard”



on page 192 discusses it in detail. Figure 8-16 shows the KDE System Guard in action.

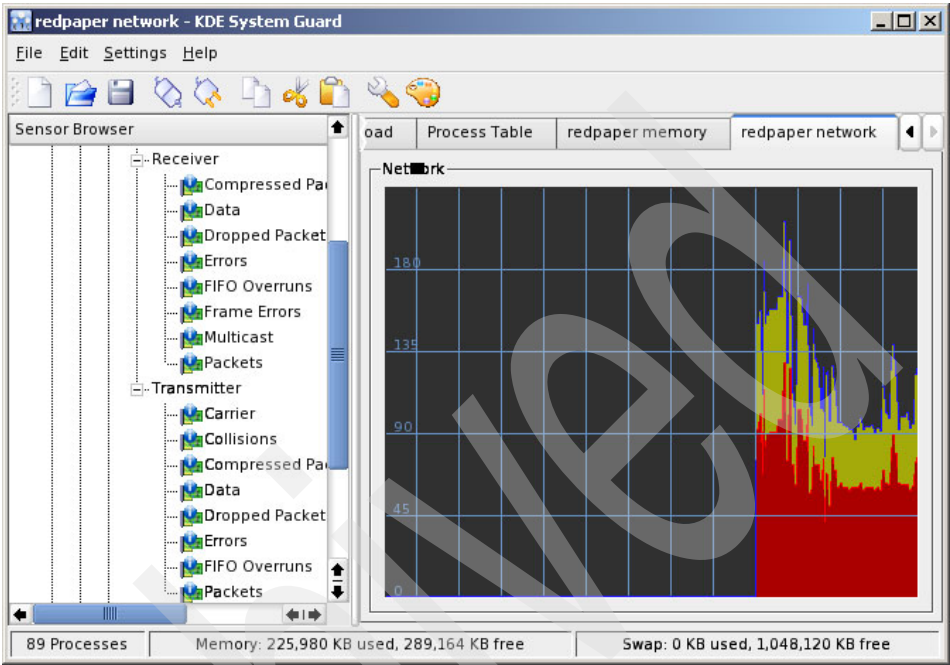


Figure 8-16 KDE System Guard network monitoring

For SUSE Linux, you can also use the **traffic-vis** package, an excellent network monitoring tool. Capture the network data and analyze the results using a Web browser. For more details on this, refer to 8.2.9, “Traffic-vis” on page 200.

Remember that numerous reasons cause performance problems, and sometimes, problems occur simultaneously, making it difficult to pinpoint the origin. The indicators in Table 8-12 provide guidance in determining the problem within the network.

Table 8-12 Indicators for network analysis

Network indicator	Analysis
Packets received Packets sent	Shows the number of packets coming in and going out of the specified network interface. Check both internal and external interfaces.
Collision packets	Collisions occur when many systems exist on the same domain. Using a hub may cause many collisions.

Network indicator	Analysis
Dropped packets	Packets may be dropped for a variety of reasons, but the result may impact performance. For example, if the server network interface is configured to run at 100 Mbps full duplex, but the network switch is configured to run at 10 Mbps, the router may have an ACL filter that drops these packets, for example: <pre>iptables -t filter -A FORWARD -p all -i eth2 -o eth1 -s 172.18.0.0/24 -j DROP</pre>
Errors	Errors occur if the communication lines, for instance, the phone line, are of poor quality. In these situations, resend the corrupted packets. This decreases network throughput.
Faulty adapters	Network slowdowns often result from faulty network adapters. When this kind of hardware fails, it may begin to broadcast junk packets in the network.

## Performance tuning options

If you encounter problems relating to network bottlenecks, perform the following tasks:

- ▶ Ensure that the network card configuration matches the router and switch configurations, for example, the frame size.
- ▶ Modify the way the subnets are organized.
- ▶ Use faster network cards.
- ▶ Tune the appropriate IPV4 TCP kernel parameters. For more details on this and some security-related parameters that improve performance, refer to 8.1, “Tuning the operating system” on page 148.
- ▶ Change the network cards if possible, and recheck the performance.
- ▶ Add network cards and bind them together to form an adapter team, if possible.

# Tuning the existing environment for better performance

This chapter provides information about tuning an existing environment for better performance. It also provides a guide for testing the application servers and components of the total serving environment that should be examined and tuned to increase the performance of the application.

**Note:** Keep in mind that performance tuning results in values that are unique to each environment. The tuning values presented in this chapter apply specifically to the environment used for this book.

This chapter contains the following sections:

- ▶ 9.1, “Testing the performance of an application” on page 224
- ▶ 9.2, “Tools of the trade” on page 225
- ▶ 9.3, “Performance monitoring guidelines” on page 238
- ▶ 9.4, “Performance tuning guidelines” on page 246

## 9.1 Testing the performance of an application

Application performance testing is an important component of the software deployment cycle. It becomes even more important with respect to Web applications, since an end user's tolerance for slow applications is generally much lower than that of a captive internal audience. Performance testing has been the subject of many products and white papers recently, and has spawned a new sector dedicated to Web application testing.

When performance testing a Web application, you must determine several requirements either by interpreting data from an existing application that performs similar work, or from best-guess estimates. These requirements are:

- ▶ User base  
This refers to the number of users who are likely to access the application. This value is generally expressed in hits per month, day, hour, or minutes, depending on the volumes.
- ▶ Total concurrent users  
During a peak interval, this refers to the maximum number of users accessing the application at the same time. You should plan for, expect, and re-evaluate this number on a regular basis.
- ▶ Peak request rate  
This rate refers to the number of pages required to be served per second. You should re-evaluate this rate regularly as well.

The user base, especially for Web applications, is a difficult number to determine, especially if it is for an application that is performing a new function on a Web site. Use existing Web server measurements to provide a *best-guess* number based on current traffic patterns for the site. If you cannot capture these numbers, the best option is to determine the breaking point for the application within the intended deployment infrastructure. This provides the ability to monitor, once the application is live, and to provide increased capacity prior to a negative user response due to load.

When dealing specifically with applications running in WebSphere Application Server, follow certain performance testing protocols. A good example of this is found in the article "Performance Testing Protocol for WebSphere Application Server-based Applications" by Alexandre Polozoff, which is found on the Web at:

[http://www.ibm.com/developerworks/websphere/techjournal/0211\\_polozoff/polozoff.html](http://www.ibm.com/developerworks/websphere/techjournal/0211_polozoff/polozoff.html)

Remember that performance tuning is more of an art than a science. However, performance tuning is an art that strictly follows the recurring, monotonous, trifold process of testing, evaluating, and tuning.

This process requires that you know your system, environment, and application thoroughly, know what you want to test, what goals you want to achieve, and the tools you are going to use for load testing. A small amount of intuition and a feeling for your work will help you find the best configuration easily. Finally, become familiar with the different load testing tools on various environments to gain comprehensive, in-depth knowledge.

It is impossible to tune the environment to reduce the impact of poor coding practices. Also keep in mind that performance tuning is an ongoing process. Continuous performance reviews should be carried out to assure that changes in load, application behavior, and site behavior have not adversely impacted the application performance. Also, there are no hard and fast rules for performance tuning. What may be appropriate tuning for one application may not be appropriate for another. It is more important to understand the concepts associated with tuning, and make adjustments based on the understanding gained from those concepts.

## 9.2 Tools of the trade

The following sections present an overview of a few load testing tools:

- ▶ Mercury LoadRunner
- ▶ Rational® TestStudio®
- ▶ Segue SilkPerformer

In our opinion, these are the most comprehensive performance testing tools, with a very broad and deep functionality and an expensive price. We neither recommend nor endorse any of these tools.

Open source licensed tools may be available that perform just as well as the other tools. Refer to 9.2.2, “ApacheBench” on page 226 and 9.2.3, “OpenSystem Testing Architecture” on page 229, which introduce open-source load testing packages. Then refer to 9.2.4, “Other testing tools” on page 237, for links to additional performance testing products and projects.

## 9.2.1 Web Performance Tools

Web Performance Tools, earlier known as *AKtools*, were developed as an internal project for IBM to provide a low-cost, highly-configurable, load-testing tools for Web sites. However, they are no longer available because their functionality overlapped with that of IBM tools such as:

- ▶ Rational Suite® TestStudio  
<http://www.ibm.com/software/awdtools/suite/>
- ▶ IBM WebSphere Studio Workload Simulator  
<http://www.ibm.com/software/awdtools/studioworkloadsimulator/>

## 9.2.2 ApacheBench

ApacheBench is a simple tool for quickly generating Hypertext Transfer Protocol (HTTP) GET and POST requests. It is a part of the Apache HTTP Server and the IBM HTTP Server powered by Apache. It is automatically installed with the IBM HTTP Server. You can find the executable in *IHS\_HOME\bin\ab*.

To access the manual and the list of options, refer to the following Web site:

<http://httpd.apache.org/docs/1.3/programs/ab.html>

**Note:** The ApacheBench executable is not a part of every IBM HTTP Server distribution. For example, it is not included in the currently available Version 1.3.28, but it is delivered with Version 2.0.42.2. However, you can download the Apache HTTP Server source code from the following Web site and compile the ApacheBench utility yourself:

<http://httpd.apache.org/download.cgi>

ApacheBench was mainly designed as a benchmarking tool. It can be used only for basic load testing because the following limitations:

- ▶ It has no functionality to record a browser click stream. Requests have to be specified on the command line.
- ▶ Although it supports the HTTP POST method, the POST request data has to be put into a file before the request is sent.
- ▶ It neither has support for distributed tests, nor does it provide any graphical representation of the test result data.
- ▶ It cannot be used for scripting scenarios or test cases. It can only test one Uniform Resource Link™ (URL) at a time.

- ▶ It cannot retrieve cookie data from the response, although cookie information can be specified on the command line, making session-aware performance testing possible in a limited manner.

**Attention:** A major limitation of ApacheBench is that it does *not* verify the HTTP response. For example, if your application server returns a “500 Internal Server Error” message in 5 ms for a page request, you may think that your system is very fast. Make sure that you always perform a test run first, with the `-v` (verbose mode) option enabled, to get the response code printed out.

ApacheBench has several advantages, some of which include:

- ▶ It is free, easy and quick to use, and requires no additional setup since it is a part of the IBM HTTP Server.
- ▶ It is useful for simple, ad hoc performance tests, where the goal is, for example, to stress a Web server or application server to obtain a rough idea about the number of requests of the same kind that can be processed per second.

It can be used to stress the application server, creating numerous HTTP sessions at a time, while using Tivoli Performance Viewer to monitor the servers' performance.

**Important:** Exercise caution when selecting the URL to test with ApacheBench. Since only one URL can be specified at a time as the input parameter, make sure that you select the correct URL that you want to test.

Sometimes, a Web page (in our example, the Trade3 main URL `http://your_host/trade/`) provides a detailed display view that is generated in the following sequence:

1. The browser retrieves the response from the server, which only contains a frameset or a redirect.
2. It creates additional requests to receive the actual content, such as images and so on.

ApacheBench is not a browser and does not know about framesets, links, images, or dynamic content like Javascript. It does not follow any redirects. In fact, it does not even resolve simple Hypertext Markup Language (HTML) `<img src>` tags to retrieve images. All this falsifies your performance results if not considered. Following our Trade3 example, use a request URL that returns meaningful data, and not just redirects or framesets, from the application server.

For example, you can use the following URLs:

- ▶ `http://your_host/trade/PingJsp.jsp`
- ▶ `http://your_host/trade/scenario`

## Command line options

The most common and useful options for ApacheBench are:

- |                         |   |
|-------------------------|---|
| <b>-h</b>               | This displays help and usage.   |
| <b>-n requests</b>      | The absolute number of requests to perform for the benchmarking session. The default is 1. Remember that ApacheBench does not implement <i>thinktime</i> . If you set the number of requests high, you will effectively be launching a denial-of-service attack on your server. |
| <b>-c concurrency</b>   | The number of simultaneous requests to perform virtual users. The default is 1, which means no concurrency.   |
| <b>-v level</b>         | This is used to set the verbosity level. Level 2 and higher print HTTP headers and the HTTP response body.  |
| <b>-A username:pass</b> | This is used to supply basic authentication credentials to the server.  |
| <b>-C name=value</b>    | This is used to add a cookie header with name and value to the requested object.  |

The number of requests is not set per concurrent user. If you specify `-n 20` and `-c 10`, each of the 10 virtual users will only perform two requests.

For example, the command for stress testing the Trade3 primitive PingJSP using ApacheBench is:

```
C:\Program Files\IBM HTTP Server 2.0\bin> ab -g test -c 5 -n 50  
"http://cluster.itso.ibm.com/trade/PingJsp.jsp"
```

In our example, we performed 50 requests using five virtual users, with each of the virtual users performing 10 requests.

An ApacheBench stress test returns values such as these:

- ▶ Time taken for tests
- ▶ Requests per second
- ▶ Time per request (mean)
- ▶ Time per request (mean, across all concurrent requests)
- ▶ Transfer rate (Kbytes/sec)
- ▶ Percentage of requests served within a certain time



### 9.2.3 OpenSystem Testing Architecture

The Open System Testing Architecture (OpenSTA) is a distributed software testing architecture based on Common Object Request Broker Architecture (CORBA). OpenSTA is an open source software licensed under the GNU General Public License. It is designed to generate realistic, heavy loads, simulating the activity of thousands of virtual users. This capability is fully realized through OpenSTA's distributed testing architecture.

OpenSTA graphs both virtual user response times and resource utilization information from all server systems under test. This feature facilitates the gathering of precise measurements during load tests and the analysis performed on these measurements.

#### Feature overview

OpenSTA stands for a collection of tools that build on the distributed architecture. These tools allow you to perform scripted HTTP load or stress tests, including performance measurements run from a single machine, or distributed and coordinated across many machines.

Obtain the software package on the Web at:

<http://opensta.org/>

You can find additional information and documentation on OpenSTA on the Web at:

<http://portal.opensta.org/>

The source code is available on the Web at:

<http://opensta.sourceforge.net/>

The current version is OpenSTA V1.4.2 contains the following features:

- ▶ Intelligent script recording: Automatic cookie-based session support in recorded scripts
- ▶ Script modeling in a BASIC like Script Control Language (SCL)
- ▶ Script debugging using single-step mode and breakpoints
- ▶ Modular test-suite creation
- ▶ Support for protocols such as HTTP 1.0, HTTP 1.1, HTTPS
- ▶ Single machine-based load generation
- ▶ Distributed load generation across multiple machines, with one controller
- ▶ Support for controlling load generation over the Internet

- ▶ Extensive data collection, simple statistics graphs provided
- ▶ Export of collected data into comma-separated text files
- ▶ Additional performance measurement data collected through Simple Network Management Protocol (SNMP) and Windows NT® performance monitor
- ▶ Tutorial, user's guide, and detailed online help available
- ▶ Online community and Web-based discussion forum available at:  
<http://portal.opensta.org/>
- ▶ Commercial support and services available
- ▶ Support for load generating platforms such as Windows NT and Windows 2000
- ▶ Manual script modeling, for example, a form-based login, required for easier to use, but more powerful features

### Prerequisites and installation

In our setup, we used the current OpenSTA V1.4.2 to test our sample topology. To view the latest release notes and the download package for the most up-to-date system requirements, visit the following Web site:

<http://www.opensta.org/download.html>

To use OpenSTA, you must meet these software requirements:

- ▶ Windows 2000
- ▶ Windows NT with service pack 5
- ▶ Microsoft Data Access Components (MDAC) V2.5 (minimum)

<http://microsoft.com/data/download.htm>

OpenSTA supports the following Web browsers for script recording:

- ▶ Microsoft Internet Explorer 4, 5, and 6
- ▶ Netscape Navigator 4.7

**Note:** Other browsers, such as Opera, can be used for script recording. However, their proxy settings should be configured manually instead of being configured automatically by the OpenSTA Script Modeler.

You install OpenSTA by running the setup wizard and following the instructions on the panels.

## The OpenSTA architecture

OpenSTA supplies a distributed software testing architecture based on CORBA. This enables the creation and running of tests across a network. For a detailed overview of the architecture and its components, see the *Getting Started Guide* and the *Users Guide* in the documentation section of the OpenSTA Web site:

<http://opensta.org>

Figure 9-1 presents an overview of the OpenSTA distributed testing architecture.

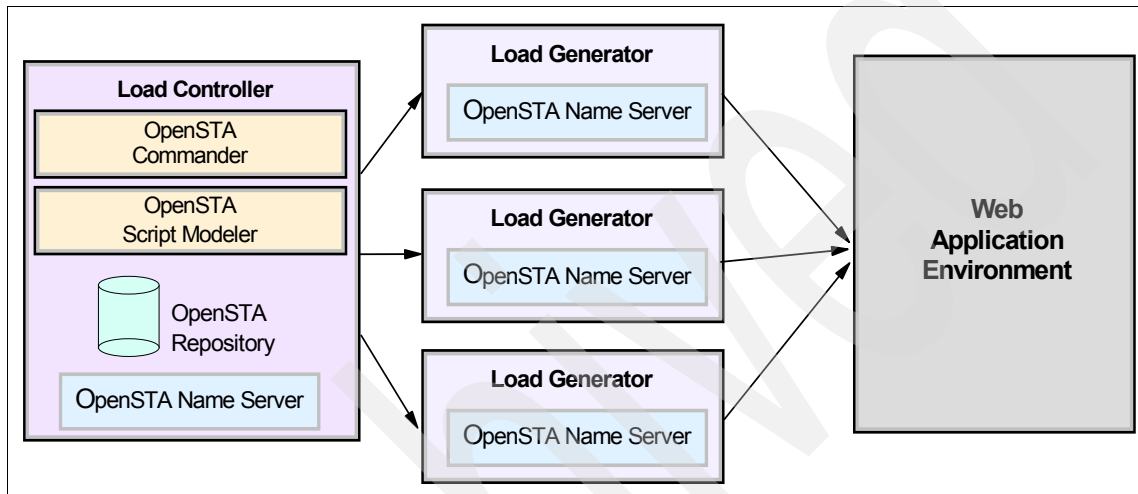


Figure 9-1 Overview of the OpenSTA distributed testing architecture

### OpenSTA Name Server

The OpenSTA Name Server configuration utility is the component that allows you to control your test environment. After installation, you will see the OpenSTA Name Server running, which is indicated by an icon in the Windows task bar. For a distributed test, every load-generating node should have the OpenSTA Name Server installed and configured to point to the controlling node, also called the *repository host*.

### OpenSTA Commander

The Commander is the graphical user interface (GUI) that functions as the front end for all test development and test coordination activity. This is the main component of the OpenSTA that you should run to perform load tests. In the left pane, a view of the OpenSTA repository containing all the defined tests, scripts, and collectors, is displayed.

## OpenSTA Script Modeler

The Script Modeler is the recording and script developing environment of the OpenSTA. It is launched through the Commander when you double-click a script name in the repository tree window.

### Recording the Trade3 script

With the Script Modeler, you can record or develop or do both, and refine your scripts. They form the content of your tests and let you generate the desired Web activity during a load test. They are stored in the repository from where they can be included into multiple tests. Perform these steps to record a new script:

1. Create a new, empty script by selecting **File** → **New Script** → **HTTP**, or by right-clicking the **scripts** node in the repository tree window and selecting **New Script** → **HTTP** from the pop-up menu. Enter the name Trade3Scenario.
2. Launch the Script Modeler by double-clicking the previously created script.
3. Select **Options** → **Browsers** and select the browser that you want to use to create the capture.
4. Select **Capture** → **Record**, or alternately, click the record icon (the red circle) in the toolbar to start recording. The settings that allow automatic cookie handling are enabled by default. For more information about recording options, refer to the online help available in the documentation section on the Web at:

<http://opensta.org>

**Note:** OpenSTA provides a proxy gateway that intercepts all HTTP traffic during recording. Script Modeler automatically configures the browser's proxy setting to point to that gateway, and reset it to the previous values after recording. By default, the proxy gateway uses port 81 on the local host.

5. After the browser window has launched, open the URL:

`http://host_name/trade/scenario`

In our case, it was:

`http://www.itso.ibm.com/trade/scenario`

Do not reload the page since only one request is to be captured. When the page has loaded completely, either close the browser window or switch back to the Script Modeler and select **Capture** → **Stop** from the menu. This terminates the recording session.

**Tip:** To navigate to a specific page on your site first, switch back to the Script Modeler window and click the **Pause** button in the toolbar. This stops the recording temporarily. After navigating to the link you want to start your script with, click the **Pause** button again to resume recording.

6. Save your script and exit Script Modeler.

A simple user session in the Trade3 application is now created. Example 9-1 shows an extract from that script. You can now view, replay, and modify the recorded script inside the Script Modeler. If you change the script manually, use the *compile* icon in the toolbar to find the syntax errors.

*Example 9-1 Excerpt of the recorded OpenSTA script Trade3Scenario*

---

```
Start Timer T_TRADE3SCENARIO

PRIMARY GET URI "http://www.itso.ibm.com/trade/scenario HTTP/1.1" ON 1&
  HEADER DEFAULT_HEADERS&
  ,WITH {"Accept: */*",&
    "Accept-Language: en-us", &
    "Connection: Keep-Alive"}

Load Response_Info Header on 1&
  Into cookie_1_0&
  ,WITH "Set-Cookie,JSESSIONID"

WAIT 6800

!Cookie : JSESSIONID=00011bubQp5N91qYm1b8KyynR58:v4fh5hfu
GET URI "http://www.itso.ibm.com/trade/style.css HTTP/1.1" ON 2  &
  HEADER DEFAULT_HEADERS&
  ,WITH {"Accept: */*",&
    "Referer: http://www.itso.ibm.com/trade/scenario", &
    "Accept-Language: en-us",&
    "Connection: Keep-Alive",&
    "Cookie: "+cookie_1_0}

DISCONNECT FROM 1
DISCONNECT FROM 2
SYNCHRONIZE REQUESTS
End Timer T_TRADE3SCENARIO
```

---

## Randomizing requests using variables

Randomization of requests is easily achieved with variables. You simply need the script editor and some knowledge of the OpenSTA scripting language. For example, you could insert two variables, such as user and password information that get filled with different account data on each request, and build a new HTTP GET string using those variables. The variable information can be read by the scripting engine from a flat file, thus simulating different, concurrently active user sessions.

## Additional Script Modeler features

These additional script modelling features are also useful:

- ▶ **Custom timers:** To evaluate the response time for business transactions, insert manual timers into the scripting code.
- ▶ **Output stream parsing:** To verify the response header and body, manually parse the output stream using the OpenSTA scripting language.

## Performance testing using OpenSTA Commander

Use the Commander to define tests. Each test can consist of one or more task groups, which are sequences of test scripts. Each task group should contain at least one script. When a script is associated with a task group, it is called a *task*.

Test execution settings such as the number of iterations or the number of virtual users (VU) can be specified per task group, along with configuring the number of iterations per task. The modular structure makes the scenario easy to accomplish, where a user logs in first, uses the Trade3 application a configurable number of times, and finally logs out, as shown in Example 9-2.

*Example 9-2 Exemplary modular test design in OpenSTA*

---

```
Test Trade3 will be repeated 10 times, consisting of:
|
|--Task group Trade3_1: 50 VU
|   |--Task-1: T3_Login, repeat count: 1
|   |--Task-2: T3_Scenario-1, repeat count: 500
|   |--Task-3: T3_Scenario-2, repeat count: 250
|   |--Task-4: T3_Logout, repeat count: 1
```

---

To run a performance test using OpenSTA, perform these steps:

1. Create the test and add the previously recorded script:
  - a. In the Commander window, select **File** → **New Test** → **Tests** and enter Trade3 as the name of the new test.
  - b. Drag the script Trade3Scenario from the repository view on the left side onto the Task1 column on the right side of the Commander window. This creates a new task group called *Trade3\_1*.
2. Specify the runtime parameters. You configure the number of iterations for the task and the number of virtual users concurrently performing the scripted task. In this example, we used 40 virtual users performing 10 iterations of the Trade3Scenario task.
  - a. Click **Trade3Scenario** in the column Task1. Change the value in the Number of times each user will run this task (iterations) field to 10.
  - b. Click the field that contains the value 1 in column VUs. Enter 40 in the Total number of virtual users for this task group field.

**Note:** Collectors are an additional OpenSTA Commander feature. They gather operating system performance data such as CPU utilization, network and disk input/output (I/O), and so on, and can be defined in the OpenSTA Commander. These collectors gather data during test execution and can then be correlated with the actual test results. The two supported collector schemes to gather data from are:

- ▶ Performance data available from the Windows System Monitor, that is, Windows Management Instrumentation (WMI)
- ▶ Simple Network Management Protocol data from SNMP-enabled hosts or devices

3. Run the test using the OpenSTA Commander. While OpenSTA Commander is running, you can view the test's progress and statistics. Change the monitoring interval to five seconds. That way the statistics are updated every five seconds. In a real-world scenario, the monitoring interval should be set even higher as to not influence the performance test by wasting CPU and network resources used for collecting performance data.
  - a. Set the monitoring interval to five seconds:
    - i. Select the **Monitoring** tab.
    - ii. Click the Tools symbol.
    - iii. Set both Task Monitoring Interval and Sampling Frequency to 5.

- b. Start the test. Select **Test** → **Execute Test** or click the green Play icon on the toolbar.

During the test, you can watch the progress and test statistics by performing the following tasks:

- i. Switch to Monitoring view by selecting the **Monitoring** register.
  - ii. On the right side of the Commander window, select the **Summary** check box to enable it and to view the following current statistics:
    - Active virtual users
    - Test duration
    - HTTP requests per second
    - Number of successful and failed HTTP requests
  - iii. Right-click inside the Summary window to select and deselect additional statistics data.
4. The test stops automatically after all the 40 VUs have cycled through their 10 iterations of the Trade3Scenario task. View the results of the data that is collected.

Switch to the Results View tab to examine the test data and statistics. You can view the reports and graphs. The most useful reports and graphs are:

- Test Summary, Audit Log, Report Log, and Test Error Log
- HTTP Data List for debugging HTTP responses
- HTTP Response Time (average per second) versus Number of Responses Graph

This graph displays the average response time for requests grouped by the number of requests per second during a test run.

- HTTP Errors versus Active Users Graph

This graph displays the effect on performance measured by the number of HTTP server errors returned, since the number of active virtual users varies during a test run.

- HTTP Responses versus Elapsed Time Graph

This graph displays the total number of HTTP responses per second during the test execution.

- HTTP Response Time versus Elapsed Time Graph

This graph displays the average response time per second of all the requests issued during the test run.

- Timer Values versus Active Users Graph



- Timer Values versus Elapsed Time Graph

The OpenSTA Commander provides two additional features to aid further data analysis:

- Exporting results data

This feature allows you to export every text-based report into a comma separated values (CSV) file. All the graphics reports can be exported directly into Microsoft Excel®, where additional statistical calculations can be carried out. Export the data by right-clicking inside the open report window, selecting **Export**, and saving the CSV file or opening it in Excel.

- URL filtering the report data

This feature is available when you right-click inside a graphics report and selecting **Filter URLs**. You can, for example, filter out URLs, so that only the servlet request to `/trade/scenario` is left. The graph then displays only the response times for the actual dynamic content of the load test.

The most interesting reports are:

- HTTP Data List
- HTTP Response Time versus Elapsed Time
- HTTP Responses versus Elapsed Time

## 9.2.4 Other testing tools

Numerous open-source or commercial products and services relating to Web application testing are available. A search on the Internet for Web Application Testing yields in excess of two million hits. We list some of these commercial tools that are available for free. The list is by no means complete, nor does it imply endorsement or recommendation. It is merely provided as an alternative to ApacheBench and OpenSTA:

- ▶ JMeter (open source software from the Apache Software Foundation)  
<http://jakarta.apache.org/jmeter/>
- ▶ TestMaker and TestNetwork (from PushToTest)  
<http://www.pushtotest.com/>
- ▶ Grinder  
<http://grinder.sourceforge.net>
- ▶ LoadRunner (from Mercury Interactive Corporation)  
<http://www.merc-int.com>

- ▶ IBM Rational Suite TestStudio (from IBM)  
<http://www.ibm.com/software/awdtools/suite/>
- ▶ Segue SilkPerformer  
<http://www.segue.com/products/load-stress-performance-testing/index.asp>
- ▶ WebLOAD (from Radview)  
<http://www.radview.com/default.asp>
- ▶ WebStone (from Mindcraft)  
<http://www.mindcraft.com>
- ▶ OpenLoad (from Opendemand Systems)  
<http://www.opendemand.com/openload/>

## 9.3 Performance monitoring guidelines

Before undertaking the tuning measures, decide which part of the system you want to tune. Simple and random tuning acts seldom give the desired effect, and sometimes even produce bad results. This section discusses the components to examine first, besides providing a practical hotlist of top ten monitors, and the tools to use.

### 9.3.1 Monitoring with Tivoli Performance Viewer and Advisors

To solve the problems pertaining to performance, use Tivoli Performance Viewer and walk through the monitoring checklist. It helps you to find those parts of your system that will gain the most by performance tuning.

Do not expect miracles by adjusting a few knobs inside WebSphere Application Server. Every system and application behaves differently, and requires different tuning measures. But the Performance Monitoring Infrastructure (PMI), Tivoli Performance Viewer, and Performance Advisors assist you in creating a system configured for optimum performance.

#### **Monitoring checklist**

Because of the sheer magnitude of monitors and tuning parameters, knowing where to start, what to monitor, and which component to tune first is difficult. The list helps check the most important counters and metrics of WebSphere Application Server. Refer to Figure 9-2 on page 240 for a graphical overview of the most important resources to check.

Consider the following points:

- ▶ If you identify something out of the ordinary, for example, an overutilized thread pool or a Java virtual machine (JVM) that spends 50% in garbage collection at peak load time, then concentrate your tuning actions on this.
- ▶ Examine the system when it is under typical, production-level load, and make notes of the values observed.
- ▶ Alternately, save Tivoli Performance Viewer sessions to the file system, where the monitoring data will be stored for recurring analysis.
- ▶ Remember that a set of tuning parameters for one application will not work in the same way for another application.

Refer to 9.4, “Performance tuning guidelines” on page 246, for tuning measures related to the following checklist:

- ▶ Servlets and Enterprise Java Beans (EJB)
  1. Average response time
  2. Number of requests per second (Transactions)
  3. Live number of HTTP sessions
- ▶ Thread pools
  4. Web server threads
  5. Web container and EJB container thread pool
  6. Datasource connection pool size
- ▶ Java virtual machine
  7. JVM memory and garbage collection statistics
- ▶ System resources on Web, application, and database servers
  8. CPU utilization
  9. Disk and network I/O
  10. Paging activity

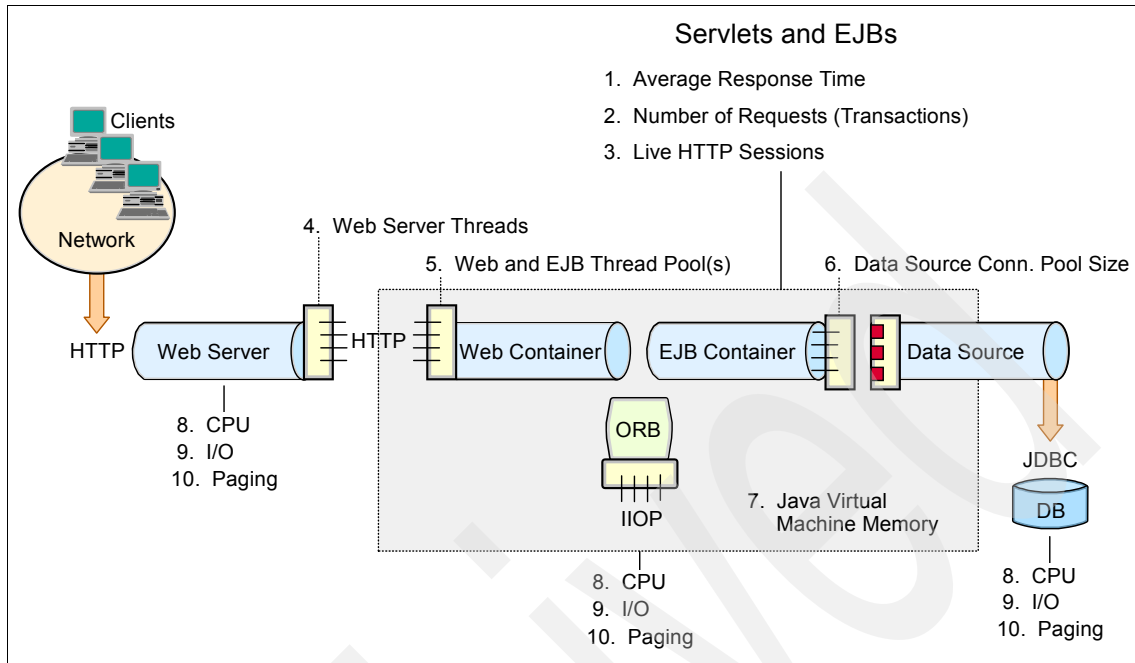


Figure 9-2 Top-ten monitoring items checklist

## Tivoli Performance Advisor

Use Tivoli Performance Advisor together with Tivoli Performance Viewer to help you tune your system, with Tivoli Performance Advisor providing recommendations on inefficient settings. View the recommendations and data from the Performance Advisor by expanding the Performance Advisor icon under Data Collection in Tivoli Performance Viewer, and selecting the server.

For detailed instructions about how to use the Performance Advisor, refer to "Using the Performance Advisor in Tivoli Performance Viewer" in the WebSphere Application Server V5.1 information center on the Web at:

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

**Hint:** Sometimes Tivoli Performance Advisor states that you should set the instrumentation level for JVM to maximum. If see the message despite having done so, activate the Java Virtual Machine Profiler Interface (JVMPi) facility.

## Runtime Performance Advisor

The Runtime Performance Advisor (RPA) service runs inside each application server process, and has to be enabled explicitly. It uses the same rules engine as Tivoli Performance Advisor, and produces the same tuning recommendations. If you experience performance problems, you can enable it and leave it running for a longer period of time even during production, since it has a small impact on performance. This way, you can get important system tuning information and advice even when you are not actively monitoring the system.

For more information about using the Runtime Performance Advisor, refer to “Using the Runtime Performance Advisor” in the WebSphere Application Server information center at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

**Tip:** It is important to configure the number of processors in the Runtime Performance Advisor configuration panel. Failure to do so can lead to inaccurate recommendations.

However, keep in mind that the advice of Tivoli Performance Advisor and RPA can be accurate only if the application is running with production-level load, and without exceptions and errors. This is best done by running a performance test, where a production-level load can be simulated in a reproducible manner. Both Tivoli Performance Advisor and RPA need high CPU utilization to provide advice, and this is best accomplished during a load and stress test.

### 9.3.2 Performance analysis

Web performance analysis describes the process of finding out how well your system, that is, a Web application or Web site, performs. It also pinpoints performance problems caused by inadequate resource utilization, such as memory leaks or over utilized or under utilized object pools, to name a few. After you identify the trouble spots of the system, you can reduce their impact by implementing the appropriate tuning actions.

#### Terminology

The following sections define the three most important concepts used in performance analysis.

##### **Load**

A Web site, especially the application running behind it, performs differently, depending on its current *load*, that is, the number of users concurrently using the Web site at the same time. This includes clients who actively perform requests at a time and clients who are currently reading a previously requested Web page.

*Peak load* refers to the maximum number of concurrent users using the site at some point in time.

### **Throughput**

A Web site can only handle a specific number of requests in parallel. *Throughput* depends on that number and on the average time a request takes to process. It is measured in requests per second. If the site can handle 100 requests in parallel and the average request takes one second, the Web site's throughput is 100 requests per second.

### **Response Time**

*Response Time* refers to the time frame between a client initiating a request and receiving a response. Typically, the time taken to display the response, usually the HTML data inside the browser window, is also accounted for in the response time.

### **Performance testing**

To determine the level of your system's performance, run a performance test to obtain an idea about the number of users who can access the site at the same time without noteworthy delays. Load tests are also helpful in determining the level of performance improvement that follows a specific tuning measure. After each tuning step, repeat the load test, sometimes 10 to 15 iterations, until the system is tuned for optimal performance.

For performance assessment and tuning, keep in mind the following test requirements:

- ▶ The load expected on the system should be definable.  
This implies that you have to create a model of your expected real-world, production-level workload, based on your experience and knowledge about your clients' behavior. Using this representative model in combination with a stress test tool, create as many test cases as needed, and combine them into a test suite that simulates the desired user behavior and performs the necessary interactions with your site during a load test. In some cases, when no good estimate of such a model can be given, even a crude approximation is better than performing no load test at all, and it can be refined for future test cycles.
- ▶ The load test should be repeatable.  
Tuning without the ability to perform repeatable tests is difficult due to lack of comparable performance data. For it to be repeatable, it is necessary to restore the initial system state after each test iteration. Usually, persistent application or transaction data is stored in databases or backend systems. This implies that a second, staging database or backend system or both have

to be used for testing purposes. Changes to the data on that system will not have consequences in the real world, that is, placing a thousand orders in an online shop during load testing will not activate the order fulfillment process in the backend.

- Find the optimum load level.

The goal is to find the site's saturation point. First, the system should be driven over the point where performance begins to degrade. This is commonly referred to as *drawing the throughput curve*, and is demonstrated in Figure 9-3.

Start a series of tests to plot the throughput curve. Begin testing with wide open server queues or pools to allow maximum concurrency in the system. Record the throughput, that is, the average requests per second, and the response time, that is, the average time for requests, increasing the concurrent user load after each test.

Figure 9-3 displays the system's saturation point, where the throughput becomes constant, that is, when the maximum throughput point is reached, but the response time begins to grow proportionally with the user load.

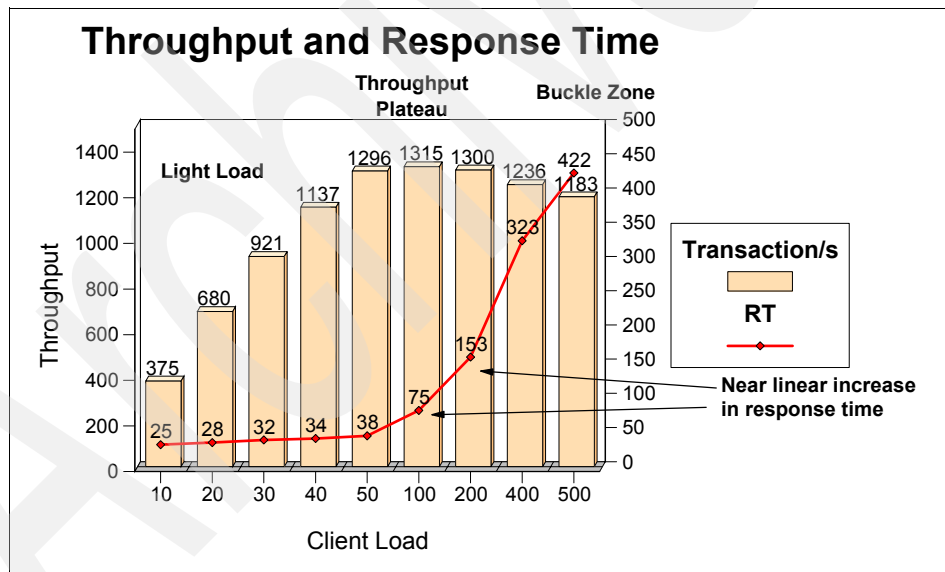


Figure 9-3 Finding the optimum load level for a system: the saturation point

Refer to 9.4.4, “Adjusting WebSphere Application Server system queues” on page 248, for more information about the queuing network and “Determining optimum queue sizes” on page 251 for detailed explanation about the steps involved in drawing the throughput curve.

**Note:** Here, the goal is to drive CPU utilization to nearly 100%. If you cannot reach that state with opened-up queues, it is likely that there is a bottleneck in your application or in your network connection, a hard limit on a resource, or an external component that you did not modify.

## The monitoring-tuning-testing cycle

Perform an initial load test to get a baseline you can refer to later, with the Performance Advisor facility enabled. Use the number of users at your saturation point as the load parameter for all future load tests. Check the monitors in Tivoli Performance Viewer according to the checklist and the Performance Advisors. After implementing an advisor's recommendation, perform another test cycle to determine whether you have gained throughput because of a decreased average response time, or freed resources such as memory, by reducing thread pools, which you could then spend on other components that need more memory resources.

Repeat this procedure to find the optimum configuration for your system and application. Performance tuning is an iterative process. Therefore, do not rely on results from a single run.

The following steps are necessary to perform a load test for tuning purposes, with Tivoli Performance Advisor:

1. Enable the JVMPI facility for the application server.
2. Enable the performance monitor interface service in the application server, and Node Agent if you are using a WebSphere Network Deployment environment, and restart both.
3. Start Tivoli Performance Viewer and set the monitoring levels to Standard.

**Note:** Some JVM rules need the instrumentation level set to maximum. The Tivoli Performance Advisor prints a message if it needs a higher monitoring level.

4. Simulate your representative production-level load using a stress test tool.
  - a. Make sure that there are no errors or exceptions during the load test.
  - b. Record throughput and average response time statistics to plot a curve at the end of all testing iterations.
5. Check Tivoli Performance Advisor and apply advice and follow your intuition.
  - a. Restart components or services, if necessary.
  - b. Reset all components, for example, the database, to the initial state.



6. Retest and repeat the steps from step 4.

### **Best practices for data capture**

To obtain accurate results, follow these best practices for data capturing:

- Measure during steady-state of the load test

Do not include ramp-up or ramp-down times in your performance data measurements and analysis (refer to Figure 9-4). Measure during the steady-state when the maximum number of users are concurrently performing requests.

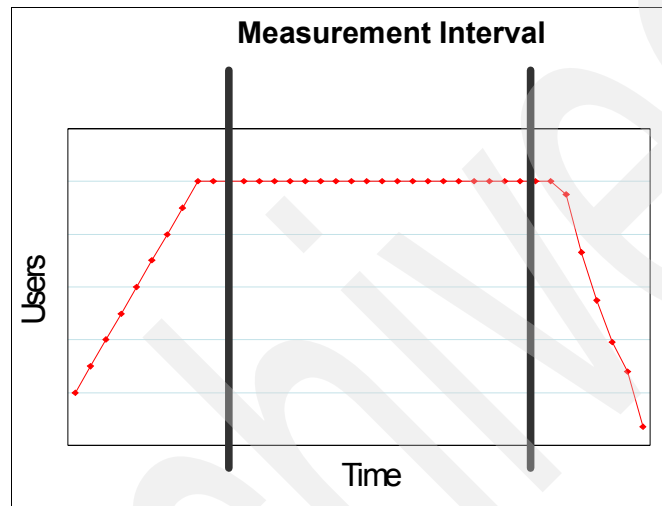


Figure 9-4 Measurement interval: Concurrently active users versus lapsed time

- Monitor machine logs and resources

Monitor important log files for exceptions or errors. Ensure that there are no exceptions or deadlocks in the database. Keep an eye on system resources such as memory, paging activity, CPU, disk IO, network utilization, socket states, and so on for bottlenecks.

The important log files are SystemOut.log and SystemErr.log. Monitor these logs to make sure your application runs without errors. SystemErr.log should typically remain free of entries. Errors logged there *must* be solved before you can capture meaningful performance data. Likewise, any sort of exception in SystemOut.log during the performance run should be solved before another run, because exception handling and the I/O necessary to write stacks to the log are expensive operations that impact performance.

For a Network Deployment cluster, you need not repeat all the monitoring steps for each cluster member if they are all set up identically. Monitoring one

or two representative cluster members is sufficient. However, it is essential to check the CPU statistics for each node to make sure that all cluster member processes are using similar amounts of CPU and there are no extra processes consuming CPU on any nodes that can interfere with the application server's CPU efficiency.

- ▶ Test on quiet systems

Avoid testing during database backups or maintenance cycles.

- ▶ Use isolated networks

Whenever possible, load driving machines should share the same network switch or router or both as your Web application servers, to rule out additional network latency and network delays.

- ▶ Performance tuning is an iterative process

Ten to fifteen test runs are quite usual during the tuning phase. Perform long-lasting runs to detect resource leaks, for example, memory leaks, where the load-tested application runs out of heap space only after a given time.

## 9.4 Performance tuning guidelines

Tuning is about using resources to their fullest potential, resulting in the fastest request processing possible. Many components in WebSphere Application Server have an impact on performance and tuning is highly application-dependent. This section discusses how to identify bottlenecks, gives a practical introduction into analyzing them, and finally gives recommendations on settings for major WebSphere environment properties.

Remember that *performance tuning is not an exact science*. Factors that influence testing vary from application to application, and also from platform to platform. This section is designed to provide a primer for you in a way that describes areas that can be tuned to increase performance.

You can find the latest performance tuning information in the WebSphere Application Server Information Center, on the Web at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

When you reach the information center, navigate to the Tuning section in the contents pane on the left.

## 9.4.1 Crucial tuning parameters

The following list of tuning suggestions is a subset of the complete list that appears in later sections in this chapter. These parameters are important since they have a crucial role to play on performance. The majority of these parameters are application-dependent, and should be adjusted only after close inspection of the testing results for the application in question.

- ▶ Hardware and capacity settings (refer to 9.4.3, “Hardware and capacity settings” on page 248)
- ▶ Java Virtual Machine heap size (refer to 9.4.6, “Java tuning” on page 274)
- ▶ Application assembly performance checklist (refer to 9.4.5, “Application assembly performance checklist” on page 266)
- ▶ Data sources connection pool and prepared statement cache (refer to “Data sources” on page 254)
- ▶ Pass by value versus Pass by reference (refer to “Pass by value versus Pass by reference (com.ibm.CORBA.iiop.noLocalCopies)” on page 309)
- ▶ IBM HTTP Server access logs (refer to “Access logs” on page 295)
- ▶ HTTP keep-alive connections (refer to “MaxKeepAliveConnections” on page 260)
- ▶ HTTP transport custom properties (refer to “HTTP transport custom properties” on page 263)
- ▶ Transaction logs (refer to 9.4.14, “Transaction service settings: Transaction log” on page 311)

## 9.4.2 Parameters to avoid failures

The following list of parameters is a subset of the entire list of parameters, and is designed to minimize application failures. A majority of these are application-specific and should be adjusted based on observed application execution and configuration.

- ▶ Number of connections to DB2 (refer to “Data sources” on page 254)
- ▶ Allow thread allocation beyond maximum (refer to “Thread pool” on page 259)
- ▶ Using TCP Sockets for DB2 on Linux (refer to Chapter 11, “Performance tuning of DB2 UDB Workgroup Server” on page 321)
- ▶ Connection Pool Size (refer to “Connection pool size” on page 254)

### 9.4.3 Hardware and capacity settings

Hardware and its capacity play an important role in performance. This book provides detailed information about how to set up a scalable WebSphere environment.

The following parameters include considerations for selecting and configuring the hardware on which the application servers can run:

- ▶ 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, use extensive messaging, or are processing workflow. Using disk I/O subsystems that are optimized for performance, for example Redundant Array of Independent Disks (RAID) array, is essential for optimum application server performance in these environments. We recommend that you spread the disk processing across as many disks as possible to avoid contention issues that typically occur with one or two disk systems.

- ▶ Processor speed

Increasing the processor speed often helps throughput and response times, once the other bottlenecks have been removed.

- ▶ System memory

Increasing the memory to prevent the system from paging memory to disk, improves performance. Allow a minimum of 256 MB of memory for each processor. We recommend 512 MB. Adjust the available memory when the system is paging and processor utilization is low because of the paging.

- ▶ 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. Also, make sure that 100 MB is in use on 10/100 Ethernet networks.

### 9.4.4 Adjusting WebSphere Application Server system queues

WebSphere Application Server establishes a queuing network, which is a group of interconnected queues that represent various components. Queues are established for the network, Web server, Web container, Enterprise JavaBeans (EJB) container, Object Request Broker (ORB), data source, and possibly a connection manager to a custom back-end system. Each of these resources represent a queue of requests waiting to use that resource.

Queues are load-dependent resources. As such, the average response time of a request depends on the number of concurrent clients. For example, think of an application consisting of servlets and EJBs that accesses a back-end database. Each of these application elements resides in the appropriate WebSphere component, for example, servlets in the Web container, and each component can handle a certain number of requests in a given time frame.

A client request enters the Web server and travels through WebSphere components to provide a response to the client. Figure 9-5 illustrates the processing path this application takes through the WebSphere components as interconnected pipes that form a large tube.

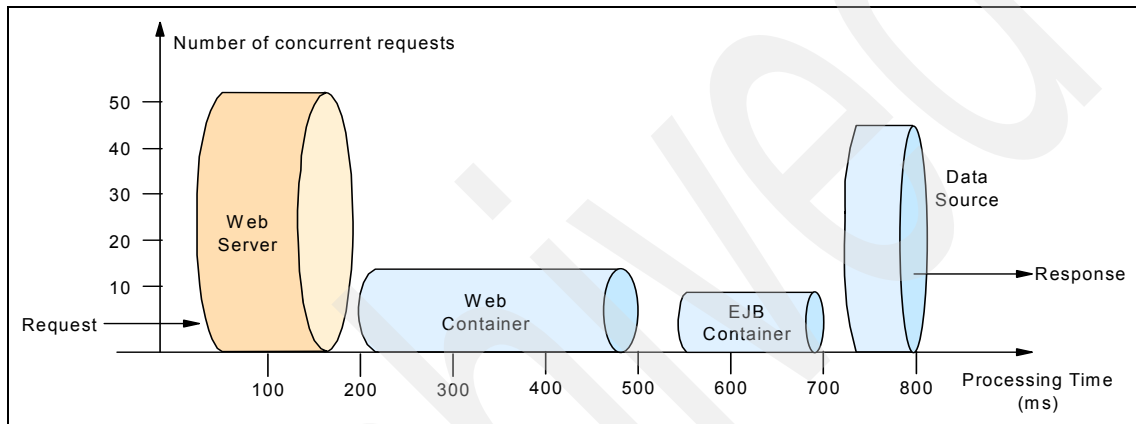


Figure 9-5 Queuing network

The width of the pipes, illustrated by height, represents the number of requests that can be processed at any given time. The length represents the processing time taken to provide a response to the request. To find processing bottlenecks, calculate a transactions per second (tps) ratio for each component. Ratio calculations for a fictional application is shown in Example 9-3.

#### Example 9-3 Transactions per second ratio calculations

The Web Server can process 50 requests in 100 ms =  $\frac{50\text{req}}{0.1\text{s}} = 500\text{tps}$

The Web container parts can process 18 requests in 300 ms =  $\frac{18\text{req}}{0.3\text{s}} = 60\text{tps}$

The EJB container parts can process 9 requests in 150 ms =  $\frac{9\text{req}}{0.15\text{s}} = 60\text{tps}$

The datasource can process 40 requests in 50 ms =  $\frac{40 \text{ req}}{0.05 \text{ s}} = 800 \text{ tps}$

---

Example 9-3 shows that the application elements in the Web container and in the EJB container process requests at the same speed. Nothing can be gained from increasing the processing speed of the servlets or Web container or both because the EJB container would still handle only 60 transactions per second. The requests normally queued at the Web container would shift to the queue for the EJB container.

This example illustrates the importance of queues in the system. Looking at the operations ratio of the Web and EJB containers, each is able to process the same number of requests over time. However, the Web container could produce twice the number of requests than the EJB container at any given time. In order to keep the EJB container fully utilized, the other half of the requests must be queued.

It is common for applications to have more requests processed in the Web server and Web container than by EJBs and back-end systems. As a result, the queue sizes are progressively smaller moving deeper into the WebSphere components. This is one of the reasons why queue sizes should not depend solely on operation ratios.

“Queuing before WebSphere” on page 250 outlines a methodology for configuring the WebSphere Application Server queues. You can change the dynamics of an individual system by moving resources, for example, moving the database server on to another machine, or providing more powerful resources, for example, a faster set of CPUs with more memory. Thus, adjustments to the tuning parameters are for a specific configuration of the production environment.

### **Queuing before WebSphere**

The first rule of tuning is to minimize the number of requests in WebSphere Application Server queues. In general, requests should wait in the network in front of the Web server, rather than the WebSphere Application Server. This configuration allows only those requests that are ready to be processed to enter the queuing network. To do this, specify that the queues furthest upstream, that is, closest to the client, are slightly larger, and that the queues further downstream, that is, furthest from the client, are progressively smaller.

For example, the queuing network becomes progressively smaller as work flows downstream. When 200 client requests arrive at the Web server, 125 requests remain queued in the network because the Web server is set to handle 75 concurrent clients. As the 75 requests pass from the Web server to the Web

container, 25 remain in queue in the Web server and the remaining 50 are handled by the Web container. This process progresses through the data source until 25 user requests arrive at the final destination, that is, the database server.

Since there is work waiting to enter a component at each point upstream, no component in this system should wait for work to arrive. The bulk of the requests wait in the network, outside the WebSphere Application Server. This type of configuration adds stability because no component is overloaded. The edge server components can be used to direct waiting users to other servers in a WebSphere Application Server cluster.

**Note:** If resources are readily available on the application server or database server, tune the system in such a way that every request from the Web server has an available application server thread, and every application server thread has an available database connection. The need for this type of configuration depends on the application and overall site design.

### Determining optimum queue sizes

A simple way to determine the right queue size for any component is to perform a number of load runs against the application server environment at a time when the queues are very large, ensuring maximum concurrency through the system.

One approach involves the following steps:

1. Set the queue sizes for the Web server, Web container, and data source to an initial value, for example, 100.
2. Simulate a large number of typical user interactions entered by concurrent users in an attempt to fully load the WebSphere environment. In this context, “concurrent users means simultaneously active users who send a request, wait for a response, and immediately resend a new request on response reception, without thinktime.

Use any stress tool such as OpenSTA to simulate this workload, as discussed in 9.1, “Testing the performance of an application” on page 224, or the tools mentioned in 9.2.4, “Other testing tools” on page 237.

3. Measure the overall throughput and determine at what point the system capabilities are fully stressed, that is, the saturation point.
4. Repeat the process, increasing the user load each time. After each run, record the throughput, that is, requests per second, and response times, that is, seconds per request, and plot the throughput curve.

The throughput of WebSphere Application Server is a function of the number of concurrent requests present in the total system. At some load point, congestion takes place due to a bottleneck, and throughput increases at a much lower rate

until it reaches a saturation point, that is, the maximum throughput value. The throughput curve should help you identify this load point.

It is better to reach the saturation point by driving CPU utilization close to 100%, since this gives an indication that a bottleneck is not caused by something in the application. If the saturation point occurs before system utilization reaches 100%, there is likely another bottleneck that is being aggravated by the application. For example, the application might be creating Java objects causing excessive garbage collection bottlenecks in Java.

**Note:** There are two ways to manage application bottlenecks, either removing the bottleneck or replicating the bottleneck. The best way to manage a bottleneck is to remove it. Use a Java-based application profiler such as WebSphere Studio Application Developer, Performance Trace Data Visualizer (PTDV), Optimizelt, JProbe, or Jinsight to examine overall object utilization.

The most manageable type of bottleneck occurs when the server CPU becomes fully utilized. To fix this type of bottleneck, add more or more powerful CPUs.

Figure 9-6 shows an example throughput curve.

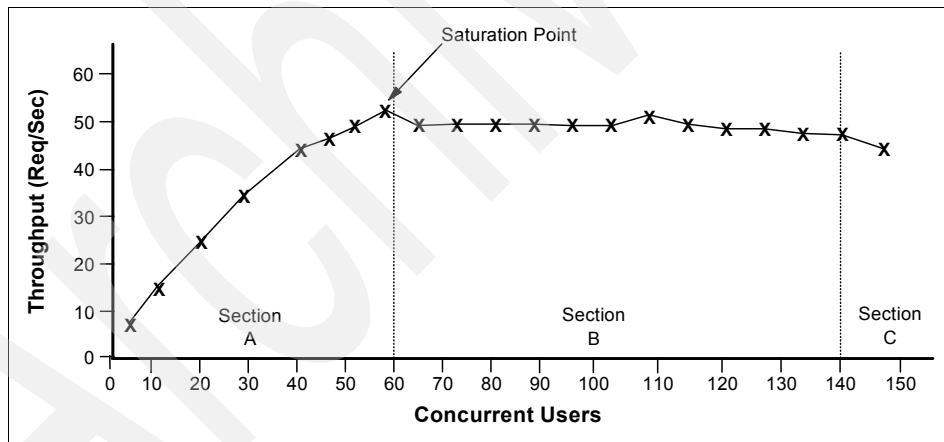


Figure 9-6 Throughput curve

In Figure 9-6, Section A contains a range of users representing a light user load. The curve in this section illustrates that as the number of concurrent user requests increase, the throughput increases almost linearly with the number of requests. You can interpret this to mean that at light loads, concurrent requests face little congestion within the WebSphere Application Server system queues.



In the heavy load zone or Section B, as the concurrent client load increases, throughput remains relatively constant. However, the response time increases proportionally to the user load. That is, if the user load is doubled in the heavy load zone, the response time doubles.

In Section C or the buckle zone, one or more of the system components have become exhausted and throughput starts to degrade. For example, the system could enter the buckle zone when the network connections at the Web server exhaust the limits of the network adapter or if the requests exceed operating system limits for file handles.

### **Determining the maximum concurrency point**

The number of concurrent users at the throughput saturation point represents the maximum concurrency of the application. For example, if the application saturated the application server at 50 users, 48 users could give the best combination of throughput and response time. This value is called the *Max Application Concurrency value*.

Max Application Concurrency becomes the preferred value for adjusting the WebSphere Application Server's system queues. It is desirable for most users to wait in the network. Therefore, queue sizes should decrease when moving downstream, that is, farther from the client. For example, given a Max Application Concurrency value of 48, start with system queues at the values of 75 for Web server, 50 for Web container, and 45 for data source. Perform additional tests, adjusting these values slightly higher and lower to find the best settings.

Use the Tivoli Performance Viewer to determine the number of concurrent users through the Servlet Engine Thread Pool Concurrently Active Threads metric. For further details, refer to "Monitoring performance with Tivoli Performance Viewer" in the WebSphere Application Server information center on the Web at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

### **Adjusting queue settings for access patterns**

In many cases, only a fraction of the requests passing through one queue enters the next queue downstream. In a site with many static pages, many requests are fulfilled in the Web server and do not go to the Web container. Under these circumstances, the Web server queue is significantly larger than the Web container queue.

In the previous section, we saw that the Web server queue was set to 75, instead of being set closer to the value of Max Application Concurrency. Similar adjustments should be made when different components have different execution times. However, as the percentage of static content decreases, a significant gap in the Web server queue and the application server queue create

poorly performing sites. Remember that different Web sites have different requirements.

For example, in an application that spends 90% of its time in a complex servlet and only 10% making a short Java Database Connectivity (JDBC™) query, on an average, 10% of the servlets use database connections at any time. Therefore, the database connection queue is significantly smaller than the Web container queue. Conversely, if much of the servlet execution time is spent making a complex query to a database, increase the queue values at both the Web container and the data source. Always monitor the CPU and memory utilization for both the WebSphere Application Server and database servers to ensure that the CPU or memory is not being saturated.

## **Configuring the queues**

Within WebSphere Application Server, the queues are represented as pooled resources, for example, thread pools or database connection pools. Pool settings determine the maximum concurrency level of a resource. The Data sources section describes how the different queues are represented in WebSphere, and details their settings.

Since the queues are most easily tuned from the inside out of the WebSphere Application Server environment, the Data sources section describes the queue properties in this order, that is, looking at the components from right to left, as shown in Figure 9-5 on page 249.

## **Data sources**

Consider the following settings while determining data source queues:

- ▶ Connection pool size
- ▶ Prepared statement cache size

### ***Connection pool size***

When accessing a database, the initial database connection is an expensive operation. WebSphere Application Server supports the JDBC V2.0 Standard Extension application programming interface (API) to provide support for connection pooling and connection reuse. Connection pool is used for direct JDBC calls within the application, as well as for enterprise beans using the database.

Tivoli Performance Viewer helps find the optimal size for the connection pool. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings. Watch the Pool Size, Percent Used, and Concurrent Waiters counters of the data source entry under the JDBC Connection Pools module. The optimal value for the pool size is that which reduces the values for

these monitored counters. If Percent Used is consistently low, consider decreasing the number of connections in the pool.

Better performance is generally achieved if the value for the connection pool size is set lower than the value for the Max Connections in the Web container. Lower settings for the connection pool size, that is, 10 to 30 connections, typically perform better than higher, that is, more than 100, settings.

On UNIX platforms, a separate DB2 process is created for each connection. These processes quickly affect performance on systems with low memory, causing errors.

Each entity bean transaction requires an additional connection to the database, to specifically handle the transaction. Take this into account when calculating the number of data source connections.

The connection pool size is set from the Administrative Console when you perform the following steps:

1. From the console navigation bar, select **Resources** → **JDBC Providers**.
2. Select the appropriate scope (cell, node, or server), depending on the configuration.
3. Click the name of the provider to open the JDBC provider configuration.
4. In the Additional Properties pane, select the **Data Sources** entry.
5. Click the data source name to open the data source configuration.
6. In the Additional Properties pane of the workspace, select the **Connection Pool** entry.
7. Use the Min connections and Max connections fields to configure the pool size.
8. Save the configuration and restart the affected application servers for the changes to take effect.

**Note:** The default values are one for Min connections and ten for Max connections.

A deadlock can occur if the application requires more than one concurrent connection per thread, and the database connection pool is not large enough for the number of threads. If each application thread requires two concurrent

database connections, the number of threads is equal to the maximum connection pool size.

A deadlock can occur when both the scenarios listed here are true:

- ▶ Each thread has its first database connection, and all are in use.
- ▶ Each thread is waiting for a second database connection, and none will be available, since all the threads are blocked.

To prevent a deadlock in this case, the value set for the database connection pool should be at least one higher than the number of waiting threads in order to have at least one thread complete its second database connection.

To avoid deadlock, code the application to use, at the most, one connection per thread. If the application is coded to require *C* concurrent database connections per thread, the connection pool must support at least the following number of connections, where *T* is the maximum number of threads:

$$T \times (C - 1) + 1$$

The connection pool settings are directly related to the number of connections the database server is configured to support. If the maximum number of connections in the pool is raised, and the corresponding settings in the database are not raised, the application fails and Structured Query Language (SQL) exception errors are displayed in the stderr.log file.

### ***Prepared statement cache size***

The data source optimizes the processing of prepared statements to help make the SQL statements process faster. Configure the cache size of the data source to gain optimal statement execution efficiency.

A *prepared statement* is a precompiled SQL statement that is stored in a prepared statement object. This object is used to efficiently execute the given SQL statement multiple times. If the JDBC driver specified in the data source supports precompilation, the creation of the prepared statement sends the statement to the database for precompilation. Some drivers may not support precompilation and the prepared statement may not be sent until the prepared statement is executed.

If the cache is not large enough, useful entries will be discarded to make room for new entries. In general, the more prepared statements the application has, the larger the cache should be. For example, if the application has five SQL statements, set the prepared statement cache size to five, so that each connection has five statements.

Tivoli Performance Viewer helps tune this setting, thereby minimizing cache discards. Use a standard workload that represents a typical number of incoming client requests, a fixed number of iterations, and a standard set of configuration settings. Watch the PrepStmt Cache Discard counter of the JDBC Connection Pools module. The optimal value for the statement cache size is the setting used to get either a value of zero or the lowest value for PrepStmt Cache Discards.

As with the connection pool size, the statement cache size setting requires resources in the database server. Specifying too large a cache can have an impact on the database server's performance. We recommend that you consult your database administrator to determine the best setting for the prepared statement cache size.

**Note:** The statement cache size setting defines the maximum number of prepared statements cached per connection. This is different from WebSphere Application Server V4, where this was specified per container.

Set the cache size from the Administrative Console by performing the following steps:

1. From the console navigation pane, select **Resources** → **JDBC Provider**.
2. From the list of JDBC providers, select the name of the provider.
3. In the Additional Properties pane, select the **Data Sources** entry.
4. Select the name of the data source.
5. Use the Statement Cache Size field to configure the total cache size.
6. Save the configuration and restart the affected application servers for the change to take effect.

## Enterprise JavaBeans container

After the EJB container is deployed, use the following parameters to make the necessary adjustments for improving performance.

### ***Cache settings (Cache size and Cleanup interval)***

To determine the cache's absolute limit, multiply the number of enterprise beans active in any given transaction by the total number of concurrent transactions expected. Then add the number of active session bean instances.

Use the Tivoli Performance Viewer to view bean performance information. The cache settings comprise two parameters:

- Cache size

The cache size specifies the number of buckets in the active instance list within the EJB container. The default value for cache size is 2053.

► Cleanup interval

The cleanup interval specifies the interval at which the container attempts to remove unused items from the cache to reduce the total number of items to the value of the cache size. The default value for cleanup interval is 3000 milliseconds.

To change these settings, from the Additional Properties pane, select **Servers** → **Application Servers** → *your servername* → **EJB Container** → **EJB Cache Settings**.

### ***Object Request Broker thread pool size***

Method invocations to enterprise beans are only queued for requests coming from remote clients going through the Remote Method Invocation (RMI) activity service. An example of such a client is an EJB client running in a separate Java Virtual Machine, that is, another address space, from the enterprise bean. In contrast, no queuing occurs if the EJB client, either a servlet or another enterprise bean, is installed in the same JVM that the EJB method runs on and the same thread of execution as the EJB client.

Remote enterprise beans communicate by using the RMI/Internet Inter-ORB Protocol (IIOP). Method invocations initiated over RMI/IIOP are processed by a server-side ORB. The thread pool acts as a queue for incoming requests. However, if a remote method request is issued and no more threads are available in the thread pool, a new thread is created. After the method request is completed, the thread is destroyed. Therefore, when the ORB is used to process remote method requests, the EJB container is an open queue, due to the use of unbounded threads.

Tivoli Performance Viewer helps tune the ORB thread pool size settings. Use a standard workload representing a typical number of incoming client requests, use a fixed number of iterations, and a standard set of configuration settings. Watch the Percent Maxed counter of the ORB submodule of the Thread Pools module. If the value of this counter is consistently in double digits, then the ORB could be a bottleneck. Under such conditions, the number of threads in the pool should be increased.

The degree to which the ORB thread pool value should be increased is a function of the number of simultaneous servlets, that is, clients, calling enterprise beans, and the duration of each method call. If the method calls are longer or the applications spend a lot of time in the ORB, consider making the ORB thread pool size equal to the Web container size. If the servlet makes only short-lived or quick calls to the ORB, the servlets can potentially reuse the same ORB thread.

In this case, the ORB thread pool can be small, perhaps even one-half of the thread pool size setting of the Web container.

To configure the ORB thread pool size from the Administrative Console, perform these steps:

1. From the console navigation pane, select **Servers** → **Application Servers**.
2. From the list of application servers in the workspace, select the name of the application server.
3. In the Additional Properties pane of the workspace, select the **ORB Service** entry.
4. In the Additional Properties pane of the workspace, select **Thread Pool**.
5. Use the Maximum Size field to configure the maximum pool size. This only affects the number of threads held in the pool. The actual number of ORB threads can be higher.
6. Save the configuration and restart the affected application server for the change to take effect.

You can also tune some additional settings related to ORB. To learn more about these additional settings, refer to 9.4.12, “Object Request Broker” on page 309.

### ***Break container-managed persistence enterprise beans into several enterprise bean modules during assembly***

To increase performance, break container-managed persistence (CMP) enterprise beans into several enterprise bean modules during assembly. To improve the load time for hundreds of beans, distribute the beans across several Java archive (JAR) files and packaging them to an enterprise archive (EAR) file. This is faster when the administrative server attempts to start the beans, for example, eight to ten minutes versus more than one hour, when one JAR file is used.

## **Web container**

To route servlet requests from the Web server to the Web containers, a transport queue between the Web server plug-in and each Web container is established. The number of client requests accepted by the container is determined by the Web container thread pool. Connection reuse is another factor that influences the number of concurrent threads processed by the Web container.

### ***Thread pool***

The Web container maintains a thread pool to process inbound HTTP(S) requests for resources in the container, that is, servlets and JavaServer Pages (JSPs). This is a closed queue, since the thread pool is bounded by the maximum thread size.

Tivoli Performance Viewer helps tune the Web container's thread pool size settings. Use a standard workload representing a typical number of incoming client requests, a fixed number of iterations, and a standard set of configuration settings. Watch the Percent Maxed and Active Threads counters of the Web container submodule of the Thread Pools module.

If the value of the Percent Maxed counter is consistently in double digits, the Web container could be a bottleneck. In such a situation, increase the number of threads. If the number of active threads are significantly lower than the number of threads in the pool, lower the thread pool size for a performance gain.

**Note:** For Linux systems, the recommended value for the Web container maximum thread pool size used to be 25. However, new Linux kernel versions might support a higher value. The WebSphere performance team is currently working on a list of Linux distributions/kernel versions that support higher values. Monitor the WebSphere Application Server information center for updates on this:

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

To configure the Web container thread pool size, perform the following steps:

1. From the console navigation pane, select **Servers** → **Application Servers**.
2. From the list of application servers in the workspace, select the name of the application server.
3. In the Additional Properties pane, select the **Web container** entry.
4. In the same pane, select the **Thread Pool** entry.
5. Use the Maximum Size field to configure the maximum pool size. Note that in contrast to the Object Request Broker (ORB), the Web container only uses threads from the pool. Therefore, a closed queue. The default value is 50.
6. Save the configuration and restart the affected application server for the change to take effect.

**Important:** Checking the Growable Thread Pool box in the Thread Pool Configuration page allows for an automatic increase in the number of threads beyond the maximum size configured for the thread pool. As a result, the system could become overloaded because too many threads are allocated.

### ***MaxKeepAliveConnections***

The *MaxKeepAliveConnections* 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 impacted because the Web server plug-in has to open a new connection for each request instead of sending multiple requests through one connection.

The **netstat** command utility helps tune the maximum keep-alive connections setting. Use a standard workload representing a typical number of incoming client requests, a fixed number of iterations, and a standard set of configuration settings. Watch the number of connections in the `TIME_WAIT` state to the application server port.

If the count of `TIME_WAIT`s is consistently in double digits, improve performance by raising the maximum keep-alive connections or maximum keep-alive request properties, that are described in “MaxKeepAliveRequests” on page 262. Use the following command, based on your platform, to retrieve the count of `TIME_WAIT`s:

- ▶ On the Windows platform  

```
netstat -na | find /i "time_wait" | find /c "9080"
```
- ▶ On the UNIX platform  

```
netstat -na | grep -i time_wait | grep -c 9080
```

Substitute the port number with the port of the specific application server you want to monitor. Having both the Web server and the application server installed on the same machine results in a double count of every connection since the `TIME_WAIT` state is listed from both the client side and server side by the **netstat** command.

To configure the maximum number of keep-alive connections allowed on the Web container, perform the following steps:

1. From the console navigation pane, select **Servers** → **Application Servers**.
2. From the list of application servers in the workspace, select the name of the application server.
3. In the Additional Properties pane, select the **Web Container** entry.
4. Again in the same pane, select **HTTP Transports**.
5. Select the host link for which the max keep-alive connections setting has to be configured, in the host column of the HTTP Transports pane in the workspace.
6. In the Additional Properties pane, select **Custom properties**.
7. Click **New**.

8. Enter `MaxKeepAliveConnections` in the Name field and an integer value in the Value field. The recommended starting value is 90% of the maximum threads in the Web container thread pool.
9. Click **OK**.
10. Save the configuration, and restart the affected application server for the change to take effect.

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 the threads could be consumed by keep-alive connections, leaving no threads available to process new connections.

The application server may not accept a new connection under a heavy load if there are too many sockets in `TIME_WAIT` state. If all client requests go through the Web server plug-in and there are many `TIME_WAIT` state sockets for the Web container port, the application server closes connections prematurely, which in turn decreases performance.

The application server closes the connection from the Web server plug-in or from any client, for any of the following reasons:

- ▶ The client request is an HTTP 1.0 request when the Web server plug-in sends HTTP 1.1 requests.
- ▶ The maximum number of concurrent keep-alive connections is reached.
- ▶ The maximum number of requests for a connection is reached.
- ▶ A timeout occurred while waiting to read the next request or read the remainder of the current request.

### ***MaxKeepAliveRequests***

*MaxKeepAliveRequests* is the maximum number of requests allowed on a single keep-alive connection. This parameter helps prevent denial of service attacks when a client holds 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.

A good starting value for the maximum number of requests allowed is 100. If the application server requests are received from the Web server plug-in only, increase this parameter's value. Use the **netstat** utility to tune the value of maximum keep-alive requests as described in “MaxKeepAliveConnections” on page 260. If the number of connections in the `TIME_WAIT` state is too high, consider raising the maximum keep-alive requests setting.

To configure the maximum number of requests allowed from the Administrative Console, perform the following tasks:

1. From the console navigation panel, select **Servers** → **Application Servers**.
2. From the list of application servers in the workspace, select the name of the application server.
3. In the Additional Properties pane, select the **Web Container** entry.
4. Again, in the same pane, select **HTTP Transports**.
5. Select the host link for which the max keep-alive requests setting have to be configured, in the host column of the HTTP Transports pane in the workspace.
6. In the Additional Properties pane, select **Custom properties**.
7. Click **New**.
8. Enter `MaxKeepAliveRequests` in the Name field and an integer value in the Value field. The recommended starting value is 100.
9. Click **OK** to store the property.
10. Save the configuration and restart the affected application server for the change to take effect.

### ***HTTP transport custom properties***

Following is a list of additional custom properties of the HTTP transport provided with the application server. These custom properties are useful for further tuning measures. As with `MaxKeepAliveRequests` and `MaxKeepAliveConnections`, these custom properties are not shown in the settings page for an HTTP transport. You must set them using the Custom properties pane the same way as explained in “`MaxKeepAliveRequests`” on page 262 and “`MaxKeepAliveConnections`” on page 260.

Following is a list of additional custom properties pertaining to the HTTP transport provided with the application server:

- ▶ **ConnectionLOTimeOut**  
This property specifies the maximum number of seconds to wait when trying to read or process data during a request. The default value is five seconds.
- ▶ **ConnectionResponseTimeout**  
This property specifies the maximum number of seconds to wait when trying to read or write data during a response. The default is 300 seconds.
- ▶ **MaxConnectBacklog**  
This property specifies the maximum number of outstanding connect requests the operating system will buffer while it waits for the application

server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request is rejected. The default setting is 511.

Set this value to the number of concurrent connections you want to allow. Keep in mind that a single client browser may need to open multiple concurrent connections, perhaps four or five. However, increasing this value consumes more kernel resources. The value of this property is specific to each transport.

► **KeepAliveEnabled**

This property specifies whether to keep connections alive, that is, persistent. The default value is true.

The Web server plug-in keeps connections open to the application server as long as it can, bounded by the following keep alive parameter settings:

– **MaxKeepAliveConnections**

This property specifies the maximum number of concurrent keep alive or persistent connections across all HTTP transports. To make a particular transport close connections after a request, set **MaxKeepAliveConnections** to zero or set **KeepAliveEnabled** to false on that transport. The default is 90% of the maximum number of threads in the Web container thread pool. The default maximum thread pool size is 50, which means that the default **MaxKeepAliveConnections** is 45.

– **MaxKeepAliveRequests**

This property specifies the maximum number of requests that can be processed on a single keep alive connection. This parameter helps prevent denial-of-service attacks when a client tries to hold on to a keep-alive connection. The default is 100 requests.

– **ConnectionKeepAliveTimeout**

This property specifies the maximum number of seconds to wait for the next request on a keep alive connection.

## **Web server**

To use WebSphere Application Server resources on request processing instead of request queuing, the client requests that arrive at the Web server when the WebSphere Application Server is saturated should be queued in the network.

All Web servers have settings for configuring the maximum number of concurrent requests accepted. For information about how to configure this setting, refer to 9.4.8, “The Web server” on page 292. For more details about the new features of IBM HTTP Server 2.0 and how to use them, refer to “IBM HTTP Server 2.0” on page 299.

### ***Limiting connections from plug-in to Web container***

In WebSphere Application Server V5.1, the plug-in has been enhanced with the MaxConnections attribute. The MaxConnections value specifies the maximum number of connections permitted concurrently to each application server. When this number of connections is reached, the plug-in automatically skips that application server on establishing new connections, and tries the next available one.

The MaxConnections attribute works best with Web servers that follow the threading model instead of the process model, and where only one process is started. This is also called a single process-multiple threads model. This is true for IBM HTTP Server V2, the recommended Web server for WebSphere Application Server V5.1.

**Important:** IBM HTTP Server V1.3.x follows the process model, in contrast to IBM HTTP Server V2.0.x. With the process model, a new process is created to handle each connection from the application server, and typically, one process handles only one connection to the application server. Therefore, the MaxConnections attribute does not have much of an impact in restricting the number of concurrent requests to the application server.

### ***Determining the Web server maximum concurrency threads setting***

A Web server monitor helps tune the maximum concurrent thread processing setting for the Web server. Use a standard workload that represents a typical number of incoming client requests, a fixed number of iterations, and a standard set of configuration settings. Watch the number of Web server threads going to the Web container and the number of threads accessing static content locally on the Web server.

Set the concurrent requests setting to the sum of the Web container thread pool size and the number of static content requests processed by the Web server. If no static content is served from the Web server, the number of requests should be about equal to the Web container thread pool size, depending on the number of clients and Web servers concurrently accessing the same Web container.

### ***Using cluster configurations***

The capability to spread workload among application servers using clustering is a valuable asset in configuring highly scalable production environments. This is especially true when the application is experiencing bottlenecks that prevent full CPU utilization of symmetric multiprocessing (SMP) servers.

When adjusting the WebSphere system queues in clustered configurations, remember that when a server is added to a cluster, the server downstream receives twice the load. This is illustrated in Figure 9-7.

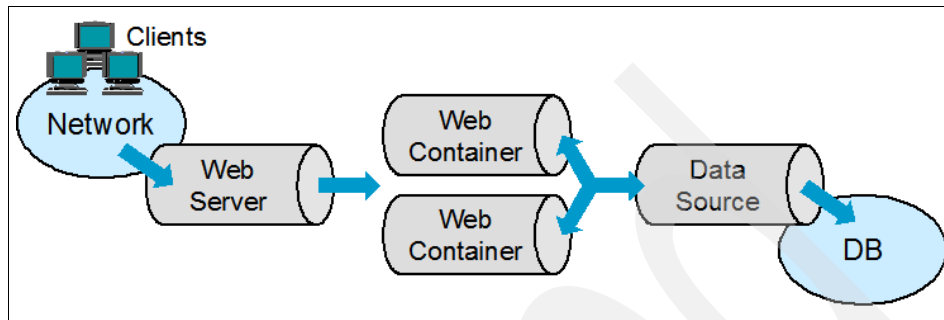


Figure 9-7 Clustering and queuing

Two Web containers within a cluster are located between a Web server and a data source. It is assumed that the Web server, Web container, and data source (but not the database) are all running on a single SMP server. Given these constraints, consider the following points regarding the queue:

- ▶ Web server queue settings can be doubled to ensure that ample work is distributed to each Web container.
- ▶ Web container thread pools can be reduced to avoid saturating a system resource such as CPU or another resource that the servlets are using.
- ▶ The data source pools can be reduced to avoid saturating the database server.
- ▶ Java heap parameters can be reduced for each instance of the application server. For versions of the JVM shipped with WebSphere Application Server, it is crucial that the heap from all JVMs remain in physical memory. Therefore, if a cluster of four JVMs are running on a system, enough physical memory must be available for all four heaps.

#### 9.4.5 Application assembly performance checklist

Application assembly tools are used to assemble J2EE components and modules into J2EE applications. Generally, this involves defining application components and their attributes, including enterprise beans, servlets, and resource references. Many of these application configuration settings and attributes play an important role in the runtime performance of the deployed application.

The most important parameters and advice for finding optimal settings are:

- ▶ Enterprise bean modules
  - Entity EJBs: Bean cache
  - Method extensions: Isolation level
  - Method extensions: Access intent
  - Container transactions
- ▶ Web modules
  - Web application: Distributable
  - Web application: Reload interval
  - Web application: Reload-enabled
  - Web application: Web components, Load on startup

## Enterprise bean modules

This section provides details about the enterprise bean module parameters.

**Note:** Although WebSphere Application Server 6 also supports EJB 2.0, the information provided here refers to EJB 1.1 settings.

### ***Entity EJBs: Bean cache***

WebSphere Application Server provides significant flexibility in the management of database data with Entity EJBs. The Entity EJBs Activate at and Load at configuration settings specify how and when to load and cache the data from the corresponding database row data of an enterprise bean. These configuration settings provide the capability to specify enterprise bean caching Options A, B, or C, as specified in the EJB 1.1 specification.

Option A provides maximum enterprise bean performance by caching database data outside of the transaction scope. Generally, Option A is only applicable where the EJB container has exclusive access to the given database. Otherwise, data integrity is compromised. Option B provides more aggressive caching of Entity EJB object instances, which can result in improved performance over Option C, but also results in greater memory usage. Option C is the most common, real-world configuration for Entity EJBs.

- ▶ Bean cache: Activate at

This setting specifies the point at which an enterprise bean is activated and placed in the cache. Removal from the cache and passivation are also governed by this setting. Valid values are *Once* and *Transaction*. *Once* indicates that the bean is activated when it is first accessed in the server process, and passivated and removed from the cache at the discretion of the container, for example, when the cache becomes full. *Transaction* indicates that the bean is activated at the start of a transaction and passivated and

removed from the cache at the end of the transaction. The default value is Transaction.

► Bean cache: Load at

This setting specifies when the bean loads its state from the database. The value of this property implies whether the container has exclusive or shared access to the database. Valid values are Activation and Transaction. Activation indicates that the bean is loaded when it is activated and implies that the container has exclusive access to the database. Transaction indicates that the bean is loaded at the start of a transaction and implies that the container has shared access to the database. The default is Transaction.

The settings of the Activate at and Load at properties govern which commit options are used.

► For Option A (exclusive database access), use Activate at = Once and Load at = Activation.

This option reduces database I/O by avoiding calls to the `ejbLoad` function, but serializes all transactions accessing the bean instance. Option A increases memory usage by maintaining more objects in the cache, but provides better response time if bean instances are not generally accessed concurrently by multiple transactions.

**Note:** When using WebSphere Network Deployment, and workload management is enabled, Option A cannot be used. You must use settings that result in the use of Options B or C.

► For Option B (shared database access), use Activate at = Once and Load at = Transaction.

Option B increases memory usage by maintaining more objects in the cache. However, because each transaction creates its own copy of an object, there can be multiple copies of an instance in memory at any given time, that is, one per transaction, requiring the database to be accessed at each transaction. If an enterprise bean contains a significant number of calls to the `ejbActivate` function, using Option B is beneficial because the required object is already in the cache. Otherwise, this option does not provide significant benefit over Option A.

► For Option C (shared database access), use Activate at = Transaction and Load at = Transaction.

This option reduces memory usage by maintaining fewer objects in the cache. However, there can be multiple copies of an instance in memory at any given time, that is, one per transaction. This option reduces transaction



contention for enterprise bean instances that are accessed concurrently, but not updated.

### ***Method extensions: Isolation level***

WebSphere Application Server enterprise bean method extensions provide settings to specify the level of transactional isolation used when accessing data. Isolation level settings specify various degrees of runtime data integrity provided by the corresponding database.

First, choose a setting that meets data integrity requirements for the given application and specific database characteristics. The valid values are:

- ▶ Serializable
- ▶ Repeatable read
- ▶ Read committed
- ▶ Read uncommitted

Isolation level also plays an important role in performance. Higher isolation levels reduce performance by increasing row locking and database overheads, while reducing data access concurrency. Various databases provide different behavior with respect to the isolation settings. In general, Repeatable read is an appropriate setting for DB2 databases. Read committed is generally used for Oracle. Oracle does not support Repeatable read and will translate this setting to the highest isolation level of Serializable.

You can specify the isolation level at the bean or method level. Therefore, it is possible to configure different isolation level settings for various methods. This is an advantage when some methods require higher isolation than others, and can be used to achieve maximum performance while maintaining integrity requirements. However, isolation cannot change between method calls within a single enterprise bean transaction. A runtime exception is thrown in this case.

The following section describes the four isolation levels:

- ▶ Serializable

This level prohibits the following types of reads:

- **Dirty reads:** A transaction reads a database row containing uncommitted changes from a second transaction.
- **Nonrepeatable reads:** One transaction reads a row, a second transaction changes the same row, and the first transaction rereads the row and gets a different value.
- **Phantom reads:** One transaction reads all the rows that satisfy a Structured Query Language (SQL) WHERE condition, a second transaction inserts a row that also satisfies the WHERE condition, and the

first transaction applies the same WHERE condition and gets the row inserted by the second transaction.

- ▶ Repeatable read  
This level prohibits dirty reads and nonrepeatable reads, but it allows phantom reads.
- ▶ Read committed  
This level prohibits dirty reads, but allows nonrepeatable reads and phantom reads.
- ▶ Read uncommitted  
This level allows dirty reads, nonrepeatable reads, and phantom reads.

The container uses the transaction isolation level attribute as follows:

- ▶ Session beans and entity beans with bean-managed persistence (BMP)  
For each database connection used by the bean, the container sets the transaction isolation level at the start of each transaction, unless the bean explicitly sets the isolation level on the connection.
- ▶ Entity beans with container-managed persistence  
The container generates database access code that implements the specified isolation level.

### **Method extensions: Access intent**

WebSphere Application Server enterprise bean method extensions provide settings to specify individual enterprise bean methods as read-only. This setting denotes whether the method updates entity attribute data or invokes other methods that can update data in the same transaction.

**Note:** This setting is applicable only for EJB 1.x-compliant beans:

- ▶ EJB 1.x compliant entity beans
- ▶ Enterprise beans with CMP Version 1.x that are packaged in EJB 2.x-compliant modules

To specify the access intent for EJB 2.x-compliant beans, select an access intent policy.

By default, all enterprise bean methods are assumed to be *update methods*. This results in EJB Entity data always being persisted back to the database at the close of the enterprise bean transaction. Marking enterprise methods that do not update entity attributes as Access Intent Read, provides a significant

performance improvement by allowing the WebSphere Application Server EJB container to skip the unnecessary database update.

A behavior for *finder methods* for CMP Entity EJBs is available. By default, WebSphere Application Server invokes a Select for Update query for CMP enterprise bean finder methods such as `findByPrimaryKey`. This exclusively locks the database row for the duration of the enterprise bean transaction. However, if the enterprise bean finder method has been marked as Access Intent Read, the container does not issue the For Update on the select, resulting in only a read lock on the database row.

### **Container transactions**

The container transaction setting specifies how the container manages transaction scopes when delegating invocation to the enterprise bean individual business method. The legal values are:

- ▶ Never
- ▶ Mandatory
- ▶ Requires New
- ▶ Required
- ▶ Supports
- ▶ Not Supported
- ▶ Bean-Managed

You can specify the container transactions attribute individually for one or more enterprise bean methods. You can configure enterprise bean methods not requiring transactional behavior as Supports to reduce container transaction management overhead.

The legal values are explained further in the following list:

- ▶ Never

This legal value directs the container to invoke bean methods without a transaction context. If the client invokes a bean method from within a transaction context, the container throws the `java.rmi.RemoteException` exception.

If the client invokes a bean method from outside a transaction context, the container behaves as it would if the Not Supported transaction attribute was set. The client must call the method without a transaction context.

- ▶ Mandatory

This legal value directs the container to always invoke the bean method within the transaction context associated with the client. If the client attempts to invoke the bean method without a transaction context, the container throws the `javax.jts.TransactionRequiredException` exception to the client. The

transaction context is passed to any enterprise bean object or resource accessed by an enterprise bean method.

Enterprise bean clients that access these entity beans must do so within an existing transaction. For other enterprise beans, the enterprise bean or bean method must implement the Bean Managed value or use the Required or Requires New value. For non-enterprise bean EJB clients, the client must invoke a transaction by using the `javax.transaction.UserTransaction` interface.

- Requires New

This legal value directs the container to always invoke the bean method within a new transaction context, regardless of whether the client invokes the method within or outside a transaction context. The transaction context is passed to any enterprise bean objects or resources that are used by this bean method.

- Required

This legal value directs the container to invoke the bean method within a transaction context. If a client invokes a bean method from within a transaction context, the container invokes the bean method within the client transaction context. If a client invokes a bean method outside a transaction context, the container creates a new transaction context and invokes the bean method from within that context. The transaction context is passed to any enterprise bean objects or resources that are used by this bean method.

- Supports

This legal value directs the container to invoke the bean method within a transaction context if the client invokes the bean method within a transaction. If the client invokes the bean method without a transaction context, the container invokes the bean method without a transaction context. The transaction context is passed to any enterprise bean objects or resources that are used by this bean method.

- Not Supported

This legal value directs the container to invoke bean methods without a transaction context. If a client invokes a bean method from within a transaction context, the container suspends the association between the transaction and the current thread before invoking the method on the enterprise bean instance. The container then resumes the suspended association when the method invocation returns. The suspended transaction context is not passed to any enterprise bean objects or resources that are used by this bean method.

► Bean-Managed

This value notifies the container that the bean class directly handles transaction demarcation. This property can be specified only for session beans and in EJB 2.0 implementations only for message-driven beans, not for individual bean methods.

## Web modules

This section explains the parameters that you can set for Web modules.

### ***Web application: Distributable***

The distributable flag for J2EE Web applications specifies that the Web application is programmed to be deployed in a distributed servlet container.

**Important:** Web applications should be marked as *Distributable* only if they will be deployed in a WebSphere Application Server clustered environment.

### ***Web application: Reload interval***

The reload interval specifies a time interval in seconds in which the Web application's file system is scanned for updated files, such as servlet class files or JSPs. You can define the reload interval at different levels for various application components.

Generally, the reload interval specifies the time the application server waits between checks to see if dependent files have been updated and need to be reloaded. Checking file system time stamps is an expensive operation and should be reduced. The default is zero. Setting this to a value of three seconds is good for a test environment because you can update the Web site without restarting the application server. In production environments, checking a few times a day is a more common setting.

### ***Web application: Reloading-enabled***

This parameter specifies whether file reloading is enabled. The default is false.

### ***Web application: Web components - Load on startup***

This parameter indicates whether a servlet is to be loaded at the startup of the Web application. The default is false.

Many servlets perform resource allocation and other up-front processing in the servlet `init()` method. These initialization routines are costly at runtime. By specifying Load on startup for these servlets, processing takes place when the application server is started. This avoids runtime delays that you may encounter on a servlet's initial access.

## 9.4.6 Java tuning

This section focuses on tuning Java memory. Enterprise applications written in Java involve complex object relationships and utilize large numbers of objects. Although Java automatically manages memory associated with an object's *life cycle*, understanding the application's usage patterns for objects is important. Ensure the following, in particular:

- ▶ The application is not over using objects.
- ▶ The application is not leaking objects, that is, memory.
- ▶ The Java heap parameters are set to handle the use of objects.

Understanding the effect of garbage collection is necessary to apply these management techniques.

### Garbage collection basics

Garbage collection algorithms, like JVMs, have evolved and become more complex to understand. Knowledge about how the Garbage Collector (GC) works is necessary for designing and tuning Java applications and application servers. Following is a broad, somewhat simplified, overview of the Mark-Sweep-Compact (MSC) garbage collection technique implemented by IBM JVMs. For an in-depth study of additional, state-of-the-art heap management and garbage collection techniques, refer to the articles mentioned in “Additional JVM and garbage collection-related resources” on page 283.

The Garbage Collector allocates areas of storage inside the Java heap, where objects, arrays, and classes are stored. An allocated object is considered *live* when there exists at least one reference to it, that means, it is used by someone, commonly, another object. Thus the object is also considered *reachable*. When this object is no longer used by anyone, all references should be removed. It is now considered *garbage*, and its allocated storage area should be reclaimed for reuse. This task is performed by the Garbage Collector.

When the JVM is unable to allocate an object from the current Java heap due to lack of free, contiguous space, a memory allocation fault or allocation failure occurs and the Garbage Collector is invoked. The GC can also be invoked by a specific function, call: `System.gc()`. However, when `System.gc()` is invoked, the JVM can simply *take it under advisement* and choose to defer the GC operation until later if the JVM has more pressing needs to manage.

The first task of the GC is to acquire all the locks required for garbage collection, and then stop all the other threads. Due to this, garbage collection is also referred to as *stop-the-world (STW) collection*.

Garbage collection then takes place in three phases:

- ▶ Mark phase
- ▶ Sweep phase
- ▶ Optionally compact phase

### ***The mark phase***

In the mark phase, all *reachable* objects that are referenced either directly by the JVM, for example, through threads stacks, or in turn by other objects, are identified. Everything else that is not marked is considered garbage.

### ***The sweep phase***

All allocated objects that are not marked are swept away, that is, the space used by them is reclaimed.

### ***The compaction phase***

When the garbage is removed from the heap, the GC considers compacting the heap, which is typically riddled with holes caused by the freed objects. When there is no chunk of memory big enough to satisfy an allocation request after garbage collection is available, the heap should be compacted.

Since heap compaction means moving objects around and updating all references to them, it is very costly in terms of time, and the GC tries to avoid it if possible. Modern JVM implementations try to avoid heap compaction by focusing on optimizing object placement in the heap.

**Note:** Do not confuse the Java heap with the native or system heap. The native heap is never garbage-collected. It is used by the JVM process and stores all the objects that the JVM itself needs during its entire lifetime. The native heap is typically much smaller than the Java heap.

### ***Heap expansion and shrinkage***

Heap expansion occurs after garbage collection if the ratio of free to total heap size falls below the value specified by the `-Xminf` parameter. The default is 0.3, that is, 30%.

Heap shrinkage occurs after garbage collection if the ratio of free to total heap size exceeds the value specified by the `-Xmaxf` parameter. The default is 0.6, that is, 60%. The amount of expansion is governed by the minimum expansion size set by the `-Xmine` parameter, and the maximum expansion size, defined by `-Xmaxe`. The defaults for `-Xmine` are 1 MB, and for `-Xmaxe` 0, which is equal to unlimited. These parameters do not have any effect on a fixed-size heap, where the `-Xms` and `-Xmx` values are equal.

### ***Parallel versus concurrent operation***

Recent releases of JVMs implement multiple helper threads that run in parallel on a multi-processor machine during the mark and sweep phases. These threads are asleep during normal operation. Only during garbage collection, the work is divided between the main GC thread and its helper threads, using all processors simultaneously.

*Parallel mark* mode is enabled by default since IBM JVM version 1.3.0, while *parallel sweep* is enabled by default since version 1.3.1. Earlier, we mentioned that garbage collection is basically a stop-the-world operation. However, this is not entirely true. IBM JVM 1.3.1 and higher know an optional *concurrent mark* mode, where a background thread is started by the JVM, and some of the mark phase's work is done concurrently while all application threads are active. Thus, the STW pause is reduced when garbage collection takes place. Concurrent mark is disabled by default. To activate it, use the `-Xgcpolicy:optavgpause` parameter.

**Note:** In some cases, concurrent mark may reduce the throughput of an application. We recommend that you compare the application performance with and without concurrent mark, using identical loads, to measure the effect on application performance.

In addition, IBM JVM 1.4 knows an *incremental compaction mode* that parallelizes the compaction phase. For more details about parallel and concurrent modes, refer to the developerWorks® article “Fine-tuning Java garbage collection performance” by Sumit Chawla that is found on the Web at:

<http://www.ibm.com/developerworks/ibm/library/i-gctroub/>

Alternately, see the IBM JVM Diagnostics Guides on the Web at:

<http://www.ibm.com/developerworks/java/jdk/diagnosis/>

### **The garbage collection bottleneck**

Examining Java garbage collection gives insight into how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects.

Garbage collection normally consumes anywhere between 5% and 20% of the total execution time of a properly functioning application. If not managed, garbage collection becomes one of the biggest bottlenecks for an application, especially when running on SMP server machines.



## The garbage collection gauge

Use garbage collection to evaluate the health of application performance. By monitoring garbage collection during the execution of a fixed workload, users gain insight as to whether the application is overutilizing objects. You can also use garbage collection to detect the presence of memory leaks.

Use the garbage collection and heap statistics in Tivoli Performance Viewer to evaluate application performance health. However, ensure that you have enabled the Java Virtual Machine Profiler Interface facility to get detailed garbage collection statistics. By monitoring garbage collection, you can detect memory leaks and overly-used objects. Following are the statistics that performance monitor interface provides through the use of JVMPI:

- ▶ Garbage collector
  - Number of garbage collection calls
  - Average time in milliseconds between garbage collection calls
  - Average duration in milliseconds of a garbage collection call
- ▶ Monitor
  - Number of times a thread waits for a lock
  - Average time a thread waits for a lock
- ▶ Object
  - Number of objects allocated
  - Number of objects freed from heap
  - Number of objects moved in heap
- ▶ Thread
  - Number of threads started
  - Number of threads died

For this type of investigation, set the minimum and maximum heap sizes to the same value. Choose a representative, repetitive workload that matches production usage as closely as possible, user errors included. To ensure meaningful statistics, run the fixed workload until the state of the application is steady. Reaching this state usually takes several minutes.

## Detecting over utilization of objects

To see if the application is overusing objects, look at the counters for the JVMPI profiler in Tivoli Performance Viewer. The average time between garbage collection calls should be five to six times the average duration of a single garbage collection. If not, it means that the application is spending more than 15% of its time in garbage collection. Also, look at the numbers of freed, allocated, and moved objects.

If the information indicates a garbage collection bottleneck, there are two ways to clear the bottleneck:

- ▶ The most cost-effective way to optimize the application is to implement object caches and pools. Use a Java profiler to determine which objects to target.
- ▶ If the application cannot be optimized, add memory, processors, and application clusters. Additional memory allows each application server in a cluster maintain a reasonable heap size. Additional processors allow the cluster members to run in parallel.

## Detecting memory leaks

Memory leaks in Java are major contributors to garbage collection bottlenecks. Memory leaks are more damaging than memory overuse, because a memory leak ultimately leads to system instability. Over time, garbage collection occurs more frequently, until finally, the heap is exhausted and Java fails with a fatal Out of Memory exception. Memory leaks occur when an object that is not needed has references that are never deleted. This occurs mostly in collection classes, such as Hashtable, because the table always has a reference to the object, even after real references are deleted.

High workload often causes many applications to crash immediately after being deployed in the production environment. This situation is especially true for leaking applications, where high workload accelerates the magnification of leakage and a memory allocation failure occurs.

Memory leak testing relates to magnifying numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage-collected. The delicate task is to differentiate these amounts from the expected sizes of useful and unusable memory. This task is achieved more easily if the numbers are magnified, resulting in larger gaps and easy identification of inconsistencies.

Following is a list of important conclusions with regard to memory leaks:

- ▶ Long-running test

Memory leak problems are manifested only after a period of time. Therefore, memory leaks are usually found during long-running tests. Short runs can lead to false alarms.

One of the problems in Java is whether to say that a memory leak is occurring when memory usage has seemingly increased either abruptly or monotonically in a given period. These kind of increases can be valid, and the objects created can be referenced at a much later time. Which method is used to differentiate the delayed use of objects from completely unused objects?

Running applications over a long period of time gets a higher confidence for whether the delayed use of objects is actually occurring. Because of this, memory leak testing cannot be integrated with some other types of tests such as functional tests that occur earlier in the process. However, stress or durability tests can be integrated.

► System test

Some memory leak problems occur only when different components of a big project are combined and executed. Interfaces between components can produce known or unknown side effects. System test is a good opportunity to make these conditions happen.

► Repetitive test

In many cases, memory leak problems occur by successive repetitions of the same test case. The goal of memory leak testing is to establish a big gap between unusable memory and used memory in terms of their relative sizes. By repeating the same scenario over and over again, the gap is multiplied in a very progressive way. This testing helps if the amount of leaks caused by the execution of a test case is so minimal that it is hardly noticeable in one run.

You can use repetitive tests at the system level or module level. The advantage with modular testing is better control. When a module is designed to keep the private module without creating external side effects such as memory usage, testing for memory leaks is much easier. First, memory usage before running the module is recorded. Then, a fixed set of test cases is run repeatedly. At the end of the test run, the current memory usage is recorded and checked for significant changes. Remember, garbage collection should be forced when recording the actual memory usage by inserting `System.gc()` in the module where garbage collection should occur or by using a profiling tool to force the event to occur.

► Concurrency test

Some memory leak problems occur only when there are several threads running in the application. Unfortunately, synchronization points are very susceptible to producing memory leaks because of the added complication in program logic. Careless programming leads to references being kept or not released. The incident of memory leaks is often facilitated or accelerated by increased concurrency in the system. The most common way to increase concurrency is to increase the number of clients in the test driver.

Consider the following points when deciding on which test cases to use for memory leak testing:

- A good test case exercises those areas of the application where objects are created. Most of the time, knowledge of the application is required. A description of the scenario suggests creation of data spaces, such as adding

a new record, creating an HTTP session, performing a transaction, and searching a record.

- Look at areas where collections of objects are being used. Typically, memory leaks are composed of objects of the same class. Also, collection classes such as Vector and Hashtable, are common places where references to objects are implicitly stored by calling corresponding insertion methods. For example, the get method of a Hashtable object does not remove its reference to the object being retrieved.

Tivoli Performance Viewer helps find memory leaks. For best results, repeat experiments with increasing duration, such as 1,000, 2,000, and 4,000-page requests. The Tivoli Performance Viewer graph of used memory should have a sawtooth shape. Each drop on the graph corresponds to a garbage collection. There is a memory leak if one of the following occurs:

- The amount of memory used immediately after each garbage collection increases significantly. The sawtooth pattern looks more like a staircase.
- The sawtooth pattern has an irregular shape.

Also, look at the difference between the number of objects allocated and the number of objects freed. If the gap between the two increases over time, there is a memory leak.

If heap consumption indicates a possible leak during a heavy workload, that is, the application server is consistently near 100% CPU utilization, yet the heap appears to recover during a subsequent lighter or near-idle workload, this is an indication of heap fragmentation. Heap fragmentation occurs when the JVM is able to free sufficient objects to satisfy memory allocation requests during garbage collection cycles, but the JVM does not have the time to compact small, free memory areas in the heap into larger, contiguous spaces.

Another form of heap fragmentation occurs when small objects, that are less than 512 bytes, are freed. The objects are freed, but the storage is not recovered, resulting in memory fragmentation.

Avoid heap fragmentation by turning on the -Xcompactgc flag in the JVM advanced settings command line arguments. The -Xcompactgc ensures that each garbage collection cycle eliminates fragmentation. However, this setting has a small performance penalty.

## Java heap parameters

The Java heap parameters also influence the behavior of garbage collection. Increasing the heap size allows more objects to be created. Because a large heap takes longer to fill, the application runs longer before a garbage collection

occurs. However, a larger heap also takes longer to compact and causes garbage collection to take longer.

For performance analysis, the initial and maximum heap sizes should be equal, since this eliminates heap growing and shrinking delays. Equating initial with maximum heap size without tuning the previous heap size, will in most cases, create an inefficiently used heap. When it is sized too big, the heap is not used by the application entirely and memory resources are wasted.

When tuning a production system, where the working set size of the Java application is not understood, a good starting value is to let the initial heap size be at 25% of the maximum heap size. The JVM then tries to adapt the size of the heap to the working set size of the application.

Run a series of test experiments that vary the Java heap settings. For example, run experiments with 128 MB, 192 MB, 256 MB, and 320 MB. During each experiment, monitor the total memory usage. If the heap is expanded too aggressively, paging occurs. Use the `vmstat` command or the Windows NT or Windows 2000 Performance Monitor to check for paging. If paging occurs, reduce the size of the heap or add more memory to the system.

**Important:** Make sure that the heap never pages, since that will introduce a enormous performance loss.

When all test runs are finished, compare the following statistics:

- ▶ Number of garbage collection calls
- ▶ Average duration of a single garbage collection call
- ▶ Average time between calls
- ▶ Ratio between the average length of a single garbage collection call and the average time between calls

If the application is not over utilizing objects and has no memory leaks, a state of steady memory utilization is reached. Garbage collection also occurs less frequently and for shorter durations.

If the heap free time settles at 85% or more, consider decreasing the maximum heap size values, because the application server and the application are underutilizing the memory allocated for heap.

The best result for the average time between garbage collections is at least five to six times the average duration of a single garbage collection. If you do not achieve this number, the JVM is spending more than 15% of its time in garbage collection. Finding this point involves the trial and error method. A longer interval

between garbage collection cycles results in longer GC runs, while very short intervals are inefficient.

### ***Heap thrashing***

Avoid heap thrashing at all costs. This is caused by a heap that is barely large enough to avoid expansion, but not large enough to satisfy future allocation failures. Usually a garbage collection cycle frees up enough space for not only the current allocation failure, but a substantial number of future allocation requests. However, when a heap is getting thrashed, each garbage collection cycle frees up barely enough heap space to satisfy just the current allocation failure. The result is that the next allocation request leads to another garbage collection cycle, and so on. This scenario also occurs due to lots of short-lived objects.

### **Tuning the IBM JVM**

Generally, every JVM offers an entire set of tuning parameters affecting the performance of application servers and applications. Since JVM tuning is a wide and complex topic, this section provides an introduction to garbage collection analysis and tuning in reference to the IBM JVM, and provides a series of links and resources for further study.

#### ***IBM Java 2 SDK 1.4.1***

Details about the SDK 1.4.1 enhancements are available on the Web at:

[http://java.sun.com/products/archive/j2se/1.4.1\\_07/changes.html](http://java.sun.com/products/archive/j2se/1.4.1_07/changes.html)

### ***Analyzing verbosegc output***

Using the **verbosegc** command as the next step after using Tivoli Performance Viewer, is a good way of seeing what is going on with garbage collection. The **verbosegc** mode is enabled by the **-verbosegc** command line option, and output is directed to the **native\_stderr.log** file. You can use this information to tune the heap size or diagnose problems.

For information about resources and illustrated examples of **verbosegc** analysis, read the following articles:

- ▶ “Sensible Sanitation: Understanding the IBM Java Garbage Collection Part 3: **verbosegc** and command-line parameters” by Sam Borman available on the Web at:  
<http://www.ibm.com/developerworks/library/i-garbage3.html>
- ▶ “Fine-tuning Java garbage collection performance” by Sumit Chawla available on the Web at:  
<http://www.ibm.com/developerworks/ibm/library/i-gctroub/>

### ***Common problems***

Common problems and suggested solutions are provided in the following list:

- ▶ The GC frequency is too high until the heap reaches a steady state.  
Use the **verbosegc** command to determine the size of the heap at a steady state and set -Xms to this value.
- ▶ The heap is fully expanded and the occupancy level is greater than 70%.  
Increase the -Xmx value so that the heap is not more than 70% occupied.
- ▶ At 70% occupancy, the frequency of GCs is too great.  
Change the setting of -Xminf. The default is 0.3, which will try to maintain 30% free space by expanding the heap. A setting of 0.4 increases this free space target to 40%, reducing the frequency of GCs.
- ▶ Pause times are too long.  
Try using -Xgcpolicy:optavgpause, which reduces pause times and makes them more consistent as the heap occupancy rises. There is a cost to pay in throughput. This cost varies and is about 5%.

### ***Additional JVM and garbage collection-related resources***

Refer to the following resources from IBM developerWorks for additional information about JVM and garbage collection:

- ▶ “Garbage collection in the 1.4.1 JVM”  
<http://www.ibm.com/developerworks/java/library/j-jtp11253/>
- ▶ “A brief history of garbage collection”  
<http://www.ibm.com/developerworks/java/library/j-jtp10283/>
- ▶ “Sensible Sanitation: Understanding the IBM Java Garbage Collection, Parts 1 and 2”  
<http://www.ibm.com/developerworks/ibm/library/i-garbage1/>  
<http://www.ibm.com/developerworks/ibm/library/i-garbage2/>
- ▶ IBM JVM Diagnostics Guides  
<http://www.ibm.com/developerworks/java/jdk/diagnosis/>
- ▶ “Fine-tuning Java garbage collection performance”  
<http://www.ibm.com/developerworks/ibm/library/i-gctroub/>
- ▶ “Mash that trash - Incremental compaction in the IBM JDK Garbage Collector”  
<http://www.ibm.com/developerworks/ibm/library/i-incrcomp/>

## Tuning the Sun JVM

The JVM offers several tuning parameters affecting the performance of WebSphere Application Server and application performance.

### ***Sun JDK 1.3 HotSpot -server warm up***

The HotSpot JVM introduces adaptive JVM technology containing algorithms for optimizing byte code execution over time. The JVM runs in two modes, -server and -client. To enhance performance significantly, run it in -server mode and allow a sufficient amount of time for a HotSpot JVM to warm up by performing continuous execution of byte code.

In most cases, -server mode should be run. This produces more efficient runtime execution over extended periods. Use the -client option if a faster startup time and smaller memory footprint are preferred, at the cost of lower extended performance.

The -server option is enabled by default and is the recommended value. Follow these steps to change the -client or -server mode:

1. Select **Servers** → **Application Servers**.
2. Select the application server you want to tune.
3. Under Additional Properties, select the link **Process Definition**.
4. Under Additional Properties, select **Java Virtual Machine**.
5. Under Additional Properties, select **Custom Properties**.
6. Click **New**.
7. In the Name field, enter HotSpotOption. In the Value field, enter -client or -server.
8. Restart the application server.

### ***Sun JDK 1.3 HotSpot new generation pool size***

Most garbage collection algorithms iterate every object in the heap to determine which objects to free. The HotSpot JVM introduces generation garbage collection, which makes use of separate memory pools to contain objects of different ages. These pools can be garbage-collected independently from one another. You can adjust the sizes of these memory pools. To avoid extra work, size the memory pools so that short-lived objects never live through more than one garbage collection cycle.



If garbage collection becomes a bottleneck, try customizing the generation pool settings. The default values are NewSize=2m, MaxNewSize=32m.

1. Select **Servers** → **Application Servers**.
2. Select the name of the application server you want to tune.
3. Under Additional Properties, select the **Process Definition** link.
4. Under Additional Properties, select **Java Virtual Machine**.
5. In the Generic JVM Arguments field, type the following values:  
-XX:NewSize (lower bound)  
-XX:MaxNewSize (upper bound)
6. Apply and save your changes.
7. Restart the application server.

We recommend that you set the new generation pool size between 25% and 50% of the total heap size.

### **Miscellaneous JVM settings**

The following sections provide several generic JVM setting recommendations that apply to most or all existing JVMs.

#### ***Just In Time compiler***

The Just In Time (JIT) compiler affects performance significantly. If you disable the JIT compiler, throughput decreases noticeably. Therefore, for performance reasons, keep JIT enabled.

To determine the setting of this parameter:

1. Select **Servers** → **Application Servers**.
2. Select the application server you want to tune.
3. Under Additional Properties, select the link **Process Definition**.
4. Under Additional Properties, select **Java Virtual Machine**.
5. Select the appropriate setting of the Disable JIT check box.
6. If changes are made, save them and restart the application server.

#### ***Heap size settings***

These parameters set the maximum and initial 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 should be low enough to contain the heap within physical memory.

The physical memory usage should be shared between the JVM and other applications running on the system, such as the database. For assurance, use a smaller heap, for example 64 MB, on machines with less memory.

Try a maximum heap of 128 MB on a smaller machine, that is, less than 1 GB of physical memory. Use 256 MB for systems with 2 GB memory, and 512 MB for larger systems. The starting point depends on the application.

If you are conducting performance runs and highly repeatable results are needed, set the initial and maximum sizes to the same value. This setting eliminates any heap growth during the run. 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 then tries to adapt the size of the heap to the working set of the Java application.

1. Select **Servers** → **Application Servers**.
2. Select the application server you want to tune.
3. Under Additional Properties, select **Process Definition**.
4. Under Additional Properties, select **Java Virtual Machine**.
5. In the General Properties configuration, enter values for the Initial Heap Size and Maximum Heap Size fields.
6. Apply and save your changes.
7. Restart the application server.

### ***Class garbage collection***

Disabling class garbage collection enables more class reuse, which, in some cases, results in small performance improvements. In most cases, run with class garbage collection turned on. This is the default.

To disable class garbage collection, enter the value `-Xnoclassgc` in the Generic JVM Arguments field of the application servers' JVM configuration as explained in the following steps.

1. Select **Servers** → **Application Servers**.
2. Select the application server you want to tune.
3. Under Additional Properties, select the link **Process Definition**.
4. Under Additional Properties, select the link **Java Virtual Machine**.
5. In the Generic JVM Arguments field, type the value `-Xnoclassgc`.
6. Apply and save your changes.
7. Restart the application server.

## 9.4.7 Operating system tuning

This section provides information about OS tuning parameters for AIX, Sun™ Solaris, Windows NT and Windows 2000. For more details, refer to the WebSphere Application Server information center on the Web at:

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

Over time, more information will be added to the information center, for example HP-UX-related tuning information. Therefore, continue checking the information center for the latest information.

### AIX

Many AIX OS settings that are not within the scope of this book should be considered. Some of the settings you can adjust are:

- ▶ Adapter transmit and receive queue
- ▶ TCP/IP socket buffer
- ▶ IP protocol mbuf pool performance
- ▶ Update file descriptors
- ▶ Update the scheduler

Two important settings are explained in the following settings.

### **AIX with DB2**

Separating your DB2 log files from the physical database files boost performance. You can also separate the logging and database files from the drive containing the Journaled File System (JFS) service. AIX uses specific volume groups and file systems for the JFS logging.

Use the AIX filemon utility to view all file system input and output, and to strategically select the file system for the DB2 logs. The default location of the files is `/home/<db2_instance>/<db2_instance>/NODExx/SQLyy/SQLLOGDIR/`.

To change the location of the files, at a DB2 command prompt, type the following command:

```
db2 update db cfg for [database_name] using newlogpath  
[fully_qualified_path]
```

We recommend that you move the logs to a separate disk when your application shows more than a 20% I/O wait time.

### **AIX file descriptors (ulimit)**

This specifies the number of open files permitted. 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.

Check the UNIX reference pages on ulimit for the syntax for different shells. For the KornShell shell (ksh), to set ulimit to 2000, type the following command:

```
ulimit -n 2000
```

Use **smiit** (or **smitty**) to permanently set this value for a user.

Use the **ulimit -a** command to display the current values for all limitations on system resources. The default setting is 2000, which is also the recommended value.

## **HP-UX 11i**

You can modify some HP-UX 11i settings to significantly improve WebSphere Application Server performance.

### ***TCP\_CONN\_REQUEST\_MAX***

When high connection rates occur, a large backlog of TCP/IP connection requests build up and client connections are dropped. Adjust this setting when clients start to timeout after waiting to connect. To verify this situation, type the following command:

```
netstat -p tcp
```

Look for the connect requests dropped due to full queue value. In most cases the default, that is, 4096, should suffice. Consider adjusting to 8192 if the default proves inadequate. Set this parameter by typing the following command:

```
ndd -set /dev/tcp tcp_conn_request_max
```

### ***JVM virtual page size***

Changing and setting the JVM instruction and data page sizes to 64 MB from the default value of 4 MB improves performance. Use the following command:

```
chattr +pi64M +pd64M  
/opt/WebSphere/AppServer/java/bin/PA_RISC2.0/native_threads/java
```

The command output provides the current OS characteristics of the process executable.

For a number of HP-UX 11i kernel parameter recommendations, refer to “Tuning operating systems” (under Tuning performance → Tuning the application serving environment) in the WebSphere Application Server information center on the Web at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

## Linux: Red Hat Advanced Server V2.1

Kernel updates for Red Hat Advanced Server V2.1 have implemented changes effecting WebSphere performance, especially memory-to-memory HTTP Session replication. If you are running any kernel prior to 2.4.9-e.23, upgrade at least to this kernel, but preferably to the latest supported.

## Linux: SUSE Linux Enterprise Server 8 SP2A

The Linux scheduler is very sensitive to excessive context switching, so fixes have been integrated into the SLES8 kernel distribution to introduce delay when a thread yields processing. This fix is automatically enabled in SLES8 SP3 but must be enable explicitly in SLES8 SP2A.

### ***sched\_yield\_scale tuning***

You can enable this fix by running the following command:

```
sysctl -w sched_yield_scale=1
```

If you are using a service pack below SP2A, upgrade to SP2A.

## Solaris

Tuning the following parameters that are a set in the startupServer.sh script, has a significant performance impact for Solaris:

- ▶ Solaris TCP\_TIME\_WAIT\_INTERVAL
- ▶ Solaris TCP\_FIN\_WAIT\_2\_FLUSH\_INTERVAL
- ▶ Solaris TCP\_KEEPALIVE\_INTERVAL

When high connection rates occur, a large backlog of the TCP connections builds up and slows server performance. The server stalls during certain peak periods. If this occurs, the `netstat` command shows that many of the sockets opened to port 80 were in the CLOSE\_WAIT or FIN\_WAIT\_2 state.

### ***Solaris file descriptors (ulimit)***

The Solaris file descriptors parameter specifies the number of open files permitted. If the value of this parameter is too low, the error message Too many files open is displayed in the WebSphere Application Server stderr.log.

Check the UNIX reference pages on `ulimit` for the syntax for different shells. For KornShell (ksh), the command is:

```
ulimit -n 1024
```

Use the `ulimit -a v` command to display the current values for all limitations on system resources.

The WebSphere Application Server startupServer.sh script sets this parameter to 1024 if its value is less than 1024.

### ***Solaris kernel semsys:seminfo\_semume***

The semsys:seminfo\_semume kernel tuning parameter limits the Max Semaphore undo entries per process and should be greater than the default, that is, 10 on Solaris 7. Since this setting specifies a maximum value, the parameter does not cause any additional memory to be used unless it is needed.

This value is displayed as SEMUME if you run the `/usr/sbin/sysdef` command. You can have an entry in the `/etc/system` file for this tuning parameter. Set it through the `/etc/system` entry as follows:

```
set semsys:seminfo_semume = 1024
```

### ***Solaris kernel semsys:seminfo\_semopm***

This setting is displayed as SEMOPM if you run the `/usr/sbin/sysdef` command. You can have an entry in the `/etc/system` file for this tuning parameter. Set it through the `/etc/system` entry as follows:

```
semsys:seminfo_semopm = 200
```

### ***Setting the virtual page size for WebSphere Application Server JVM***

On Solaris, if you want to increase the virtual page size for the WebSphere Application Server JVM, enter the following command:

```
chattr +pi64M +pd64M  
/opt/WebSphere/AppServer/java/bin/PA_RISC2.0/native_threads/java
```

Here, 64M stands for 64 MB, the recommended value.

### ***Other Solaris TCP parameters***

Customers have reported success with modifying other Solaris TCP parameters, including the following parameters:

- ▶ tcp\_conn\_req\_max\_q
- ▶ tcp\_comm\_hash\_size
- ▶ tcp\_xmit\_hiwat

Although significant performance differences have not been seen after raising these settings, the system may benefit.

Many other TCP parameters exist and can affect performance in a Solaris environment. For more information about tuning the TCP/IP Stack, refer to the article “Tuning your TCP/IP Stack and More” available on the Web at:

<http://www.sean.de/Solaris/soltune.html>

## Windows NT or Windows 2000 TCP/IP parameters

Two important tuning parameters exist for Windows NT and Windows 2000:

- ▶ `TcpTimedWaitDelay`
- ▶ `MaxUserPort`

These parameters are set in the Windows registry.

**Important:** When tuning WebSphere Application Server, use these two parameters together. We recommend that you tune both parameters at the same time.

### ***TcpTimedWaitDelay***

The `TcpTimedWaitDelay` parameter determines the time that must elapse before TCP releases a closed connection and reuses its resources. This interval between closure and release is known as the `TIME_WAIT` state or `2MSL`, that is, twice the maximum segment lifetime, state. During this time, reopening the connection to the client and server costs less than establishing a new connection. Reducing the value of this entry allows TCP to release closed connections faster, providing more resources for new connections.

Consider adjusting this value when the application that is running requires rapid release and creation of new connections, and there is a low throughput due to many connections sitting in `TIME_WAIT`. The default value is `0xF0` (240 seconds = 4 minutes), while the recommended value is the minimum value of `0x1E` (30 seconds).

To view or set this parameter:

1. Use the `regedit` command and access `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters`.
2. Create a new `REG_DWORD` named `TcpTimedWaitDelay`.
3. Set the value to decimal 30, which is Hex `0x0000001e`.
4. Restart the system.

To see that there are fewer connections in `TIME_WAIT`, use the `netstat` command.

### ***MaxUserPort***

The `MaxUserPort` parameter determines the highest port number TCP can assign when an application requests an available user port from the system. We

recommend that you set this value to at least decimal 32768 using the following steps:

1. Using the **regedit** command and access HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters.
2. Create a new REG\_DWORD named MaxUserPort.
3. Restart the system.

## 9.4.8 The Web server

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.

This section discusses some of the performance tuning settings associated with the Web servers. In addition to the settings mentioned here, you can find additional information about Web server tuning in the WebSphere Application Server information center on the Web at:

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

### Web server configuration reload interval

WebSphere Application Server administration tracks a variety of configuration information about WebSphere Application Server resources. Some of this information, such as URLs pointing to WebSphere Application Server resources, should 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 allow you to add new servlet definitions without having to restarting any of the WebSphere Application Server. However, the dynamic regeneration of this configuration information is costly in terms of performance. Try to adjust this value in a stable production environment. The default reload interval setting is 60 seconds.

The parameter `<RefreshInterval=xxxx>`, where `xxxx` is the number of seconds, is specified in the `<WAS_HOME>/config/plugin.xml` file. Increase the reload interval to a value that represents an acceptable wait time between the servlet update and the Web server update.



## IBM HTTP Server 1.3

The IBM HTTP Server v1.3.x is a multi-process, single-threaded server, except on the Windows platform. Some factors relating to the IBM HTTP Server's performance are covered in "Web server" on page 264.

### *Modifying the WebSphere plug-in to improve performance*

Improve the performance of IBM HTTP Server (with the WebSphere Web server plug-in) by modifying the plug-in's RetryInterval configuration. The RetryInterval is the length of time to wait before trying to connect to a server that has been marked as temporarily unavailable. Making this change helps the IBM HTTP Server to scale user levels that are higher than 400.

The plug-in marks a server that is temporarily unavailable if the connection to the server fails. Although the default value is 60 seconds, we recommend that you lower this value in order to increase throughput under heavy load conditions. Lowering the RetryInterval is important for IBM HTTP Server 1.3 on UNIX OS that have a single thread per process, and for IBM HTTP Server 2.0 if it is configured to have fewer than 10 threads per process.

How can lowering the RetryInterval affect throughput? If the plug-in attempts to connect to a particular application server while the application server threads are busy handling other connections, which happens under heavy load conditions, the connection times out and the plug-in marks the server temporarily unavailable. If the same plug-in process has other connections open to the same server and a response is received on one of these connections, the server is marked again. However, when you use the IBM HTTP Server 1.3 on a UNIX OS, there is no other connection since there is only one thread and one concurrent request per plug-in process. Therefore, the plug-in waits for the RetryInterval before attempting to connect to the server again.

Since the application server is only busy, and not really down, requests are typically completed within a small amount of time. The application server threads become available to accept more connections. A large RetryInterval causes application servers that are marked temporarily unavailable, resulting in more consistent application server CPU utilization and a higher sustained throughput.

**Note:** Although lowering the RetryInterval improves performance, if all the application servers are running, a low value has an adverse affect when one of the application servers is down. In this case, each IBM HTTP Server 1.3 process attempts to connect and fails more frequently, resulting in increased latency and decreased overall throughput.

### ***TCP initial connect timeout***

If an application server is really unavailable because of a node outage, the time that each plug-in process waits for a connect is defined by the underlying operating system's *TCP initial connect timeout*. Each plug-in process has to wait for that timeout before marking an application server temporarily unavailable.

This timeout value differs for every operating system. In Windows 2000, it is comparatively short and adjusted per TCP session to match the characteristics of the connection. For detailed information, refer to Microsoft Knowledge Base article #170359, "How to modify the TCP/IP maximum retransmission timeout" on the Web at:

<http://support.microsoft.com/kb/170359/en-us>

On AIX, this value can be queried and set using the **no** command. The default value for AIX 5.2, for example, is 75 seconds.

In a scenario where a single-threaded, multi-process Web server is used, for example, IBM HTTP Server 1.3, the plug-in processes do not share information about application server status. Therefore, each process tries to connect to the application server until it times out. Lowering the TCP initial connect timeout in AIX to a smaller value improves this specific behavior.

Keep in mind that, when changing this value, all processes using the TCP stack are affected, and this may produce unwanted side effects. Using a value that is too small may accomplish the opposite, when connections are no longer established under high load because the client thinks the connection partner is unavailable, when in fact, the latter is merely busy and will respond in due time. If you change this parameter, perform a load test and watch how your systems behave under peak load.

For detailed instructions on using the **no** command, refer to the AIX Man page or AIX administration manual.

**Note:** The value for `tcp_keepinit` should be specified in *halfseconds*. Refer to Example 9-4, where the TCP initial connect timeout value is set from 75 to 10 seconds. For instructions on how to accomplish this in other operating systems, refer to their respective administration manuals.

---

#### *Example 9-4 Querying and setting network options in AIX*

```
$ no -o tcp_keepinit
tcp_keepinit = 150

$ no -o tcp_keepinit=20
```

```
$ no -o tcp_keepinit  
tcp_keepinit = 20
```

---

**Attention:** The WebSphere Application Server V5.x the Web server plug-in support the new ConnectTimeout parameter. This attribute is valid inside the <server> tag of plugin-cfg.xml, and lets you bypass the operating system's TCP initial connect timeout. Specify the duration of waiting time until the currently queried application server is marked unavailable.

This is the procedure we recommend and prefer it over changing the OS' TCP initial connect timeout value. Refer to "Tuning failover" in *IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series*, SG24-6198, for a detailed description of the ConnectTimeout parameter.

### **Access logs**

All incoming HTTP requests are logged here. Logging degrades performance because I/O operation overhead causes logs to grow significantly in a short time.

To turn logging on or off, perform these tasks:

1. Edit the IBM HTTP Server httpd.conf file located in the IBM HTTP Server Home/conf directory.
2. Comment out the line that has the text CustomLog.
3. Save and close the httpd.conf. file.
4. Stop and restart the IBM HTTP Server.

By default, logging is enabled, but for better performance, we recommend that you disable the access logs.

### **MaxClients/ThreadsPerChild**

The value of the MaxClients parameter significantly impacts the application, particularly if it is too high. More is not always better. The optimum value depends on the application. Setting MaxClients too high causes swapping if there is insufficient RAM. Set this value to prevent bottlenecks, allowing just enough traffic through to the application server.

The maximum number of concurrent requests accepted by the IBM HTTP Server is configured by a parameter in the httpd.conf configuration file. The parameter sets the number of concurrent threads running at any one time within the IBM HTTP Server. The default value is 150.

The parameter keyword is different on different platforms. On IBM HTTP servers running on the UNIX platform, the keyword is MaxClients. On the Windows platform, the configuration parameter keyword is ThreadsPerChild. Refer to “ThreadsPerChild” on page 297 for more information about this parameter on Windows platforms.

To configure the number of maximum concurrent allowed connections, open the httpd.conf file and change the value of ThreadsPerChild in the case of Windows, or MaxClients in the case of UNIX, to the appropriate value, as shown in Example 9-5. Save the changes and restart the IBM HTTP server.

*Example 9-5 ThreadsPerChild/MaxClients parameter in httpd.conf*

---

```
# Number of concurrent threads at a time (set the value to more or less  
# depending on the responsiveness you want and the resources you wish  
# this server to consume).
```

```
MaxClients 50
```

---

***MinSpareServers, MaxSpareServers, and StartServers***

These preallocate and maintain 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. Specifying similar values reduces the CPU usage for creating and destroying HTTP daemon (HTTPD) processes. You must adjust this parameter if the time waiting for IBM HTTP Server to start more servers, so that it can handle HTTP requests, is not acceptable.

For optimum performance, specify the same value for the MaxSpareServers and the StartServers parameters. If MaxSpareServers is set to less than MinSpareServers, IBM HTTP Server resets MaxSpareServer=MinSpareServer+1. Setting the StartServers too high causes swapping if memory is not sufficient, thus degrading performance.

To view or change these values, edit the following directives in the httpd.conf file located in the IBM HTTP Server Home/conf directory:

- ▶ MinSpareServers
- ▶ MaxSpareServers
- ▶ StartServers

The default values for these servers are five, ten, and five respectively.

For more information about tuning IBM HTTP Server, refer to “Hints on Running a High-Performance Web Server” on the Web at:

<http://www.ibm.com/software/webservers/httpservers/doc/v136/misc/perf.html>

### **IBM HTTP Server 1.3: Linux**

The instructions in the following section are important when running the IBM HTTP Server on Linux.

#### ***MaxRequestsPerChild***

The MaxRequestsPerChild directive sets a limit on the number of requests that an individual child server process handles. After the number of requests reaches the value set for the MaxRequestsPerChild parameter, the child process dies. If there are no known memory leaks with Apache and Apache’s libraries, set this value to zero.

By default, this value is set to 500. To change this value:

1. Edit the IBM HTTP server file httpd.conf located in the directory IBM HTTP Server Home/conf.
2. Change the value of the MaxRequestsPerChild parameter.
3. Save the changes and restart the IBM HTTP server.

### **IBM HTTP Server 1.3: Windows 2000 or Windows 2003**

This section discusses the IBM HTTP Server tuning specifics when running on Windows NT or Windows 2000. Contrary to other platforms, IBM HTTP Server is a single-process, multi-threaded server on the Windows platform.

#### ***ThreadsPerChild***

This parameter sets the number of concurrent threads running at any one time within the IBM HTTP Server. Refer to “MaxClients/ThreadsPerChild” on page 295 for information about how to change this value.

There are two ways to determine how many threads are in use:

- Use the Windows 2000 or Windows 2003 Task Manager:
  - a. On your desktop, right-click the Status bar and select **Task Manager**.
  - b. Select the **Processes** tab.
  - c. Select **View** → **Select Columns**.
  - d. Select **Thread Count**.
  - e. Click **OK**.

- Use IBM HTTP Server server-status. This choice works on all platforms, not just Windows 2000 or Windows 2003. To enable it, use the following steps depending on your Web server version:

- IBM HTTP Server 1.3.x:

- i. Edit the httpd.conf file located in the IBM\_HTTP\_Server\_Home/conf directory and remove the comment character “#” from the lines shown in Example 9-6.

*Example 9-6 Enabling HTTP Server 1.3 server-status in httpd.conf file*

---

```
#LoadModule status_module modules/ApacheModuleStatus.dll
#<Location/server-status>
#SetHandler server-status
#</Location>
```

---

- ii. Save the changes and restart the IBM HTTP server.

- iii. In a Web browser, go to the following URL:

`http://your_host/server-status`

Click **Reload to update status**.

Alternatively, if the browser supports refresh, go to the following URL to refresh every five seconds:

`http://your_host/server-status?refresh=5`

- IBM HTTP Server 2.0:

- i. Edit httpd.conf located in the IBM\_HTTP\_Server\_Home/conf directory and remove the comment character “#” from the lines shown in Example 9-7.

*Example 9-7 Enabling HTTP Server 2.0 server-status in httpd.conf file*

---

```
#LoadModule status_module modules/mod_status.so#
#<Location /server-status>
#    SetHandler server-status
#</Location>
```

---

- ii. Save the changes and restart the IBM HTTP server.

- iii. In a Web browser, go to the following URL:

`http://your_host/server-status`

Click **Reload to update status**.

Alternatively, if the browser supports refresh, go to the following URL to refresh every five seconds:

`http://your_host/server-status?refresh=5`

The default value of the `ThreadsPerChild` parameter is 50 for IBM HTTP Server 1.3.28, and 250 for IBM HTTP Server 2.0.

**Note:** If you are using the `server-status` module on other platforms, additional changes may be needed, and modules will not be in the form of a dynamic link library (DLL). Consult the administrator's guide for specific information.

## IBM HTTP Server 2.0

Although IBM HTTP Server 1.3.28 is installed as the default Web server if you select a full installation of WebSphere Application Server V5.1, IBM introduced plug-in support for IBM HTTP Server 2.0 in Application Server V5.0. To use the IBM HTTP Server 2.0 Web server, download and install IBM HTTP Server 2.0 from the following Web site:

<http://www.ibm.com/software/webservers/httpservers/>

For details about installing and upgrading from IBM HTTP Server V1.3.x to V2.0.x, refer to Installing “IBM HTTP Server powered by Apache 2.0”. Select **Installation** → **Getting Started** → **Preparing to install and configure a Web server** in the WebSphere Application Server information center on the Web at:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

The easiest way, which we recommend, is to install IBM HTTP Server 2.0 before you install WebSphere Application Server. Then, select the plug-in for V2.0 to have the installer configure IBM HTTP Server 2.0 to work with WebSphere Application Server.

For information about IBM HTTP Server 2.0 tuning, refer to “Tuning IBM HTTP Server 2.0” on page 302.

### ***New features in IBM HTTP Server V2.0***

IBM HTTP Server 2.0 has numerous new features. This section describes specific issues to help you determine whether to migrate your system from IBM HTTP Server 1.3 to 2.0. A complete list of features about the Apache 2.0 is available on the Web at:

[http://httpd.apache.org/docs-2.0/new\\_features\\_2\\_0.html](http://httpd.apache.org/docs-2.0/new_features_2_0.html)

Some of the new features include:

- IBM HTTP Server 2.0 is a thread-based Web server

Although IBM HTTP Server 1.3 is thread-based on the Windows platform, it is a process-based Web server on all UNIX and Linux platforms. That means, it

implements the *multi-process, single-thread* process model. For each incoming request, a new child process is created or requested from a pool to handle it.

IBM HTTP Server 2.0 is now a fully thread-based Web server on all platforms. This gives you the following advantages:

- Each request does not require its *own* HTTP daemon (HTTPD) process anymore, and less memory is needed.
- Overall performance improves because, in most cases, new HTTPD processes do not have to be created.
- The plug-in load balancing and failover algorithms work more efficiently in a single-process, multi-threaded HTTP server. If one plug-in thread marks an application server unavailable, all other connection threads of the plug-in share that knowledge, and do not try to connect to this particular application server again before the `RetryInterval` has elapsed.

Refer to *IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series*, SG24-6198, and *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392 for information about the `RetryInterval` parameter and plug-in load balancing and failover issues.

**Note:** On the UNIX platform, IBM HTTP Server 2.0 also allows you to configure more than one process to be started.

► The `mod_deflate` module

This module allows supporting browsers to request that content be compressed before delivery. It provides the *Deflate* output filter that lets output from the server be compressed before it is sent to the client over the network. Some of the most important benefits of using the `mod_deflate` module are:

- Saves network bandwidth during data transmission
- Shortens data transmission time
- Generally improves overall performance

You can find detailed information about configuring and using `mod_deflate` at:

[http://httpd.apache.org/docs-2.0/mod/mod\\_deflate.html](http://httpd.apache.org/docs-2.0/mod/mod_deflate.html)

► Request and response filtering

Apache and IBM HTTP Server modules may now be written as filters which act on the stream of content as it is delivered to or from the server.

► No GUI-based administration tool



IBM HTTP Server 2.0 does not contain a GUI-based administration tool anymore. Changes have to be performed manually on the `httpd.conf` file using an editor. If browser-based administration is imperative for you, do *not* upgrade to IBM HTTP Server 2.0.

### **Configuration migration**

Review your IBM HTTP Server 1.3.x `httpd.conf` configuration file to see which modules are loaded and what directives are set, due to the following reasons:

- ▶ Many IBM HTTP Server 1.3 directives remain unchanged in IBM HTTP Server 2.0. Notable exceptions relate primarily to new IBM HTTP Server 2.0 functions, such as filtering and UNIX thread support.
- ▶ IBM HTTP Server 2.0 contains some new directives.
- ▶ In IBM HTTP Server 2.0, some 1.3 directives have been deprecated.

**Important:** Do not overwrite the new IBM HTTP Server 2.0 `httpd.conf` with your IBM HTTP Server 1.3.x one because this will not work.

After you install WebSphere Application Server V5.1, verify that all the needed WebSphere plug-in directives are added to the IBM HTTP Server 2.0 configuration file. Example 9-8 shows the WebSphere plug-in directives that must be added to `httpd.conf` for IBM HTTP Server 2.0 running on UNIX. This is similar on the Windows platform.

#### *Example 9-8 Plug-in directives*

---

```
Alias /IBMWebAS/ <wasroot>/web/  
Alias /WSsamples <wasroot>/WSsamples/  
LoadModule was_ap20_module <wasroot>/bin/mod_was_ap20_http.so  
WebSpherePluginConfig <wasroot>/config/cells/plugin-cfg.xml
```

---

### **Single-processing versus multiprocessing**

Apache version 2 (and thus IBM HTTP Server 2) achieves efficient support of different OS by implementing a multiprocessing modules (MPM) architecture, allowing it to, for example, use native networking features instead of going through an emulation layer in V1.3.

For detailed information about MPM, refer to the Apache HTTP Server documentation on the Web at:

<http://httpd.apache.org/docs-2.0/mpm.html>

MPMs are chosen at compile time and differ between OS, which implies that the Windows version uses a different MPM module than the AIX or Linux version.

The default MPM for Windows is mpm\_winnt, where the default module for AIX is mpm\_worker.

For a complete list of available MPMs, refer to the Apache MPM documentation on the Web at:

<http://httpd.apache.org/docs-2.0/mpm.html>

To identify which was MPM compiled into an Apache 2.0 or IBM HTTP Server 2.0 Web server, run the **httpd -l** command, which prints out the module names. Look for the module name worker, or a name starting with the mpm prefix, as shown in Example 9-9.

*Example 9-9 Listing compiled-in modules for IBM HTTP Server 2.0 on AIX*

---

```
root@app2:/usr/IBMIHS/bin $ ./httpd -l
```

Compiled in modules:

```
core.c
worker.c
http_core.c
mod_suexec.c
mod_so.c
```

---

► mpm\_winnt module

This multiprocessing module is the default for the Windows operating system. It uses a single control process which launches a single child process which, in turn, creates all the threads to handle requests.

► mpm\_worker module

This multiprocessing module implements a hybrid multiprocess, multithreaded server. This is the default module for AIX. By using threads to serve requests, it is able to serve a large number of requests with less system resources than a process-based server. Yet, it retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.

### ***Tuning IBM HTTP Server 2.0***

This section gives you configuration tips for the UNIX and Windows platform that provide a good starting point for Web server tuning, and an explanation about the new configuration directives used. Every system and every site has different requirements. Therefore, make sure to adapt these settings to your needs.

► ThreadsPerChild

Each child process creates a fixed number of threads as specified in the ThreadsPerChild directive. The child creates these threads at startup and

never creates more. If you are using an MPM such as `mpm_winnt`, where there is only one child process, this number should be high enough to handle the entire load of the server. If using an MPM-type worker, where there are multiple child processes, the total number of threads should be high enough to handle the common load on the server.

► **ThreadLimit**

This directive sets the maximum configured value for `ThreadsPerChild` for the lifetime of the Apache process. You can modify `ThreadsPerChild` during a restart up to the value of this directive.

► **MaxRequestsPerChild**

This directive controls after how many requests a child server process is recycled and a new one is launched.

► **MaxClients**

This directive controls the maximum total number of threads that may be launched.

► **StartServers**

The `StartServers` directive sets the number of processes that will initially be launched.

► **MinSpareThreads and MaxSpareThreads**

During operation, the total number of idle threads in all processes will be monitored, and kept within the boundaries specified by `MinSpareThreads` and `MaxSpareThreads`.

► **ServerLimit**

The `ServerLimit` directive sets the maximum number of processes that can be launched.

Example 9-10 shows a sample UNIX configuration.

*Example 9-10 Sample configuration for the UNIX platform*

---

```
<IfModule worker.c>
ServerLimit 1
ThreadLimit 2048
StartServers 1
MaxClients 1024
MinSpareThreads 1
MaxSpareThreads 1024
ThreadsPerChild 1024
MaxRequestsPerChild 0
</IfModule>
```

---

**Attention:** We recommend that you do not use a ThreadsPerChild value greater than 512 on the Linux and Solaris platform. If 1024 threads are needed, we recommend that you increase the ServerLimit value to two to launch two server process with 512 threads each.

Example 9-11 shows a sample Windows configuration.

*Example 9-11 Sample configuration for the Windows platform*

---

```
<IfModule mpm_winnt.c>
ThreadsPerChild 2048
MaxRequestsPerChild 0
</IfModule>
```

---

## **Sun ONE Web Server (formerly iPlanet)**

The default configuration of the Sun ONE™ Web server, Enterprise Edition, provides a single-process, multi-threaded server.

### **Active Threads**

The Active Threads value specifies the current number of threads active in the server. After the server reaches the limit set with this parameter, the server stops servicing new connections until it finishes old connections. Thus, if this setting is too low, the server can become throttled, resulting in degraded response times.

To tell if the Web server is being throttled, consult its perfdump statistics. Look at the following data:

- ▶ **WaitingThreads count**  
If the WaitingThreads count is getting close to zero, or is zero, the server is not accepting new connections.
- ▶ **BusyThreads count**  
If the WaitingThreads count is close to zero, or is zero, BusyThreads is probably very close to its limit.
- ▶ **ActiveThreads count**  
If the ActiveThreads count is close to its limit, the server is probably limiting itself.

Use the Maximum number of simultaneous requests parameter in the Enterprise Server Manager interface to control the number of active threads within Sun ONE Web server, Enterprise Edition. This setting corresponds to the RqThrottle parameter in the magnus.conf file.

The default value is 512. We recommend that you increase the thread count until the active threads parameters show optimum behavior.

## **Microsoft IIS**

The Web server has several properties that dramatically affect the performance of the application server.

### ***IIS permission properties***

The default settings are usually acceptable. However, because other products can change the default settings without the user's knowledge, make sure that you check the IIS settings for the Home Directory permissions of the Web server. The permissions should be set to Script and not to Execute. If the permissions are set to Execute, no error messages are returned, but the performance of WebSphere Application Server is decreased.

To check or change these permissions, perform the following procedure in the Microsoft management console:

1. Select the Web site, usually the default Web site. Right-click and select the **Properties** option. Click the **Home Directory** tab.
2. Under Application settings, in the Permissions list, ensure that the **Script** check box is selected and that the **Execute** check box is deselected.

**Note:** It may be necessary to check the permissions of the sePlugin to confirm that the Execute permissions are set to Execute. This is done from the same console by expanding the Web server and selecting the properties for the sePlugin object.

### ***Number of expected hits per day***

The Number of expected hits per day parameter controls the memory that IIS allocates for connections.

Using the performance window, set the parameter to More than 100000 in the Web site properties panel of the Microsoft management console. The default for this value is Fewer than 100000.

### ***ListenBackLog parameter***

If you are using IIS on Windows 2000 or Windows 2003, the server is likely to encounter failed connections under heavy load conditions, typically more than 100 plus clients. This condition commonly results from IIS rejecting connections. Alleviate the condition by using the ListenBackLog parameter to increase the number of requests IIS keeps in its queue. If you intermittently receive the error

message Unable to locate server in a Netscape browser when running a heavy load, consider raising this parameter.

Use the **regedit** command to access the registry. In the registry window, locate the parameter in the HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\ ListenBackLog directory. Right-click the parameter to modify it. Adjust the setting according to the server load.

The default value is 25 (decimal). You can set the ListenBackLog parameter to as high as 200 without negative impact on performance, and an improvement in load handling.

### 9.4.9 Dynamic Cache Service

The Dynamic Cache Service improves performance by caching the output of servlets, commands, and JSP files. WebSphere Application Server consolidates several caching activities, including servlets, Web services, and WebSphere commands into one service called the dynamic cache. These caching activities work together to improve application performance, and share many configuration parameters, which are set in an application server's dynamic cache service.

The dynamic cache works within an application server JVM, intercepting calls to cacheable objects, for example, through a servlet's `service()` method or a command's `execute()` method, and either stores the object's output to or serves the object's content from the dynamic cache. Since J2EE applications have high read/write ratios and can tolerate small degrees of latency in the currency of their data, the dynamic cache creates an opportunity for significant gains in server response time, throughput, and scalability.

For more detailed information about the functionality and usage of the WebSphere dynamic cache services, refer to *IBM WebSphere V5.1 Performance, Scalability, and High Availability WebSphere Handbook Series*, SG24-6198.

### 9.4.10 Security settings

This section discusses how various settings relating to security affect performance. Refer to *IBM WebSphere V5.0 Security WebSphere Handbook Series*, SG24-6573, for more information about WebSphere security.

#### Disabling security

Security is a global setting. When security is enabled, performance can be decreased between 10% to 20%. Therefore, disable security when not needed.

In the Administrative console, select **Security** → **Global Security**. The Enabled and Enforce Java 2 Security check boxes control global security settings. By default, security is not enabled.

### **Fine-tuning the security cache timeout for the environment**

If WebSphere Application Server security is enabled, the security cache timeout influences performance. The timeout parameter specifies how often you should refresh the security-related caches.

Security information pertaining to beans, permissions, and credentials is cached. When the cache timeout expires, all cached information become invalid. Subsequent requests for the information result in a database lookup. Sometimes, acquiring the information requires invoking a Lightweight Directory Access Protocol (LDAP)-bind or native authentication. Both invocations are relatively costly operations for performance.

Determine the best trade-off for the application by looking at usage patterns and security needs for the site.

Use the Administrative console to change this value. To do so, select **Security** → **Global Security**. Enter an appropriate value in seconds in the Cache Timeout field. The default is 600 seconds.

## **9.4.11 Tuning Secure Sockets Layer**

Following are two types of Secure Sockets Layer (SSL) performance:

- ▶ Handshake
- ▶ Bulk encryption/decryption

### **Overview of handshake and bulk encryption and decryption**

When an SSL connection is established, an SSL handshake occurs. After a connection is made, SSL performs bulk encryption and decryption for each read/write. The performance cost of an SSL handshake is much larger than that of bulk encryption and decryption.

### **Enhancing SSL performance**

To enhance SSL performance, decrease the number of individual SSL connections and the handshakes. Decreasing the number of connections increases performance for secure communication through SSL connections, as well as non-secure communication through simple TCP connections.

One way to decrease individual SSL connections is to use a browser that supports HTTP 1.1. Decreasing individual SSL connections can be impossible for some users if they cannot upgrade to HTTP 1.1.

Another common approach is to decrease the number of connections, both TCP and SSL, between two WebSphere Application Server components. The following guidelines help to ensure that the HTTP transport of the application server is configured so that the Web server plug-in does not repeatedly reopen new connections to the application server:

- ▶ The maximum number of keep-alives should be, at a minimum, as large as the maximum number of requests per thread of the Web server or maximum number of processes for IBM HTTP Server on UNIX.

Make sure the Web server plug-in is capable of obtaining a keep-alive connection for every possible concurrent connection to the application server. Otherwise, the application server closes the connection after a single request is processed. This adds performance overhead due to the creation and destruction of the connection, and the handshaking associated with SSL transactions.

Also, the maximum number of threads in the Web container thread pool should be larger than the maximum number of keep-alives, in order to prevent the Web container threads from being consumed with keep-alive connections.

- ▶ You can increase the maximum number of requests per keep-alive connection. The default value is 100, which means the application server will close the connection from the plug-in after 100 requests. The plug-in then has to open a new connection.

The purpose of this parameter is to prevent denial of service attacks when connecting to the application server and continuously sending requests in order to tie up threads in the application server.

- ▶ Use a hardware accelerator if the system performs several SSL handshakes. Hardware accelerators currently supported by WebSphere Application Server only increase the SSL handshake performance, and not the bulk encryption or decryption.

An accelerator typically benefits only the Web server because Web server connections are short-lived. All other SSL connections in WebSphere Application Server are long-living.

- ▶ Use an alternative cipher suite with better performance. The performance of a cipher suite is different with software and hardware. Just because a cipher suite performs better in software does not mean that it will perform better with hardware. Some algorithms are typically inefficient in hardware, for example,



DES and 3DES. However, specialized hardware provides efficient implementations of these same algorithms.

The performance of bulk encryption and decryption is affected by the cipher suite used for an individual SSL connection.

### 9.4.12 Object Request Broker

Several settings are available for controlling internal Object Request Broker processing. Use these to improve application performance in the case of applications containing enterprise beans.

Change these settings for the default server or any application server configured in the administrative domain from the Administrative console.

#### ***Pass by value versus Pass by reference (com.ibm.CORBA.iiop.noLocalCopies)***

For EJB 1.1 beans, the EJB 1.1 specification states that method calls are to be Pass by value. For every remote method call, the parameters are copied on to the stack before the call is made. This is expensive. You can specify the Pass by reference, which passes the original object reference without making a copy of the object.

For EJB 2.0 beans, interfaces can be local or remote. For local interfaces, method calls are Pass by reference, by default.

If the EJB client and EJB server are installed in the same WebSphere Application Server instance, and the client and server use remote interfaces, specifying Pass by reference can improve performance up to 50%. Pass by reference helps performance only when non-primitive object types are being passed as parameters. Therefore, int and floats are always copied, regardless of the call model.

**Important:** Pass by reference can be dangerous and lead to unexpected results. If an object reference is modified by the remote method, the change may be seen by the caller.

Use the Administrative console to set this value by performing the following steps:

1. Select **Servers** → **Application Servers**.
2. Select the application server you want to change.
3. From the Additional Properties panel, select **ORB Service**.
4. Select the check box **Pass by Reference**.
5. Click **OK** and click **Apply** to save the changes.

6. Stop and restart the application server.

The default is Pass by value for remote interfaces and Pass by reference for EJB 2.0 local interfaces.

If the application server expects a large workload for enterprise bean requests, the ORB configuration is critical. Note of the properties that are explained in the following sections.

### ***com.ibm.CORBA.ServerSocketQueueDepth***

The `com.ibm.CORBA.ServerSocketQueueDepth` property corresponds to the length of the TCP/IP stack listen queue. It also prevents WebSphere Application Server from rejecting requests when there is no space in the listen queue.

If there are many simultaneous clients connecting to the server-side ORB, you can increase this parameter to support a heavy load of up to 1000 clients. The default value is 50.

To set the property (in our setup, we set it to 200), perform these steps:

1. Select **Servers** → **Application Servers**.
2. Click the application server you want to tune.
3. Under Additional Properties, select **Process Definition**.
4. Under Additional Properties, select **Java Virtual Machine**.
5. In the Generic JVM Properties field, type  
-Dcom.ibm.CORBA.ServerSocketQueueDepth=200.

### ***Object Request Broker connection cache maximum (com.ibm.CORBA.MaxOpenConnections)***

This property has two names and corresponds to the size of the ORB connection table. The property sets the standard for the number of simultaneous ORB connections that can be processed.

If there are many simultaneous clients connecting to the server-side ORB, you can increase this parameter to support a heavy load of up to 1000 clients. The default value is 240.

To change this value, perform the following steps:

1. Select **Servers** → **Application Servers**.
2. Select the application server you want to tune.
3. Under Additional Properties, select **ORB Service**.
4. Update the Connection cache maximum field and click **OK**.
5. Click **Apply** to save the changes.
6. Restart the application server.

### ***ORB thread pool size***

Refer to “Enterprise JavaBeans container” on page 257 to learn about the ORB thread pool size property.

## **9.4.13 Extensible Markup Language parser selection**

Add Extensible Markup Language (XML) parser definitions to the `jaxp.properties` file and `xerces.properties` file found in the `<WAS_HOME>/jre/lib` directory to help facilitate server startup. You may have to change the `XMLParserConfiguration` value since new versions of Xerces are provided.

In both files, insert the lines shown in Example 9-12.

### ***Example 9-12 XML parser definitions***

---

```
javax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFact  
oryImpl  
javax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentB  
uilder  
FactoryImpl  
org.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.p  
arsers.  
StandardParserConfiguration
```

---

## **9.4.14 Transaction service settings: Transaction log**

When an application running on WebSphere accesses more than one resource, WebSphere stores transaction information to properly coordinate and manage the distributed transaction. In a higher transaction load, this persistence slows down performance of the application server due to its dependency on the operating system and the underlying storage systems.

To achieve better performance, move the transaction log files to a storage device with more physical disk drives, or preferably RAID disk drives. When the log files are moved to the file systems on the RAID disks, the task of writing data to the physical media is shared across the multiple disk drives. This allows more concurrent access to persist transaction information and faster access to that data from the logs. Depending on the design of the application and storage subsystem, performance gains range from 10% to 100%, or even more in some cases.

This change is applicable only to the configuration where the application uses distributed resources or XA transactions, for example, multiple databases and

resources are accessed within a single transaction. Consider setting this property when the application server shows one or more of the following signs:

- ▶ CPU utilization remains low despite an increase in transactions.
- ▶ Transactions fail with several timeouts.
- ▶ Transaction rollbacks occur with “unable to enlist transaction” exception.
- ▶ Application server hangs in the middle of a run and requires the server to be restarted.
- ▶ The disk on which an application server is running shows higher utilization.

Use the WebSphere Administrative console to change the location of the transaction logs.

1. Select **Servers** → **Application Servers** → **AppServer name**.
2. From the Additional Properties pane, select **Transaction Service**.
3. In the Transaction log directory field, enter the new path. By default, the transaction log has a size of 1 MB and is stored in the directory `WAS_HOME/tranlog/AppServer name`.

We recommend that you create a file system with at least three to four disk drives RAIDed together in a RAID-0 configuration. Then, create the transaction log on this file system with the default size. When the server is running under load, check the disk input and output. If disk input and output time is more than 5%, consider adding more physical disks to lower the value. If disk input and output is low, but the server is still high, consider increasing the size of the log files.

### 9.4.15 Additional reference materials

Consult the following resources for additional information about performance tuning and WebSphere Application Server:

- ▶ IBM WebSphere Application Server Library, including monitoring and troubleshooting documentation  
<http://www.ibm.com/software/webservers/appserv/was/library/index.htm>
- ▶ WebSphere Application Server white papers  
<http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&dc=DA480+DB100&dtm>
- ▶ iSeries performance documents, including *WebSphere Application Server for iSeries Performance Considerations* and links to the PTDV tool, Workload Estimator tool, and other documents

<http://www.ibm.com/servers/eserver/series/software/websphere/wsappserver/product/PerformanceConsiderations.html>

- ▶ *IBM WebSphere Application Server Advanced Edition Tuning Guide* (Version 4.02)

[http://www.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/was/pdf/nav\\_Tuneguide.pdf](http://www.ibm.com/software/webservers/appserv/doc/v40/ae/infocenter/was/pdf/nav_Tuneguide.pdf)

- ▶ *J2EE Application Development: One or many applications per application server?* (Nov. 2002)

[http://www.ibm.com/developerworks/websphere/techjournal/0211\\_alcott/alcott.html](http://www.ibm.com/developerworks/websphere/techjournal/0211_alcott/alcott.html)

- ▶ *Handling Static content in WebSphere Application Server* (Nov. 2002)

[http://www.ibm.com/developerworks/websphere/techjournal/0211\\_brown/brown.html](http://www.ibm.com/developerworks/websphere/techjournal/0211_brown/brown.html)

- ▶ WebSphere Application Server zone on from IBM developerWorks

<http://www.ibm.com/developerworks/websphere/zones/was/>



# Performance tuning of WebSphere MQ

This chapter provides assistance in performance tuning WebSphere MQ that is installed on the in-store processor as part of the WebSphere Remote Server software stack.

**Attention:** The performance and tuning information presented in this chapter was developed and documented in a WebSphere Remote Server V5.1.2.1 environment.

This chapter contains the following sections:

- ▶ 10.1, “Performance factors” on page 316
- ▶ 10.2, “Types of WebSphere MQ applications” on page 317
- ▶ 10.3, “Optimization techniques” on page 318

## 10.1 Performance factors

*Performance* is generally defined as response time. From the end-user perspective, response time is the time taken for the results to be visible from the time an application is triggered.

The key elements that impact the response time for a WebSphere MQ application include:

- ▶ Processing time
- ▶ I/O time
- ▶ Network transport time
- ▶ Message contention

### 10.1.1 Processing time

Processing time is the time required for the central processing unit (CPU) to service the application plus any operating system overhead. The major component of a WebSphere MQ application that requires processing is the type and number of Message Queue Interface (MQI) calls. The calls in order of CPU requirements are:

- ▶ **MQCONN**: This call creates a connection to the queue manager and the required task structures and control blocks.
- ▶ **MQOPEN**: This call opens a specific queue for processing. It acquires control blocks and may lock the required resources.
- ▶ **MQCLOSE**: This call closes the queue and commits the resources. It frees all locks and releases control blocks.
- ▶ **MQPUT**: This puts a message on a queue. It may require recovery processing.
- ▶ **MQGET**: This gets a message from a queue. It may require recovery processing.

On OS/390® platforms, most CPUs are charged to the calling application. On distributed systems, an agent process is used to communicate to WebSphere MQ. The agent consumes most of the CPU cycles.

### 10.1.2 I/O time

This is the time that the application spends waiting for I/O operations to be completed. This can be a major component of a WebSphere MQ application, since all processed messages must be logged to ensure guaranteed delivery.



WebSphere MQ must ensure that the log is committed prior to the completion of a work unit.

### 10.1.3 Network transport time

A key attribute of WebSphere MQ is its ability to tie together multiple disparate platforms. Since the vast majority of messages are transferred across a network, speed, volume, and reliability are critical to the performance of WebSphere MQ applications.

### 10.1.4 Message contention

Message contention relates to the server's ability to process incoming messages at the rate in which they are received. If messages arrive faster than they can be processed, the incoming messages wait in the queue and thus, increase the overall response time. Depending on the application design, additional servers may be added as the demand increases.

## 10.2 Types of WebSphere MQ applications

The different types of WebSphere MQ applications each of these unique performance characteristics. This section describes several application types and their particular attributes with regard to performance.

- ▶ **Asynchronous send**

An asynchronous send application sends messages based on some external activity. This type of application is primarily concerned with throughput because it needs to keep up with the number of events coming in from the external source. There is no concern about how long the message takes to reach its final destination.

- ▶ **Synchronous send and receive**

This is an asynchronous send application that expects a timely response to its message. The key performance aspect in this type of application is the response time to the sent messages.

- ▶ **Server**

The server application is complementary to the asynchronous and synchronous types of applications. The server processes WebSphere MQ messages, performs local processing, and may send responses. There can be multiple servers to share the workload.

- ▶ **Client**

A WebSphere MQ client can be an implementation of any of the previous applications. Since there is no local queue manager, all the requests should pass over the network to an associated server that acts as a queue manager. The components of a client's response time include the number of requests to be processed, the speed at which the request and response are delivered, and any additional processing time on the server.

## 10.3 Optimization techniques

This section provides options for improving the performance of a WebSphere MQ application. Unless you are upgrading the hardware, attempting to improve the performance of one application may lead to degraded performance in another. Ensure that you weigh the costs and benefits of each option before making any changes to a production environment.

### 10.3.1 Additional system resources

The most common approach to improving performance is to add additional resources. Moving the queue manager to a faster server with more memory improve the performance of an overburdened WebSphere MQ application.

### 10.3.2 Application design

Follow the techniques listed here to relate to the design of a WebSphere MQ application:

- ▶ Avoid unnecessary calls.
- ▶ Reduce message size.
- ▶ Compress messages.
- ▶ Reduce number of messages.
- ▶ Use intermittent commits for large numbers of messages.
- ▶ Use nonpersistent messages.

### 10.3.3 Logging

The following techniques relate to how WebSphere MQ logging is configured and administered.

- ▶ Separate logging from data volumes
- ▶ Distribute active and archive logs

### 10.3.4 WebSphere MQ configuration

The following techniques relate to the configuration of WebSphere MQ to optimize performance.

#### Channel parameters

There are two main channel tuning parameters:

- ▶ Batch size
- ▶ Batchint

*Batch size* defines the maximum number of messages sent within a batch. Batching records together reduce the amount of channel processing required. If an associated transmission queue is empty, WebSphere MQ ends the batch. However, depending on the message arrival rate, use *batchint* to indicate that WebSphere MQ should wait before ending the batch.

If a message arrives within the specified time span, it is added to the batch, and so on, until the batch size is reached. If the message rate is low, this causes an unacceptable transmission delay. Consider an application that sends a single message to a remote server and expects a single reply. If the batchint value is one second for both channels, the minimum turnaround for the reply is two seconds.

It is important that none of the messages are visible on the remote queue manager until the entire batch is sent and committed, thus increasing the batch size results in message spikes.

#### Fast messages

The primary benefit of using fast messages is that nonpersistent messages bypass significant channel processing. Without fast message support, nonpersistent messages are logged and included within channel recovery processing. Since nonpersistent messages are lost in the case of a queue manager restart, the overhead of channel recovery could be considered unnecessary.

When active, fast message support transfers nonpersistent messages over the channel outside the normal batch process. Thus, the message does not count against batch values. Since the message is placed outside the batch, it is immediately visible on the receiving side, creating the possibility of nonpersistent messages being processed out of order, along with persistent messages.

Fast message support can be activated on a channel-by-channel basis. They must be active on both the sender and the receiver channels.



# Performance tuning of DB2 UDB Workgroup Server

This chapter provides assistance in performance tuning for the DB2 UDB Workgroup Server that is installed on the in-store processor as part of the WebSphere Remote Server software stack.

This chapter contains the following sections:

- ▶ 11.1, “Overview of performance tuning” on page 322
- ▶ 11.4, “Monitoring and tuning tools” on page 337
- ▶ 11.5, “Application tuning” on page 356
- ▶ 11.6, “System tuning” on page 375

## 11.1 Overview of performance tuning

Performance tuning begins at the design phase. You must specify the performance goals of both the system and the application in the project requirements. You must also consider the performance requirements in the design and implement them while the application is being developed.

In this chapter, we show you the performance considerations and tuning methods of a database and an application starting from the design stage. We introduce some tools that are useful in monitoring and tuning the system. These tools are all part of DB2 Universal Database (UDB), so there is no need to spend extra money buying these software. Then we discuss database- and application-related performance topics that database administrators (DBAs) and developers should consider while building and developing the system.

Finally, we show how you to administer a running system. We show you where you can find the information that you need to monitor your running system. And we describe what you should look at and how to tune it if there is problem.

## 11.2 Performance related aspects of DB2 architecture

In this section, we provide a deeper view of the performance-related aspects of the DB2 UDB architecture. This prepares you for the sections later in this chapter where we show you how to tune the database server.

### 11.2.1 Buffer pool

When an application accesses a row of a table for the first time, the database manager places the page containing that row in the buffer pool. The next time any application requests data, the database manager first looks for the buffer pool. If the requested data is in the buffer pool, it can be retrieved without disk access, resulting in faster performance.

The buffer pool is an important area for data accessing performance. It is a good practice to have as much data as possible that the application frequently needs in the buffer pool. However, allocating excessive amounts of memory for the buffer pool for an application can impact the system performance. In 11.6.1, “Tuning the buffer pools” on page 375, we show you how to monitor and tune the buffer pool to achieve your performance goal.

A buffer pool is assigned to one or many table spaces. You may define more than one buffer pool in a database. If you have a table with data that is constantly

used, it may be useful to place this table in a tablespace with its own buffer pool to make the data memory resident.

With DMS table spaces, the table data, indexes, and long data can be placed in separate table spaces. This makes it possible to have a separate buffer pool for an index to keep the index memory resident and improve the performance.

By default the prefetcher reads contiguous pages from disk and may write them into non-contiguous pages in the buffer pool. With DB2 V8, you can configure a *block-based buffer pool*. When this feature is activated (during creation of a buffer pool or with the ALTER buffer pool command), the contiguous pages from disk are written into contiguous pages of the buffer pool, when available.

We recommend that you use a separate buffer pool for temporary table spaces. This increases performance for queries that require temporary storage, especially sort-intensive queries.

**Note:** The default buffer pool size is small, therefore, there is a need to tune. In UNIX systems, the default size is 1000 4 K pages. In Windows systems, it is 250 4 K pages.

### 11.2.2 Asynchronous read/write

Input/output (I/O) is a complex part and important for performance. As described earlier, it is good to have prefetchers running. These agents are responsible for accessing data before an application needs it. Processes known as *I/O servers* handle the prefetching. The number of processes can be configured using the configuration parameter NUM\_IOSERVERS. It is better to configure more I/O servers than needed than not to have enough of them. The impact of having excessive I/O servers is almost none because the memory used for the I/O servers is paged out. However, the performance can be dramatically decreased if I/O servers are not enough.

Figure 11-1 illustrates the steps of how DB2 UDB uses the I/O server and prefetching process. It illustrates the process starting from a client request for data pages until DB2 brings the data from the database server back to the client.

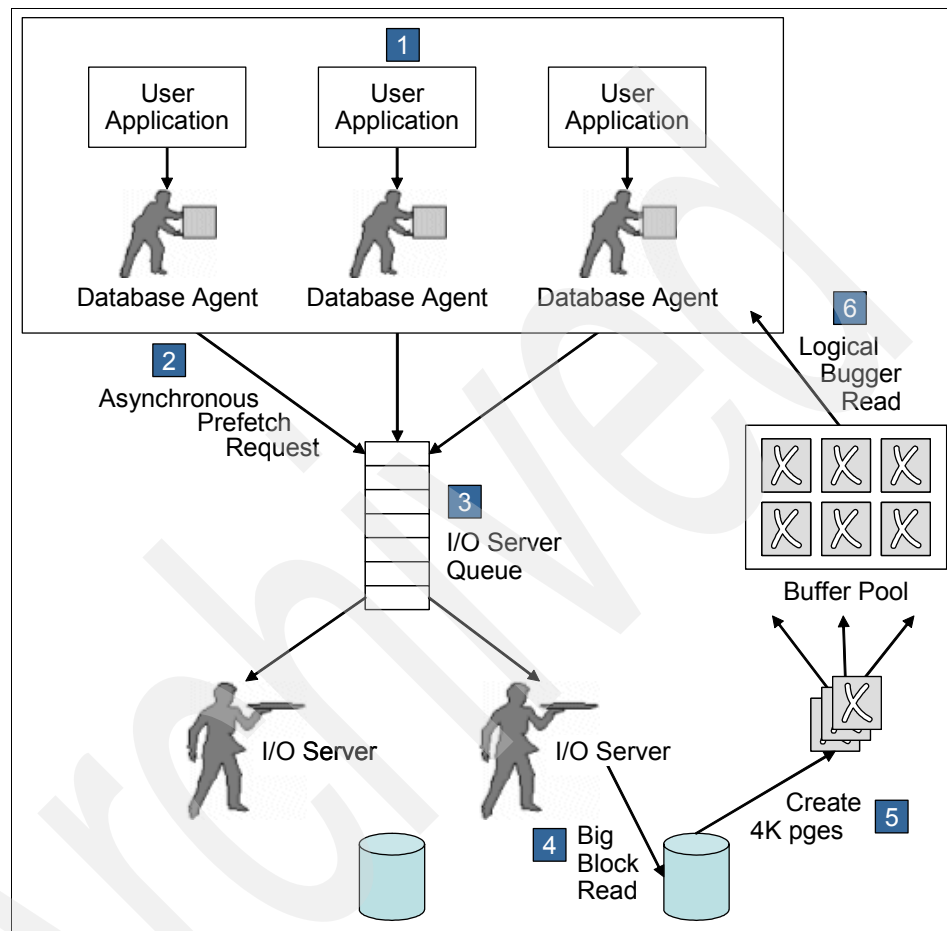


Figure 11-1 Prefetching data using I/O servers

The actions shown in Figure 11-1 follow this sequence:

1. The user application passes the Structured Query Language (SQL) request to the database agent that has been assigned to the user application by the database manager.
2. The database agent determines that prefetching should be used to obtain the data required to satisfy the SQL request.
3. Then the database agent writes a prefetch request to the I/O server queue.
4. The first available I/O server reads the prefetch request from the queue.



5. Then the I/O server reads the data from the tablespace into the buffer pool.  
The number of I/O servers that can fetch data from a tablespace at the same time depends on the number of prefetch requests in the queue and the number of I/O servers configured by the NUM\_IOSERVERS database configuration parameter.
6. The database agent performs the necessary operations on the data pages in the buffer pool and returns the result to the user application.

### 11.2.3 Tablespaces and containers

A tablespace can have several containers. The data is distributed in a round-robin technique across the available containers. Figure 11-2 shows how the extents are allocated. Extent 0 is allocated in the first container, extent 1 in the second container, and so forth.

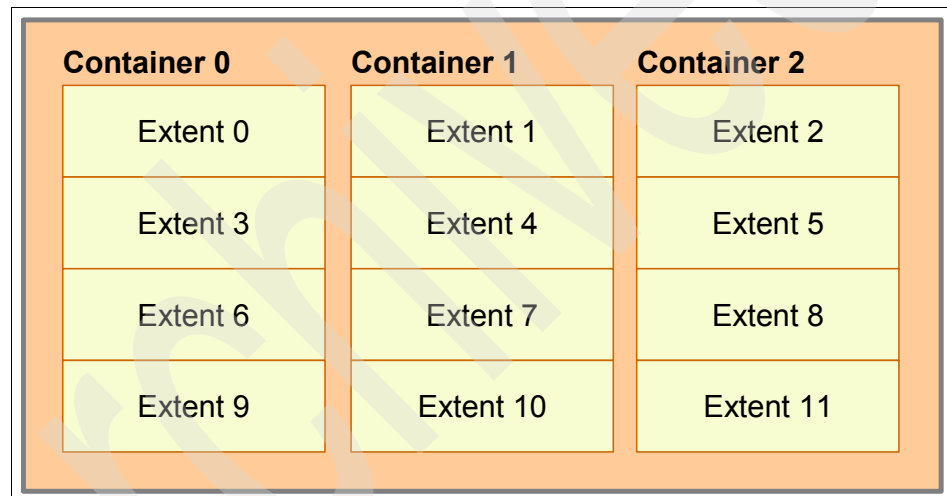


Figure 11-2 Distribution of extents in a container

DB2 V8 introduces some new container management features. For example, you can now alter the size of a container. You can also turn off rebalancing while adding a container. In the previous version, the container rebalance process was automatic. This process can impact overall system performance if the tablespace is big. For detailed information about how to add or extended the containers, refer to Chapter 5 in the *IBM DB2 UDB Administration Guide: Planning V8.2*, SC09-4822.

It is important to distribute the container over several disks to allow the database manager to do parallel I/O. If you are using striped devices such as Redundant

Array of Independent Disks (RAID) devices, we recommend that you use only one container per tablespace. The size of an extent should be:

$n \times \text{RAID stripe size}$

In this equation,  $n$  is greater than or equal to 1.

The prefetch size should be:

$(n \times \text{extentsize}) + (n \times \text{RAID stripe size} \times \text{count of devices})$

Again,  $n$  must be greater than or equal to 1.

**Note:** DB2 cannot determine if a RAID device is used. It is important to set the DB2\_PARALLEL\_IO registry variable to enable parallel I/O. DB2\_PARALLEL\_IO=\* enables parallel I/O for all table spaces.

## 11.2.4 Database agents

Agents are processes, and they need system resources. For a good performance, it is necessary to handle database agents economically with system resources. The number of available agents depends on the database manager configuration parameters MAXAGENTS and NUM\_POOLAGENTS.

For Internet applications with many relatively transient connections, or similar kinds of applications, the connection concentrator improves performance by allowing many more client connections to be processed efficiently. It also reduces memory use for each connection and decreases the number of context switches.

The connection concentrator is new in DB2 UDB V8 and works like a transaction monitor. Therefore, there is no more need to buy a commercial product to reduce the number of connections to the database. It is now possible to handle thousands of concurrent connections with hundreds of agents.

**Note:** The connection concentrator is enabled when the value of the database manager parameter MAX\_CONNECTIONS is greater than the value of the DBM parameter MAX\_COORDAGENTS.

Figure 11-3 shows the concept of the connection concentrator. The left side of the figure shows that, without the connection concentrator, the number of coordinator agents is as many as the number of client connections. This consumes a lot of system resources if there are many connections active on a database server. When the connection concentrator is activated, the number of coordinator agents decreases because the coordinator agents are within a pool

and accessible for different connections. When the number of coordinator agents shrinks, the number of subagents also reduces, as shown in right side of Figure 11-3.

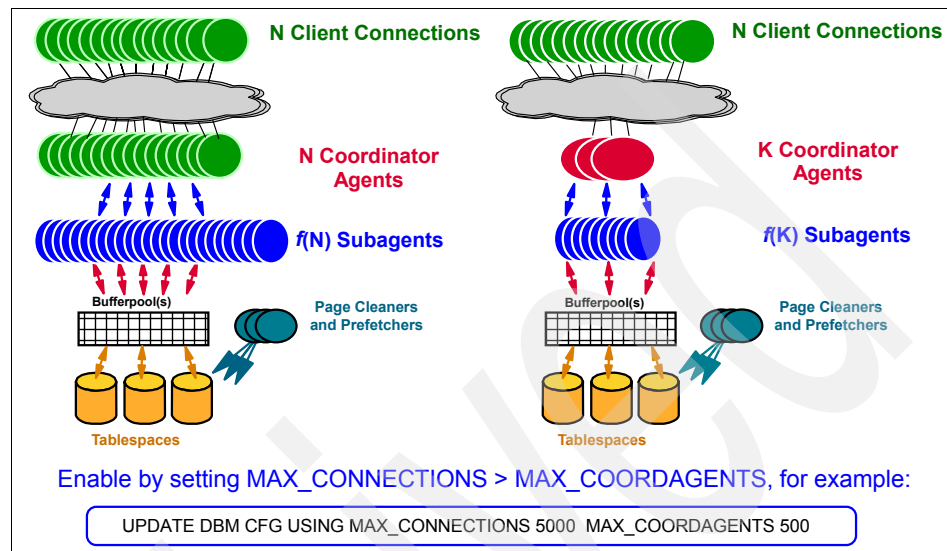


Figure 11-3 Connection concentrator concept

## 11.2.5 Concurrency

Concurrency can be a reason for an application with poor performance. DB2 UDB places locks on data used by an application according to the *isolation level*. How restrictive these locks are depends on the isolation level. The isolation level can be set for an application and depends on the kind and the needs of the application. These levels are from restrictive to non-restrictive:

- Repeatable Read (RR)

RR is the highest isolation level. Locks are held on all referenced rows of a result set within a unit of work. RR guarantees that no changes can be made on the referenced data until the unit of work terminates. For example, if you scan 10,000 rows and apply predicates to them, locks are held on all 10,000 rows, even though only 10 rows qualify.

- Read Stability (RS)

RS locks only the rows that an application retrieves in a unit of work. The Read Stability isolation level ensures that all returned data remains unchanged until the time the application sees the data, even when temporary tables or row blocking is used. Other applications can make changes to other parts, so if the cursor is repositioned, the result may be different.

► Cursor Stability (CS)

CS locks any row accessed by a transaction of an application while the cursor is positioned on the row. This lock remains in effect until the next row is fetched or the transaction is terminated. However, if any data on a row is changed, the lock must be held until the change is committed to the database.

► Uncommitted Read (UR)

UR allows applications to retrieve uncommitted data. Therefore, it may occur that the applications see changes or inserts that will become undone.

Table 11-1 summarizes the isolation levels.

*Table 11-1 Summary of isolation levels*

Isolation level	Access to uncommitted data	Nonrepeatable read	Phantom read phenomenon
RR	Not possible	Not possible	Not possible
RS	Not possible	Not possible	Possible
CS	Not possible	Possible	Possible
UR	Possible	Possible	Possible

The impact of the isolation level on the performance that may be seen in an application includes deadlocks and lock timeouts or lock waits. To learn how to set the isolation level, refer to 11.5, “Application tuning” on page 356. This referenced section also explains how to prevent a lock escalation.

Table 11-2 outlines the different locks that DB2 UDB uses.

*Table 11-2 Lock modes shown in order of increasing control over resources*

Lock mode	Applicable object type	Description
IN (Intent None)	Table spaces, tables	The lock owner can read any data in the table, including uncommitted data, but cannot update any of it. No row locks are acquired by the lock owner. Other concurrent applications can read or update the table.
IS (Intent Share)	Table spaces, tables	The lock owner can read data in the locked table, but not update this data. When an application holds the IS table lock, the application acquires an S or NS lock on each row read. In either case, other applications can read or update the table.

Lock mode	Applicable object type	Description
NS (Next Key Share)	Rows	The lock owner and all concurrent applications can read, but not update, the locked row. This lock is acquired on rows of a table, instead of an S lock, where the isolation level is either RS or CS on data that is read.
S (Share)	Rows, tables	The lock owner and all concurrent applications can read, but not update, the locked data. Individual rows of a table can be S locked. If a table is S locked, no row locks are necessary.
IX (Intent Exclusive)	Table spaces, tables	The lock owner and concurrent applications can read and update data in the table. When the lock owner reads data, an S, NS, X, or U lock is acquired on each row read. An X lock is also acquired on each row that the lock owner updates. Other concurrent applications can both read and update the table.
SIX (Share with Intent Exclusive)	Tables	The lock owner can read and update data in the table. The lock owner acquires X locks on the rows it updates, but acquires no locks on rows that it reads. Other concurrent applications can read the table.
U (Update)	Rows, Tables	The lock owner can update data in the locked row or table. The lock owner acquires X locks on the rows before it updates the rows. Other units of work can read the data in the locked row or table, but cannot attempt to update it.
NX (Next Key Exclusive)	Rows	The lock owner can read but not update the locked row. This mode is similar to an X lock, except that it is compatible with the NS lock.
NW (Next Key Weak Exclusive)	Rows	This lock is acquired on the next row when a row is inserted into the index of a non-catalog table. The lock owner can read but not update the locked row. This mode is similar to X and NX locks, except that it is compatible with the W and NS locks.
X (Exclusive)	Rows, tables	The lock owner can both read and update data in the locked row or table. Tables can be Exclusive locked, meaning that no row locks are acquired on rows in those tables. Only uncommitted read applications can access the locked table.

Lock mode	Applicable object type	Description
W (Weak Exclusive)	Rows	This lock is acquired on the row when a row is inserted into a non-catalog table. The lock owner can change the locked row. This lock is similar to an X lock except that it is compatible with the NW lock. Only uncommitted read applications can access the locked row.
Z (Superexclusive)	Table space, tables	This lock is acquired on a table in certain conditions, such as when the table is altered or dropped, an index on the table is created or dropped, or a table is reorganized. No other concurrent application can read or update the table.

## 11.2.6 SQL

In the following sections, we list some concepts with high performance issues and some tools introduced in version 8.

### Data types

Understanding your data and defining the proper data type to store the data is another element that affects performance. Improper data type design in an application can result in bad performance. For example, it is not a good practice to use VARCHAR whenever a string needs to be stored. Numbers should be stored in number fields and not in character fields. In 11.5.2, “SQL tuning” on page 367, we give advice on how to choose the correct data types.

### String data

This section describes the three string data types: CHAR, VARCHAR and LONG VARCHAR. Each character string is further defined as one of the following types:

- ▶ **Bit data:** Data that is not associated with a code page
- ▶ **Single-byte character set (SBCS) data:** Data in which every character is represented by a single byte
- ▶ **Mixed data:** Data that may contain a mixture of characters from a single-byte character set and a multi-byte character set (MBCS)

When defining the length of a string data type, it is important to keep in mind that the length represents the number of bytes and not the number of characters. If

you have a multi-byte code set, such as *Unicode*, you need to reserve more space.

- ▶ **CHAR(n)** specifies a fixed-length column for character string data. The maximum length is 254 bytes.
- ▶ **VARCHAR(n)** specifies a varying-length column for character string data. The maximum length of the string is 4000 bytes. If the length is greater than 254, the column is a long-string column.
- ▶ **LONG VARCHAR** specifies a varying-length column for character string data. The maximum length of a column of this type is 32700 bytes.

### ***Numeric data***

There are six data types that can be used to store numeric values. The data types are used to store different numeric types and precision. The data is stored using a fixed amount of storage for all numeric data types. The amount of storage required increases as the precision of the number increases.

- ▶ **Small integer (SMALLINT)** specifies a small integer. Values in a column of this type can range from -32768 through +32767.
- ▶ **Large integer (INTEGER)** specifies a large integer. Values in a column of this type can range from -2147483648 through +2147483647.
- ▶ **Big integer (BIGINT)** is available to store 64-bit integers and can range from -9,223,372,036,854,775,808 to +9,223,372,036,775,807. When platforms include native support for 64-bit integers, the processing is much faster than with decimal, and more precise than double or real.
- ▶ **Single-precision floating-point (REAL)** is a 32-bit approximation of a real number. The number can be zero or can range from -3.402E+38 to -1.175E-37, or from 1.175E-37 to 3.402E+38.
- ▶ **Double-precision floating-point (DOUBLE or FLOAT)** specifies a floating-point number that is 64 bits long. Values in a column of this type can range from -1.79769E+308 to -2.225E-307 or +2.225E-307 to +1.79769E+308, or they can be 0.
- ▶ **Decimal (DECIMAL or NUMERIC)** specifies a mainframe packed decimal number with an implicit decimal point. The position of the decimal point is determined by the precision and scale of the number. The scale, which is the numbers to the right of the decimal point, cannot be negative or greater than the precision. The maximum precision is 31 digits. Note that numbers that require decimal precision greater than 15 digits may be subject to rounding and conversion errors.

## Dates, times, and timestamps

Three DB2 data types represent dates and times:

- ▶ DATE specifies date values in various formats, as determined by the country code of the database.
- ▶ TIME specifies time values in a three-part format. The values range from 0 to 24 for hours (hh) and from 0 to 59 for minutes (mm) and seconds (ss).
- ▶ TIMESTAMP combines a date and time and adds an optional microsecond to make a seven-part value of the format yyyy-mm-dd-hh.mm.ss[.nnnnnn].

## Indexes

Indexes are used to speed up the retrieval of data. An index usually takes less space than a table and has a tree structure to reduce the lookup time. But indexes also need space from your storage. What indexes need to be created depends on the SQL the application uses. We introduce a tool later in this chapter that helps you to determine which indexes may be useful.

DB2 Version 8 adds support for type-2 indexes. Here are the main advantages of type-2 indexes:

- ▶ They improve concurrency because the *next-key locking* is reduced to a minimum. Most next-key locking is eliminated by marking the key as having been deleted instead of physically removing the key from the index page.
- ▶ An index can be created on columns that have a length greater than 255 bytes.
- ▶ The *in-place table reorg* and online table load can be used against a table that has only type-2 indexes defined on it. They are required for the new multidimensional clustering facility.

**Attention:** All new indexes are created as type-2 indexes, except when you add an index on a table that already has type-1 indexes. In this case, the new index will also be a type-1 index, because you cannot mix type-1 and type-2 indexes on the same table.

All indexes created before DB2 Version 8 are type-1 indexes. Use the REORG INDEXES command to convert type-1 indexes to type-2 indexes. Use the INSPECT command to ascertain the type of index defined on a table. After this conversion, run the **runstats** command.

## Stored procedures

Store procedures can help to reduce network traffic. They can contain several SQL commands and some business logic. Stored procedures are stored in the database server. Calling a stored procedure takes only one call command, and



the result is sent back to the application after executing the SQLs in a stored procedure at the host. If you run all the SQL commands within your application, the communication between your application and the database is much higher.

## **Multidimensional clustering**

Multidimensional clustering is a new feature of DB2 UDB V8. It enables a table to be physically clustered on more than one key or dimension simultaneously. In versions prior to Version 8, DB2 UDB only supports single-dimensional clustering of data via clustering indexes. Using a clustering index, DB2 UDB attempts to maintain the physical order of data on pages in the key order of the index as records are inserted and updated in the table. Clustering indexes greatly improves the performance of range queries that have predicates containing the key (or keys) of the clustering index. With good clustering, only a portion of the table needs to be accessed, and, when the pages are sequential, more efficient prefetching can be performed.

With multidimensional clustering, these benefits are extended to more than one dimension or clustering key. In the case of query performance, range queries involving any, or a combination of, specified dimensions of the table will benefit from clustering. These queries will access only those pages having records with the correct dimension values, and the qualifying pages will be grouped by extents. Furthermore, although a table with a clustering index can become unclustered over time as space fills up in the table, a multidimensional clustering table can maintain its clustering over all dimensions automatically and continuously. By maintaining its clustering, the table eliminates the need for reorganization to restore the physical order of the data.

## **Declared temporary tables**

A declared temporary table is defined by the `DECLARED GLOBAL TEMPORARY TABLE` statement. Declared temporary tables are used to store data that does not need to be persistent. A declared temporary table is a temporary table that is only accessible to SQL statements that are issued by the application that created the temporary table. A declared temporary table does not persist beyond the duration of the connection of the application to the database. The creation of a temporary table is not stored in the catalog. In comparison to regular tables, DB2 does not lock declared temporary tables or their rows. Also, if you specify the `NOT LOGGED` parameter when you create it, DB2 does not log declared temporary tables or their contents.

In DB2 UDB V8, a new feature is added to the declared temporary tables that allows the creation of an index on temporary tables. In versions prior to version 8, the temporary table needs to be scanned completely, which is time consuming if the table is big. With indexes, retrieving data from a temporary table is much

faster. For better optimization, it is now also possible to update the statistic of temporary tables with the **runstats** command.

## 11.2.7 Maintenance

Performance tuning is a constant task. An administrator can do maintenance tasks regularly to keep good performance.

### Runstats

DB2 UDB has an optimizer that looks for the best way to retrieve data. The optimizer uses statistical data to calculate the lowest price for retrieval. With the **runstats** command, the statistical data is collected and stored in the catalog. The optimizer knows, for example, how many rows a table has and can determine if an index scan is useful or to just use a table scan.

The statistic data must be updated from time to time. If you have tables with many changes, run the **runstats** command at regular intervals.

### Reorg

In tables where data is frequently inserted and deleted, the tablespace becomes more and more fragmented. This has an impact on the performance. To defragment the data, DB2 UDB offers the **reorgchk** command, to determine if a table requires re-organization, and the **reorg** command, to organize the data.

### Rebind

When using static SQL, the best access plan is determined during the bind time and saved as a package. If your data changes and you run the **runstats** command, the static code that was bound to the server does not pick up the new statistic. Therefore, it is necessary to perform a rebind to force a new calculation of the access plan.

## 11.2.8 Application design

Keep in mind that the tuning of the server is important. However, even the best-tuned server is slow if the database design or the application is badly designed.

The physical and logical designs of the database and the application are also critical for performance. The database needs to become normalized to eliminate redundant data. For performance, it is sometime useful to not have normalized data. We discuss this later in the chapter.

## 11.3 Performance optimization tools

DB2 UDB provides some tools to help you in several tasks, such as performance optimization and maintenances tasks. In the following sections, we introduce four of these tools.

### 11.3.1 IBM DB2 UDB Performance Expert for Multiplatforms

IBM DB2 UDB Performance Expert for Multiplatforms offers a comprehensive view that consolidates, reports, analyzes and recommends changes on DB2 UDB performance-related information. The tool includes a Performance Warehouse that stores performance data and analysis tools. It also includes a Buffer Pool Analyzer that collects data and provides reports on related event activity.

With online monitoring capabilities, DB2 UDB Performance Expert monitors DB2 Connect gateways, including application and system-related information. Building on IBM autonomic computing and on demand expertise, DB2 UDB Performance Expert also offers recommendations for system tuning to gain optimum throughput.

You can use DB2 UDB Performance Expert to perform the following tasks:

- ▶ Monitor the DB2 UDB database and DB2 applications online and by evaluating report sets
- ▶ View and examine the status of a DB2 instance and its applications while they are currently active, or investigate events and performance problems that happened in the past
- ▶ Monitor individual nodes or an entire system
- ▶ Recognize event exceptions and take appropriate actions by means of a user exit
- ▶ Obtain tuning recommendations
- ▶ Identify trends and anticipate potential bottlenecks

### 11.3.2 IBM DB2 UDB Recovery Expert for Multiplatforms

DB2 UDB Recovery Expert for Multiplatforms provides targeted, flexible, and automated recovery of database assets. Its environment is easier to use and enables even less experienced DBAs to successfully complete highly sophisticated and efficient recovery techniques in minimal time.

DB2 UDB Recovery Expert provides intelligent analysis and diagnostics of altered, incorrect, or missing database assets including tables, indexes, or data. It also automates the process of rebuilding those assets to a correct “point-in-time”, often without taking the database or e-business operations offline.

### **11.3.3 IBM DB2 High Performance Unload for Multiplatforms**

An IBM DB2 UDB tool that helps reduce the maintenance window is DB2 UDB High Performance Unload for Multiplatforms. This product gives customers a fast and efficient tool for unloading and extracting data for movement across the enterprise or for reorganization in place.

DB2 UDB High Performance Unload for Multiplatforms delivers high levels of parallelism and distribution of data across multiple logical or physical partitions. It supports DB2 UDB Enterprise Edition/Enterprise Server Edition (symmetric multiprocessing (SMP)). It also supports data that has been distributed across multiple logical or physical partitions on a massively parallel processing (MPP) system, such as DB2 Enterprise-Extended Edition/Database Partitioning Feature.

### **11.3.4 IBM DB2 UDB Table Editor for Multiplatforms**

IBM DB2 UDB Table Editor for Multiplatforms quickly and easily accesses, updates, and deletes data across multiple DB2 database platforms, including IBM Informix Dynamic Server 9.x. Through integration with the Control Center, or Web-based interfaces, the tool supports high function editing with full support for DB2 security and user profiles.

In addition to editing raw data, you can create editing forms with drag-and-drop graphical user interfaces. This ability gives end users the ability to perform more of the data maintenance activities, therefore, freeing up DBA resources.

### **11.3.5 IBM DB2 UDB Web Query Tool for Multiplatforms**

IBM DB2 UDB Web Query Tool for Multiplatforms makes it easy for users at all levels to access enterprise data for query, data comparison, or reporting through multiple interfaces. These interfaces can include traditional Web-based browsers, e-mail clients, and wireless devices such as personal digital assistants (PDAs), text pagers, and Wireless Application Protocol (WAP)-enabled cellular phones.

## 11.4 Monitoring and tuning tools

DB2 UDB offers several tools that help you monitor and tune your databases and applications. Before you start tuning your system, you must have an idea of what to tune. Monitoring helps you to determine what is going on in your system, for example, if you have a long-running transaction and need to know why it is running so long. One method to handle the problem is to try something and see if it helps. The better method is to monitor the system, pinpoint the problem, and resolve the problem.

Monitoring consumes system resources. Therefore, it is a good idea to do it within a test environment and not in a production system. If you cannot recreate the problem in a test environment or if the production system and the test system are not comparable, then start monitoring in production. Use monitoring only as long as you need to and then switch off the monitoring after you finish tuning.

The database manager parameter `MON_HEAP_SZ` configures the amount of memory reserved for monitoring within DB2 UDB. The default value is not designed for high monitor activities. Therefore, it might be necessary to change the parameter to a higher value if you plan to monitor.

### 11.4.1 Snapshot monitor

The *snapshot monitor* gives you the status of your database server for a specific point in time. DB2 UDB offers several monitors that can be turned on or off separately.

To view the available monitoring options and the status, use the command:

```
DB2 GET MONITOR SWITCHES
```

In Figure 11-4, you can see an example output. By default, the monitor switches are turned off except for the monitor for timestamp information that is turned on by default. If a switch is turned on, you can see from the output of the `DB2 GET MONITOR SWITCHES` command when the monitor was activated or reset.

**Note:** Any switch (except `DFT_MON_TIMESTAMP`) that is turned on instructs the database manager to collect monitor data related to that switch. Collecting additional monitor data increases database manager overhead that can impact system performance. Turning off the `dft_mon_timestamp` switch becomes important as central processing unit (CPU) utilization approaches 100 percent.

The values collected from the monitor output are accumulated from the moment the monitor switch is activated or reset. The time displayed behind an active monitor shows the timestamp for the monitor that was activated. With the reset command this time is not updated.

```

$ db2 get monitor switches

      Monitor Recording Switches

Switch list for db partition number 0
Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information                 (LOCK) = OFF
Sorting Information              (SORT) = OFF
SQL Statement Information        (STATEMENT) = OFF
Table Activity Information        (TABLE) = OFF
Take Timestamp Information        (TIMESTAMP) = ON  10-24-2003 10:34:43.403708
Unit of Work Information         (UOW) = OFF

$

```

Figure 11-4 Status of monitor switches for snapshot monitoring

To change the status of a monitor, change the switches with the command shown in Example 11-1.

*Example 11-1 Command syntax for UPDATE MONITOR SWITCHES*

---

```

UPDATE MONITOR SWITCHES USING {switch-name {ON | OFF} ...} [AT
DBPARTITIONNUM db-partition-number | GLOBAL]
switch-name: BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, TIMESTAMP, UOW

```

---

Example 11-2 shows how to turn on the monitor switches that are turned off by default. Any combination is possible with ON or OFF.

*Example 11-2 Command to activate snapshot monitoring*

---

```

-- activates all monitor switches except TIMESTAMP that is on by
default
UPDATE MONITOR SWITCHES USING BUFFERPOOL ON LOCK ON SORT ON STATEMENT
ON TABLE ON UOW ON;

```

---

The switches for snapshot monitoring are also part of the database manager configuration. In the *Default database monitor switches* section of the DB2 GET DBM CFG command output, you can see the startup configuration of the monitor switches. Using the command UPDATE DBM CFG USING ..., you can configure the monitor switches to be active from the startup of the server. You should turn

on a switch in the configuration file only if you want to collect data starting from the moment the database manager is started. Otherwise, each monitoring application should set its own switches, and the data collected is relative to the time its switches are set.

In Table 11-3, you see the monitor switches along with the information that they provide. The DBM parameters shown in the table are online configurable. No **db2stop** and **db2start** commands are needed to activate the changes.

*Table 11-3 Data returned by the snapshot monitor*

Group	Information provided	Monitor switch	DBM parameter
Buffer pools	Number of reads and writes from and to the buffer pool and disk; time taken	BUFFERPOOL	DFT_MON_BUFPOOL
Locks	Number of locks held; number of deadlocks; what is locked and which lock mode is used	LOCK	DFT_MON_LOCK
Sorts	Number of heaps used; sort overflows; performance	SORT	DFT_MON_SORT
SQL statements	Start and stop time	STATEMENT	DFT_MON_STMT
Tables	Rows read; rows written	TABLE	DFT_MON_TABLE
Timestamps	Timestamp information	TIMESTAMP	DFT_MON_TIMESTAMP
Unit of work	Start and end time; completion status	UOW	DFT_MON_UOW

To retrieve the collected information, use the DB2 GET SNAPSHOT command, which is quite complex. Example 11-3 shows the complete syntax.

*Example 11-3 Command syntax for get snapshot*

---

```
>>-GET SNAPSHOT FOR----->

>--+--DATABASE MANAGER--+--WRITE TO FILE ----->
| +-DB MANAGER-----+
| |-DBM-----'
+-ALL--+-----+--DATABASES-----+
|      '-DCS-'
+-ALL--+-----+--APPLICATIONS-----+
|      '-DCS-'
+-ALL BUFFERPOOLS-----+
+-+-----+--APPLICATION--+--APPLID--appl-id-----+--+
| '-DCS-'          '-AGENTID--appl-handle-' |
+-FCM FOR ALL DBPARTITIONNUMS-----+
+-LOCKS FOR APPLICATION--+--APPLID--appl-id-----+--+
|          '-AGENTID--appl-handle-' |
+-ALL REMOTE_DATABASES-----+
+-ALL REMOTE_APPLICATIONS-----+
'-+--ALL-----+--ON--database-alias---'
+-+-----+--+--DATABASE--+--+
| '-DCS-' '-DB-----' |
+-+-----+--APPLICATIONS--+
| '-DCS-' |
+-TABLES-----+
+-TABLESPACES-----+
+-LOCKS-----+
+-BUFFERPOOLS-----+
+-REMOTE_DATABASES-----+
+-REMOTE_APPLICATIONS-----+
'-DYNAMIC SQL-----'

>--+-----+-----><
+-AT DBPARTITIONNUM--db-partition-number--+
'-GLOBAL-----'
```

---

The information that you can receive from the snapshot is extensive. Based on the options, you can take a snapshot from several points of view, starting from a snapshot of the database manager with a global view to a granular snapshot of dynamic SQL statements. You can also take snapshots when the monitor switches are switched off. Some information is collected all of the time, and for



those that are not collected by default, the values are marked NOT COLLECTED.

Example 11-4 shows snapshot output for a database manager. For more details, we recommend that you use the snapshot monitor and online documentation.

*Example 11-4 Output of a DBM snapshot*

---

Database Manager Snapshot

Node name	=
Node type	= Enterprise Server Edition with local
and remote clients	
Instance name	= db2inst1
Number of database partitions in DB2 instance	= 1
Database manager status	= Active
Product name	= DB2 v8.1.1.24
Service level	= s030728 (U488481)
Private Sort heap allocated	= 0
Private Sort heap high water mark	= 277
Post threshold sorts	= 0
Piped sorts requested	= 4
Piped sorts accepted	= 4
Start Database Manager timestamp	= 10-24-2003 10:34:43.403708
Last reset timestamp	= 11-05-2003 09:35:14.167372
Snapshot timestamp	= 11-05-2003 13:16:11.390812
Remote connections to db manager	= 4
Remote connections executing in db manager	= 0
Local connections	= 0
Local connections executing in db manager	= 0
Active local databases	= 1
High water mark for agents registered	= 13
High water mark for agents waiting for a token	= 0
Agents registered	= 13
Agents waiting for a token	= 0
Idle agents	= 8
Committed private Memory (Bytes)	= 4947968
Switch list for db partition number 0	
Buffer Pool Activity Information (BUFFERPOOL)	= ON 11-05-2003 09:33:14.184819

Lock Information	(LOCK)	= ON	10-24-2003 11:08:45.878709
Sorting Information	(SORT)	= ON	11-05-2003 09:33:14.184821
SQL Statement Information	(STATEMENT)	= ON	11-05-2003 09:33:14.184815
Table Activity Information	(TABLE)	= ON	11-05-2003 09:33:14.184817
Take Timestamp Information	(TIMESTAMP)	= ON	10-24-2003 10:34:43.403708
Unit of Work Information	(UOW)	= ON	11-05-2003 09:33:14.184812

Agents assigned from pool	= 355
Agents created from empty pool	= 15
Agents stolen from another application	= 0
High water mark for coordinating agents	= 13
Max agents overflow	= 0
Hash joins after heap threshold exceeded	= 0

Total number of gateway connections	= 0
Current number of gateway connections	= 0
Gateway connections waiting for host reply	= 0
Gateway connections waiting for client request	= 0
Gateway connection pool agents stolen	= 0

#### Memory usage for database manager:

Memory Pool Type	= Database Monitor Heap
Current size (bytes)	= 311296
High water mark (bytes)	= 344064
Maximum size allowed (bytes)	= 540672

Memory Pool Type	= Other Memory
Current size (bytes)	= 5423104
High water mark (bytes)	= 5439488
Maximum size allowed (bytes)	= 15958016

---

Example 11-5 shows output of a snapshot for a database.

*Example 11-5 Output of a database snapshot*

---

Database Snapshot

Database name	=	TRADE3DB
Database path	=	
/home/db2inst1/db2inst1/NODE0000/SQL00002/		
Input database alias	=	TRADE3DB
Database status	=	Active
Catalog database partition number	=	0
Catalog network node name	=	
Operating system running at database server	=	AIX
Location of the database	=	Remote
First database connect timestamp	=	10-24-2003 11:08:45.657390
Last reset timestamp	=	11-05-2003 09:35:14.167372
Last backup timestamp	=	
Snapshot timestamp	=	11-05-2003 13:51:35.622079
High water mark for connections	=	9
Application connects	=	4
Secondary connects total	=	1
Applications connected currently	=	4
Appls. executing in db manager currently	=	0
Agents associated with applications	=	4
Maximum agents associated with applications	=	9
Maximum coordinating agents	=	9
Locks held currently	=	0
Lock waits	=	0
Time database waited on locks (ms)	=	0
Lock list memory in use (Bytes)	=	3420
Deadlocks detected	=	0
Lock escalations	=	0
Exclusive lock escalations	=	0
Agents currently waiting on locks	=	0
Lock Timeouts	=	0
Number of indoubt transactions	=	0
Total Private Sort heap allocated	=	0
Total Shared Sort heap allocated	=	0
Shared Sort heap high water mark	=	0
Total sorts	=	4
Total sort time (ms)	=	0

Sort overflows	= 0
Active sorts	= 0
Buffer pool data logical reads	= 31
Buffer pool data physical reads	= 3
Asynchronous pool data page reads	= 0
Buffer pool data writes	= 0
Asynchronous pool data page writes	= 0
Buffer pool index logical reads	= 2
Buffer pool index physical reads	= 0
Asynchronous pool index page reads	= 0
Buffer pool index writes	= 0
Asynchronous pool index page writes	= 0
Total buffer pool read time (ms)	= 9
Total buffer pool write time (ms)	= 0
Total elapsed asynchronous read time	= 0
Total elapsed asynchronous write time	= 0
Asynchronous data read requests	= 0
Asynchronous index read requests	= 0
LSN Gap cleaner triggers	= 0
Dirty page steal cleaner triggers	= 0
Dirty page threshold cleaner triggers	= 0
Time waited for prefetch (ms)	= 5
Unread prefetch pages	= 0
Direct reads	= 0
Direct writes	= 0
Direct read requests	= 0
Direct write requests	= 0
Direct reads elapsed time (ms)	= 0
Direct write elapsed time (ms)	= 0
Database files closed	= 0
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
Host execution elapsed time	= 0.027284
Commit statements attempted	= 161
Rollback statements attempted	= 159
Dynamic statements attempted	= 293
Static statements attempted	= 2
Failed statement operations	= 0
Select SQL statements executed	= 209
Update/Insert/Delete statements executed	= 16

DDL statements executed	= 0
Internal automatic rebinds	= 0
Internal rows deleted	= 0
Internal rows inserted	= 0
Internal rows updated	= 0
Internal commits	= 0
Internal rollbacks	= 0
Internal rollbacks due to deadlock	= 0
Rows deleted	= 1
Rows inserted	= 3
Rows updated	= 12
Rows selected	= 551
Rows read	= 707
Binds/precompiles attempted	= 0
Log space available to the database (Bytes)	= 20400000
Log space used by the database (Bytes)	= 0
Maximum secondary log space used (Bytes)	= 0
Maximum total log space used (Bytes)	= 8393003
Secondary logs allocated currently	= 0
Log pages read	= 0
Log pages written	= 9
Package cache lookups	= 225
Package cache inserts	= 20
Package cache overflows	= 0
Package cache high water mark (Bytes)	= 3186052
Application section lookups	= 293
Application section inserts	= 20
Catalog cache lookups	= 48
Catalog cache inserts	= 0
Catalog cache overflows	= 0
Catalog cache high water mark	= 0
Workspace Information	
Shared high water mark	= 0
Corresponding shared overflows	= 0
Total shared section inserts	= 0
Total shared section lookups	= 0
Private high water mark	= 838854
Corresponding private overflows	= 0

Total private section inserts	= 20
Total private section lookups	= 205
Number of hash joins	= 0
Number of hash loops	= 0
Number of hash join overflows	= 0
Number of small hash join overflows	= 0
Memory usage for database:	
Memory Pool Type	= Backup/Restore/Util
Heap	
Current size (bytes)	= 16384
High water mark (bytes)	= 16384
Maximum size allowed (bytes)	= 20660224
Memory Pool Type	= Package Cache Heap
Current size (bytes)	= 4096000
High water mark (bytes)	= 4505600
Maximum size allowed (bytes)	= 4294950912
Memory Pool Type	= Catalog Cache Heap
Current size (bytes)	= 638976
High water mark (bytes)	= 638976
Maximum size allowed (bytes)	= 4294950912
Memory Pool Type	= Buffer Pool Heap
Current size (bytes)	= 4341760
High water mark (bytes)	= 4341760
Maximum size allowed (bytes)	= 4294950912
Memory Pool Type	= Buffer Pool Heap
Current size (bytes)	= 671744
High water mark (bytes)	= 671744
Maximum size allowed (bytes)	= 4294950912
Memory Pool Type	= Buffer Pool Heap
Current size (bytes)	= 409600
High water mark (bytes)	= 409600
Maximum size allowed (bytes)	= 4294950912
Memory Pool Type	= Buffer Pool Heap
Current size (bytes)	= 278528
High water mark (bytes)	= 278528
Maximum size allowed (bytes)	= 4294950912

Memory Pool Type	= Buffer Pool Heap
Current size (bytes)	= 212992
High water mark (bytes)	= 212992
Maximum size allowed (bytes)	= 4294950912
Memory Pool Type	= Lock Manager Heap
Current size (bytes)	= 458752
High water mark (bytes)	= 458752
Maximum size allowed (bytes)	= 638976
Memory Pool Type	= Database Heap
Current size (bytes)	= 1490944
High water mark (bytes)	= 1490944
Maximum size allowed (bytes)	= 6668288
Memory Pool Type	= Other Memory
Current size (bytes)	= 0
High water mark (bytes)	= 0
Maximum size allowed (bytes)	= 12517376

---

## 11.4.2 Event monitor

The *event monitor* is a tool that allows you to monitor ongoing actions in the database server. It is a good tool for checking problems that are difficult to handle using the snapshot monitor. One example is to find the reason for a deadlock situation.

A *deadlock event monitor* waits for a deadlock to occur. When a deadlock occurs, the monitor collects information about the applications involved and the locks in contention. Monitoring a deadlock situation with a snapshot monitor could be difficult. DB2 rolls back any transactions that are part of the deadlock situation except one that is allowed to finish the transaction. The timeframe for this action is very short. Therefore, a tool like Event monitor that records actions within a period of time is better than one that collects the state in a point of time.

To create an event monitor, use the following statement:

```
CREATE EVENT MONITOR SQL
```

Event monitors collect event data only when they are active. To activate or deactivate an event monitor, use the following statement:

```
SET EVENT MONITOR STATE SQL
```

The status of an event monitor (whether it is active or inactive) can be determined by the SQL function `EVENT_MON_STATE`.

Another way to create and control event monitors is by using the Control Center. Figure 11-5 shows the Control Center with the Event Monitors view open. This is a good way to see whether the monitors are running.

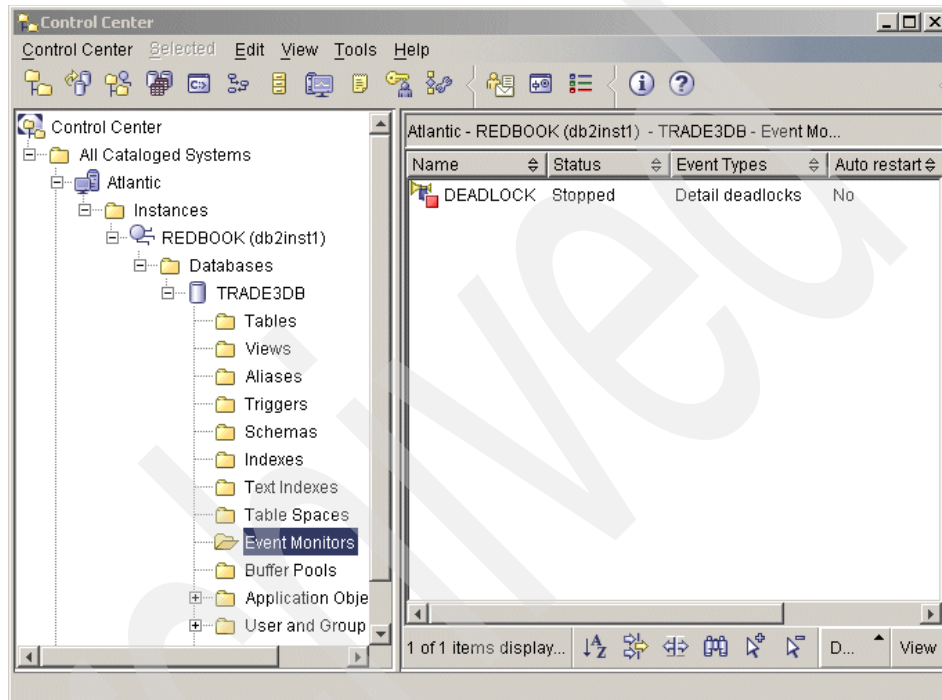


Figure 11-5 Event monitors from within Control Center

With DB2 UDB V8, you can store the monitor output in tables. Other options are to use files or named pipes. Be careful because event monitors gather a large amount of data. You should reserve enough space in the output directory or use tables to store the data. Using tables allows you to define exactly which data elements you want. Data that is not useful can be discarded. Once the data is stored in tables, you can use SQL to retrieve the information, which is a powerful and flexible way.



```

Example 11-6 Syntax of the CREATE EVENT MONITOR statement
>>-CREATE EVENT MONITOR--event-monitor-name--FOR----->

      .,-----|
      V
>--+--DATABASE-----+--+----->
      |
      | +-TABLES-----+
      | +-DEADLOCKS--+-----+
      | |               '-WITH DETAILS-' |
      | +-TABLESPACES-----+
      | '-BUFFERPOOLS-----'
      | '-CONNECTIONS--+--+-----+
      | +-STATEMENTS---+ '-WHERE--| Event Condition |-
      | '-TRANSACTIONS-'

>--*--WRITE TO--+--TABLE--| Table Options |-----+--*----->
      +-PIPE--pipe-name-----+
      '-FILE--path-name--| File Options |-

      .-MANUALSTART-.
>--+-----+--+----->
      '-AUTOSTART---'

                                     .-LOCAL--.
>--+-----+--+-----*--+----->
      '-ON DBPARTITIONNUM--db-partition-number-'      '-GLOBAL-'

Event Condition

      .-AND | OR-----|
      V
|-----+-----+--+--APPL_ID--+--+-----+--comparison-string--+--+
| '-NOT-' | +-AUTH_ID---+ | (1) |
|         | '-APPL_NAME-' | +-<-----+
|         |               | +->-----+
|         |               | | (1) |
|         |               | +->=-----+
|         |               | +-<-----+
|         |               | | (1) |
|         |               | +-<=-----+
|         |               | +-LIKE-----+

```



```
DB2 SELECT * FROM SYSCAT.EVENTTABLES WHERE EVMONNAME = 'dlmon'
```

Alternatively, use DB2 LIST TABLES and look for such tables as these:

- ▶ connheader\_dlmon
- ▶ conn\_dlmon
- ▶ deadlock\_dlmon
- ▶ dlconn\_dlmon
- ▶ dllock\_dlmon
- ▶ control\_dlmon

After you choose the table names, you can start analyzing the data that the event monitor has collected. For example, if you want to look for the long-running statements, use:

```
SELECT * FROM STMT_TEST WHERE START_TIME + 60 SECOND < STOP_TIME
```

If you chose to use a file to store the output of the event monitor, you can analyze the results by using the **db2evmon** command. This method is also useful but not as powerful as the one that uses SQL to analyze the collected data.

### 11.4.3 Explain utilities

When DB2 UDB is requested to retrieve data, it uses an optimizer to determine which access plan is the most efficient one. It is possible to see the access plan that DB2 UDB created and to view how expensive the retrieval is.

The optimizer uses several inputs to calculate the best access plan. Such information as the speed and count of CPUs and disks is important. DB2 calculates the CPU speed during the installation and stores it in the database manager configuration parameter CPUSPEED. Do not change this value unless you are modeling a different hardware environment. If you change the CPUs in a machine, you can set the value to -1 to have DB2 compute the new CPU speed. The access plans in your development environment can differ from the ones you have in the productive environment. If you deploy your database to a productive system, it is necessary to look at the access plans again. Access plans can differ due to speed and number of CPUs and disks changed.

By default, DB2 UDB assumes that you have very slow disks and calculates the access plan with that assumption. To have the best plan, you need to tell DB2 the characteristics of your disks. For detailed information, refer to Chapter 4 of the *Table space impact on query optimization in Administration Guide: Performance V8.2*, SC09-4821-01.


Other very important information that the optimizer uses is the statistics of the tables. It is necessary to keep this information up to date and to rebind your applications once the statistics are re-collected.

It is possible to configure how intensively DB2 UDB should search for the best access plan. You can choose between 10 optimization classes, where 0 means minimal optimization and 9 stands for using all available optimization techniques. By default, DB2 UDB uses the optimization class 5 configured with the database configuration parameter DFT\_QUERYOPT. You can change this default value or use DB2 SET CURRENT QUERY OPTIMIZATION = *value* when accessing the data with such statements as this:

```
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES WHERE QUERYOPT = CURRENT  
QUERY OPTIMIZATION.
```

The **db2exfmt** command and Visual Explain utilities require the DB2 Explain tables to be built for each user who is executing the explain commands. The **db2exp1n** utility dynamically explains static SQL and does not require the DB2 Explain tables. A variation of the **db2exp1n** utility, **dynexp1n**, allows dynamic SQL to be explained without using the DB2 Explain tables.

You can find further details about the **db2exp1n** and **dynexp1n** utilities by using the online help available within the DB2 Command Line Processor, by entering **db2exp1n -h** or **dynexp1n -h**. These two explain utilities can be useful if the Explain tables are not created or if a user needs to run an explain but does not own a DB2 Explain table schema.

The easiest way to create an access plan is from the Command Center by choosing the menu **Interactive → Create Access Plan** or by click the  icon. The explain tables are created automatically if they are not still present. A message informs you that the explain tables have been created. Another option to create explain tables is to run the EXPLAIN.DDL script that is located under sqllib\misc.

To view the access plan, you have several options. You can see the graphical version in the Command Center by choosing **Interactive → Create Access Plan**

from the menu bar. Figure 11-6 shows a simple example of the output for the statement:

```
SELECT * FROM ORDEREJB WHERE ORDERSTATUS = 'closed'
```

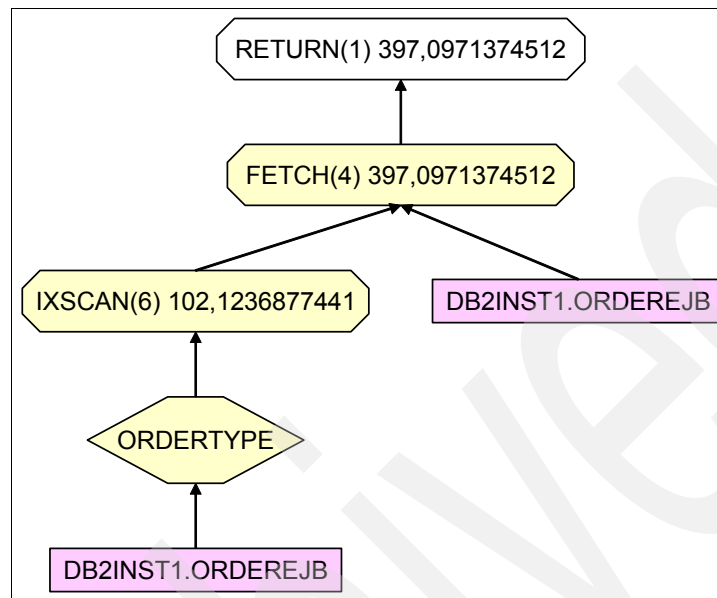


Figure 11-6 Visual Explain output

The **db2exfmt** utility formats the data contained in the DB2 Explain tables in a textual report that explains the access plans selected for each SQL statement that is contained in the DB2 Explain tables.

For more details, we recommend that you refer to the following DB2 UDB manuals:

- ▶ *IBM DB2 Universal Database Command Reference V8.2*, SC09-4828-01
- ▶ *IBM DB2 Universal Database Administration Guide: Performance V8.2*, SC09-4821-01

#### 11.4.4 DB2 diagnostic log (DB2DIAG.LOG)

DB2 UDB provides a diagnostic capturing mechanism. It writes information into a file called DB2DIAG.LOG. By default, this file is written to the following paths:

- ▶ On UNIX platforms, in the \$INSTHOME/sql/lib/db2dump directory, where INSTHOME is the database instance home directory
- ▶ On Intel platforms:

- If the DB2INSTPROF environment variable is not set, x:\Program Files\IBM\SQLLIB\%\DB2INSTANCE%
- If DB2INSTPROF is set, x:\%DB2INSTPROF%\%DB2INSTANCE%

Here x:\SQLLIB is the drive reference and directory specified in the DB2PATH registry variable. DB2INSTANCE is the name of the instance owner and DB2INSTPROF is the name of the instance profile directory.

In most situations, the default paths are adequate and do not need to be changed.

In DB2DIGA.LOG, you can find information such as values of parameters that need to be increased, errors that occurred, or lock escalations. The value of the database manager parameter DIAGLEVEL defines the level of diagnostic information that will be captured. The diagnostic levels are:

- ▶ 0: No diagnostic data captured
- ▶ 1: Severe errors only
- ▶ 2: All errors
- ▶ 3: All errors and warning (default)
- ▶ 4: All errors, warnings, and informational messages

In general, level 3 is sufficient. In a developing environment, it is sometimes helpful to change to level 4 to have all the details captured. With level 4, you can, for example, see what Java environment is used by DB2. IBM Support also needs this information. If possible use a level 4 diagnostic level when you need support. The DB2DIGA.LOG file is managed manually by the administrator. The administrator can delete this file without impacting the DB2 UDB system. DB2 UDB recreates the file if it is not exited.

### 11.4.5 Health Center and Memory Visualizer

The *Health Center* is a new feature of DB2 UDB Version 8. The *Health Monitor* gathers information about the health of the system using new interfaces that do not impose a performance penalty. It allows you to monitor the state and utilization of many parts of the database manager and databases. It automatically evaluates a set of health indicators.

If the Health Monitor detects an abnormal state for an object, the Health Monitor raises an alert. The health state of the system is presented using traffic light type icons. Green indicates the state is normal. If you find a yellow or red state, you have to tune the system. You can set up the Health Center to inform you by e-mail if an alert occurs. You can also define the actions that should occur in case of an alert so that the system can heal itself.

The *Memory Visualizer* is a part of the Health Center. With the Memory Visualizer, you can monitor the memory usage such as sort heaps, buffer pools, and caches. It displays the memory allocation for a particular instance and all of its databases. Because many of the memory configuration parameters are dynamic, you can change these parameters and monitor whether your change has the expected result.

### 11.4.6 Design Advisor

Indexes play an important part in tuning the performance. An index is good when retrieving data because data access is, in general, much faster through an index than scanning an entire table to find matching rows. However, having indexes is also an overhead. An index needs storage, and every insert and delete needs a write operation on the table and on the index. If an update changes the value of an index field, an update on the index is also necessary. Indexes are important but not every index is helpful.

The *Design Advisor* is a wizard that you can start from the Control Center or from the command line using the `db2adviz` command. The Design Advisor can help you design and define suitable indexes for your data. It can perform the following actions:

- ▶ Find the best indexes for a problem query
- ▶ Find the best indexes for the set of queries that define in a workload, subject to resource limits that are optionally applied
- ▶ Test an index or materialized query table on a workload without having to create the index or materialized query table

### 11.4.7 Configuration Advisor

The *Configuration Advisor* is another helpful tool for DBAs to evaluate and tune the database system. This graphical tool guides you step-by-step in gathering the information about your system requirements. When the process is completed, the Configuration Advisor provides you with the recommended database configuration parameter values for an optimal performance. These values can be applied immediately or later by Task Center or through the command line processor.

Figure 11-7 shows the recommendations panel of the Configuration Advisor.

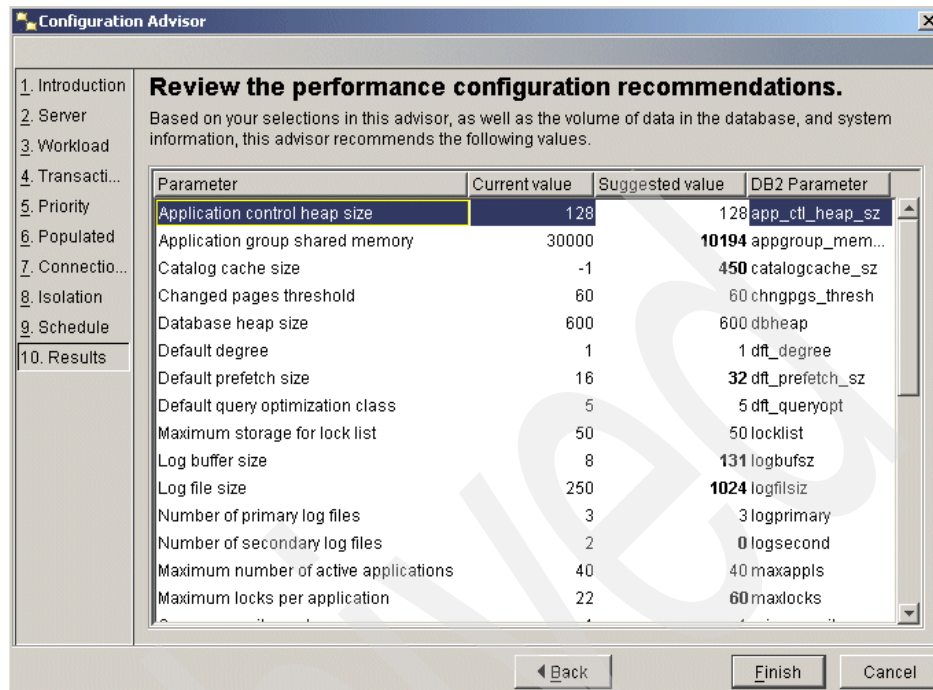


Figure 11-7 Configuration Advisor

## 11.5 Application tuning

Tuning the DB2 UDB server is important to achieve the performance goal. Well-designed databases and a performance-optimized application play a major role in reaching the performance tuning goal. From our experiences, often the application or the database design are the reasons for a performance problem.

In this section, from a performance-tuning point of view, we discuss some major topics and the best practices for database design and explain what to keep in mind when developing an application. Performance tuning is an on-going task. These guidelines are also applicable for database and application tuning when the application is in production.



## 11.5.1 Database design

Before you start the database design, you need to understand application requirements and performance goals. The database logical and physical design as well as the database server system resource needed are based on the application requirement and performance goal. If you can request the system you need, examine the performance requirement carefully and order hardware with enough capacity. If you are given a system, check whether the system has enough capacity to meet the performance goal.

Often the database server resides on a separate server. In this case, you can use all the resources available. You have to share the resources with the application if you have a single server containing the application and database server. We recommend that you have a separate server for DB2 UDB when you have an environment that includes WebSphere application server.

### Information gathering

To gather the application requirements for setting up a database, look for the following items:

- ▶ The number of users that will connect to DB2 UDB  
This number may be different from the number of real users when using connection pooling. The more connections you have, the more memory is necessary.
- ▶ The number of instances and databases that will be on the system  
The configuration is different when more than one instances or databases are on the server. Think about the number of instances to create and where to place the databases.
- ▶ The volume size of the data

Consider the following resources for DB2 UDB database server:

- ▶ Number and speed of CPU  
It is better to have more than one CPU to allow parallel processing. The speed of the CPU is necessary for the optimizer. During installation, DB2

UDB measures the speed. Any change afterwards need to be configured in the DBM CFG.

- ▶ The amount of memory that is available or required (if one machine, the amount available for DB2)

You need to know how much memory is available when configuring the database server.

- ▶ The number of disks and the kind of storage

How the data is stored is important. Only one disk can slow down the system because I/O becomes the bottleneck. Many disks allow parallel processing.

Consider the performance from the beginning on both the logical and physical database design. A lot of literature is available that covers the database design theory. We cannot discuss them one by one but we provide information here to give you a basic understanding of this topic.

## **Logical database design**

DB2 UDB is a relational database server. A relation can be displayed as a table with two dimensions. The first dimension is the fields of the table, while the second dimension is the values. Each field has a data type.

### ***Data types***

Define data types to minimize disk storage consumption and any unnecessary processing, such as data type transformations. Some of the main recommendations as they apply to the choice of data types required by the application are provided in the following list:

- ▶ Use SMALLINT rather than INTEGER if SMALLINT can contain the size of the number.
- ▶ Use the DATE and TIME data types rather than CHAR for date and time data.
- ▶ Use NOT NULL for columns wherever possible.
- ▶ Choose data types that can be processed most efficiently.
- ▶ Use CHAR rather than VARCHAR for any narrow columns, for example, those that are less than 50 characters wide.
- ▶ Choose VARCHAR instead of LONG VARCHAR when the row size is less than 32 K.
- ▶ Use FLOAT rather than DECIMAL if exact precision is not required.
- ▶ Use IDENTITY columns where appropriate.

## Normalization

Normalization is important in designing tables. In general, *normalization* means more tables with fewer columns, and *denormalization* means the opposite, fewer tables with more columns. There are several levels of normalization to guarantee consistency of the data and to reduce redundant data. Figure 11-8 shows how tables are normalized and denormalized.

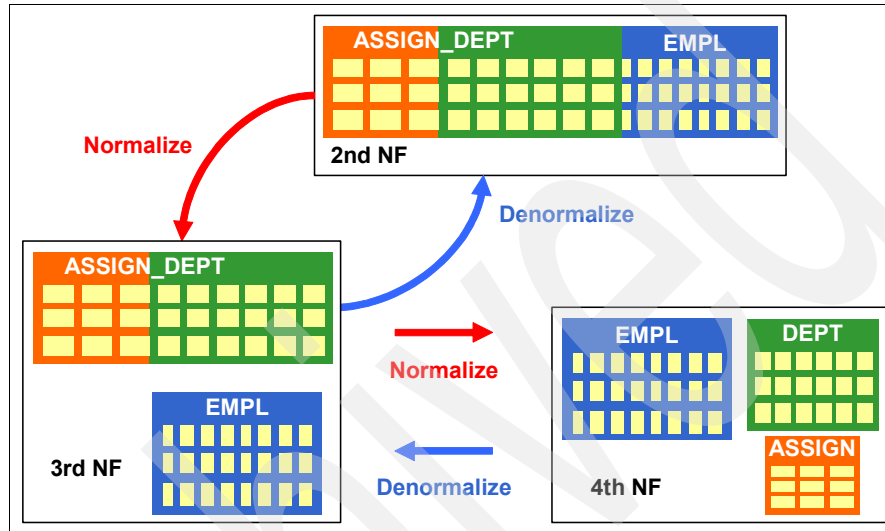


Figure 11-8 Normalization and denormalization

The level that is used most often is called the *third normal form*. For performance reasons, sometimes it makes sense to have some data redundancy to speed up access, especially in data warehouses where only read accesses occur. Redundant data has the following characteristics:

- ▶ More disk space is needed.
- ▶ There is more network traffic and memory usage when inserting data.
- ▶ Data can become inconsistent because of failure during insert or update.
- ▶ When redundant data is updated, all occurrences need an update.
- ▶ It can be faster for accessing data because less optimization is needed and data can be accessed with less read operations.

After the tables are normalized, consider defining constraints and primary keys to let DB2 UDB take care of data consistency. The *constraint* is a relation put in place between two tables and is one of the good practices in database design. We do not discuss this topic here because it is not a performance factor.

## **Index**

Indexes are important for a good performance. We cover indexes in more detail in 11.6, “System tuning” on page 375. You can see a need for indexes when you design the application. In general, the DBA creates indexes.

Take the following considerations into account during the application design:

- ▶ Define primary keys and unique keys, wherever possible, by using the CREATE UNIQUE INDEX statement. That helps by avoiding some sorts.
- ▶ To improve join performance with a multiple-column index, if you have more than one choice for the first key column, use the column that is most often specified with the “=” predicate or the column with the greatest number of distinct values as the first key.
- ▶ To improve data retrieval, add INCLUDE columns to unique indexes. Good candidates are columns that have the following characteristics:
  - Are accessed frequently and would benefit from index-only access
  - Are not required to limit the range of index scans
  - Do not affect the ordering or uniqueness of the index key

The DBA can monitor the system to see if an index is necessary. The DBA also has to decide where to place the index (when using DMS) and which buffer pool to use. Good teaming between developer and DBA is the best way to find a solution for good performance.

## **Physical database design**

After specifying the logical database design, the DBA must consider the placement of the data. The DBA must understand the files that will be created to support and manage the database, understand the amount of space that is required to store the data, and determine how to use the table spaces that are required to store the data.

The placement of the data is another performance consideration. For example, consider a database that stores all the data on one physical disk. You can certainly understand that there is no chance to allow parallel processing in such a design and that I/O will become the bottleneck in the system.

## **Buffer pool**

The buffer pool is the memory area where DB2 UDB caches the data that is requested, updated, or inserted by client applications. The reason for buffer pools is better performance, because the access to memory is much faster than to a disk. Therefore, it is good for performance if you can keep as much data, as needed by the client applications, as you can in the memory. If the data requested by a client is not in the buffer pool, it is retrieved from disk into the

buffer pool. If the buffer pool has no more space left, the data currently retrieved from disk pushes some data out of the buffer pool.

Buffer pools are assigned to table spaces, with a single buffer pool supporting one or more table spaces. This flexibility allows the database designer to group data table spaces and indexes table spaces into separate buffer pools (only possible with DMS, not with SMS). It also allows the designer to assign small lookup tables to a small buffer pool to keep the pages memory resident, maintain a separate buffer pool for the system catalog tables, among other things. These techniques can minimize disk I/O by allowing better reuse of data from the buffer pool or pools, which provides another mechanism to improve overall DB2 system performance.

Since the space for a buffer pool is allocated at database startup at the full amount of memory and pages cannot be shared between buffer pools, it is a good idea to start with a small amount of buffer pools. Upon further analysis and monitoring, additional buffer pools may be added. However, initially you should keep the number from one to three.

If the database server is small, a single buffer pool is usually the most efficient choice. Large systems, with applications that use dynamic SQL to do table scans intermixed with index access, usually benefit from multiple buffer pools by separating the indexes and data for frequently accessed tables. This configuration prevents a table scan from “flushing” the index pages from the buffer pool to make room for additional data pages. Since most table accesses are through an index. This allows the index data to remain in the buffer pool because it is sharing space with other index pages only, not data pages.

Fewer shared buffer pools allow for some margin of error in sizing, when multiple table spaces are sharing a buffer pool. If one tablespace is used less than initially thought, and another is used more, the overall buffer pool impact is easier to balance. Single tablespace buffer pools can waste server memory with little or no performance benefit to the database, if that tablespace is not frequently read by the application.

## **Buffer pool data page management**

Pages in the buffer pool are either in use, dirty, or clean:

- ▶ *In-use pages* are currently being read or updated. They can be read, but not updated, by other agents.
- ▶ *Dirty pages* contain data that has been changed but has not yet been written to disk.
- ▶ After a changed page is written to disk, a *clean page* remains in the buffer pool until its space is needed for new pages. Clean pages can also be migrated to an associated extended storage cache, if one is defined.

When the percentage of space occupied by changed pages in the buffer pool exceeds the value specified by the CHNGPGS\_THRESH configuration parameter, page-cleaner agents begin to write clean buffer pages to disk.

### ***Page-cleaner agents***

Page-cleaner agents perform I/O as background processes and allow applications to run faster because their agents can perform transaction work. Page-cleaner agents are sometimes referred to as *asynchronous page cleaners* or *asynchronous buffer writers* because they are not coordinated with the work of other agents and work only when required.

To improve performance in update-intensive workloads, configure more page-cleaner agents. Performance improves if more page-cleaner agents are available to write dirty pages to disk. This is also true when there are many data-page or index-page writes in relation to the number of asynchronous data-page or index-page writes.

### ***Table spaces***

For the physical design, you have to decide whether to use DMS or SMS table spaces. The arguments for SMS are the ease of administration and that only the space that is needed becomes allocated. Administration is easier because you only have to tell DB2 UDB in which directory to store the data. The only time you have to worry about space for an SMS tablespace is when the device is full. Especially if the database temporarily needs a high amount of disk space, it makes sense to use SMS because the space is released after completion. With DMS table spaces, the space is preallocated and does not increase automatically.

From the performance point of view, we recommend that you use DMS table spaces using raw devices. With this option, DB2 UDB handles the storage itself without the operating system and can store table data in contiguous pages on the disk. The operating system cannot assure this, and the data could be fragmented, resulting in worse performance during read and write operations.

With DMS table spaces, it is possible to separate the index data and long data from the regular table data. Using this feature can result in better performance. If the index data and the regular data are stored in separated table spaces, for example, it is possible to assign a separate buffer pool for the index data to have the index resident in memory.

By default, DB2 uses three SMS table spaces. One tablespace is to store the system catalog, one is for temporarily needed space (during a sort, for example), and one is for the user data. Example 11-7 shows the output of DB2 LIST TABLESPACES.

Tablespaces for Current Database

Tablespace ID	= 0
Name	= SYSCATSPACE
Type	= System managed space
Contents	= Any data
State	= 0x0000
Detailed explanation:	
Normal	

Tablespace ID	= 1
Name	= TEMPSPACE1
Type	= System managed space
Contents	= System Temporary data
State	= 0x0000
Detailed explanation:	
Normal	

Tablespace ID	= 2
Name	= USERSPACE1
Type	= System managed space
Contents	= Any data
State	= 0x0000
Detailed explanation:	
Normal	

---

You must consider two key factors when designing SMS table spaces:

► Containers for the tablespace

When specifying the number of containers for the tablespace, it is important to identify all the containers required up front, since containers cannot be added or deleted after the SMS tablespace has been created. Each container used for an SMS tablespace identifies an absolute or relative directory name. Each of these directories can be located on a different file system (or physical disk).

► Extent size for the tablespace

The extent size can only be specified when the tablespace is created. Because it cannot be changed later, it is important to select an appropriate value for the extent size.

To calculate the optimal size for the tablespace extent, calculate the row length of the tables within the tablespace. Also, determine the number of rows that will

fit on each data page. Remember there is a maximum of 255 rows per page. And, determine the number of rows that will fill up the selected extent size:

number of rows per page x extent size

Then estimate the update or insert activity of the tables to determine how often an extent will be filled. For example, if you assume the default extent size of 32 pages, and a particular table row length allows one hundred (100) rows per page, it would take 3200 rows of data to fill an extent. If this particular table had a transaction rate of 6000 rows per hour, it would take roughly one half-hour to fill the extent. Therefore, the default extent size of 32 would be adequate for this table.

Generally, the default extent size of 32 has been shown to be the optimal extent size through numerous DB2 UDB benchmark tests. The only time that the extent size should be set differently is if the extent size is too small and will fill up every several minutes. This would cause the database manager to allocate an additional extent every few minutes. Since some overhead is involved in the extent allocation process, this could impact overall database performance.

When choosing an extent size, if it is difficult to determine which of two extent sizes is best (for example, 16 and 32). It is better to choose the larger extent size. If an extent size is too large, the worst scenario is that part of the last extent for the table is unused. That is, the table is small, with only 40 rows of 50 bytes each. If an extent size is too small, the database manager has to allocate new extents frequently, which could cause a performance impact.

Using DMS table spaces makes it necessary to think about sizing because you have to preallocate the space for DMS table spaces. From the Control Center, you can access, for example, a table. Right-click the table and select **ESTIMATE SIZE...** to estimate the size of the table and the indexes by entering the expected row count. This feature helps you to decide how much space is needed.

This same feature is also helpful for choosing the right page size. By default, the page size is 4 KB. Each page (regardless of page size) contains 68 bytes of overhead for the database manager. This leaves 4028 bytes to hold user data (or rows), although no row on a 4 KB page can exceed 4005 bytes in length. A row does not span multiple pages. You can have a maximum of 500 columns when using a 4 KB page size.

If you have rows that need more space, DB2 UDB offers the ability to create table spaces with page sizes of 8 KB, 16 KB, or 32 KB. The maximum row sizes for these page sizes are:

- ▶ 8 KB: Up to 8101 bytes
- ▶ 16 KB: Up to 26293 bytes
- ▶ 32 KB: Up to 32677 bytes



A buffer pool assigned to a tablespace must have the same page size. During installation, a small buffer pool of each page size is created that cannot be seen and is not big enough to support a tablespace. It is possible to use the same buffer pool for each tablespace. But when you have tables that are more frequently accessed than others, it may be useful to use a separate buffer pool to have much of the data resident in the memory. Or if you have a large table that is not as frequently accessed, you can assign a small buffer pool to the tablespace where the table is located because it is no problem when this data is not present in the memory.

If you have only one large table in a tablespace, it might be useful to use a bigger page size for the tablespace. The bigger the page size is, the more data a table can contain. When using small and large tables within one tablespace, use the smallest page size fitting your large tables. For small tables, use a 4 KB page size so that you do not waste memory, as the following note explains.

**Note:** There is a maximum of 255 rows per page. If the row size is smaller than the page size divided by 255, space is wasted on each page.

Refer to Chapter 5, “Physical Database Design”, in the *IBM DB2 Universal Database Administration Guide: Planning*, SC09-4822, for a complete description.

As you can see from Example 11-7 on page 363, DB2 UDB creates, by default, a system temporary tablespace. This tablespace is used to store internal temporary data required during SQL operations such as sorting, reorganizing tables, creating indexes, and joining tables. Although you can create any number of system temporary table spaces, we recommend that you create only one, using the page size that the majority of your tables use.

User temporary table spaces are used to store declared global temporary tables that store application temporary data. User temporary table spaces are not created by default at database creation time.

Separating an index from the data makes sense if you have a table with many columns that is frequently accessed and has many rows. In this case, the tablespace for the indexes can have a separate buffer pool to make the most of the index pages resident in memory. Think about the feature of putting additional columns on the index to avoid accessing the table.

Figure 11-9 shows an example of how tablespace can be arranged.

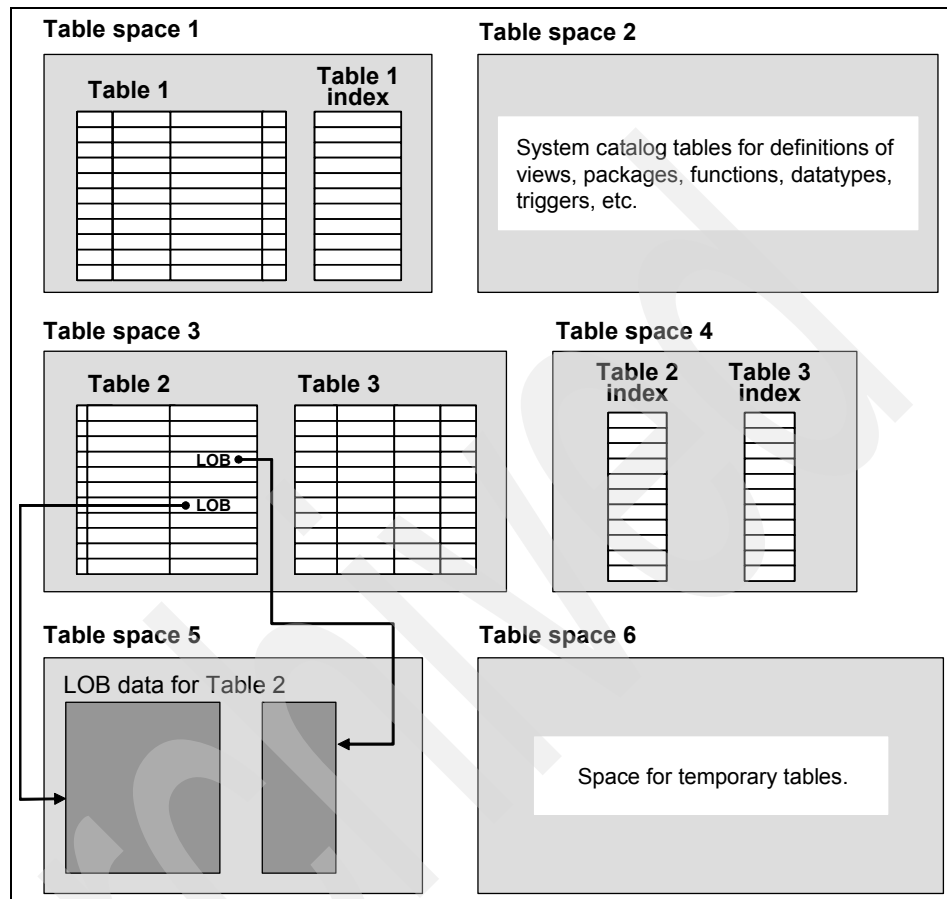


Figure 11-9 Tablespaces

### Container

*Tablespaces* are a logical view while a *container* is a physical storage device. Each tablespace has at least one container, and a container can belong only to one tablespace. When using more than one container per tablespace, DB2 UDB uses a striping mechanism to distribute the data across the containers.

If you have several disks, then create the containers for a tablespace on different disks to allow parallel processing. We recommend that you use striped devices, such as RAID systems, because a RAID system is already striped. We also recommend that you use only one container per tablespace. If you use a RAID system, you have to set the registration variable `DB_PARALLEL_IO=*` to allow parallel data access from all table spaces. The size of an extent should be a

multiple of the RAID stripe size. The prefetch size should be a multiple of the extent size and a multiple of the RAID stripe size multiplied by the count of devices.

For the physical design, it is also important to consider the location of the log files. Because any change to the data is written to the log files, the number of I/Os on these files is high. It is a good practice to have a different disk on which to put the logs. If you have to use any disk for your data, use the disk with the lowest I/O for the log files.

The path to the log files is stored in the database configuration. Use the command DB2 GET DB CFG and look for PATH TO LOGFILE. To change the path, update the NEWLOGPATH parameter in the database configuration. The new setting does not become the value of logpath until both of the following actions occur:

- ▶ The database is in a consistent state, as indicated by the DATABASE\_CONSISTENT parameter from the output of DB2 GET DB CFG command.
- ▶ All users are disconnected from the database.

When the first new connection is made to the database, the database manager moves the logs to the new location specified by logpath.

A set of *system catalog tables* is created and maintained for each database. These tables contain information about the definitions of the database objects (for example, tables, views, indexes, and packages) and security information about the type of access that users have to these objects. These tables are stored in the SYSCATSPACE tablespace.

**Note:** The catalog tables are frequently accessed. We recommend that you use a separate disk or a disk with less activity for the SYSCATSPACE.

## 11.5.2 SQL tuning

The next step is to implement the application. SQL is the language to speak with relational databases such as DB2 UDB. It is used to select, insert, and update the data. DB2 UDB uses an SQL compiler to produce an access plan. Figure 11-10 illustrates the steps for the SQL compiler.

Like every compiler, the SQL compiler checks the syntax and the semantic first. If one of the checks fails, an error is returned to the application. If these checks are passed, the compiler tries to rewrite the statement if necessary. The compiler uses the global semantics stored in the query graph model to transform the

query into a form that can be optimized more easily and stores the result in the query graph model.

The next step is to optimize the query. How intensive the optimizations will be is configurable. You can learn more about optimizing the query in 11.4.3, “Explain utilities” on page 351.

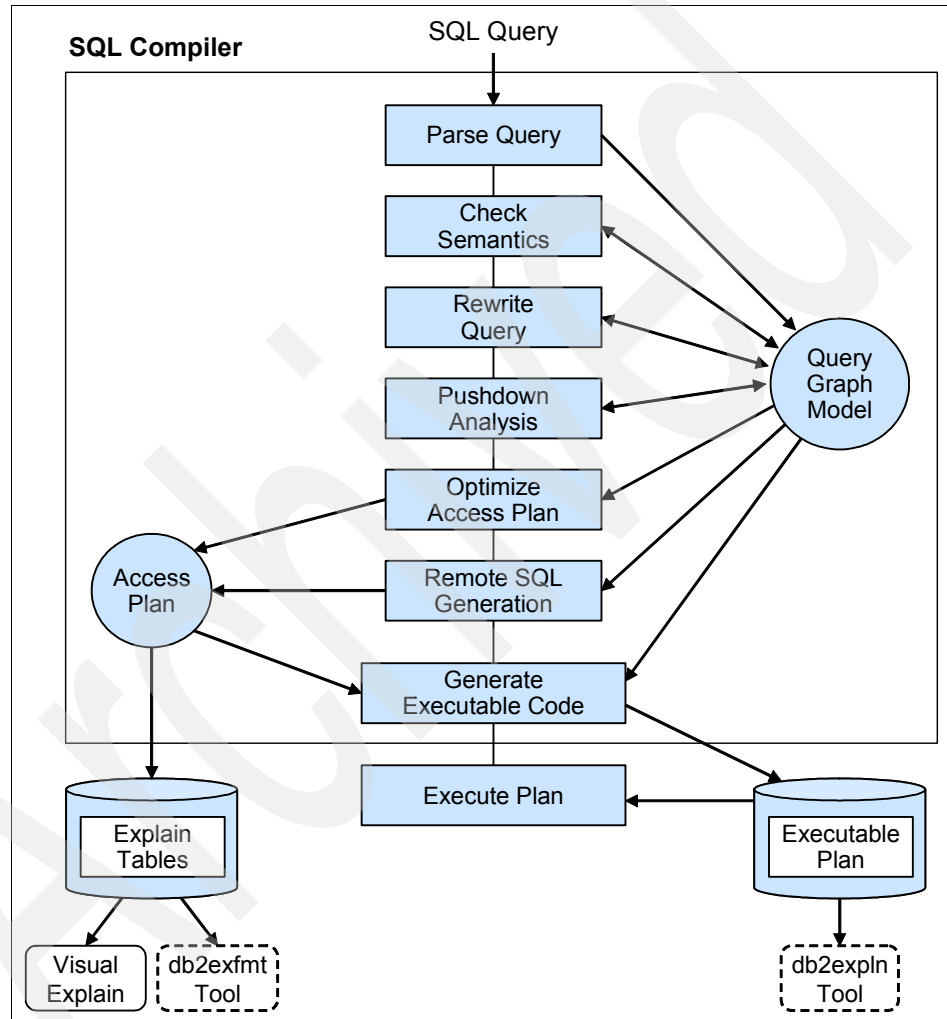


Figure 11-10 Steps that the SQL compiler takes

## Static or dynamic SQL

With DB2 UDB, you have the ability to use static SQL or dynamic SQL. The difference is that *static SQL* uses a precalculated access plan, which allows

faster access because there is no need to optimize the query every time. If you look at Figure 11-10, you see a process that needs to be done every time a dynamic SQL statement requests data from the server. For static SQL statements, this step is done only once at creation time. However, you need to rebind the code every time the conditions change and the optimizer might find another access plan. Do a rebind after the following situations occur:

- ▶ The package is marked as invalid.
- ▶ An index is created or removed.
- ▶ The statistics of the tables used in the statement become updated.
- ▶ A fix pack is installed.

### **Efficient SELECT statements**

The optimizer can also optimize the select statement. You have to take care of the following points that the optimizer cannot do for you:

- ▶ Specify only the columns needed. Unnecessary columns waste CPU cycles and slow down the network traffic.
- ▶ Avoid selecting more rows than needed by specifying all applicable predicates in the query, to reduce the answer set to a minimum. Do not discard rows in your application code, let DB2 do it for you.
- ▶ When the number of rows you need is significantly less than the total number of rows that might be returned, specify the OPTIMIZE FOR clause.
- ▶ Use the FOR READ ONLY or FOR FETCH ONLY clause to avoid exclusive lock and to take advantage of row blocking.
- ▶ Use the FOR UPDATE OF clause for update cursors to prevent deadlocks.
- ▶ Avoid numeric data type conversion whenever possible.
- ▶ Use such operations as DISTINCT and ORDER BY only if necessary. Sorting is very CPU intensive. If the order of the result set is insignificant, do not order it.
- ▶ Use COUNT(\*) FROM only when needed.
- ▶ Use WHERE COL1 IN (1,2,5) instead of WHERE COL1= 1 OR COL1 = 2 OR COL1 = 5.
- ▶ Do not use user-defined functions (UDF) in join conditions. If you need a function in a join condition, then create a view with an additional field including the UDF and join the view with the other table.

It is a good practice to use the same code page on both the client and the server site because the conversion of data types slows down the application. Character conversion occurs in the following circumstances:

- ▶ When a client or application runs in a code page that is different from the code page of the database that it accesses

The conversion occurs on the database server machine that receives the data. If the database server receives the data, character conversion is from the application code page to the database code page. If the application machine receives the data, conversion is from the database code page to the application code page.

- ▶ When a client or application that imports or loads a file runs in a code page different from the file being imported or loaded

### 11.5.3 Stored procedures

*Stored procedures* are programs that run in the database environment. They can be written in SQL, C, and Java and help to reduce the network traffic. In any case, procedures are typically written to contain multiple SQL Data Manipulation Language (DML) statements and procedure logic constructs such as loops and if/else statements. Therefore, stored procedures are conceptually similar to “small” applications, providing useful, database-intensive business service to multiple applications. These applications, which typically are remote from the database management system (DBMS) itself, invoke the procedure with a single call statement.

Stored procedures can require parameters for input (IN), output (OUT), or input and output (INOUT). They may return one or more sets of results.

Keep in mind that calling a stored procedure has overhead. You should understand that it makes no sense to have a single statement inside the stored procedure. The benefit arises if the network traffic reduces significantly. This can be the case if the stored procedure includes several SQL statements. This is especially true when the result set of the stored procedure is smaller than the result set that the stored procedure itself has to handle, or if the stored procedure has loops including SQL statements.

#### **FENCED versus NOT FENCED: Security versus performance**

A stored procedure can run in two modes, FENCED and NOT FENCED. A stored procedure running in fenced mode uses a different address space than the database server. A NOT FENCED stored procedure runs in the same process as the database manager. We recommend that you use FENCED stored procedures only because an error in a NOT FENCED stored procedure

can accidentally or maliciously corrupt the database or damage the database control structure.

The benefit of a NOT FENCED stored procedure is better performance. Therefore, for performance reasons, it might be useful to use a NOT FENCED stored procedure, but again it is a high risk. Use the stored procedure as a FENCED one first and later. If you know that the stored procedure runs stably and has no errors, create the stored procedure in a NOT FENCED mode.

## Java routines

For Java routines running on UNIX platforms, scalability may be an issue if NOT FENCED is specified. This is due to the nature of the DB2 UNIX process model, which is one process per agent. As a result, each invocation of a NOT FENCED Java routine requires its own Java virtual machine (JVM). This requirement can result in poor scalability, because JVMs have a large memory footprint. Many invocations of NOT FENCED routines on a UNIX-based DB2 server use a significant portion of system memory.

Consumption of system memory is not the case for Java routines running on Windows NT and Windows 2000, where each DB2 agent is represented by a thread in a process shared with other DB2 agent threads. This model is scalable, because a single JVM is shared among all the DB2 agent threads in the process.

If you intend to run a Java routine with large memory requirements, we recommend that you register it as FENCED NOT THREADSAFE. For FENCED THREADSAFE Java routine invocations, DB2 attempts to choose a threaded Java fenced mode process with a Java heap that is large enough to run the routine. Failure to isolate large heap consumers in their own process may result in “out of Java heap” errors in multi-threaded Java **db2 fmp** processes. If your Java routine does not fall into this category, FENCED routines run better in thread safe mode where they can share a small number of JVMs.

## C or C++ routines

C or C++ routines are generally faster than Java routines, but are more prone to errors, memory corruption, and crashing. For these reasons, the ability to perform memory operations makes C or C++ routines risky candidates for THREADSAFE or NOT FENCED mode registration. These risks can be mitigated by adhering to programming practices for secure routines, and thoroughly testing your routine.

For more information about secure routines, refer to the DB2 Information Center in the topic “Security Considerations for Routines” by selecting **Security** → **Database systems** → **DB2 Universal Database** → **User responsibilities for security** → **Application considerations** → **Security and routines** at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp>

### SQL-bodied routines

SQL-bodied routines are also generally faster than Java routines, and usually share comparable performance with C routines. SQL routines always run in NOT FENCED mode, providing a further performance benefit over external routines.

**Note:** To create SQL stored procedures, you need a C compiler.

## 11.5.4 Declared global temporary tables

The DECLARE GLOBAL TEMPORARY TABLE statement defines a temporary table for the current session. The benefit of temporary tables is that their description does not appear in the system catalog and that any action on them is not logged. A temporary table consists only as long as the session exists that created it. If the session terminates, the temporary is gone.

For some circumstances, temporary tables are helpful, and in some cases, they can become very big. With DB2 UDB Version 8, you can now optimize access to the data in temporary tables since you can create indexes and update the statistics on them. With these options, temporary tables are now much more powerful than in previous versions.

Think about a very large temporary table in your system. Every time you need some data from that table, a tablescan on this temporary table is necessary. With an index and statistical data, it is now possible to have the same performance as with regular tables.

## 11.5.5 Concurrency

When designing applications, it is important to consider concurrency. There is a need to balance between the requirement of the application for consistency when accessing data and the impact of performance when the isolation level is set too high.

Generally, in a production environment, the DBA is the one who binds the system utilities such as the call level interface (CLI) and so on. Therefore, the DBA sets the default isolation level for the applications that use those utilities. To set a proper default isolation level, the DBA should understand the applications



running on the server. Good communication between the DBA and the application designer or programmer is necessary. More importantly, the developer should always explicitly set the isolation best suited for the application.

## Isolation level

DB2 UDB offers four isolation levels:

- ▶ Repeatable read (RR)
- ▶ Read stability (RS)
- ▶ Cursor stability (CS)
- ▶ Uncommitted read (UR)

Each of these isolation levels allows the user and application to control the number and duration of read (share) locks held within a unit of work. By setting the appropriate isolation level based on a particular application's requirement, lock resources can be minimized and user concurrency (application's tendency to multiple users) can be increased as much as possible for a particular application. Each of the isolation levels and their definitions, along with a brief example, is shown in the Table 11-4 (from highest to lowest locking impact). The example is based on the following information:

- ▶ A table (EMPLOYEES) that contains the LAST\_NAME column with these values: ARNOLD, BLACK, BROWN, BRUEGGEMAN, BRUTY, HANCOCK, HURT, JAYNE, JONES, LAW, MORGAN, NOBLES, NOVACEK, OATES, STAUSKAS, and TOBIAS

The table has no index on LAST\_NAME.

- ▶ The SQL query being executed:

```
SELECT LAST_NAME, EMPLOYEE_NUMBER  
FROM A.EMPLOYEES  
WHERE LAST_NAME LIKE 'B%'
```

Table 11-4 Isolation levels with an example

Isolation level	Brief description	Example that illustrates impact on share (read) locking
Repeatable read (RR)	<p>The same query can be run multiple times within the same unit-of-work, and the results of the query is identical every time (repeatable).</p> <p>A share lock is held on each row and examined during the execution of a SQL statement. For table scans, this encompasses each row within the table.</p>	A share lock is held on every row within the table for the entire unit-of-work, effectively preventing updates (X locks) from occurring until this SQL statement is committed.
Read stability (RS)	<p>The same query returns the same base set of information each time the query is run in a unit-of-work, but additional rows may be included with each subsequent execution.</p> <p>A share lock is held on every row that satisfies the result set of the query (is part of the answer).</p>	A share lock is held on the rows that contain BLACK, BROWN, BRUEGGEMAN and BRUTY, but all other rows can be available for update and read.
Cursor stability (CS)	<p>The data is repeatable on the current position of the cursor only. Previously read rows or to-be-read rows allow updates, reads, or both.</p> <p>A share lock is held on the current position of the cursor only.</p>	A share lock is held on the rows containing BLACH, BROWN, BRUEGGEMAN and BRUTY, but only when the cursor is positioned on that row. After the cursor is repositioned, the row is eligible for a read, update, or both by another user.
Uncommitted read (UR)	<p>No share locks are held on any row by this query and updates can take place. Only super exclusive (Z) locks prevent UR queries.</p> <p>No shared locks are held at all.</p>	No locks are held on any rows of the table and the data may be in the process of being changed as it is read.

In looking at the isolation levels and their descriptions, it is easy to see that using an unnecessary isolation level (too high a locking impact) impacts performance of a database environment that supports both database read and write transactions. A good rule to go by in selecting the isolation level is to use the lowest locking level that will support the application requirements.

For some reporting applications, if the answer set does not have to be exact (but rather an approximation), UR might be the proper isolation level. There are few instances where the RR isolation level is required. The application requirements should be verified to confirm that RR isolation is necessary (due to the potential for lock waits and lock escalation).

## Locking options

DB2 UDB offers options in the area of locking. The options help to reduce DB2 UDB resource consumption for the lock list memory area and reduce the chance of lock escalation for the entire database or an individual database agent.

The LOCKSIZE of the table can be changed from the default of row level locking to table level locking through the ALTER TABLE statement. This allows read-only tables to hold less DB2 UDB lock resources (as share locks behave the same at the table or row level) due to the fact that one lock (at the table level) is all that is required on the table. Depending on the isolation level setting, this can reduce the memory consumption within the lock list considerably.

Another locking option to perform essentially the same function of reducing the memory consumption within the lock list focuses on the application level by providing the LOCK TABLE statement. This statement can be issued by an application process or user to lock a table in either Share or Exclusive (X) mode. The use of this option allows locks to be reduced for a single application task or process. It also prevents other applications or tasks from accessing the table at all (through the use of Exclusive (X) mode).

Carefully consider the database usage requirements and application design before using either of these locking options. Since the lock granularity is at the table level, this type of locking strategy can have a major impact on overall database performance, if it is used in an inappropriate situation.

## 11.6 System tuning

We have made as many performance considerations as we can during the application development and have put them in place. Once the whole system is up and running, we need to monitor and adjust the system or application to ensure that optimal performance is reached.

In this section, we show you how to tune a running DB2 UDB system. We go through the key performance areas and describe what to look for and how to set parameters to fit the requirements for your system.

### 11.6.1 Tuning the buffer pools

When creating a buffer pool (DB2 CREATE BUFFERPOOL), you have to specify the size of a buffer pool. The size specifies the amount of pages to use. When using -1, the DB2 BUFPAGE parameter is used to specify the size of the buffer pool. To change the size, use DB2 ALTER BUFFERPOOL. When the size of the buffer pool is -1, then change the BUFPAGE parameter.

**Note:** The DB parameter BUFFPAGE controls the size of any buffer pool with the size set to -1. If you want different sizes, you have to set the size using the CREATE BUFFERPOOL or ALTER BUFFERPOOL command.

In previous DB2 UDB versions, you must increase the DBHEAP parameter when using more space for the buffer pool. With Version 8, nearly all buffer pool memory, including page descriptors, buffer pool descriptors, and the hash tables, comes out of the database shared-memory set and is sized automatically.

To determine how well your buffer pool is designed, use the snapshot monitor. The monitor switch for BUFFERPOOLS must be set to ON. Check the status by using the following command:

```
DB2 GET MONITOR SWITCHES
```

If the switch for BUFFERPOOL is OFF, use the following command to turn it on:

```
DB2 UPDATE MONITOR SWITCHES USING BUFFERPOOL ON
```

Or use the following command to set the DBM parameter DFT\_MON\_BUFPOOL to ON:

```
DB2 UPDATE DBM CFG USING DFT_MON_BUFPOOL ON
```

When the system is running for a while with the monitor switch for buffer pools set to on, then use the following command to see the current status of the buffer pools:

```
DB2 GET SNAPSHOT FOR ALL BUFFERPOOLS
```

Example 11-8 shows output of a snapshot for buffer pools. As you can see, the systems have the standard buffer pool with the standard size (for UNIX systems) of 1000 4 K pages.

*Example 11-8 Sample output for a snapshot of a buffer pool*

---

\$ db2 get snapshot for all bufferpools

Bufferpool Snapshot

Bufferpool name	= IBMDEFAULTBP
Database name	= TRADE3DB
Database path	=
/home/db2inst1/db2inst1/NODE0000/SQL00002/	
Input database alias	=
Snapshot timestamp	= 11-17-2003 16:30:46.883397
Buffer pool data logical reads	= 12823
Buffer pool data physical reads	= 279
Buffer pool data writes	= 0
Buffer pool index logical reads	= 222
Buffer pool index physical reads	= 68
Total buffer pool read time (ms)	= 775
Total buffer pool write time (ms)	= 0
Asynchronous pool data page reads	= 88
Asynchronous pool data page writes	= 0
Buffer pool index writes	= 0
Asynchronous pool index page reads	= 0
Asynchronous pool index page writes	= 0
Total elapsed asynchronous read time	= 22
Total elapsed asynchronous write time	= 0
Asynchronous data read requests	= 5
Asynchronous index read requests	= 0
Direct reads	= 17210
Direct writes	= 0
Direct read requests	= 7545
Direct write requests	= 0
Direct reads elapsed time (ms)	= 3564
Direct write elapsed time (ms)	= 0
Database files closed	= 0
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	= 0

Vectored I/Os	= 5
Pages from vectored I/Os	= 88
Block I/Os	= 0
Pages from block I/Os	= 0
Physical page maps	= 0
Node number	= 0
Tablespaces using bufferpool	= 5
Alter bufferpool information:	
Pages left to remove	= 0
Current size	= 1000
Post-alter size	= 1000

---

From the output of the DB2 GET SNAPSHOT FOR ALL BUFFERPOOLS command, you can determine how good the hit ratio of your buffer pools is. The more logical reads and the less physical reads the buffer pool has, the better the ratio will be. To determine the ratio for the buffer pool and the index pool, use the following formulas:

$$\text{BufferPoolHitRatio} = \frac{\Sigma \text{LogicalReads} - \Sigma \text{PhysicalReads}}{\Sigma \text{LogicalReads}} \times 100$$

$$\text{IndexPoolHitRatio} = \frac{\Sigma \text{IndexLogicalReads} - \Sigma \text{IndexPhysicalReads}}{\Sigma \text{IndexLogicalReads}} \times 100$$

In our example, the overall buffer pool ratio is 97.82, which means that the buffer pool is big enough. The hit ratio for indexes is below 70. The small value of the indexes hit ratio show that the indexes are in less use. This could mean that the system is in the point of diminishing returns for a over sized buffer pool. With this ratio, we could try to reduce the buffer pool size to set memory free to improve the system performance.

In general, a good value for the hit ratio is a value more than 90 percent for both buffer pool hit ratio and index pool hit ratio. When the value is smaller, then increase the buffer pool size and continue monitoring. Be sure not to allocate so much memory that the operating systems starts paging. In AIX, use either the **1sps -s** or **topas** command to monitor the memory usage and the paging activities.

To monitor the memory of the database, use the DB2 Memory Visualizer.

## 11.6.2 Table management

The optimizer uses the table statistics to calculate the best access plan. The statistics will become out-of-date invalid because of insert, update, and delete operations performed on tables. It is necessary to update the statistics from time to time by using the **runstats** command. After the **runstats** command is run on a table, you must *rebind* the applications that access the table or tables. When an application with static SQL is bound to the database, the access plan gets calculated using the current statistics. This access plan is updated as the application is rebound due to the statistics change, or an index is created.

### Table reorganization

If your SQL codes slows down and this performance problem does not improve after running the **runstats** command and rebinding the applications, the SQL code performance might be helped by reorganizing the tables. When inserting new data into tables, it is often not possible to place them in a physical sequence that is the same as the logical sequence defined by the index (unless you use clustered indexes). When accessing the data, many more operations are necessary than with having the data on sequential pages.

**Note:** With DB2 UDB Version 8, REORG becomes an online feature. The REORG can be paused and restarted. The IN-PLACE REORG operation reorganizes only a small portion of data. Therefore only a small amount of additional space is necessary, and only a few pages are locked for updates at a time.

For more details about how to determine if a REORG is necessary, search the DB2 Information Center or the DB2 Online Help for the REORGCHK command on the Web at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp>

After the table is reorganized, run the update the statistic by running the **runstats**, and then re-bind the static applications. To reduce the need for reorganizing a table, perform these tasks after you create the table:

1. Alter the table to add PCTFREE.
2. Create a clustering index with PCTFREE on index.
3. Sort the data.
4. Load the data.

After you perform these tasks, the table with its clustering index and the setting of PCTFREE on the table help to preserve the original sorted order. If enough space is allowed in the table pages, new data is inserted into the pages to maintain the clustering characteristics of the index. As more data is inserted and

the pages of the table become full, records are appended to the end of the table so that the table gradually becomes unclustered.

### 11.6.3 Index management

The most important thing for performance in the database design is the index. When designing the index on tables, you need to understand how the tables will be accessed by the application. First, determine which queries will be used on the database. Then analyze the statements by yourself or use the *Design Advisor* provided by DB2 UDB.

After the data are loaded into the tables, run the **runstats** utility to update the statistics on the tables so the optimizer can use the indexes. For frequently updated tables, run the **runstats** command regularly to update the table statistics to ensure that the best access plan will be used in your queries.

If you notice that there are long-running queries in an existing system, use the Explain tool to analyze the access path that the optimizer created. If the table scan is used, it could be an indication that an index is needed. In some situations, it is better to use a table scan than an index if the table has only a few rows or the query needs most of the rows as a result set. Therefore, we recommend that you use the Design Advisor because you can test if a new index will help to achieve a better performance.

If you see that the access plan uses a table scan and you have already had an index that fits, then it may be that the statistics of the table are *not* up-to-date. Use the **runstats** utility to update the statistics and let DB2 UDB create a new access plan. If you are testing with static SQL, do not forget to rebind your application to generate a new access plan.

Each index entry contains a search-key value and a pointer to the row containing that value. If you specify the ALLOW REVERSE SCANS parameter in the CREATE INDEX statement, the values can be searched in both ascending and descending order. Therefore, it is possible to bracket the search, given the right predicate. An index can also be used to obtain rows in an ordered sequence, eliminating the need for the database manager to sort the rows after they are read from the table.

Consider the following suggestions for using and managing indexes:

- Specify parallelism at index creation.

When you create indexes on large tables hosted by an SMP machine, consider setting the DBM configuration parameter INTRA\_PARALLEL to YES (1) or SYSTEM (-1) to take advantage of parallel processing for performance improvements. Multiple processors can be used to scan and sort data.



- Specify separate table spaces for indexes.

Indexes can be stored in a different tablespace from the table data. 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. In addition, you can assign the index tablespace to a different buffer pool, which might keep the index pages in the buffer longer because they do not compete with table data pages.

- Ensure the degree of clustering.

Your SQL statement may require ordering, such as ORDER BY, GROUP BY, and DISTINCT, even though an index might satisfy the ordering. In this case, the optimizer might not choose the index if clustering is poor or the table is so small that it is cheaper to scan the table and sort the answer set in memory.

After you create a clustering index, perform a REORG TABLE in classic mode, which creates a perfectly organized index. In general, a table can only be clustered on one index.

- Use *volatile tables* for tables that vary widely in size.

A volatile table is one that might vary in size at run time from empty to very large. For this kind of table, in which the cardinality varies greatly, the optimizer might generate an access plan that favors a table scan instead of an index scan.

Declaring a table *volatile* using the ALTER TABLE...VOLATILE statement allows the optimizer to use an index scan on the volatile table.

**Note:** In DB2 UDB V8.1 and later, all new indexes are created as type-2 indexes. The one exception is when you add an index on a table that already has type-1 indexes. In this case only, the new index will also be a type-1 index. To find out which type of index exists for a table, use the INSPECT command. To convert type-1 indexes to type-2 indexes, run the REORG INDEXES command. To see the advantages of the new index, refer to Chapter 8, “Operational performance,” in *DB2 UDB Administration Guide: Performance*, SC09-4821.

## Agents

Some database manager configuration parameters influence the number of agents created and the way they are managed. The parameters include:

- MAXAGENTS

This is the number of agents that can be working at any one time. This value applies to the total number of agents that are working on all applications, including coordinator agents, subagents, inactive agents, and idle agents.

► NUM\_POOLAGENTS

This refers to the total number of agents, including active agents and agents in the agent pool, that are kept available in the system. The default value for this parameter is half the number specified for maxagents.

► NUM\_INITAGENTS

When the database manager is started, a pool of worker agents is created based on this value. This speeds up performance for initial queries. The worker agents all begin as idle agents.

► MAX\_CONNECTIONS

This parameter specifies the maximum number of connections allowed to the database manager system on each partition.

► MAX\_COORDAGENTS

This parameter is for partitioned database environments and environments with intra-partition parallelism enabled when the connection coordinator is enabled. This value limits the number of coordinating agents.

► MAXCAGENTS

This parameter controls the number of tokens permitted by the database manager. For each database transaction (unit of work) that occurs when a client is connected to a database, a coordinating agent must obtain permission to process the transaction from the database manager. This permission is called a *processing token*. The database manager permits only agents that have a processing token to execute a unit of work against a database. If a token is not available, the agent must wait until one is available to process the transaction.

This parameter can be useful in an environment in which peak usage requirements exceed system resources for memory, CPU, and disk. For example, in such an environment, paging might cause performance degradation for peak load periods. You can use this parameter to control the load and avoid performance degradation, although it can affect either concurrency or wait time, or both.

The snapshot monitor provides some information about the activity of the agents. Example 11-9 shows the information that you see from taking a snapshot of the database manager and *grep* for agent.

*Example 11-9 Agent information from the snapshot monitor*

---

```
db2 get snapshot for dbm | grep -i agent
```

High water mark for agents registered	= 16
High water mark for agents waiting for a token	= 0
Agents registered	= 16
Agents waiting for a token	= 0
Idle agents	= 10
Agents assigned from pool	= 92
Agents created from empty pool	= 18
Agents stolen from another application	= 0
High water mark for coordinating agents	= 16
Max agents overflow	= 0
Gateway connection pool agents stolen	= 0

---

#### 11.6.4 Prefetcher

Refer to 11.2, “Performance related aspects of DB2 architecture” on page 322, to learn how the prefetcher works. Prefetching can be configured using the `NUM_IOSERVERS` database parameter. Use as many prefetchers as the agents allow DB2 to perform the prefetching requests in parallel.

Use the snapshot monitor for buffer pools and calculate the difference of *buffer pool data physical reads* and *asynchronous pool data page reads* to see how well the prefetchers are working. It is better to define too many prefetchers than too few prefetchers. If you specify extra I/O servers, these servers are not used, and performance does not suffer. Each I/O server process is numbered. The database manager always uses the lowest numbered process, so some of the upper numbered processes might never be used.

Configuring enough I/O servers with the `NUM_IOSERVERS` configuration parameter can greatly enhance the performance of queries for which prefetching of data can be used. To maximize the opportunity for parallel I/O, set `NUM_IOSERVERS` to at least the number of physical disks in the database.

#### 11.6.5 Cleaner

To improve performance in update-intensive workloads, configure more page-cleaner agents. Performance improves if more page-cleaner agents are

available to write dirty pages to disk. This is also true when there are many data-page or index-page writes in relation to the number of asynchronous data-page or index-page writes.

Consider the following factors when setting the value for DB parameter NUM\_IOCLEANERS:

- ▶ Application type
  - If a query-only database will not have updates, set this parameter to be zero (0). 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 be between one and the number of physical storage devices used for the database.
- ▶ Workload

Environments with high update transaction rates may require more page cleaners to be configured.
- ▶ Buffer pool sizes

Environments with large buffer pools may also 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:

- ▶ Reduce the parameter 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.
- ▶ Increase the parameter if either of the following conditions are true:
  - pool\_data\_writes is much greater than pool\_async\_data\_writes
  - pool\_index\_writes is much greater than pool\_async\_index\_writes.

### 11.6.6 Sort heap

The *sort heap* is a part of memory where DB2 UDB stores data during a sort. When the sort heap is big enough, the sort can be done in one operation called a *pipelined sort*. If the sort heap is not big enough to store the whole data for the sort, then a temporary table is created in the buffer pool and a sort is divided in several pieces, which is more time consuming.

Try to avoid sorts that do not fit into the sort heap. The application developer has to verify if the sort is necessary or whether a search query can be performed

without an order by clause. The DBA should monitor if the sort heap is big enough. If not, the DBA must increase the heap size when better performance is necessary and more memory is available.

Be careful when increasing the sort heap. Because the heap is part of the agent memory, for each connection, the space is allocated and that multiplies the memory required by the number of agents.

To determine whether you have a sort performance problem, look at the total CPU time spent on sorting compared to the time spent for the whole application using, for example, the snapshot monitor for database and database manager. If total sort time is a large portion of CPU time for the application, then look at the following values, which are also shown by default:

- ▶ Percentage of overflowed sorts

This variable (on the performance details view of the Snapshot™ Monitor) shows the percentage of sorts that overflowed. If the percentage of overflowed sorts is high, increase the SORTHEAP. To learn whether there were any post threshold sorts, use the Snapshot Monitor.

- ▶ Post threshold sorts

If post threshold sorts are high, it indicates that more sort memory is requested than defined with the SHEAPTHRES parameter. Any sort following after this value that is reached receives less sort memory than defined by the SORTHEAP parameter. To avoid this situation, increase SHEAPTHRES, decrease SORTHEAP, or do both.

In general, overall sort memory available across the instance (SHEAPTHRES) should be as large as possible without causing excessive paging. Although a sort can be performed entirely in sort memory, this might cause excessive page swapping. In this case, you lose the advantage of a large sort heap. For this reason, use an operating system monitor to track changes in system paging whenever you adjust the sorting configuration parameters. Also, note that in a piped sort, the sort heap is not freed until the application closes the cursor associated with that sort. A piped sort can continue to use up memory until the cursor is closed.

## 11.6.7 Locking

A lock time-out or long response times reported from the users indicate problems. The LOCKTIMEOUT database parameter specifies the amount of time that an application should wait for a resource that is locked by another application. The value -1 defines an infinite wait time. Any positive values specify the wait time in seconds.

The LOCKTIMEOUT has nothing to do with deadlocks. DB2 recognizes deadlocks using separate process. Setting LOCKTIMEOUT to infinite waiting will cause the application to wait forever, but will not result in a deadlock situation.

## Monitoring locks

Use the snapshot monitor as explained in 11.4.1, “Snapshot monitor” on page 337, to look for lock time-out and deadlock situations. The locks currently held can be monitored by using the following command:

```
DB2 GET SNAPSHOT FOR LOCKS ON database name
```

Example 11-10 shows the first part of the snapshot output of our sample database TRADE3DB. This part shows the current lock situation on the database. You can see that currently three locks are held and that no other application is waiting for any of the locked objects.

*Example 11-10 Snapshot for locks (database part)*

---

Database Lock Snapshot

Database name	=	TRADE3DB
Database path	=	
/home/db2inst1/db2inst1/NODE0000/SQL00002/		
Input database alias	=	TRADE3DB
Locks held	=	3
Applications currently connected	=	3
Agents currently waiting on locks	=	0
Snapshot timestamp	=	11-24-2003 15:08:46.134255

---

From the same output, you can see which application is currently locking one or more objects in the database. The output in Example 11-11 shows the application that held the three locks. The status of the application is UOW Waiting. DB2 UDB is waiting for the user to do something. The output of the snapshot shows all connected applications, so you have a chance to see which application is waiting for a lock.

*Example 11-11 Snapshot for locks (application part)*

---

Application handle	= 18
Application ID	= G9012775.MA0D.017404230648
Sequence number	= 0018
Application name	= javaw.exe
CONNECT Authorization ID	= DB2INST1
Application status	= UOW Waiting
Status change time	= 11-24-2003 15:08:43.201674
Application code page	= 1208
Locks held	= 3
Total wait time (ms)	= 0

---

The list of currently held locks follows the list of connected applications as shown in Example 11-12. From the output, you can see the object that is locked and its mode.

*Example 11-12 Snapshot for locks (list of locks)*

---

List Of Locks

Lock Name	= 0x000000001000000010001940056
Lock Attributes	= 0x00000000
Release Flags	= 0x40000000
Lock Count	= 1
Hold Count	= 0
Lock Object Name	= 0
Object Type	= Internal V Lock
Mode	= S
Lock Name	= 0xA6B2A69FA4A17C7D9175505041
Lock Attributes	= 0x00000000
Release Flags	= 0x40000000
Lock Count	= 1
Hold Count	= 0
Lock Object Name	= 0
Object Type	= Internal P Lock
Mode	= S
Lock Name	= 0x0003000800000000000000000054
Lock Attributes	= 0x00000000
Release Flags	= 0x00000001
Lock Count	= 1
Hold Count	= 1
Lock Object Name	= 8
Object Type	= Table
Tablespace Name	= TRADEDAT

Table Schema	= DB2INST1
Table Name	= ACCOUNTEJB
Mode	= IN

---

If a lot of locks are held or if DB2 UDB finds a lot of deadlock situations, it indicates that something with the application is wrong. From the database side, you can check if there is a long-running transaction that causes the problem. Try to tune the system to make this transaction faster. On the application side, check if the isolation level is needed and try to have a short transaction or to commit as often as possible to reduce locking. When changing the isolation level, both the DBA and application developer should work together to experiment with different isolation levels.

### Lock escalation

Some error situations occur because of lock escalation used by DB2 UDB to reduce locks. Two parameters in the database configuration are responsible for lock escalation. With LOCKLIST, you define the amount of memory used for the locks. The MAXLOCKS parameter indicates the percentage an application can hold of the available locks. If an application requests more locks than specified by this parameter, then lock escalation occurs.

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 stops. If not, it continues until the percentage of the lock list held is below the value of MAXLOCKS.

The MAXLOCKS parameter multiplied by the MAXAPPLS parameter cannot be less than 100. We recommend that you set MAXLOCKS to:

$2 * 100 / \text{MAXAPPLS}$

Lock escalation can result in deadlock and time-out situations. Look into the DB2DIAG.LOG file and search for lock escalation, especially when deadlocks or time-outs are reported.



## 11.6.8 Logging

The LOGBUFSZ database parameter allows you to specify the amount of the database heap (defined by the DBHEAP database 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 situations occurs:

- ▶ A transaction commits or a group of transactions commit, as defined by the MINCOMMIT configuration parameter.
- ▶ The log buffer is full.
- ▶ Some other internal database manager event occurs.

This parameter must also be less than or equal to the DBHEAP database parameter. Buffering the log records results in a more efficient logging file I/O because the log records are written to disk less frequently and more log records are written at each time.

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, also consider the DBHEAP parameter since the log buffer area uses space controlled by the DBHEAP parameter.

You can use the snapshot monitor for the application to determine how much of the log buffer space is used for a particular transaction (or unit of work). Refer to the *unit of work log space used* monitor element.

## 11.6.9 Tablespace

When using DMS tablespace, check if there is enough space available in the tablespaces. Use the snapshot monitor for table spaces and look for the available and used pages in the containers. You can add containers to a tablespace by using the following command:

```
ALTER TABLESPACE name_of_tablespace ADD CONTAINER
```

**Note:** With DB2 UDB V8, it is now possible to drop containers if there is enough space in the other containers to store the data of the container you want to drop. You can also reduce the size of containers and add containers without rebalancing the data.





## Part 5

# Appendixes



## Additional material

This IBM redbook refers to additional material that can be downloaded from the Internet as described below.

### Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG247184>

Alternatively, you can go to the IBM Redbooks Web site at:

[ibm.com/redbooks](http://ibm.com/redbooks)

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-7184.

## Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<b><i>File name</i></b>	<b><i>Description</i></b>
<b>sam.policies-1.2.2.0-05201.i386.rpm</b>	This RPM contains default policies provided by Tivoli System Automation
<b>cluster.zip</b>	Script used for the high availability scenario with Distributed Replicated Block Device (DRBD)
<b>cfgwas_scen5</b>	Script that creates all the necessary Tivoli System Automation resources for WebSphere Application Server for advanced setup
<b>wasctrl-asconx</b>	Update control script for WebSphere Application Server that we used in the advanced setup

## System requirements for downloading the Web material

The following system configuration is recommended:

**Hard disk space:** 1 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.

# Glossary

**authorization role:** A role assigned to Tivoli administrators to enable them to perform their assigned systems management tasks. A role can be granted over the entire Tivoli management region (TMR) or over a specific set of resources, such as those contained in a policy region. Examples of authorization roles include super, senior, admin, and user.

**bulletin board:** The primary mechanism by which the Tivoli Management Framework and Tivoli applications communicate with Tivoli administrators. The bulletin board is represented as an icon on the Tivoli desktop through which the administrators can access notices. Tivoli applications use the bulletin board as an audit trail for important operations that the administrators perform.

**CIM:** See *Common Information Model*.

**classpath:** In the execution environment, an environment variable keyword that specifies the directories in which to look for class and resource files.

**collection:** A container that groups objects on a Tivoli desktop, thus providing the Tivoli administrator with a single view of related resources. Either the Tivoli Management Framework or a Tivoli administrator can create a collection. The contents of a collection are referred to as its members. Examples of collections include the administrator collection, the generic collection, and the monitoring collection; the administrator collection is an example of a collection generated by the Tivoli Management Framework.

**Common Information Model (CIM):** CIM defines the format of system management information. It includes specification about how the information should be written. It also includes a base data structure (schema) for what should be defined. The structure is extensible for any implementation-specific information as indicated in the specification. This CIM-based common definition enables different products and platforms to exchange management information seamlessly.

**configuration repository:** In the Tivoli environment, the relational database that contains information collected or generated by Tivoli applications. Examples of the information stored in the configuration repository include: Tivoli Inventory stores information concerning hardware, software, system configuration, and physical inventory. Tivoli Software Distribution stores information concerning file package operations. Tivoli Enterprise Console stores information concerning events.

**data packet:** A data packet is a combination of the actual data, plus additional information. This can comprise the following information: priority, size, and encryption level.

**default policy:** A set of resource property values that are assigned to a resource when the resource is created.

**deployment request:** A request created by the solution deployer in the enterprise to arrange a deployment in the outlet.

**Distributed Management Task Force, Inc.**

**(DMTF):** DMTF provides systems management specifications for the standardization of managing an IT environment. It provides industry standards for systems management of back-office environments and Web-services-based environments. The standards are platform-independent and not aligned to a specific technology implementation. They include information models, control protocols, and management services specifications. For more information about DMTF, visit their Web site: <http://www.dmtf.org>

**DMTF:** See *Distributed Management Task Force*.

**downcall:** A method invocation from the Tivoli server or gateway “down” to an endpoint.

**employee handheld:** Any handheld wireless device.

**endpoint:** (1) A Tivoli client that is running an endpoint service (or daemon). Typically, an endpoint is a machine that is not used to perform daily management operations. An endpoint communicates only with its assigned gateway. (2) Any managed resource that represents the final destination for a profile distribution. Tivoli Distributed Monitoring also uses the concept of a proxy endpoint, which is a representation for a non-Tivoli entity (such as a network device or a host) that functions as a subscriber for Tivoli Distributed Monitoring profiles. A Tivoli administrator associates each proxy endpoint with a managed node; several proxy endpoints can be associated with a single managed node. (3) In the Tivoli environment, a managed node, PC managed node, or SQL server on which a task is run.

**endpoint client:** See *endpoint*.

**endpoint gateway:** See *gateway*.

**endpoint list:** A list of all endpoints in the Tivoli management region (TMR) and their assigned gateways. This list is maintained by the endpoint manager.

**endpoint manager:** A service on the Tivoli server that controls and configures gateways and endpoints, assigns endpoints to gateways, and maintains the endpoint list.

**endpoint method:** A method that runs on an endpoint as the result of a request from other managed resources in the Tivoli management region (TMR). Results of the method are forwarded to the gateway and then to the calling managed resource.

**enterprise application:** An enterprise application is an application that is installed in the enterprise. Its design and operation are out of scope of this solution.

**enterprise:** An enterprise (or company) consists of all the establishments that operate under the ownership or control of a single organization. For the purpose of this solution, enterprise refers to a corporation, retailers or banks, with large numbers of stores/branches throughout a specific region or throughout the world. Each retail store or bank branch is referred as an outlet.

**event repository:** In the Tivoli environment, the relational database in which Tivoli Enterprise Console stores Tivoli Enterprise Console events.

**fanout:** In communications, the process of creating copies of a distribution to be delivered locally or to be sent through the network.

**gateway method:** A method that runs on the gateway’s proxy-managed node on behalf of the endpoint. Results of the method are forwarded to the calling managed resource.

**gateway:** A service running on a managed node that provides all communications between a group of endpoints and the rest of the Tivoli environment. The gateway also has the multiplexed distribution (MDist) repeater functionality built in, enabling it to act as the fanout point for distributions to a very large number of endpoints.



**Health Policy:** A policy that determines the health of an application.

**ILGW:** See *initial login gateway*.

**initial login gateway:** The initial gateway that all endpoints initially broadcast to be assigned a standard gateway.

**instance:** One occurrence of a resource.

**job:** A resource consisting of a task and its preconfigured execution parameters. Among other things, the execution parameters specify the set of hosts on which the job is to execute.

**Kerberos:** The security system of the Massachusetts Institute of Technology's (MIT's) Project Athena. It uses symmetric key cryptography to provide security services to users in a network.

**Kerberos master machine:** In Kerberos, the host machine on which the Kerberos database resides.

**Kerberos master password:** In Kerberos, the password required to change or access the Kerberos database.

**Kerberos principal:** In Kerberos, a service or user that is known to the Kerberos system. See also *principal name*.

**Kerberos realm:** In Kerberos, a set of managed nodes that share the same Kerberos database.

**kiosk:** Stand-alone, self-service machines.

**managed node:** Any managed resource on which the Tivoli Management Framework is installed.

**Managed Object Format (MOF):** The format for CIM-based information. It is based on the Interface Definition Language (IDL) from the Object Management Group. This allows for the definition of object classes and instances in plain text format to provide readability for both the human reader and computer parsing. MOF-based files can be edited by any text editor.

**managed resource:** In the Tivoli environment, any hardware or software entity (machine, service, system, or facility) that is represented by a database object and an icon on the Tivoli desktop. Managed resources must be a supported resource type in a policy region and are subject to a set of rules. Managed resources include, but are not limited to, managed nodes, task libraries, monitors, profiles, and bulletin boards.

**MDist:** See *multiplexed distribution*.

**MOF:** See *Managed Object Format*.

**multiplexed distribution (MDist):** A service that enables efficient distribution of large amounts of data across complex networks.

**name registry:** A name service consisting of a two-dimensional table that maps resource names to resource identifiers and corresponding information within a Tivoli management region (TMR).

**NetWare managed site:** A resource that represents a Novell NetWare server on which the Tivoli NetWare Repeater is installed, and one or more clients. A NetWare-managed site enables profiles to be distributed through the NetWare server to one or more specified client PCs using either TCP/IP or IPX.

**notice:** A Tivoli message generated by a systems management operation that contains information about an event or the status of an application. Notices are stored in notice groups. See also *bulletin board*.

**notice groups:** An application-specific or operation-specific container that stores and displays notices pertaining to specific Tivoli functions. The Tivoli bulletin board consists of notice groups. A Tivoli administrator can subscribe to one or more notice groups; the administrator's bulletin board contains only the notices that reside in a notice group to which the administrator is subscribed.

**object path:** An absolute or relative path to a Tivoli object, similar to paths in file systems.

**object reference:** The object identifier (OID) given to an object during its creation.

**oserv:** The name of the CORBA-compliant object request broker used by the Tivoli environment. The oserv runs on the Tivoli server and each of its clients.

**PC agent:** Software installed on a client PC that enables Tivoli operations to execute on the PC. See also, *PC-managed node*.

**PC-managed node:** An object that represents a client PC. The Tivoli Management Framework can communicate with the client PC only if the PC agent is installed on the PC. Client PCs are most often referred to as PC-managed nodes.

**Point of Sale (POS):** POS technology uses laser scanners and bar codes to speed checkout.

**policy region:** A group of managed resources that share one or more common policies. Tivoli administrators use policy regions to model the management and organizational structure of a network computing environment. The administrators can group similar resources, define access to and control the resources, and associate rules for governing the resources. The policy region contains resource types and the list of resources to be managed. A policy region is represented on the Tivoli desktop by an icon that resembles the Capitol building (dome icon). When a Tivoli management region (TMR) is created, a policy region with the same name is also created. In this case, the TMR has only one policy region. However, in most cases, a Tivoli administrator creates other policy regions and subregions to represent the organization of the TMR. A TMR addresses the physical connectivity of resources while a policy region addresses the logical organization of resources.

**policy subregion:** A policy region created or residing in another policy region.

**policy:** A set of rules that are applied to managed resources. A specific rule in a policy is referred to as a policy method.

**POS:** See *Point of Sale*.

**principal name:** A name by which the Kerberos principal is identified. The principal name consists of three parts: a service or user name, an instance name, and a realm name.

**principal password:** The password that corresponds to the principal name. This password is used to authenticate services and users to each other.

**profile:** A container for application-specific information about a particular type of resource. A Tivoli application specifies the template for its profiles. The template includes information about the resources that can be managed by that Tivoli application. Examples of profiles include:

- ▶ In Tivoli Distributed Monitoring, a monitor
- ▶ In Tivoli Net.Commander, a host namespace profile
- ▶ In Tivoli Software Distribution, a file package
- ▶ In Tivoli User Administration, a user or group profile

A profile is created in the context of a profile manager. The profile manager links a profile to the Tivoli resource, for example, a managed node, that uses the information contained in the profile. A profile does not have any direct subscribers.

**profile manager:** A container for profiles that links the profiles to a set of resources, called subscribers. Subscribers can be managed nodes or other profile managers. A profile manager can contain profiles of multiple types or multiple profiles of the same type. Tivoli administrators use profile managers to organize and distribute profiles. A profile manager is created in the context of a policy region and is a managed resource in a policy region. See also *subscription list*.

**proxy-managed node:** A managed resource that provides communication between the PC and the Tivoli server.

**pull:** An operation, for example, a Tivoli UserLink file-package request that initiates an action by requesting it of a resource.

**push:** An operation, for example, a file package distribution, that sends information to other resources.

**query:** In the Tivoli environment, a combination of statements that are used to search the configuration repository for systems that meet certain criteria.

**query facility:** In the Tivoli Management Framework, a facility that enables you to use SQL functions to access information in an RDBMS Interface Module (RIM) repository.

**query library:** In the Tivoli environment, a facility that provides a way to create and manage Tivoli queries.

**radio frequency identification (RFID):** RFID tagging enables non-contact, wireless, transmission of product information such as price, manufacturer, expiration date, and product weight. RFID tags, or smart tags, provide the company with a method for managing inventory and the distribution process.

**RDBMS Interface Module (RIM):** In the Tivoli Management Framework, the module in the distributed object database that contains information about the installation of the relational database management system (RDBMS).

**RDBMS:** See *relational database management system*.

**recovery policy:** A recovery policy describes actions that can be performed to recover from a specified failure event situation.

**registered name:** The name by which a particular resource is registered with the name registry when it is created.

**relational database management system (RDBMS):** A collection of hardware and software that organizes and provides access to a relational database.

**repeater range:** The Tivoli clients that receive data from a repeater site.

**repeater site:** In a Tivoli management region (TMR), a managed node that is configured with the multiplexed distribution (MDist) feature. A repeater site receives a single copy of data and distributes it to the next tier of clients.

**resource:** See *managed resource*.

**resource type:** One of the properties of a managed resource. Resource types are defined in the default policy for a policy region.

**RIM:** See *RDBMS Interface Module*.

**RIM repository:** In the Tivoli environment, the relational database management system (RDBMS) database that a Tivoli application accesses through the RIM API. Examples of the information stored in the RIM repository include:

- ▶ Tivoli Inventory
- ▶ Tivoli Software Distribution
- ▶ Tivoli Enterprise Console
- ▶ Tivoli Data Warehouse

**root administrator:** An account for the initial Tivoli administrator that is created during the installation of the Tivoli Management Framework.

**scalable:** Pertaining to the capability of a system to adapt readily to a greater or lesser intensity of use, volume, or demand. For example, a scalable system can efficiently adapt to working with larger or smaller networks, performing tasks of varying complexity.

**scenario:** A set of one-or-more typical interaction dialogs between the users of a system (people or other systems) and a proposed system that is about to be developed. Scenarios are developed during the analysis phase of system development to assist in understanding business events, objects, and interactions. Scenarios document specific transaction sequences, transformations, interfaces, and information exchange. They represent cases that should be included in the Software Quality Assurance Test Plan, and might be used for end user training after the system is completed. Use case scenarios facilitate communication between the people who request a system, analysts, developers, and testers. They are used to validate understanding, and to identify normal and special use situations. Scenarios clarify an evolving agreement between requesters and development teams

Courtesy of the University of Louisiana at Lafayette Information Networks glossary at:  
<http://info.louisiana.edu/dept/gloss.html>

**software package:** The software to be attached to the deployment request.

**software package block (.spb):** A software package block bundles all the resources necessary to execute the actions contained in the software package into a standard zipped format. At distribution time, the resources do not need to be collected from the source host; they are already contained in the software package block.

However, the software package block must reside on the source host. When the software package block is distributed to an endpoint, it is not stored on the endpoint, but is unzipped in the target directory. By unpacking the zipped file immediately, there is no need for additional disk space on the endpoint for the .spb file. A software package that contains all its resources is in the built format. The maximum size of a software package block is 2 GB.

**subscriber:** A managed node or a profile manager that is subscribed to a profile manager. Although profiles are distributed to a subscriber, the subscriber may or may not be the final destination of the profile distribution. See also *endpoint*.

**subscription:** The process of identifying the managed resources to which profiles will be distributed. Managed resources and profiles are associated with each other in profile managers.

**subscription list:** A list that identifies the managed resources that are subscribed to a profile manager. These managed resources, which can include other profile managers, are the recipients of profile distributions. Including a profile manager on a subscription list (in effect, a list within a list) is a way of subscribing several managed resources simultaneously, rather than adding each one individually. In Tivoli Plus modules, a profile manager functions as a subscription list.

**TAP:** See *Tivoli Authentication Package*.

**TAP user:** An optional Windows login and password provided to the TAP. This login is used when Tivoli programs require access to a network resource.

**task:** The definition of an action that must be routinely performed on various managed nodes throughout the network. A task defines the executables to be run when the task is executed, the authorization role required to execute the task, and the user or group name under which the task will execute.

**task endpoint:** See *endpoint*.

**task library:** A container in which a Tivoli administrator can create and store tasks and jobs.

**Tivoli ADE:** Tivoli Application Development Environment. A Tivoli toolkit that contains the complete application programming interface (API) for the Tivoli Management Framework. This toolkit enables customers and Tivoli Partners to develop their own applications for the Tivoli environment.

**Tivoli administrator:** In the Tivoli environment, a system administrator who has been authorized to perform systems management tasks and manage policy regions in one or more networks. Each Tivoli administrator is represented by an icon on the Tivoli desktop.

**Tivoli AEF:** Tivoli Application Extension Facility. A Tivoli toolkit that enables customers to extend the capabilities of Tivoli applications. For example, they can add fields to a dialog box, create custom attributes and methods for application resources, or create custom icons of the Tivoli environment.

**Tivoli Authentication Package (TAP):** A dynamically linked library (DLL) installed by Tivoli, capable of creating Windows security tokens for a different user context. Such tokens can be used for accessing network resources or creating processes in a different user context.

**Tivoli client:** A client of a Tivoli server. Any computer, except the Tivoli server, on which the Tivoli Management Framework is installed. A Tivoli client runs the oserv and maintains a local object database. Contrast with Tivoli server.

**Tivoli desktop:** In the Tivoli environment, the desktop that system administrators use to manage their network computing environments.

**Tivoli Distributed Monitoring:** A Tivoli product that monitors system resources, initiates any necessary corrective actions, and informs system administrators of potential problems. Tivoli Distributed Monitoring consists of a group of monitors that are installed on each Tivoli client that is to be monitored. It resolves some events on its own and can send others to Tivoli Enterprise Console.

**Tivoli EIF:** Tivoli Event Integration Facility. A Tivoli toolkit that provides a simple application programming interface (API) to enable customers and Tivoli Partners to develop new event adapters that can forward events to Tivoli Enterprise Console. A customer can also translate events from third-party or in-house applications.

**Tivoli Enterprise Console:** A Tivoli product that collects, processes, and automatically initiates corrective actions for system, application, network, and database events; it is the central control point for events from all sources. Tivoli Enterprise Console provides a centralized, global view of the network computing environment; it uses distributed event monitors to collect information, a central event server to process information, and distributed event consoles to present information to system administrators.

**Tivoli environment:** The Tivoli applications, based on the Tivoli Management Framework, that are installed at a specific customer location. These address network computing management issues across many platforms. In a Tivoli environment, a system administrator can distribute software, manage user configurations, change access privileges, automate operations, monitor resources, and schedule jobs.

**Tivoli Inventory:** A Tivoli product that enables system administrators to gather hardware and software information for a network computing environment. It scans the managed resources and stores inventory information in the configuration repository.

**Tivoli Management Framework:** The base software that is required to run any of the Tivoli systems management applications. This software infrastructure enables the integration of systems management applications from Tivoli and the Tivoli Partners. The Framework includes:

- ▶ Object request broker (oserv)
- ▶ Distributed object database
- ▶ Basic administration functions
- ▶ Basic application services
- ▶ Basic desktop services such as the graphical user interface (GUI)

In a Tivoli environment, the Tivoli Management Framework is installed on every client and every server with these exceptions:

- ▶ Tivoli Management Framework is never installed on a client PC; rather, the PC agent is installed on the PC.
- ▶ The Tivoli server is the only server that contains the full object database.

**Tivoli management region (TMR):** In the Tivoli environment, a Tivoli server and the set of clients that it servers. An organization can have more than one TMR. A TMR addresses the physical connectivity of resources, while a policy region addresses the local organization of resources.

**Tivoli NetWare repeater:** In the Tivoli environment, a server application that is installed on a Novell NetWare server and that maintains a list of available clients for the server. The Tivoli NetWare repeater works with the NetWare managed site to perform profile distribution.

**Tivoli Remote Execution Service:** A service that enables the Tivoli environment to perform remote operations on machines. These operations include: remotely installing clients, connecting TMRs, and starting oserv from a remote machine.

**Tivoli server:** The server that holds or references the complete set of Tivoli software, including the full object database. Contrast with Tivoli client.

**Tivoli Software Distribution:** A Tivoli product that automates software distribution to clients and servers in a network computing environment. An organization can use this product to install and update applications and software in a coordinated, consistent manner across a network. Tivoli Software Distribution creates file packages and distributes them to predefined subscribers.

**Tivoli User Administration:** A Tivoli product that provides a GUI for centralized management of user and group accounts. It offers efficient, automated management of user and system configuration parameters, secure delegation of administrative tasks, and centralized control of all user and group accounts in a network computing environment.

**TMR:** See *Tivoli management region* (TMR).

**transaction:** A specific set of input data that triggers execution of a specific process or job; a message destined for an application.

**UNIX:** A highly portable operating system originally developed by Bell Laboratories that features multiprogramming in a multiuser environment. UNIX is implemented in the C language. UNIX was originally developed for use on minicomputers, but has been adapted on mainframes and microcomputers. It is especially suitable for multiprocessor, graphics, and vector-processing systems.

**upcall:** A method invocation from an endpoint *up* to the gateway.

**user login map:** A mapping that associates a single user login name with a user account on a specified operating system. User login maps enable Tivoli administrators to log in to the Tivoli environment or perform operations within the Tivoli environment with a single user login name, regardless of the system they are currently using.

**UserLink/DHCP service:** A Tivoli service that provides IP address synchronization between the PC agent and its associated PC managed node.

**validation policy:** A policy that ensures that all resources in a policy region comply with the region's established policy. Validation policy prevents Tivoli administrators from creating or modifying resources that do not conform to the policy of the policy region in which the resources were created.

**virtual login:** See *virtual user*.

**virtual user:** A user ID (UID) mapping set up in the Tivoli Management Framework. A single UID can be mapped to different actual users on different types of architectures. For example, the virtual user \$root\_user can be mapped to root on UNIX machines and Administrator on Windows machines.

**WBEM** See *Web-Based Enterprise Management*.

**Web-Based Enterprise Management (WBEM):** WBEM extends systems management to include Web-based management. Since this is a standard developed by DMTF, it is primarily based on the CIM data model. WBEM provides additional specification for XML encoding for CIM, called *xmlCIM*. It also provides specification for transporting the CIM information over HTTP. The xmlCIM specification defines the XML elements in a standard Document Type Definition (DTD) format. It represents CIM classes and instances, not in MOF format, but XML. The XML format allows easier transport over HTTP, because it is a natural extension to the HTML format.

**Webpad:** A Webpad is a wireless, portable, low-cost, easy to use, tablet-shaped information device with a touch-screen and browser-based interface. Larger than other handheld devices, Webpads typically have 10-inch displays.

**Windows repeater:** The first Windows machine on which the Tivoli Remote Execution Service is installed. Using fanout, the Windows repeater distributes the Tivoli Remote Execution Service to all other Windows servers during the client installation process.





# Abbreviations and acronyms

<b>AIX</b>	Advanced Interactive eXecutive	<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>AMI</b>	application messaging interface	<b>DMA</b>	direct memory access
<b>AMI</b>	Application Messaging Interface alternate mark inversion	<b>DML</b>	Data Manipulation Language
<b>API</b>	application programming interface	<b>DMS</b>	Data Management System
<b>ARTS</b>	Association for Retail Technology Standards	<b>DNS</b>	Domain Name System
<b>ASCII</b>	American Standard Code for Information Interchange	<b>DRBD</b>	Distributed Replicated Block Device
<b>ATA</b>	AT Attachment	<b>DRM</b>	Direct Rendering Manager
<b>ATAPI</b>	AT Attachment with Packet Interface	<b>EAR</b>	enterprise archive
<b>BIGINT</b>	Big integer	<b>EIDE</b>	Enhanced Integrated Drive Electronics
<b>BIOS</b>	Basic Input/Output System-level	<b>EJB</b>	Enterprise JavaBeans
<b>BMP</b>	bean-managed persistence	<b>FIN</b>	SWIFT's store-and-forward message-processing service defining message standards and protocols. SWIFTNet FIN
<b>CD</b>	compact disc	<b>FOR</b>	file-owning region
<b>CD-ROM</b>	compact-disc read-only memory	<b>FS</b>	Fibre Channel service
<b>CICS</b>	Customer Information Control System	<b>FTP</b>	File Transfer Protocol
<b>CLI</b>	call level interface command line interface	<b>GB</b>	gigabyte
<b>CMP</b>	container-managed persistence	<b>GC</b>	Garbage Collector
<b>CORBA</b>	Common Object Request Broker Architecture	<b>GIF</b>	Graphics Interchange Format
<b>CPU</b>	central processing unit	<b>GNOME</b>	GNU Network Object Model Environment
<b>CSD</b>	corrective service disk	<b>GUI</b>	graphical user interface
<b>CSS</b>	cascading style sheets	<b>HA</b>	high availability
<b>CSV</b>	comma-separated-values	<b>HADR</b>	High Availability Disaster Recovery
<b>CVF</b>	Compress VFAT	<b>HAGS</b>	High Availability Group Services
<b>DB</b>	database	<b>HATS</b>	High Availability Topology Services Host Access Transformation Services
<b>DBA</b>	database administrator	<b>HD</b>	half-duplex
<b>DBM</b>	Database management	<b>HDX</b>	half-duplex
<b>DBMS</b>	database management system	<b>HTML</b>	Hypertext Markup Language
<b>DDL</b>	Data Definition Language	<b>HTTP</b>	Hypertext Transfer Protocol
<b>DES</b>	Data Encryption Standard	<b>HTTPD</b>	HTTP daemon

<b>HTTPS</b>	HTTP over SSL Hypertext Transfer Protocol Secure	<b>JVMPI</b>	Java Virtual Machine Profiler Interface
<b>I/O</b>	input/output	<b>KB</b>	kilobyte
<b>IBM</b>	International Business Machines Corporation	<b>KDE</b>	K Desktop Environment
<b>ICMP</b>	Internet Control Message Protocol	<b>LAN</b>	local area network
<b>ID</b>	identifier	<b>LDAP</b>	Lightweight Directory Access Protocol
<b>IDE</b>	integrated development environment Integrated Drive Electronics	<b>LKCD</b>	Linux Kernel Crash Dump
<b>HS</b>	IBM HTTP Server	<b>MAC</b>	Medium Access Control
<b>IIOP</b>	Inter-ORB Protocol	<b>MB</b>	megabyte
<b>IIS</b>	Internet Information Server	<b>MBCS</b>	multi-byte character set
<b>IMS</b>	Information Management System	<b>MDAC</b>	Microsoft Data Access Components
<b>INOUT</b>	input and output	<b>MDC</b>	multidimensional clustering
<b>IP</b>	Internet Protocol	<b>MPM</b>	multiprocessing modules
<b>IPL</b>	initial program load	<b>MPP</b>	massively parallel processing
<b>IPX</b>	Internet Packet Exchange	<b>MQ</b>	message queueing
<b>RES</b>	IBM Retail Environment for SUSE Linux	<b>MQI</b>	Message Queue Interface
<b>IRQ</b>	interrupt request	<b>MSC</b>	Mark, Sweep, Compact
<b>ISP</b>	In-Store Processor	<b>NCQ</b>	Native Command Queuing
<b>ISV</b>	independent software vender	<b>NFS</b>	Network File System
<b>IT</b>	information technology	<b>NLD</b>	Novell Linux Desktop 9
<b>TCAM</b>	IBM Tivoli Composite Application Manager	<b>NLPOS</b>	Novell Linux Point of Service
<b>TCM</b>	IBM Tivoli Configuration Manager	<b>NT</b>	network termination
<b>ITSO</b>	International Technical Support Organization	<b>NTP</b>	Network Time Protocol
<b>JAR</b>	Java archive	<b>NUMA</b>	nonuniform memory access-based
<b>JDBC</b>	Java Database Connectivity	<b>ORB</b>	Object Request Broker
<b>JDK</b>	Java Development Kit	<b>OS</b>	operating system
<b>JFS</b>	Journaled File System	<b>PCI</b>	Peripheral Component Interconnect
<b>JIT</b>	Just In Time	<b>PDF</b>	Portable Document Format
<b>JMS</b>	Java Message Service	<b>PID</b>	process ID
<b>JMX</b>	Java Management Extensions	<b>PM</b>	project manager
<b>JRE</b>	Java Runtime Environment	<b>PMI</b>	Performance Monitoring Infrastructure
<b>JSP</b>	JavaServer Pages	<b>POS</b>	point of sale
<b>JVM</b>	Java Virtual Machine	<b>POST</b>	power-on self test
		<b>PRI</b>	primary rate interface
		<b>PTDV</b>	Performance Trace Data Visualizer
		<b>PXE</b>	Pre-boot eXecution Environment
		<b>RAID</b>	Redundant Array of Independent Disks

<b>RAM</b>	random access memory	<b>TCQ</b>	Tagged Command Queuing
<b>RFID</b>	radio frequency ID	<b>TFTP</b>	Trivial File Transfer Protocol
<b>RM</b>	resource managers	<b>TLOG</b>	Transaction log
<b>RMA</b>	Remote Management Agent	<b>TPA</b>	Tivoli Performance Advisor
<b>RMC</b>	Resource Monitoring and Control	<b>TPV</b>	Tivoli Performance Viewer
<b>RMI</b>	Remote Method Invocation	<b>TSA</b>	Tivoli Systems Automation
<b>RMI/IIOP</b>	Remote Method Invocation over Internet InterORB Protocol	<b>UDB</b>	Universal Database
<b>RPA</b>	Runtime Performance Advisor	<b>UDDI</b>	Universal Description, Discovery and Integration
<b>RPM</b>	Redhat Package Manager	<b>UDF</b>	Universal Disk Format
<b>RR</b>	Repeatable Read	<b>UDF</b>	user-defined functions
<b>RSCT</b>	Reliable Scalable Cluster Technology	<b>UDP</b>	User Datagram Protocol
<b>RSS</b>	Retail Store Solutions	<b>UI</b>	unique identifier
<b>SATA</b>	Serial ATA	<b>UML</b>	Unified Modeling Language
<b>SBCS</b>	Single-byte character set	<b>UR</b>	Uncommitted Read
<b>SBCS</b>	Single-byte character set	<b>URI</b>	Uniform Resource Identifier
<b>SCL</b>	Script Control Language	<b>URL</b>	Uniform Resource Locator
<b>SCSI</b>	Small Computer System Interface	<b>UTC</b>	Universal Time Coordinated
<b>SDK</b>	Software Developer's Kit	<b>VM</b>	virtual machine
<b>SET</b>	Secure Electronic Transaction	<b>WAN</b>	wide area network
<b>SLES</b>	SUSE Linux Enterprise Server	<b>WMI</b>	Windows Management Interface
<b>SMALLINT</b>	Small integer	<b>WRS</b>	WebSphere Remote Server
<b>SMP</b>	symmetric multiprocessing	<b>WSDL</b>	Web Services Description Language
<b>SMS</b>	Short Message Service	<b>WSIL</b>	Web Services Inspection Language
<b>SNMP</b>	Simple Network Management Protocol	<b>XA</b>	extended architecture
<b>SOA</b>	Service Oriented Architecture	<b>XFS</b>	Extended File System
<b>SOAP</b>	Simple Object Access Protocol	<b>XHTML</b>	Extensible Hypertext Markup Language
<b>SPX</b>	Sequenced Packet Exchange protocol	<b>XML</b>	Extensible Markup Language
<b>SQL</b>	Structured Query Language	<b>XSL</b>	Extensible Stylesheet Language
<b>SSL</b>	Secure Sockets Layer		
<b>SUSE</b>	Software Und System Entwicklung		
<b>SWAT</b>	Samba Web Administration Tool		
<b>TB</b>	terabyte		
<b>TCP</b>	Transmission Control Protocol		
<b>TCP/IP</b>	Transmission Control Protocol/Internet Protocol		



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 414. Note that some of the documents referenced here may be available in softcopy only.

- ▶ WebSphere
  - *DB2 UDB V8 and WebSphere V5 Performance Tuning and Operations Guide*, SG24-7068
  - *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
  - *WebSphere Application Server V6 System Management & Configuration Handbook*, SG24-6451
  - *WebSphere Application Server V6.1 Security Handbook*, SG24-6316
  - *WebSphere Studio Application Developer Version 5 Programming Guide*, SG24-6957
- ▶ Tivoli
  - *All About IBM Tivoli Configuration Manager Version 4.2*, SG24-6612
  - *Certification Study Guide: IBM Tivoli Configuration Manager (ITCM) Version 4.2.2*, SG24-6691
  - *Deployment Guide Series: IBM Tivoli Configuration Manager*, SG24-6454
  - *IBM Tivoli Monitoring Version 5.1.1 Creating Resource Models and Providers*, SG24-6900
  - *Implementing a Tivoli Solution for Central Management of Large Distributed Environments*, SG24-6468
  - *Certification Study Guide for IBM Tivoli Configuration Manager 4.2*, REDP-3946
  - *Certification Guide Series: IBM Tivoli Monitoring V 6.1*, SG24-7187

- ▶ DB2 UDB
  - *DB2 Performance Expert for Multiplatforms V2.2*, SG24-6470
  - *DB2 II: Performance Monitoring, Tuning and Capacity Planning Guide*, SG24-7073
- ▶ Linux
  - *Linux Handbook A Guide to IBM Linux Solutions and Resources*, SG24-7000
  - *Tuning SUSE LINUX Enterprise Server on IBM @server xSeries Servers*, REDP-3862
- ▶ Store Integration Framework
  - *Enable the On Demand Store with IBM Store Integration Framework*, SG24-6698
- ▶ xSeries servers
  - *Tuning IBM System x Servers for Performance*, SG24-5287
  - *Help Me Find My IBM @server xSeries Performance Problem!*, REDP-3938

## IBM product documentation

Even if you are a regular user of IBM Redbooks, you should be aware that there is a vast amount of knowledge available only in the product documentation. The product documentation PDFs and online help are updated periodically and can be downloaded from the following locations:

- ▶ IBM Tivoli Configuration Manager  
<http://www.ibm.com/software/tivoli/products/config-mgr/>
- ▶ IBM Tivoli Monitoring  
<http://www.ibm.com/software/tivoli/products/monitor/>
- ▶ IBM Tivoli Monitoring for Business Integration  
<http://www.ibm.com/software/tivoli/products/monitor-integration/>
- ▶ IBM Tivoli Monitoring for Databases  
<http://www.ibm.com/software/tivoli/products/monitor-db/>
- ▶ IBM Tivoli Monitoring for Web Infrastructure  
<http://www.ibm.com/software/tivoli/products/monitor-web/>

- ▶ IBM WebSphere Application Server  
<http://www.ibm.com/software/webservers/appserv/was/>
- ▶ IBM WebSphere MQ  
<http://www.ibm.com/software/integration/wmq/>
- ▶ IBM DB2 Universal Database  
<http://www.ibm.com/software/data/db2/udb/>
- ▶ IBM Store Integration Framework  
<http://www.ibm.com/products/retail/products/software/sif/index.html>
- ▶ IBM Retail Environment for SUSE Linux  
<http://www.ibm.com/products/retail/products/software/ires/>

## Other publications

These publications are also relevant as further information sources:

- ▶ DB2 UDB
  - *IBM DB2 Universal Database Federated Systems Guide Version 8 Release 1*, GC27-1224
  - *IBM DB2 Universal Database Installation and Configuration Supplement V8.2*, GC09-4837-01
  - *IBM DB2 Universal Database Quick Beginnings for DB2 Clients V8.2*, GC09-4832-01
  - *IBM DB2 Universal Database Quick Beginnings for DB2 Servers V8.2*, GC09-4836-01
  - *IBM DB2 Universal Database Administration Guide: Implementation V8.2*, SC09-4820-01
  - *IBM DB2 Universal Database Administration Guide: Performance*, SC09-4821
  - *IBM DB2 Universal Database Administration Guide: Planning V8.2*, SC09-4822-01
  - *IBM DB2 Universal Database Application Development Guide: Building and Running Applications V8.2*, SC09-4825-01
  - *IBM DB2 Universal Database Application Development Guide: Programming Client Applications V8.2*, SC09-4826-01
  - *IBM DB2 Universal Database Application Development Guide: Programming Server Applications*, SC09-4827

- *IBM DB2 Universal Database Command Reference V8.2*, SC09-4828-01
- *IBM DB2 Universal Database Data Movement Utilities Guide and Reference V8.2*, SC09-4830-01
- *IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference*, SC09-4831
- *IBM DB2 Universal Database SQL Reference, Volume 1 V8.2*, SC09-4844-01
- *IBM DB2 Universal Database SQL Reference, Volume 2 V8.2*, SC09-4845-01
- *IBM DB2 Universal Database System Monitor Guide and Reference V8.2*, SC09-4847-01
- *IBM DB2 Universal Database What's New V8.2*, SC09-4848-01
- *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1 V8.2*, SC09-4849-01
- *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2 V8.2*, SC09-4850-01
- *IBM DB2 Universal Database Guide to GUI Tools for Administration and Development*, SC09-4851
- *IBM DB2 Universal Database Data Warehouse Center Application Integration Guide Version 8 Release 2*, SC27-1124-01
- *IBM DB2 XML Extender Administration and Programming*, SC27-1234
- ▶ Tivoli
  - *End-to-End Automation Management User's Guide and Reference*, SC33-8211-02
  - *IBM Tivoli System Automation for Multiplatforms: Base Component User's Guide, Version 2.1*, SC33-8210-04

## Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ “Fine-tuning Java garbage collection performance” by Sumit Chawla  
<http://www.ibm.com/developerworks/ibm/library/i-gctroub/>
- ▶ “High-availability middleware on Linux, Part 1: Heartbeat and Apache Web server” by Hidayatullah Shaikh  
<http://www.ibm.com/developerworks/library/l-halinux/?ca=dgr-lnxw03HiLinP1>



- ▶ “Performance Testing Protocol for WebSphere Application Server-based Applications” by Alexandre Polozoff  
[http://www.ibm.com/developerworks/websphere/techjournal/0211\\_polozoff/polozoff.html](http://www.ibm.com/developerworks/websphere/techjournal/0211_polozoff/polozoff.html)
- ▶ Association for Retail Technology Standards (ARTS)  
<http://www.nrf-arts.org/>
- ▶ DB2 Information Center  
<http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp>
- ▶ DRBD  
<http://www.drbd.org>
- ▶ IBM Linux Technology Center  
<http://ltc.linux.ibm.com/>
- ▶ IBM Retail Store Solutions Publications  
<http://www.clearlake.ibm.com/store/support/html/pubs.html>
- ▶ IBM Tivoli Monitoring for Web Infrastructure Information Center  
<http://publib.boulder.ibm.com/infocenter/tiv3help/index.jsp?toc=/com.ibm.itmwi.doc/toc.xml>
- ▶ IBM WebSphere Business Integration Information Center  
<http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp>
- ▶ Novell Linux Point of Service documentation  
<http://www.novell.com/documentation/nlpos9/index.html>
- ▶ SUSE Linux Enterprise Server 9 documentation  
<http://www.novell.com/documentation/sles9/index.html>
- ▶ Tivoli System Automation  
<ftp://ftp.software.ibm.com/software/tivoli/products/sys-auto-linux/>
- ▶ Tivoli System Automation for Multiplatforms downloads  
<http://www.ibm.com/software/tivoli/products/sys-auto-linux/downloads.html>
- ▶ WebSphere Application Server V5.1 Information Center  
<http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1/index.jsp>
- ▶ WebSphere MQ Information Center  
<http://www.ibm.com/software/integration/wmq/library/infocenter/>

## 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](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## Numerics

3DES 309

## A

access intent 267, 270  
    read 270  
access log 295  
access patterns 253  
access plan 334, 351, 379  
activate 347  
Activate at 267  
    once 267  
    transaction 267  
active threads 260, 304  
ActiveThreads count 304  
administrative server 45  
    operating system images 51  
    system time 49  
agent 381  
agent, page-cleaner 383  
AKtools 226  
Apache 297, 299  
Apache HTTP Server 226  
application assembly performance 266  
application design 318, 334  
application performance  
    health 277  
    testing 224  
application programming interface (API) 17  
application server 4, 10, 15  
application tuning 356  
ARTS (Association for Retail Technology Standards) 413  
Association for Retail Technology Standards (ARTS) 413  
asynchronous buffer writers 362  
asynchronous page cleaners 362  
asynchronous pool data page reads 383  
asynchronous read/write 323  
asynchronous send 317

## B

bdf flush 172  
bean cache 267  
bean-managed persistence (BMP) 270  
block size 175  
BMP (bean-managed persistence) 270  
bottleneck 249, 251–252, 258, 260, 265  
    analyzing performance 207  
    CPU 211  
    disk 216, 218  
    disk actions 220  
    garbage collection 276, 285  
    identification of 207  
    memory 213  
    network 220  
    processing 249  
branch server 45  
    automated remote installation 45  
    NTP service 50  
    POS device 53  
    RSYNC directory 52  
    TFTP service 50  
browser image 51  
buffer pool 322, 360, 375  
    data page management 361  
    data physical reads 383  
BUFFPAGE parameter 375  
build server 45  
bulk encryption/decryption 307  
Business Integration 10, 410  
BusyThreads count 304

## C

C or C++ routine 371  
C++ 371  
cache 256, 268  
    discards 257  
cache size 256–257  
caching 306  
capacity 357  
    planning on WebSphere Remote Server 136  
catalog table 367  
cell 113

- central office 44
- channel parameters 319
- cipher suite 308
- class 352
  - garbage collection 286
- clean page 361
- client, WebSphere MQ 317
- CLOSE\_WAIT 289
- cluster 266
- clustering 381
- CMP (container-managed persistence) 270–271
- CMP enterprise beans 259
- code page 370
- com.ibm.CORBA.MaxOpenConnections 310
- com.ibm.CORBA.ServerSocketQueueDepth 310
- concurrency 327, 372
- concurrency test memory leak 279
- concurrent user 251
- concurrent waiters 254
- Configuration Advisor 355
- configuration file 47
- connection
  - keep-alive 308
  - manager 248
  - pool 254–257
    - size 254
  - pooling 254
- ConnectTimeout 295
- consistency 359
- constraint 359
- container 325, 363, 366
  - bean managed transaction 271, 273
  - transaction
    - mandatory 271
- container transaction 267, 271
  - never 271
  - NotSupported 271–272
  - required 271–272
  - RequiresNew 271–272
  - support 271–272
- container-managed persistence (CMP) 270–271
- CORBA 229, 231
- count 351
- CPU 245, 357
  - bottleneck 211
  - subsystem 161
  - utilization 244
- cursor stability 373
- CustomLog 295

## D

- data source queues 254
- data transmission time 300
- data type 358
- database agent 326
- database design
  - information gathering 357
  - logical 358
  - physical 360
- database row lock 271
- database server 165
- date data type
  - DATE 332
  - TIME 332
  - TIMESTAMP 332
- DB2
  - architecture, performance aspects 322
  - diagnostic log 353
  - performance, repeatable read 269
- DB2 High Performance Unload for Multiplatforms 336
- DB2 Memory Visualizer 378
- DB2 UDB 8.2.4 Workgroup Server
  - overview 18
- DB2 UDB configuration parameter
  - BUFFPAGE 375–376
  - CHNGPGS\_THRESH 362
  - CPUSPEED 351
  - DFT\_MON\_BUFPOOL 376
  - DFT\_MON\_TIMESTAMP 337
  - DFT\_QUERYOPT 352
  - DIAGLEVEL 354
  - INTRA\_PARALLEL 380
  - LOCKSIZE 375
  - LOGBUFSZ 389
  - MAX\_CONNECTIONS 382
  - MAX\_COORDAGENTS 382
  - MAXAGENTS 381
  - MAXCAGENTS 382
  - MAXLOCKS 388
  - MON\_HEAP\_SZ 337
  - NUM\_INITAGENTS 382
  - NUM\_IOSERVERS 383
  - NUM\_POOLAGENTS 382
  - SHEAPTHRES 385
  - SORTHEAP 385
- DB2 UDB monitoring and tuning tools 337
- DB2 UDB Recovery Expert for Multiplatforms 335
- DB2 UDB Table Editor for Multiplatforms 336

- DB2 UDB Web Query Tool for Multiplatforms 336
- DB2 UDB Workgroup Server performance tuning 322
- DB2\_PARALLEL\_IO 326
- db2advis 355
- DB2DIAG.LOG 353
- db2evmon 351
- db2exfmt utility 352–353
- db2expln 352
- dbname 5
- dbpassword 5
- dbuser 5
- deactivate 347
- deadlock 245, 255–256, 347
- declared global temporary table 372
- declared temporary table 333
- denormalization 359
- DependsOn relationship type 112
- DES 309
- deserializable dirty read 269
- Design Advisor 355, 380
- diaglevel 354
- dirty page 361
- dirty read 269
- disk 358
- disk bottleneck 216
  - iostat command 218
  - vmstat command 218
- disk IO 245
- disk subsystem, adding drives 220
- disk technology 166
- distributable Web application module 267, 273
- dmesg command 184–185
- DMS 361
- dynamic cache service 306
- dynamic SQL statement 340, 369
- dynexpln 352

## E

- EJB (Enterprise JavaBean) 258
  - caching option A 267–268
  - caching option B 267–268
  - caching option C 267–268
  - client 258
  - container 257
    - cache settings 257
    - cache size 257
    - cleanup interval 257

- EJB 2.0 273
- ejbActivate 268
- elevator algorithm 170
- e-mail server 165
- end-to-end management 24
- Enterprise JavaBean (EJB) 258
  - option A caching 267–268
  - option B caching 267–268
  - option C caching 267–268
- Enterprise Server
  - defined 27
- Enterprise server 43
- Enterprise Server Manager interface 304
- entity bean 255
- Entity EJBs
  - Activate at 267
  - Load at 267
- error 354
- event monitor 347
- exception 245
- execute() 306
- extent size 363

## F

- fast messages 319
- FENCED stored procedure 370
- file and print server 165
- file system, tuning 164, 169
- filemon 287
- FIN\_WAIT\_2 289
- findByPrimaryKey 271
- firewall 15
- flexibility 361
- formula 378
- free command 199

## G

- garbage collection 252, 274, 277, 280
  - algorithm 284
  - bottleneck 276, 278
  - class 286
  - common problems 283
  - compaction phase 275
  - concurrent mark mode 276
  - gauge 277
  - generation 284
  - HotSpot JVM 284
  - incremental compaction mode 276

- mark phase 275
- monitoring 277
- parallel mark mode 276
- parallel sweep mode 276
- phases 275
- sweep phase 275
- garbage collection call 277
  - duration of 281
  - length of 281
  - number of 281
  - time between calls 281
- Garbage Collector 274
  - Mark-Sweep-Compact (MSC) technique 274
- garbage object 274
- General Agent 30
  - service 33
- generation garbage collection 284
- global 340
- GNU General Public License 229
- grinder 237

## H

- HA (high availability) 45
- handshake 307
- hardware
  - accelerator 308
  - considerations for Linux 165
- hashtable 278, 280
- Health Center 354
- Health Monitor 354
- heap 246, 266, 277
  - compaction 275
  - consumption 280
  - expansion 275
  - fragmentation 280
  - parameters 274, 280
  - shrinkage 275
  - thrashing 282
  - Xcompactgc 280
  - Xgcpolicy 276
  - Xmaxe 275
  - Xmaxf 275
  - Xmine 275
  - Xminf 275
  - Xms 275
  - Xmx 275
- heap size 278, 285
  - initial 281, 285

- maximum 277, 281, 285
- minimum 277
- tuning 282
- heavy load conditions 305
- high availability (HA) 45
- High Performance Unload for Multiplatforms 336
- hits per
  - day 224
  - hour 224
  - minute 224
  - month 224
- host\_name 5
- HotSpot JVM, Sun JDK 1.3 HotSpot 284
- HP-UX 11i 288
  - tuning JVM virtual page size 288
  - tuning TCP\_CONN\_REQUEST\_MAX setting 288
- HTTP GET 226, 234
- HTTP POST 226
- HTTP transport 264
  - custom properties 263
- httpd command 302
- HTTPD process 300
- httpd.conf 295, 298, 301
- Hypertext Markup Language (HTML) 16
- hyper-threading 161

## I

- I/O bound 216
- I/O server 383
- I/O time 316
- IBM DB2 UDB Performance Expert for Multiplatforms 335
- IBM HTTP Server 265, 293, 297
  - performance tuning 297
- IBM HTTP Server 1.3.28 299
- IBM HTTP Server 2.0 299
  - directives 301
  - mod\_deflate 300
  - multiprocessing 301
  - tuning
    - MaxClients 303
    - MaxRequestsPerChild 303
    - MaxSpareThreads 303
    - MinSpareThreads 303
    - ServerLimit 303
    - StartServers 303
    - ThreadLimit 303

- ThreadsPerChild 302
- IBM HTTP Server powered by Apache 226
- IBM product documentation 410
- IBM Rational Suite TestStudio 238
- IBM Retail Environment for SUSE Linux 39, 46
  - architecture 41–46
  - deployment 43
  - overview 40
  - point-of-sale images 51
  - requirements 54–59
  - Role-based configuration 47
  - software stack 41
- IBM Tivoli Composite Application Manager for Web-Sphere
  - overview 21
- IBM Tivoli Monitoring
  - overview 19
- IDENTITY 358
- IIS permission properties 305
- index 332, 355, 360
  - management 380
  - type-1 332
  - type-2 332
- indicator 354
- init() 273
- initial heap size 281
- initialization routines 273
- in-place table reorg 332
- in-store application 32
- In-Store Processor
  - defined 9
- in-store processor 32
  - branch server 46
  - minimum hardware requirements 29
- in-store processor (ISP) 43
- intra-partition parallelism 382
- in-use page 361
- iostat command 188
  - disk bottleneck 218
- iPlanet 304
- isolation level 267, 269, 327, 373
  - database overhead 269
  - row locking 269
- ISP
  - See In-Store Processor

## J

- J2EE application 306

## Java

- object 32
- performance, swapping to disk 285
- profiler 278
- routine 371
- Java Management Extension (JME) 32
- Java memory tuning 274
- Java Message Service (JMS) 17
- Java Runtime Environment (JRE) 30
- Java virtual machine (JVM) 32, 371
- Java Virtual Machine Profiler Interface (JVMPi) 244, 277
- java.rmi.RemoteException 271
- javax.jts.TransactionRequiredException 271
- javax.transaction.UserTransaction 272
- JDBC Provider
  - Connection pool settings 255
- JDBC provider 255
- JFS (Journaled File System) 287
- JIT (Just in Time) compiler 285
- JME (Java Management Extension) 32
- JMeter 237
- JMX specification 32
- Journaled File System (JFS) 287
- journaling 164
- JRE (Java Runtime Environment) 30
- Just in Time (JIT) compiler 285
- JVM (Java virtual machine) 32, 371
- JVM mode
  - client 284
  - server 284
- JVM tuning 284
- JVMPi (Java Virtual Machine Profiler Interface) 244, 277
  - profiler 277

## K

- KDE System Guard (KSysguard) 192
- keep-alive 308
  - connection 308
- kernel parameter 155
- KSysguard (KDE System Guard) 192

## L

- latency 306
- LDAP 307
  - directory 45
- Linux

- hardware considerations 165
- init command 152
- runlevel command 152
- uname command 155
- version of kernel 154
- zombie processes 188
- Linux kernel 154
  - file system tuning 169
- Linux scheduler 289
- Linux tools
  - dmesg 184–185
  - iostat 188
  - nice command 188
  - top 185
  - uptime 184
- ListenBackLog 305
- Load at 267
- load at 268
  - activation 268
  - transaction 268
- load on startup 267, 273
- load testing tools 225
- LoadRunner 237
- lock 385
  - escalation 388
  - list memory area 375
- logging 164
- logging file systems 318
- logical read 378
- long-running test, memory leak 278
- lspc command 378

## M

- magnus.conf 304
- maintenance 334
- mandatory 271
- Mark-Sweep-Compact garbage collection 274
- Master Agent 30
- Max Application Concurrency value 253
- Max Connections 255
- MaxClients 295–296, 303
- maximum concurrency level 254
- maximum concurrency point 253
- maximum heap size 277, 281
- maximum number of threads 303
- maximum size 259
- MaxKeepAliveConnections 260, 262
- MaxKeepAliveRequests 262–263

- MaxRequestsPerChild 297, 303
- MaxSpareServers 296
- MaxSpareThreads 303
- MBean 31–32, 34
  - server 34
- memory 245, 355, 358
  - allocation error 287
  - allocation fault 274
  - utilization 276
- memory bottleneck 213
- memory leak 241, 274, 277–278, 281
  - concurrency test 279
  - long-running test 278
  - repetitive test 279
  - system instability 278
  - system test 279
  - testing 278–279
  - Tivoli Performance Viewer 280
- memory overuse 278
- Memory Visualizer 355
- menu 364
- Mercury LoadRunner 225
- message 352
  - contention 317
- method extension 269
- Microsoft Data Access Components 230
- Microsoft Internet Information Server (Microsoft IIS) 305
- middleware 11
- min connections 255
- minimum heap size 277
- MinSpareServers 296
- MinSpareThreads 303
- mod\_deflate 300
- monitor 376
- monitoring tools 337
- MPM (multiprocessing modules) architecture 301
- mpm\_winnt 302–303
- mpm\_worker 302
- mpstat command 206
- MQCLOSE 316
- MQCONN 316
- MQGET 316
- MQOPEN 316
- MQPUT 316
- multidimensional clustering 333
  - facility 332
- multiprocessing modules (MPM) architecture 301



## N

- named pipe 348
- native\_stderr.log 282
- netstat command 261–262, 288
- network bandwidth 300
- network bottleneck 220
- network subsystem 179
- network transport time 317
- network utilization 245
- never container transaction 271
- next\_key locking 332
- nice level 187
- NLPOS
  - See Novell Linux Point of Service
- no command 294
- node\_name 5
- nonrepeatable read 269
- normalization 359
- NOT FENCED stored procedure 370
- NOT NULL 358
- NotSupported 271–272
- Novell Linux Desktop (NLD) 58
- Novell Linux Point of Service 41, 43
  - components 46–51
- numeric data 331

## O

- object overuse 277, 281
- object pool 241
- Object Request Broker (ORB) 248, 258–260, 309–310
- online 341
- Open System Testing Architecture (OpenSTA) 229
- OpenLoad 238
- OpenSTA (Open System Testing Architecture) 229
- OpenSTA Commander 234
- operating system 30
  - performance tuning 287
- Operating System (OS) 16
- operations ratio 250
- optimization 352
- optimizer 351–352, 379
- option A caching 267–268
- option B caching 267–268
- option C caching 267–268
- Oracle performance, read committed 269
- ORB (Object Request Broker) 248, 258–260, 309
  - connection cache maximum 310

- ORB service 259
- ORB thread pool size 258, 311
- out of Java heap error 371
- Out of Memory exception 278
- overflow 385
- overhead 337, 355
- over-using objects 277, 281

## P

- page 364
  - clean 361
  - dirty 361
  - in-use 361
- page-cleaner agent 362, 383
- pageout 213
- paging activity 245
- paging indicator 215
- parallelism 380
- pass by reference 309
- pass by value 309
- PCTFREE 379
- peak load 242
- percent
  - maxed 258, 260
  - used 254
- performance
  - access intent read 270
  - analyzing bottlenecks 207
  - breaking point 224
  - compared with security 370
  - factors for WebSphere MQ 316
  - garbage collection 274
  - hardware capacity 248
  - hits per day 224
  - hits per hour 224
  - hits per minute 224
  - hits per month 224
  - load testing tools 225
    - ApacheBench 226
      - advantages 227
      - limitations 226
      - options 228
    - Mercury LoadRunner 225
  - OpenSTA
    - Architecture 231
    - Collectors 235
    - Commander 231
    - create test 235

- define test 234
- features 229
- name server 231
- prerequisites 230
- randomization of requests 234
- record a script 232
- repository 231
- repository host 231
- Script Modeler 232
- Task Group 234
- view test results 236
- Rational TestStudio 225
- Segue SilkPerformer 225
- maintenance 334
- maximum number of users 224
- monitoring - tuning - testing 244
- optimization techniques 318
- Oracle read committed 269
- peak interval 224
- peak request rate 224
- poor coding practices 225
- preventing a decrease 179
- testing tools
  - grinder 237
  - IBM Rational Suite TestStudio 238
  - JMeter 237
  - LoadRunner 237
  - OpenLoad 238
  - Segue SilkPerformer 238
  - TestMaker 237
  - TestNetwork 237
  - WebLOAD 238
  - WebStone 238
- total concurrent users 224
- traffic patterns 224
- tuning parameter hotlist 247
- user base 224
- performance advisor 238, 244
- performance analysis 241
  - load test
    - measure steady-state 245
    - ramp-down time 245
    - ramp-up time 245
  - production level workload 242
  - repeatable tests 242
  - saturation point 243
  - stress test tool 242
  - terminology
    - load 241
    - peak load 242
    - requests/second 242
    - response time 242
    - throughput 242
- Performance Expert for Multiplatforms 335
- performance monitor interface 244
- performance optimization tool 335
  - IBM DB2 High Performance Unload for Multiplatforms 336
  - IBM DB2 UDB Performance Expert for Multiplatforms 335
  - IBM DB2 UDB Recovery Expert for Multiplatforms 335
  - IBM DB2 UDB Table Editor for Multiplatforms 336
  - IBM DB2 UDB Web Query Tool for Multiplatforms 336
- performance testing 224, 242
- performance tuning 223, 322
  - access log 295
  - AIX 287
  - AIX file descriptors 287
  - AIX ulimit 287
  - AIX with DB2 287
  - DB2 log files 287
  - disabling security 306
  - disk speed 248
  - dynamic cache service 306
  - existing environment 223
  - full duplex 248
  - HTTP transport custom properties 263
    - ConnectionIOTimeout 263
    - ConnectionKeepAliveTimeout 264
    - ConnectionResponseTimeout 263
    - KeepAliveEnabled 264
    - MaxConnectBacklog 263
    - MaxKeepAliveConnections 264
    - MaxKeepAliveRequests 264
    - Number of concurrent connections 264
  - IBM HTTP Server Linux 297
  - IBM HTTP Server WebSphere plug-in 293
  - IBM HTTP Server Windows 297
    - ThreadsPerChild 297
  - Java memory 274
  - Linux IBM HTTP Server 297
  - logging 295
  - MaxClients 295
  - MaxConnections Web server plug-in 265
  - MaxSpareServers 296

- Microsoft IIS 305
- Microsoft IIS expected hits per day 305
- Microsoft IIS ListenBackLog 305
- Microsoft IIS memory allocation 305
- MinSpareServers 296
- network 248
- number of requests 249
- ongoing process 225
- operating system 287
- options 215
- ORB 309
- parameter hotlist 247
- pass by reference 309
- pass by value 309
- processing time 249
- processor speed 248
- retryInterval 293
- security 306
- Solaris file descriptor 289
- Solaris kernel semsys 290
- Solaris Max Semaphore 290
- Solaris tcp\_conn\_req\_max\_q 290
- SolarisTCP\_FIN\_WAIT\_2\_FLUSH\_INTERVAL 289
- Solaris TCP\_KEEPALIVE\_INTERVAL 289
- Solaris TCP\_TIME\_WAIT\_INTERVAL 289
- Solaris tcp\_comm\_hash\_size 290
- Solaris tcp\_xmit\_hiwat 290
- Solaris ulimit 289
- Solaris virtual page size 290
- SSL 307
- SSL hardware accelerator 308
- StartServers 296
- Sun JDK 1.3 HotSpot 284
- Sun ONE Web Server 304
- Sun ONE Web Server Active Threads 304
- system memory 248
- Testing - Evaluating - Tuning process 225
- ThreadsPerChild 296
- top-ten monitoring list
  - average response time 239
  - CPU utilization 239
  - datasource connection pool size 239
  - disk and network I/O 239
  - EJB container thread pool 239
  - garbage collection statistics 239
  - JVM memory 239
  - live number of HTTP sessions 239
  - number of requests per second 239
  - paging activity 239
  - Web container thread pool 239
  - Web server threads 239
- transaction log 311
- Web server 292
- Web server reload interval 292
- WebSphere MQ 315
- Windows 291
- Windows 2MSL 291
- Windows MaxUserPort 291
- Windows TcpTimedWaitDelay 291
- Windows ThreadsPerChild 297
- Windows TIME\_WAIT state 291
- XML parser selection 311
- permission 382
- pervasive device 19
- phantom read 269
- physical design 357
- physical read 378
- physical storage device 366
- pipe 202
- piped sort 384
- pmap command 203
- PMI 238
- point-of-sale (POS) 40
- pool size 254
- POS (point-of-sale) 40
- POS client 41
  - image 41, 52
- POS controller 43
- POS device 47, 53
  - flat load definition template 53
- POS image 45
- POS terminal 41
  - remote loading 41
- precompilation 256
- prefetcher 383
- prepared statement 256
  - Cache size 256
- PrepStmt Cache Discard 257
- primary key 359
- process priority 187
- processing bottlenecks 249
- processing time 316
- processing token 382
- processing, up-front 273
- processor subsystem 161
- product documentation 410
- provider, JDBC 255

## Q

- query 355
- queue 251, 254
  - size 250–251
- queue manager 17
- queue, system 248
- queuing before WebSphere 250
- queuing network 248, 250

## R

- RAID 311
  - array 248
- ratio 378
- ratio calculation 249
- Rational TestStudio 225
- reachable object 274
- read committed 269–270
  - Oracle performance 269
- read stability 373
- read uncommitted 269–270
- read/write ratio 306
- rebind 334, 379
- Recovery Expert for Multiplatforms 335
- Red Hat Advanced Server 2.1 289
- Redbooks Web site 414
  - Contact us xvi
- reduce memory usage 268
- reference model 19
- RefreshInterval 292
- ReiserFS, tuning 171
- reload enabled 267, 273
- reload interval 267, 273, 292
- Remote Management Agent 30
  - configuring high availability 130–132
  - overview 30
- Remote Management Viewer 32
- Remote Method Invocation (RMI) 258
- Remote Method Invocation over Internet InterORB Protocol (RMI/IIOP) 258
- reorg command 334
- reorgchk command 334
- repeatable read 269–270
  - DB2 269
- repeatable test 242
- repetitive test
  - memory leak 279
  - module level 279
  - system level 279

- request filtering 300
- required container transaction 271–272
- RequiresNew container transaction 271–272
- ResierFS 168
- resource allocation 273
- resource leaks 246
- response filtering 300
- retail store 10
- retrieve 348
- RetryInterval 293, 300
- RMI (Remote Method Invocation) 258
- RMI/IIOP (Remote Method Invocation over Internet InterORB Protocol) 258
- Role-based configuration 47
- rollback 347
- round robin 325
- RqThrottle 304
- RSYNC directory 52
- runstats command 334, 379
- runtime delays 273
- Runtime Performance Advisor 241

## S

- sar command 191
- saturation point 243–244, 251
  - maximum concurrency point 253
- scalability 371
- Secure Sockets Layer (SSL) 17
- security
  - cache timeout 307
  - compared with performance 370
- Segue SilkPerformer 225, 238
- SELECT statement 369
- seminfo\_semume 290
- serializable 269
  - dirty read 269
  - nonrepeatable reads 269
  - phantom reads 269
- server application 317
- server\_name 5
- ServerLimit 303
- service goal 136
- service() 306
- service-level agreement 136
- severe error 354
- Simple Object Access Protocol (SOAP) 16
- sleeping program 213
- SLES

- See SUSE Linux Enterprise Server
- SLRS
  - See SUSE Linux Retail Solution
- smit 288
- smitty 288
- SMS 361
- snapshot 337, 376
- SNMP
  - adapter 35
  - bridge 34
  - event 34
- SNMP Trap Mapper
  - overview 34
- SOAP
  - See Simple Object Access Protocol
- socket state 245
- software
  - requirement 32
- software component 29
- software distribution 19
- Solaris
  - file descriptor 289
    - seminfo\_semume 290
  - kernel semsys:seminfo\_semopm 290
  - Max Semaphore 290
  - tcp\_conn\_req\_max\_q 290
  - TCP\_FIN\_WAIT\_2\_FLUSH\_INTERVAL 289
  - TCP\_KEEPALIVE\_INTERVAL 289
  - TCP\_TIME\_WAIT\_INTERVAL 289
  - ttcp\_comm\_hash\_size 290
  - ttcp\_xmit\_hiwat 290
  - virtual page size 290
- sort 355
- sort heap 384
- speed 351
- SQL (Structured Query Language) 256, 307, 330, 333, 340, 367
- SQL-bodied routine 372
- SSL
  - bulk encryption/decryption 307
  - cipher suite 308
  - handshake 307
  - hardware accelerator 308
- StartServers 296, 303
- state 354
- Statement Cache Size 257
- static SQL 368–369
- steady memory utilization 281
- stop-the-world (STW) collection 274
- Store Integration Framework 32
  - general understanding 6
  - key component 10
- store procedure 332
- stored procedure 370
  - FENCED or NOT FENCED 370
  - security versus performance 370
- strace command 203
- stress, Web Performance Tools 251
- string data 330
- stripe 367
- Structured Query Language (SQL) 256, 330, 333, 340
  - connections 307
  - performance 307
  - tuning 367
- STW (stop-the-world) collection 274
- Sun JDK 1.3 HotSpot 284
  - new generation pool size 284
  - server warm-up 284
- SUN JVM 284
- Sun ONE Web Server 304
- support, container transaction 271–272
- SUSE Linux
  - ReiserFS 168
- SUSE Linux Enterprise Server (SLES) 30, 41
- SUSE Linux Enterprise Server 8 SP2A 289
- SUSE Linux Retail Solution 42
- swap partition 177
- swapping indicator 215
- swapping to disk 285, 296
- switch 337, 376
- synchronization points 279
- synchronous send and receive 317
- SYSCATSPACE table space 367
- system catalog table 367
- system queue 248
- system resources 318
- system test memory leak 279
- System.gc() 274, 279

**T**

- table 358
  - management 379
  - reorganization 379
- Table Editor for Multiplatforms 336
- table space 325, 362, 366
- Tagged Command Queuing (TCQ) 174

- task 234
- TCP initial connect timeout 294
- TCP stack 294
- tcp\_keepinit 294
- TCQ (Tagged Command Queuing) 174
- temporary table 365
  - declared global 372
- TestMaker 237
- TestNetwork 237
- third normal form 359
- thread 255–256, 258–260, 265, 277, 295, 371
  - Web server 265
- thread pool 254, 258–259, 265
  - Web container 259–260, 266
- thread waits 277
- thread-level parallelism 162
- ThreadLimit 303
- threads
  - idle 303
  - maximum number 303
- THREADSAFE 371
- ThreadsPerChild 296–297, 302
- throughput 242
- throughput curve 243, 251–252
- TIME\_WAIT 261–262
- Tivoli Configuration Manager
  - overview 19
- Tivoli Enterprise 19
- Tivoli Enterprise Console 34–35
  - overview 23
- Tivoli Monitoring
  - overview 19
- Tivoli Monitoring for Databases
  - overview 20
- Tivoli Omegamon XE for Messaging
  - overview 22
- Tivoli Performance Advisor 240
- Tivoli Performance Viewer 238, 244, 253–254, 257–258, 260, 277, 280
- top command 185, 212
- topas command 378
- total memory usage 281
- traffic-vis 200, 221
- transaction 311
- transaction isolation level
  - read committed 269
  - read uncommitted 269
  - repeatable read 269
  - serializable 269

- transaction log 311
- Transaction Log (TLOG) 10
- transaction monitor 326
- transaction rollback 312
- transactions per second (tps) ratio 249
- tuning
  - elevator algorithm 170
  - file system 164
  - file system in a Linux kernel 169
  - network subsystem 179
  - options for performance 215
  - ReiserFS 171
- tuning tools 337
- type-1 index 332
- type-2 index 332

## U

- UDDI 16
- ulimit command 204, 287, 289
- UML 16
- uncommitted read 373
- Unified Modeling Language (UML) 16
- unique key 360
- unit-of-work 373
- up-front processing 273
- uptime command 184, 211
- user ID 5
- utilization 354

## V

- vector 280
- verbosegc 282
- virtual page size 290
- vmstat command 190, 281
  - disk bottleneck 218
- volatile table 381

## W

- WaitingThreads count 304
- WAS\_installdir 4
- Web container 259–260
  - requests 250
- Web performance analysis 241
- Web Performance Tools (WPT) 226
  - stress 251
- Web Query Tool for Multiplatforms 336
- Web server 165

- maximum concurrency thread setting 265
  - process-based 299
  - reload interval 292
  - thread-based 299
- Web server plug-in 259–262, 295
- Web service 15
- Web services 14–16
- Web Services Description Language (WSDL) 16
- Web Services Inspection (WS-I) 16
- Web Services Inspection Language (WSIL) 16
- Web site 17, 19–20, 24, 33
- WebLOAD 238
- WebSphere Application Server 4, 14, 31
  - architecture 6
  - Express 14
  - family 14
  - Network Deployment 15
    - overview 15
  - overview 14
  - product 15
  - system queues 248
- WebSphere Central Site Server 14
- WebSphere MQ 10
  - application types 317
  - client 317
  - configuration 319
  - important feature 17
  - overview 17
  - performance optimization techniques 318
  - performance tuning 315
- WebSphere MQ queue manager 17
- WebSphere Remote Server 9–37
  - Advanced 13
  - Basic 13
  - capacity planning 135
  - components 11
  - Entry 12
  - overview 10
  - tiered offerings 12
  - WebSphere Central Site Server 14
  - WebSphere Systems Management Accelerator 14
- WebSphere Systems Management Accelerators
  - components 29
  - hardware requirements 27–28
  - overview 14, 24
- WebStone 238
- wizard 355
- workload 355, 362
- Workload management 17
- WPT (Web Performance Tools) 226, 251
- WSDL 16
- WSIL 16

**X**

- XA transactions 311
- XHTML 16
- XML parser 311
- xmlCIM 403
- Xnoclassgc 286
- XSL 16

**Z**

- zombie process 188







## HA Architectures & Capacity Planning with WebSphere Remote Server V6

(0.5" spine)  
0.475" <-> 0.875"  
250 <-> 459 pages







**Redbooks**

# Highly Available Architectures and Capacity Planning with WebSphere Remote Server V6

**High availability architecture options in WebSphere Remote Server**

**Capacity planning guidelines for WebSphere Remote Server**

**Performance tuning for WebSphere Remote Server**

IBM WebSphere Remote Server V6 delivers a fully-integrated platform that helps manage remote environments such as retail stores and branch offices. It is a key component of the Store Integration Framework and on demand operating environment. This infrastructure offering extends IBM Enterprise Business Integration technology to distributed locations.

WebSphere Remote Server helps retailers to manage their business more cost-effectively, increase employee productivity, and create a unique shopping experience for their customers. Employees have better access to customers, products, and sales information, thereby increasing productivity and providing better service to customers.

This IBM Redbook introduces highly available architectures using IBM WebSphere Remote Server, capacity planning for store environments, and performance tuning for operating systems and WebSphere Remote Server. It can help IBM Clients and Business Partners integrate these tools into enterprise retail environments. The highly available architecture scenarios and the performance and tuning scenarios were developed and documented in a WebSphere Remote Server V5.1.2.1 environment.

This book also discusses the underlying and related technologies, including the installation and configuration processes. In addition, this book will help you configure and optimize the new release of WebSphere Remote Server V6 in an IBM Retail Environment for SUSE Linux V2 environment.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)