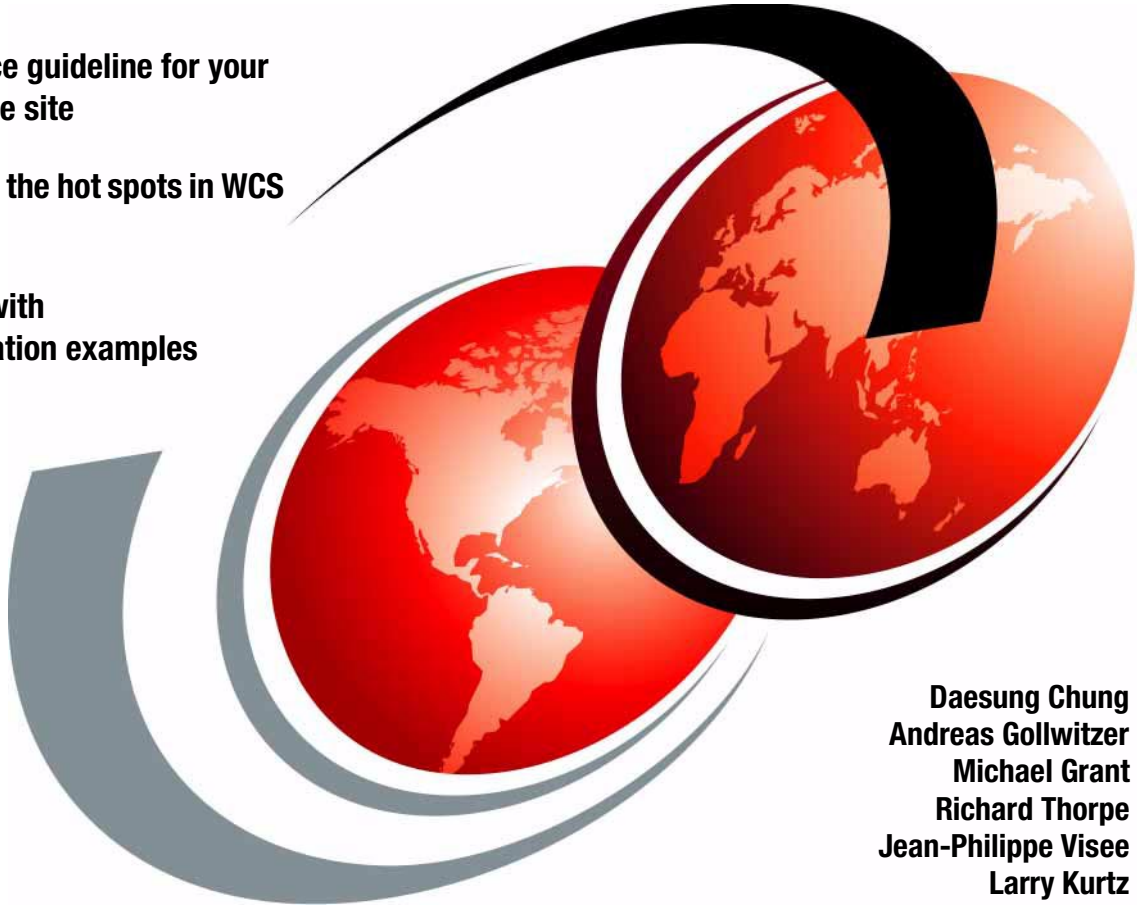
IBM

WCS V5.1 Performance Tuning

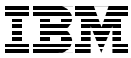
- Performance guideline for your e-commerce site
- Understand the hot spots in WCS
- Explained with implementation examples



Daesung Chung
Andreas Gollwitzer
Michael Grant
Richard Thorpe
Jean-Philippe Visee
Larry Kurtz

Redbooks

ibm.com/redbooks



International Technical Support Organization

WCS V5.1 Performance Tuning

July 2001

SG24-6258-00

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 207.

First Edition (July 2001)

This edition applies to Version 5.1 of IBM WebSphere CommerceSuite Pro, 13P0853 for use with the AIX 4.3.3

This document created or updated on July 6, 2001.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This IBM Redbook provides detailed information on how to tune WebSphere Commerce Suite V5.1. The biggest innovation introduced in WCS V5 is that it is based on pure-Java programming model. Its business components have been rewritten in Java servlets and EJBs. IBM WebSphere Application Server provides the underlying framework for the new programming model. For this reason, the redbook project team devoted considerable amount of time to study the impact of the new Java architecture from the performance point of view. We put emphasis on verifying whether the tuning techniques developed for WebSphere Application Server could be also applied to V5.1. We also introduce various performance monitoring tools that can be used in the tuning process. Some knowledge of WebSphere Application Server V3.5 and DB2 UDB is assumed.

AIX (or NT) only!

A word to note regarding platform specific information. The standard operating system environment the project team tested was AIX. However, we think most of the technical tips and parameter names presented in this book can be equally applied to Windows NT environment. Tips that are applicable to only one operating system are marked as shown on the left.

The team that wrote this redbook

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

Daesung Chung is working at ITSO, Austin Center, and is in charge of e-business solutions on IBM @server pSeries and RS/6000. He writes extensively and teaches IBM classes worldwide on e-business solutions and UNIX. His areas of study are implementation, capacity planning, and performance tuning of WebSphere family products. He also has years of experience in UNIX system management. Before joining ITSO, he was a Consulting IT specialist at IBM Korea.

Andreas Gollwitzer is working at Haitec AG business partner company in Germany as an e-business consultant and system engineer. He has more than two years of experience in WebSphere family products. His areas of expertise include architecture, development, and performance tuning as well as monitoring of WebSphere products based on AIX and Solaris.

Michael Grant is an Advisory Software Engineer at the IBM Toronto Lab in Canada with close to 10 years of experience in the IT field. He has worked at IBM for three years in site performance and capacity planning for WebSphere Commerce Suite solutions. Before joining the IBM Toronto Lab, he was a contract consultant for the Chicago Stock Exchange based in the Middle East, and Team Leader at IBM Global Services in Canada for the Securities Information Services group.

Richard Thorpe is a New Media Developer working in the e-Business Innovation Centre at Hursley in the UK. He has worked at IBM for five years, performing a number of technical roles. Richard has been developing on-line shops for the past three years. He has worked with many different customers, and has a wide range of experience with internet applications.

Jean-Philippe Visee is an IT Specialist in IBM France. He has worked for one and a half years for IBM in the e-Business Innovation Center in Paris. His areas of expertise include Lotus Domino on line web applications and WebSphere Commerce Suite site design and development.

Larry Kurtz is a managing partner of The Preferred Solutions Group, Inc. (www.tpsgi.com), an IBM Premier business partner based in Chicago, Illinois, USA. His background includes over 18 years of programming, system design, and system performance tuning for a variety of clients. Larry currently focuses on designing and supporting secure e-business solutions for his clients using IBM's WebSphere family of products.

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

WebSphere Commerce Suite Development, IBM Toronto Lab
Maurus Cappa, Don Bourne, Joseph Fung, Ivan Lew

AIX Solutions Performance Team, IBM Austin
Bob Minns

IBM EMEA Technical Advocate, DB2 Servers & Architecture
Adrian Lee

International Technical Support Organization, Austin Center
Ernest A. Keenan, Matthew Parente

Special notice

This publication is intended to help customers, business partners, and IBM professionals tune WebSphere Commerce Suite V5.1.The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere V3. See the PUBLICATIONS section of the IBM Programming Announcement for Commerce Suite V5.1 for more information about what publications are considered to be product documentation.

IBM Trademarks

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

IBM ®
e (logo)® 
AIX
MQSeries
Netfinity
pSeries
SP
Wizard
Redbooks
Redbooks Logo 
DB2 Universal Database
Net.Data
NetVista
RS/6000
WebSphere

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an Internet note to

  redbook@us.ibm.com

- Mail your comments to address on Page ii

1

Enhancements in WebSphere Commerce Suite 5.1

WebSphere Commerce Suite has evolved in many ways with the introduction of Version 5.1. It has been completely re-developed to become a full Java application.

The programming model has been re-architected and enhanced to simplify the procedure required for customization. However, V5.1 still provides the same functionality such as product catalog, packages and bundles and shopping carts, and so on. WebSphere Commerce Suite V5.1 also provides the following new functions:

- Multicultural support
- Mobile e-commerce enablement
- A better user management model to allow hierarchies between users inside an organization
- Support of marketing campaigns
- Business intelligence tools

In this chapter we discuss the architectural changes in WebSphere Commerce Suite 5.1 and their implication to performance. Giving detailed information on the new functions is beyond the scope of this book. You can find more information about the new functions in the following sources.

- *What's new in Version 5.1*. The PDF copy of this document is included in product CD under Commerce_Suite_install_path \doc \<locale>\WhatsNew.pdf. It can also be downloaded from `http://www-4.ibm.com/software/webservers/commerce/wcs_pro/WhatsNew.pdf`
- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167

1.1 New architecture of WebSphere Commerce Suite 5.1

WebSphere Commerce Suite has revamped itself from a C++ / Net.Data centric application to a 100% Java based application running on top of WebSphere Application Server. As a consequence, its main architecture is now very similar to a typical WebSphere Application Server application. Taken out of the box, it can be installed in a single tier configuration. But WebSphere Commerce Suite can also be installed in a multi-tier configuration, like any other application running on top of WebSphere Application Server. The primary purpose of this book is to study architectural and configurational issues affecting performance. We mostly study the question of whether tuning techniques developed for WebSphere Application Server can also be applied to WebSphere Commerce Suite 5.1.

By default, WebSphere Commerce Suite is installed with IBM DB2 and IBM HTTP Server. It is beyond the scope of this book to study performance variation caused by using non-standard software components such as Oracle and Netscape iPlanet Server. You may also install optional products provided with WebSphere Commerce Suite such as Commerce Integrator (basically MQSeries), Blaze Advisor, Macromedia LikeMinds, or Payment Manager. However, none of them will be covered in this book.

In the following sections, we are going to review the standard WebSphere Commerce Suite architecture and how we modified it to get better performances.

1.1.1 The standard topologies of WebSphere Commerce Suite

1-Tier topology

If you choose the default settings during installation, WebSphere Commerce Suite will be installed as a single tier, with IBM HTTP Server, IBM WebSphere Application Server, and IBM DB2 Universal Database installed on the same machine.

Figure 1-1 on page 4 presents a diagram of 1-tier configuration.

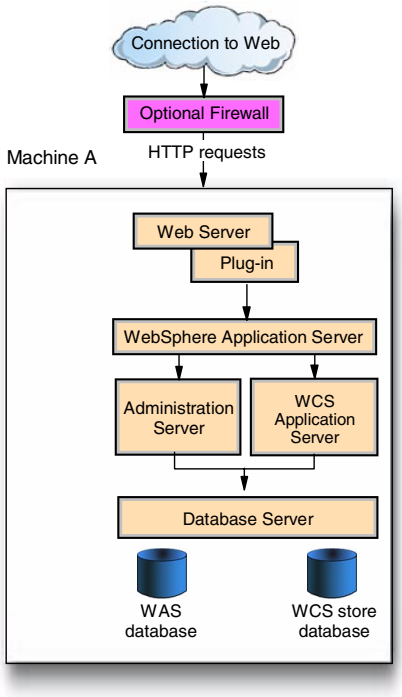


Figure 1-1 1-tier topology

We do not recommend this architecture for a production environment because it does not scale well under high volume of transactions.

2-Tier topology

WebSphere Commerce Suite Version 5.1 allows you to easily implement 2-tier topology, installing a DB2 client on the Web server and the DB2 server on a separate machine. Figure 1-2 on page 5 presents 2-tier topology.

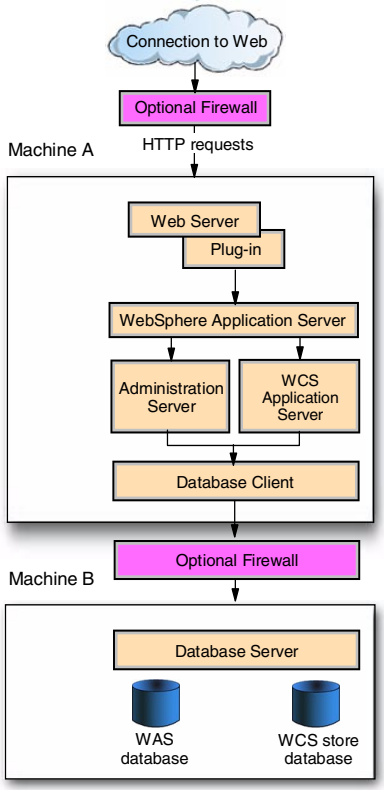


Figure 1-2 2-tier topology

This second architecture allows you to relieve the Web server machine from the load of processing database transactions. You can expand processing power by adding more application server machines in the second tier. The configuration can easily be expanded to clustered configuration balancing workload through WLM.

Figure 1-3 on page 6 shows a horizontal scalability of WebSphere Commerce Suite V5.1. The test was done by gradually incrementing the number of WCS machines in the second tier. As you see in the graph, V5.1 provides excellent horizontal scalability.

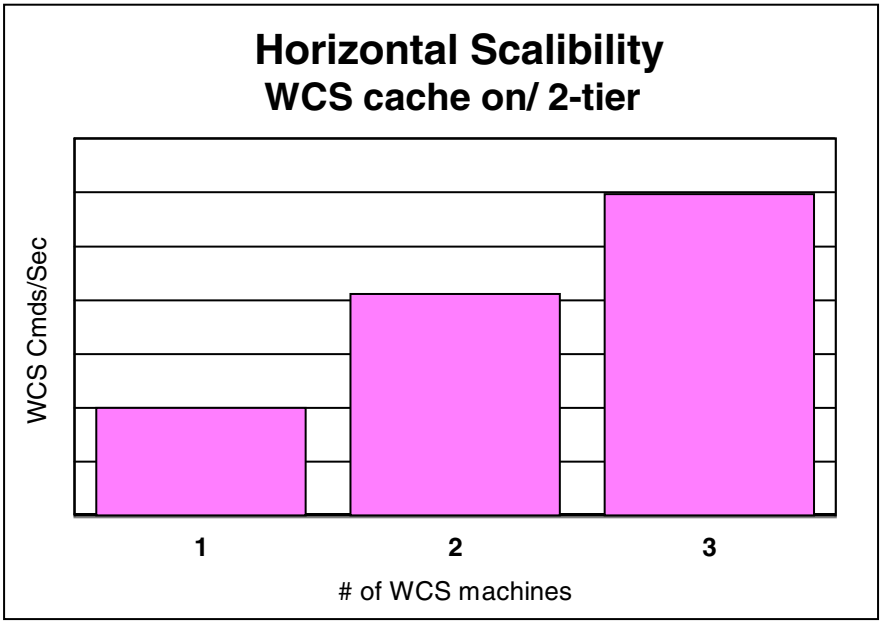


Figure 1-3 Horizontal scalability

3-Tier topology

The third option is to install WebSphere Commerce Suite in a 3-tier topology. This consists of installing IBM HTTP Server and WebSphere Application Server on two separate machines forming the first and second tier. The machine running IBM DB2 is the third tier. Besides superior scalability, this option also provides better network security. You can reinforce the protection level for the WCS machine by adding another firewall between the web server and the WCS server. This is a new feature of WebSphere Commerce Suite 5.1 introduced with the integration into WebSphere Application Server. This 3-tier topology allows to give dedicated machines to each major component of the standard WebSphere Commerce Suite installation. Figure 1-4 on page 7 describes a 3 tier configuration.

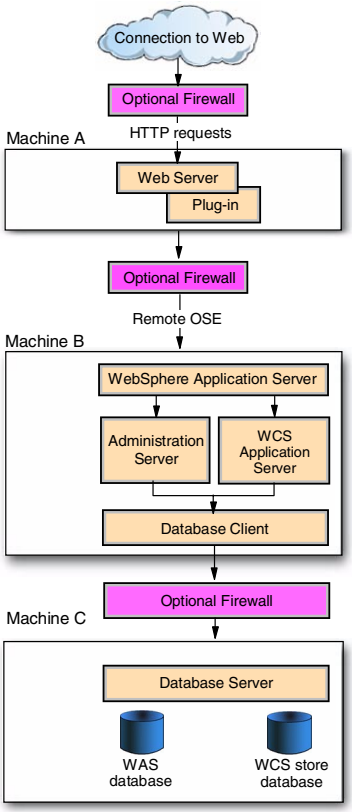


Figure 1-4 3-tier topology

A large-scale topology

If your commerce site needs to process high volume of transactions, you will want to have a more scalable architecture than what was described previously. You may also want build more redundancy in your architecture to provide non-stop service. The architecture shown in Figure 1-5 on page 8 can address such needs.

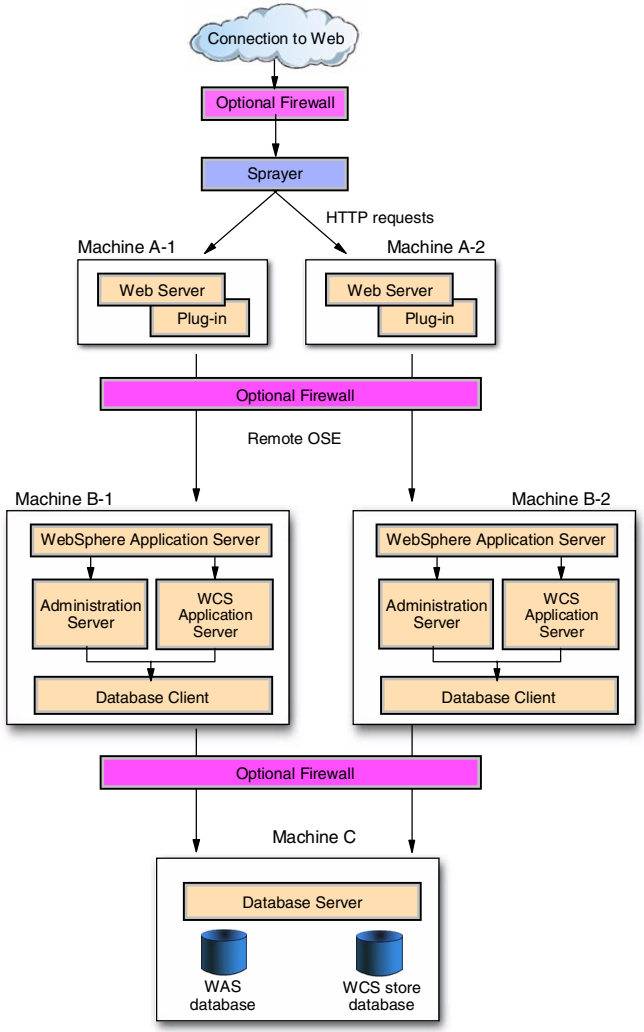


Figure 1-5 Large scale topology using multiple WAS machines

One of the advantages of the above architecture is scalability. This configuration can support far heavier load by accommodating multiple web application servers.

This architecture consists of the following components:

- Application layer:
  - Web browser
  - IP sprayer

  - HTTP server
  - Application server
  - Database server
- Data exchange layer:
  - Network between the web browser and the data flow dispatcher (if any)
  - Network between the IP sprayer and the HTTP server
  - Network or the queue between the HTTP server and the application server
  - Network or the queue between the application server and the database server

Using an IP sprayer such as IBM Network Dispatcher provides both scalability and load balancing to your commerce application. Basically, this product allows you to have multiple servers working in parallel.

Note: Network Dispatcher is now a part of IBM WebSphere Edge Server product. Further information on this product can be found on `http://www-4.ibm.com/software/webservers/edgeserver/`

The HTTP servers forms the front-end layer that handles all the requests for your commerce application, and therefore can be a bottle necking factor. How to tune the IBM HTTP Server is covered in Chapter 6, “Tuning Web Server” on page 127.

On a typical commerce site, most of the pages are dynamic. Requests for dynamic contents on your commerce site are processed by WebSphere Application Server. In a 3-tier topology, HTTP server runs on a machine different from the machine on which WebSphere Application Server is running. WebSphere Application Server provides two options for dispatching the requests from the HTTP server. The first option is to use Servlet Redirector and the second option is to use Open Servlet Engine (OSE) Remote protocol. There are advantages and disadvantages for each of the them. In terms of performance, OSE Remote is typically faster by 15-30% than Servlet Redirector, and we recommend you to use OSE Remote whenever possible. Because WebSphere Application Server is sophisticated middleware, there are many other tunable parameters in the tool. Chapter 5, “Tuning WebSphere Application Server” on page 77 of this book is dedicated to performance tuning of WebSphere Application Server.

Because all of information on your commerce site is stored in a database and every shopping transaction theoretically retrieves data from the database, managing the database performance is critical. Even though WebSphere Commerce Suite provides various ways to minimize database transactions, we think database tuning is one of the most important tasks you should pay attention to. Tuning of IBM DB2 Version 7.1 is described in Chapter 4, “Database tuning” on page 47.

Networking is also an important factor, and can cause a performance bottleneck. However, we excluded network tuning because it is too broad a subject to be contained in this book, and network tuning requires wholly different set of disciplines from that of WebSphere administrator.

Note: From a viewpoint of performance tuning, application tuning is another important topic. Often performance problems originate from a poorly designed piece of code. However, the subject is not discussed in this book, either, because we decided to concentrate on the architectural and component-level perspective.

1.1.2 Our test environment

In order to tune WebSphere Commerce Suite, we have tested several different configurations. All our tests were done on AIX machines connected to a token-ring network. The load was generated by SilkPerformer running on Windows NT machines.

Our test environment is detailed below.

Hardware

The following are the machines we used to do our tests. We used them in many different test scenarios.

- Machine A is a S7A with 1 GB of memory and 12 processors.
- Machine B is a F50 with 1 GB of memory and 4 processors.
- Machine C is a F50 with 1 GB of memory and 4 processors.
- Machine D is a F50 with 1 GB of memory and 2 processors.
- Machine E is a F50 with 512 Mb of memory and 2 processors.
- Machine F is a 44 P 170 with 768 MB of memory and 1 processor.
- Machine G is a 44 P 140 with 768 MB of memory and 1 processor.
- Machine H is a 43 P 140 with 768 MB of memory and 1 processor.

- Machine I is a Netfinity 3000 with 196 MB of memory and 1 processor.
- Machine J is a NetVista with 512 MB of memory.
- Machine K is @server pSeries 640 with 4GB memory, 4 processors, and 33 SSA disks

Software

Here is the list of the software we used for the making of this book:

- WebSphere Commerce Suite Version 5.1:
  - WebSphere Application Server Version 3.5 for AIX with FixPack 2 and eFixes.
  - IBM DB2 UDB Version 7.1 for AIX
  - IBM HTTP Server Version 1.3.12 for AIX
  - WebSphere Commerce Server Version 5.1
- Performance monitoring tools:
  - WebSphere Application Server Resource Analyzer
  - IBM Performance Toolbox 2.2 for AIX
- Workload generator:
  - Segue SilkPerformer Version 3.5.1 for Windows NT

Operating systems

- IBM AIX Version 4.3.3.0 with maintenance level 6
- Microsoft Windows NT 4 with Service Pack 5

Network

All the server machines were on the same network segment except the Windows NT machines that were used as SilkPerformer testing clients.

1.2 WebSphere Commerce Suite application architecture

WebSphere Commerce Suite integrates with many software server components such as the IBM HTTP server, IBM WebSphere Application Server and IBM DB2 Universal Database. In the rest of this chapter, we will see how these components interact with each other in WebSphere Commerce Suite. Figure 1-6 on page 12 summarizes the architectural differences between V4.1 and V5.1.

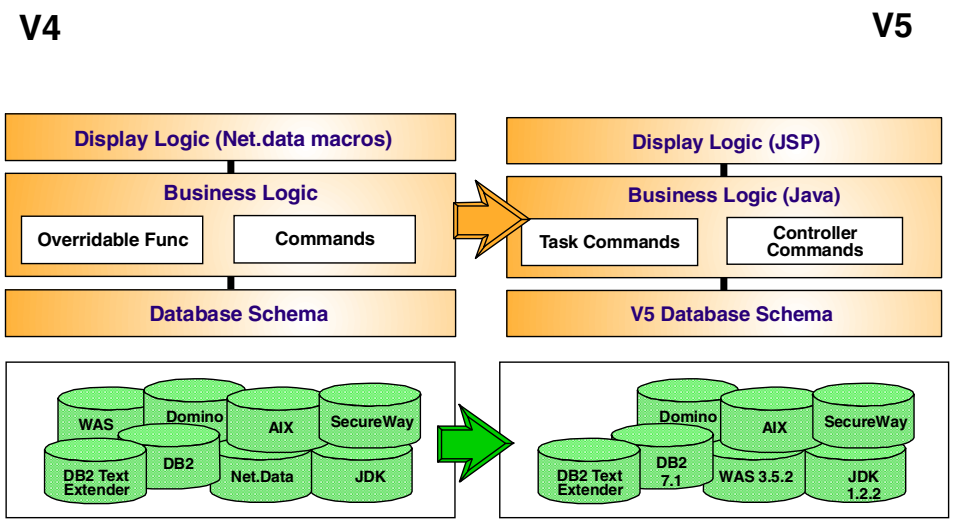


Figure 1-6 Software components of V 5.1 vs. V4.1

1.2.1 WebSphere Commerce Suite and the HTTP Server

In WebSphere Commerce Suite V5.1, a web server provides the communications link between browser-based applications and the other components of WebSphere Application Server. A web server plug-in is installed with the web server that detects and forwards requests for the services of an application server defined in WebSphere Application Server. The application server receives the requests from the plug-in and coordinates services of servlets and Enterprise JavaBeans (EJBs).

The web server plug-in plays a role in the following two respects. First, the plug-in is linked to a WebSphere Commerce Suite's plug-in, which is used by the standard WebSphere Commerce Suite cache.

Note: For this reason, if you want to switch on or off WebSphere Commerce Suite cache, you have to restart your WebSphere Commerce Suite instances in both WebSphere Application Server and your web server.

Second, the plug-in also takes part in workload balancing work under the WLM environment. It looks up a list of available WebSphere Application Server clones, then directs the incoming requests to the most available server.

1.2.2 WebSphere Commerce Suite and WebSphere Application Server

WebSphere Commerce Suite (WCS) runs as an application in WebSphere Application Server. It uses both Enterprise JavaServer (EJS) Servlet Engine and EJB Container. In the traditional Java programming model, JavaServer Pages (JSPs), and Servlets are used in the presentation layer and the EJBs are usually used to implement business logic. However, unlike conventional WebSphere Application Server applications, WCS uses EJBs to access the database whereas it uses servlets and WCS commands to implement business logic. WCS commands are implemented as JavaBeans. You can think of WCS as a servlet/command based engine that uses EJBs to access data and JSPs to implement the presentation layer applications.

The EJB Container

WebSphere Commerce Suite uses EJBs to access all application data. As a consequence, there are nearly 240 EJBs in the WebSphere Commerce Suite for more than 290 tables. Because the number of EJBs are large, initializing the EJBs could be a problem. To solve this problem, WebSphere Application Server Version 3.5 Advanced uses a “lazy initialization” mechanism by default to improve the speed of EJB initialization.

Note: The lazy initialization on EJBs in WebSphere Application Server.

When you call a finder method of an EJB home, WebSphere Application Server responds with an enumeration of “empty” elements. In fact those are true Entity Beans taken from the EJB instances pool, but their state is not yet “ready”. WebSphere Application Server does not do the fetch in the database until a business method has been called on one of the instances brought back in the enumeration.

The EJBs in WebSphere Commerce Suite V5.1 do not contain any business logic. The business logic is written in “commands” and “tasks” Java classes. This reduces the frequency of using EJBs to a minimum value. WebSphere Commerce Suite uses only entity beans and stateless session beans. The stateless session beans are used to process complex SQL requests. In order to simplify the access to the EJBs and optimize their use, WebSphere Commerce Suite accesses EJBs through access beans. Access Beans store EJB values in caches, thus reducing access time.

The bigger the number of jar files containing EJBs, the longer the startup time of the EJB container will be. To resolve this performance issue, all WebSphere Commerce Suite EJBs are packed in a single jar file to achieve best startup performance. The name of the jar file is /usr/lpp/CommerceSuite/lib/wcsejsdeployed.jar.

1.2.3 WebSphere Commerce Suite and DB2

WebSphere Commerce Suite version 5.1 now relies completely on WebSphere Application Server for database connections. The database schema was redesigned completely to support new functions and to fit better with the new persistent objects layer that is now the base of WebSphere Commerce Suite programming model.

In addition to the WebSphere Commerce Suite database, which contains product and shopper information, two additional databases are used; one by WebSphere Administration Server and the other to store persistent session information. Note that persistent session management database is optional, whereas the WebSphere Administration Server database is mandatory.

In case that IBM Payment Manager is used, you need to manage one more database. Payment Manager uses database tables to maintain information on the status of transactions, approval requests, and deposit requests. The records in these tables are kept for working purposes and for tracking and record-keeping. The database also contains configuration information. Monitor the performance of this database as well as the performance of the other two databases. For further information on IBM Payment Manager, refer to `http://www.ibm.com/software/webservers/commerce/paymentmanager`.

2

Quick reference guide

The purpose of the Quick reference section is to provide you with an overview of many of our tuning observations during the writing of this Redbook. The most important observation to point out is that WebSphere Commerce Suite V5.1 is a database application and should be tuned in that fashion. Additionally, tuning is not a quick fix for application performance improvements, but a continual process that needs to be repeated and reviewed on a regular basis. The characteristics of transactions processed by your system site and the performance goal of your site are major factors determining how often you need to review the various tuning ideas presented within this redbook.

The tips presented here are meant to be used as a guideline and not hard and fast rules. Every change that is made will have an effect on other parts of the system. These effects may not be apparent at first, but may become evident over time as the system or database loads increase. Understanding the relationship between the different building blocks of WebSphere Commerce Suite will help you in determining what tuning changes may need to be implemented and what effect they may have on other components.

Some of the quick tips covered in this chapter include:

- HTTP server tuning tips
- Application server tips
- CommerceSuite server tips
- Database tuning tips

- Operating system tips

Some tuning changes can have an immediate effect, while others may not be noticed at all initially. The key point in performance tuning is to take steps and review the results each time. And the first step in any performance tuning exercise is understanding the base line, or how the system is performing before any changes are made. While it is unlikely that a system will perform at its best right out of the box, you should at least test and document what the base line performance is for your system installation. That way once changes are tried and documented, there is something to compare your new results against.

2.1 Overview of tuning procedures

One of the first steps in tuning is to first establish a baseline so that you will know what changes have improved or degraded the system performance.

Tuning the HTTP server is a good next step, as all users who will access your system have to first connect through the HTTP server.

Working with the application server or the middle tier of the application should be next. Because most all HTTP requests will be serviced by the application server, adjusting values here should be made in conjunction with the HTTP server.

Database tuning takes into account many aspects of system performance, from buffer pools to disk layout to indexes. Each one needs to be individually reviewed and monitored as changes are made to maximize your database servers throughput.

2.2 HTTP server tuning tips

There are some performance gains that can be achieved from some simple changes within the HTTP server. The exact values that you will use may vary in your installation based upon your type of installation such as in a 2-tier or 3-tier installation. But even the amount of memory and the number of CPU's installed can have an effect on this value.

MaxClients

But as a general rule, the *MaxClients* parameter should be changed from the default value of 150 to around 50. Again, even a number of 50 may be too high in your environment. This parameter is the maximum number of HTTP server processes running in memory, or the maximum number of web clients making requests at any given time.

The goal in changing this setting is to achieve close to 85 to 90 percent CPU utilization while not waiting on I/O from disks. You can use the **vmstat** command to display average CPU usage for a given time period.

StartServers

This parameter determines the number of HTTP daemons started initially. Its default value is 5. This can be controlled using the *StartServers* directive in the httpd.conf file. Setting this value equal to the value of MaxClients will start the maximum number of HTTP daemons at once.

MaxSpareServer

Setting the *MaxSpareServer* equal to the *MaxClients* setting will keep all of the HTTP servers running and available, reducing startup and shutdown times.

More about HTTP tuning can be found in Chapter 6, “Tuning Web Server” on page 127.

2.3 Application server tips

The WebSphere Application Server offers a number of parameters that can be adjusted that will have an effect on the overall performance of the system, good or bad. Each should be reviewed carefully and changed and tested for your actual system implementation. While most of these settings will provide positive performance gains, your actual system environment will dictate what settings will work best.

JVM Heap

Increasing the JVM heap size is one of the values that is usually changed first. The idea behind this is that if more memory is made available to the JVM, then the underlying Java application will perform better. As a rough sizing guideline, we found 512 MB of physical memory per processor would be optimal.

But more memory is not always better when it comes to performance tuning. Changes should be made and then monitored to determine if any gains have been achieved. There is no reason to have a 1GB heap size if only 512MB is needed. Section 5.2, “Tuning JVM” on page 89 details additional JVM tuning parameters and methods that can be used to monitor JVM performance.

2.3.1 Adjusting Queue Sizes

WebSphere Commerce Suite is a transaction oriented system with requests that are received and queued at various points in the system. Understanding the different queues and how they interact can provide the greatest performance improvements. Setting the queues incorrectly can result in a poorly performing system. Understanding how to determine your system's saturation point will help in determining the proper queue sizes for your system. The following topics highlight some of the areas that should be checked when determining the overall system performance of your site.

Max Connections

Reducing the *Max Connections* of a servlet engine so it equals the maximum number of users that can be supported has been shown to be a far better setting then trying to support something larger. Initially, you will want to open up all of the queue sizes to get the baseline performance for the system and then adjust the queue sizes down from there. This is described in further detail in Section 5.1, "Adjusting queue sizes" on page 78.

Database Connection pool

Changing the minimum and maximum database connection pool size to a value that can sustain DB requests from servlets will keep a constant number of connects established, thus reducing the time it takes to establish a connection. Do not set this number higher then what your database server can support or what is reasonable. This is described in further detail in Section 5.1, "Adjusting queue sizes" on page 78.

Auto-reload

Auto-reload should be disabled or at the very least set to a high time-out value to maximize system performance. If your code is stable and in production, then it is unlikely that it is changing that much and shouldn't need to reloaded. Section 5.3, "Relaxing auto reloads" on page 96 describes in detail how this setting can be changed and what effects it will have on your system.

Transport queue types

If your system is a 2-tier environment, where the HTTP server and Application server are on the same machine you may benefit from the use of Local Pipes. In our test runs on AIX, we observed about a 30 percent increase in total throughput when using Local Pipes. But Local Pipes do not scale very well as the workload increases, so use this setting with caution.

INET Socket is the default for WebSphere Commerce Suite 5.1 and typically performs better on heavy load. INET socket is mandatory in any 3-tier system, where the HTTP server and Application server are on different systems. This is described in further detail in Section 5.1.5, “Adjusting transport queue type” on page 88.

2.4 Commerce Suite server tips

One of greatest performance gains that you can achieve in WebSphere CommerceSuite is enabling the WebSphere Commerce Suite cache. While this is the default setting during installation, it should be checked just the same to insure that it had not been disabled. Our results during testing showed a 200 percent performance improvement when the cache was enabled. To tune other components of WebSphere Commerce Suite, we recommend to turn off the Commerce Suite cache. The reason is that its effect is so big that the effect of tuning other components cannot easily be detected. But it should be re-enabled when your system is deployed in a production environment.

One of the useful tools that we used for monitoring WebSphere Commerce Suite was WebSphere Commerce Suite Performance Monitor. This performance monitoring tool shows what resources are being used in a running application server. Additional information about this tool can be found in Section “WCS Performance Monitor” on page 138.

Cache Wizard

The feature that has the greatest impact on performances is surely the WebSphere Commerce Suite cache. Enabling WCS cache could achieve significant performance improvement, so turn on the cache if possible. However, this technique is not always applicable. For instance, e-commerce sites with heavily personalized pages won’t benefit much from turning on WCS cache.

Note: By default, WebSphere Commerce Suite cache is turned on.

WebSphere Commerce Suite cache allows you to cache pages generated by the URL you specify. By default, only the following commands are cached, but your can add others; StoreCatalogDisplay, TopCategoriesDisplay, CategoryDisplay, ProductDisplay.

WebSphere Commerce Suite provides a Cache Wizard that assists you in adding a new URL to the cache. Section 3.3, “Caching custom WCS commands” on page 35 provides further information on the Cache Wizard and how adding your commands to it will greatly improve the overall system performance.

Call-by-reference

WebSphere Commerce Suite makes use of the call-by-reference setting, which can improve performance by up to 50 percent in some cases. This is the default setting during installation and should not be changed. However, any custom application code that is developed and installed should take this calling method into consideration so as not to modify an object reference, which could cause undesirable results. This is described in further detail in Section 5.7, “Call-by-reference” on page 112.

WebSphere Commerce Suite session management

Web browsers and e-commerce sites use HTTP to communicate. Because HTTP is a stateless protocol, WebSphere Commerce Suite needs a way to manage sessions between the browser and server sides. There are two possible choices for configuring session management in WebSphere Commerce Suite. The first choice is to use cookie-based WCS session management, which is the default option provided by WebSphere Commerce Suite. The other choice is to use WebSphere Application Server’s session management. Switching to WAS session management will have other performance impacts on the systems as well, therefore requiring appropriate tuning. Section 5.5, “Effect of enabling WAS session management” on page 105 provides a detailed description on the benefits of WebSphere Commerce Suite session management versus WAS session management, and how to switch between them.

Prepared statement cache

WebSphere Commerce Suite also makes use of a PreparedStatement cache, which stores the database access plan so that it doesn’t have to be recomputed each time. It improves database access performance. Section 5.6, “Prepared statement cache” on page 110 describes how you can calculate the size of the prepared statement cache, and where to define it.

EJB Cache

EJBs reside in the Enterprise JavaBean container of the WebSphere Application Server, ensuring that the cache settings are important for an application like WebSphere Commerce Suite. WebSphere Application Server has different options associated with the EJB container and 5.4, “Tuning EJB performance” on page 99 provides some examples in determining what settings should be used.

Cloning

One of the goals in performance tuning is to maximize the use of the processor, provided the system is not paging or waiting on I/O from disk. One way to utilize more of the available CPU is to clone your application server on the existing server, otherwise known as vertical scaling. Again, your system environment and

the overall performance that you are achieving will dictate if cloning is needed. Cloning has another benefit as well; that being reliability. Because you have an additional copy of your application server running, it provides failover protection for your application.

2.5 Database tuning tips

Database tuning can be as simple as adding a new index to a table or as complex as separating heavily used database tables to different hard drives, and everything in between. The key point with database tuning is to first understand the data that is being stored and how it is accessed and used. A system created that does mostly reading of data would be tuned differently then a transaction originated system.

Also, a system that is designed to support a small online store would be tuned differently then a database containing a complete listing of phone numbers and addresses of everyone that has a telephone number. But the general tuning principles apply to both scenarios.

Since a database is a collection of data that is stored on a disk sub-system somewhere, there are a number of operating tools that can be used which will provide you details about the disk operation.

The *iostat* and *vmstat* commands can be used to show the amount of reads and writes that various disks are performing, as well as the amount of CPU time that is spent waiting for I/O. In a properly tuned WebSphere Commerce Suite system, there should be very little time waiting on I/O. It is because on a production system WebSphere Commerce Suite cache is turned on to minimize the number of queries passed to the database. Even with the cache turned on, writing to the database is inevitable for certain type of transactions. This is where database buffers and proper database indexes can improve performance. However, a poorly written query that performs table scans cannot be fixed by changing the database buffers. The key point here is to understand the application and what it is doing. That way you can make the correct changes that will have the greatest impact on performance.

2.5.1 Key database tuning parameters

There are several database tuning parameters that effect the operation of the database by altering the amount of memory that can be used or by changing the way the database engine works against the tables that it is managing. Chapter 4, “Database tuning” on page 47 provides many suggestions on various database tuning parameters that can improve performance. A few key observations are listed here as a reference.

Database bufferpool

A database bufferpool allocates memory space for a portion of the database records that are currently being read and stores them in memory, allowing for faster data access to the data. The amount of memory to allocate to this buffer depends on your system environment and performance objectives. Section 4.5, “Adjusting database bufferpool size” on page 63 provides further detail on setting and monitoring the performance of the database buffer cache.

applheapsz

The DB2 application heap size allocates the a block of memory that is used for processing commands, such as updates and queries. The DB2 default value is 128 and it is recommended that this number be increased to 256 for use with WebSphere Commerce Suite. Additional information about the changing the applheapsz can be found in Section 4.8.1, “applheapsz” on page 73. However, to update the applheapsz parameter, use the following command:

```
db2 update db cfg for <database name> using applheapsz <block amount>
```

pckcachesz

The package cache size is a cache which stores database access plans. This reduces the database manager overhead, as it does not have to access the system catalogs or recompile dynamic SQL. As WebSphere Commerce Suite performs a lot of repetitive queries, this parameter is important when tuning the application. Section 4.8.2, “pckcachesz” on page 73 provide additional information about this parameter. However, to update this setting, use the following command:

```
db2 update db cfg for <database name> using pckcachesz <cache size>
```

maxappls

This database parameter defines the maximum number of concurrent applications that can connect to a database. This includes both remote and local applications. The following needs to be considered when choosing this parameter:

- The application data source maximum connection pool size
- The number of cloned servers
- Other database connections, for example, from the DB2 command line
- Maximum connection pool size of session datasource (if persistent session management of WebSphere Application Server is going to be used).

As you can see, the value selected here is related to the values described in Section 5.1, “Adjusting queue sizes” on page 78. Choose a value that makes sense for your environment.

locklist

Locking is the mechanism that the database manager uses to control concurrent access to data in the database. Both rows and tables can be locked.
Section 4.8.4, “locklist” on page 74 describes in more detail the effects of the locklist and the impact it can have with WebSphere Commerce Suite. The goal in setting this parameter is to make certain there are enough entries in the locklist to avoid unnecessary table locks, which occur if the locklist is full. Once a table lock is established other applications will be blocked from updating that table. As WebSphere Commerce Suite relies on several key tables, a heavily loaded system could generate many locks. By increasing the locklist and maxlocks values you can avoid performance costly table locks or deadlocks.

maxlocks

The maxlocks parameter defines the *percentage* of the locklist held by an application. When the number of locks held by any one application reaches this value, the database manager will perform lock escalation for the locks held by that application

2.5.2 Database utilities

The following database utilities alter the database by either reorganizing or removing data that is not longer necessary. Listed below are some key DB2 tuning utilities that can be used to monitor and update the tables that are being managed by DB2.

runstats

The DB2 *runstats* utility updates statistics about the characteristics of a database table and the associated indexes. Whenever a table has a large amount of updates to it, like during an mass import, these statistics should be updated to optimize how the data will be accessed. Section 4.6.1, “runstats” on page 66 describes how this utility can be used.

reorg

The DB2 *reorg* utility can be used to reorganize database tables and eliminating fragmented data, and compacting data. When you run this utility you can specify whether you want to physically order the data in the table by a named index, or simply compact the data without any reordering. This is described in further detail in Section 4.6.2, “reorg” on page 67.

2.5.3 dbclean

With an active system there will come a time when you will need to remove some data that is no longer necessary. dbclean is a flexible and extensible tool for cleaning up your database. It removes obsolete records from WebSphere Commerce Suite database. It is a utility shipped in WebSphere Commerce Suite V5.1. Individual tables can be selected while maintaining referential integrity. How often this is run is really dependent on many factors, the primary one being the amount of data that is required to be maintained and over what period of time. This is described in further detail in Section 4.7.1, “Running dbclean” on page 69. You can also use dbclean to clean up the tables you have added to expand the functionality of your e-commerce site.

2.5.4 Most frequently accessed tables

WebSphere Commerce Suite uses some database tables more than others, and these tables are the most important to clean up over time. If there are excessive old rows, the extra data will reduce the speed of queries against that table, which has a negative effect on performance. Section 4.7.2, “Identifying most frequently accessed tables” on page 70 describes how to determine which database tables are being used the most. After identifying the busiest tables, analyze why they are heavily used. If your custom-built applications are causing excessive transactions on the tables, think about optimizing your codes. If the access to the tables are legitimate, then consider allocating more system resource that will improve performance. Relocating the tables to dedicated disk drives could be one of the good solutions.

2.6 Network tuning

This section applies only to AIX

There are many options for network performance tuning available in AIX. But we will introduce only the most important and significant tuning parameters. For more information on this topic, refer to *IBM Certification Study Guide AIX Performance and System Tuning,* SG24-6184.

2.6.1 Full duplex mode

There is always heavy network traffic between the database server, web server, and WCS server. Network configuration is very important in this respect to achieve good performance. Performance could get worse when you move from 1 tier configuration to 2 or 3 tier configuration. Therefore, be sure to use a network with a large bandwidth such as Fast Ethernet to connect those machines.

When using Fast Ethernet, be sure to check every Ethernet adapter is set to full-duplex mode. Full-duplex mode gives you the maximum throughput. If one the ethernet adapters happens to set to half-duplex mode, the network performance of between the two machines will be degraded.

2.6.2 Maximum Transfer Unit size

The Maximum Transmission Unit (MTU) specifies the maximum size of packets (including all the protocol headers) that can be transmitted on a network. It is important to make sure all systems on the same physical network have the same MTU. The MTU can be displayed using the **netstat -i** command. Table 2-1 gives an overview of common network adapters and their related MTU sizes.

To obtain the current setting:

```
# lsattr -E -l <interface_name>
```

To change the value:

```
# chdev -l <interface_name> -P -a mtu=<newvalue>
```

After changing the value, you should reboot the system to make the change effective.

Because all systems on the same physical network should have the same MTU, any changes made should be made simultaneously. The change is effective across system boots.

Table 2-1 MTU size by network type

Network Type	Default MTU	Maximum MTU	Optimal
Fast Ethernet	1500	1500	1500
Token Ring	1492	17284	4096
FDDI	4352	4352	4352
Gigabit Ethernet	9000		9000

2.6.3 thewall

Specifies the maximum amount of memory, in KB, that is allocated to the memory pool. In AIX Version 4.3.2 and later, the default value is 1/2 of real memory or 1048576 (1 GB), whichever is smaller. thewall is a runtime attribute. Changes take effect immediately and remain in effect until the next reboot. For tuning, increase size, preferably in multiples of 4 KB.

AIX maintains various network buffer areas in memory pool. But AIX has the ability to self-tune those buffers, so there is no need to tune each network buffer parameter. The only option which is not self-tuned is thewall. If the system memory requirements exceed thewall, then it will start to drop packets. If dropped packets are found, then increase the size of the value of thewall parameter. The following shows how to adjust the parameter.

```
no -o thewall
thewall = 262124
#
# no -o thewall=300000
#
# no -o thewall
thewall = 300000
```

2.6.4 rfc1323

Value of 1 indicates that tcp_sendspace and tcp_recvspace sizes can exceed 64 KB. If the value is 0, the effective tcp_sendspace and tcp_recvspace sizes are limited to a maximum of 65535. If you are setting tcp_recvspace and tcp_sendspace greater than 65536, you need to set rfc1323=1 on each side of the connection. Without having rfc1323 set on both sides, the effective values for tcp_recvspace and tcp_sendspace will be 65536. For better performance, we recommend that this always be set to 1. Changes take effect immediately for new connections and remain in effect until the next reboot. The following command can be used to set this option:

```
# no -o rfc1323=1
```

3

Tuning WCS instance

The WebSphere Commerce Suite (WCS) cache offers significant performance improvements. Many of the people using websites only ever browse, without actually buying anything. This browsing activity can result in significant overhead for your servers without generating any sales. The WebSphere Commerce Suite cache reduces this overhead dramatically by caching product and category pages, resulting in reduced overhead on your servers, increased server throughput, and faster response times to the client. When comparing test results with the cache turned on and off, we observed performance improvements between 200% and 300%.

This chapter discusses the following:

- Tuning the WebSphere Commerce Suite cache
- Caching custom commands
- The differences between session dependent caching and session independent caching

3.1 Tuning the WCS cache

When a shopper browses a product or category page, most of the processing time is spent parsing the HTTP request, accessing the database, and dynamically creating the page the shopper requested. These dynamic contents are generated by a few WCS commands. These commands retrieve information from the WCS database, then display the information as a JavaServer Page (JSP). If your product and category information has not changed since the page was last viewed, there is no need to re-create the same contents from the database. WCS maintains a cache for this purpose, and serves previously generated pages much faster by retrieving from the cache. The cached contents are saved as files under *<WCS_install_path>*/instances/*<instance_name>*/cache/ directory.

WebSphere Commerce Suite cache can cache in the following two ways:

1. Session independent caching

   Session independent caching can be used where the same file in the cache is served to each user requesting that page. Session independent caching stores the results of a user specified list of commands and serves these cache pages in response to subsequent requests using the same commands. This method is usually used for applications that do not have personalized pages for individual users.

   In general, session independent caching performs far better than session dependent caching, so you should consider enabling session independent caching first, then enable session dependent caching only when you need to use multiple currencies or member groups. We have observed 200% to 300% overall system performance improvement by enabling caching.

2. Session dependent caching

   Session dependent caching caches pages by session. Use this method for sites with distinct pages for member groups, multiple languages, or multiple currencies. WebSphere Commerce Suite cache looks up the parameters such as the language, currency, or member group when generating or retrieving pages. Then WebSphere Commerce Suite cache manager will check to which member group the user requesting the dynamic contents belongs and figure out in which language the dynamic pages should be displayed in. Once the page has been generated, the same contents will be retrieved directly from the cached files managed by WebSphere Commerce Suite in the next transaction.

In session dependent caching, CacheCommand is called to retrieve custom dynamic page index parameters. Implementations of the CacheCommand interface can be used to provide additional information for the indexing of WCS dynamic cache pages. The CacheCommand implementation class is called during the invocation of every command whose sessionDependent attribute is set to "true" in the Cache section of the WCS configuration file.

During to our tests we observed that session dependent caching only produced a 10% to 15% performance improvement over no caching. This result is not strange considering the fact that more system resources are used in session dependent caching to retrieve extra information from the database as well as to communicate with the application server. Enabling session dependent cache is beneficial if it allows you to cache pages that otherwise would not be cacheable due to dependency on information not contained in the HTTP request. Some personalized pages may belong to this case depending on how they are personalized.

By default, the commands CategoryDisplay, ProductDisplay, TopCategoriesDisplay, and StoreCatalogDisplay are enabled for session dependent caching. You can use any combination of the two caching methods for any of your cacheable commands.You can also enable caching for your custom-built commands. The procedure for this is explained in Section 3.3, “Caching custom WCS commands” on page 35.

In order for a command to be cacheable, the HTML result of that command must not vary greatly for different users viewing the same command with the same parameters. When you apply personalization to your site, consider the following trade-offs between personalization and performance:

- Personalized pages

  When you have pages that are fed by the personalization engine, every page may be unique for each user. Depending on the level of personalization, these differences could be great. Although the WCS cache could cache these pages, it would not provide any benefit, as the cache would have to store a copy of each page for each user. As the personalization rules may cause pages to change depending on user actions, these pages could become invalid after being used once, or cached, but never requested again by the user. In this case the cache would add extra overhead for the server, slowing things down. It is therefore advisable not to cache WCS pages that personalize content. The exception would be where pages have simple personalization implemented, where more than one user would see the same page, and the pages can be distinctly separated by parameters in the URL.

- Catalogs from outside sources

  When catalog pages are not generated by WCS. The WCS cache works by caching WCS commands. This effectively means URLs served through the WCS request servlet. If catalog pages are generated without using the request servlet, then the cache will not be able to detect these pages being requested, and therefore not cache them.

The WebSphere Commerce Suite cache can be tuned to improve performance. There are several parameters to be aware of that can affect the way the cache works. These parameters can be changed either in the WebSphere Commerce Suite configuration manager, or by editing the *<instance_name>*.xml file in the corresponding instance directory. The following table shows the tunable parameters defined in <WCS_install_path>/instances/<instance_name>/xml/*<instance_name>*.xml.

Table 3-1 WebSphere Commerce Suite cache parameters.

Parameter	Default value	Recommended value
Cache enabled	Yes	Yes
CacheDirsPerMember	100	100
AutoPageInvalidation	True	True
Cache invalidation triggers	Not enabled	Application dependent
MaxObjectsPerMember	500	0
CacheFilePath	*<WCS_install_path>*/instances/*<instance_name>*/cache	a directory name in a filesystem residing on dedicated disks to achieve better performance

The main thing to consider when tuning the cache is to cache as many pages as possible. Caching helps minimize the number of database transactions to retrieve information from the database. Retrieving information from the database consumes much more system resource than retrieving from the cached files. If you have a system with a new WCS instance, it is a good idea to run test scripts to generate and store sufficient amount of cached files on your WCS system before the system is put into a production environment.

3.1.1 Enabling Cache

By default, the WebSphere Commerce Suite cache is NOT enabled. Therefore, be sure to enable caching in production mode. To enable caching, invoke WCS configuration manager and click **Caching Sub System** -> **Advanced**, then check the **Cache Enabled** checkbox as seen in Figure 3-1 on page 31.

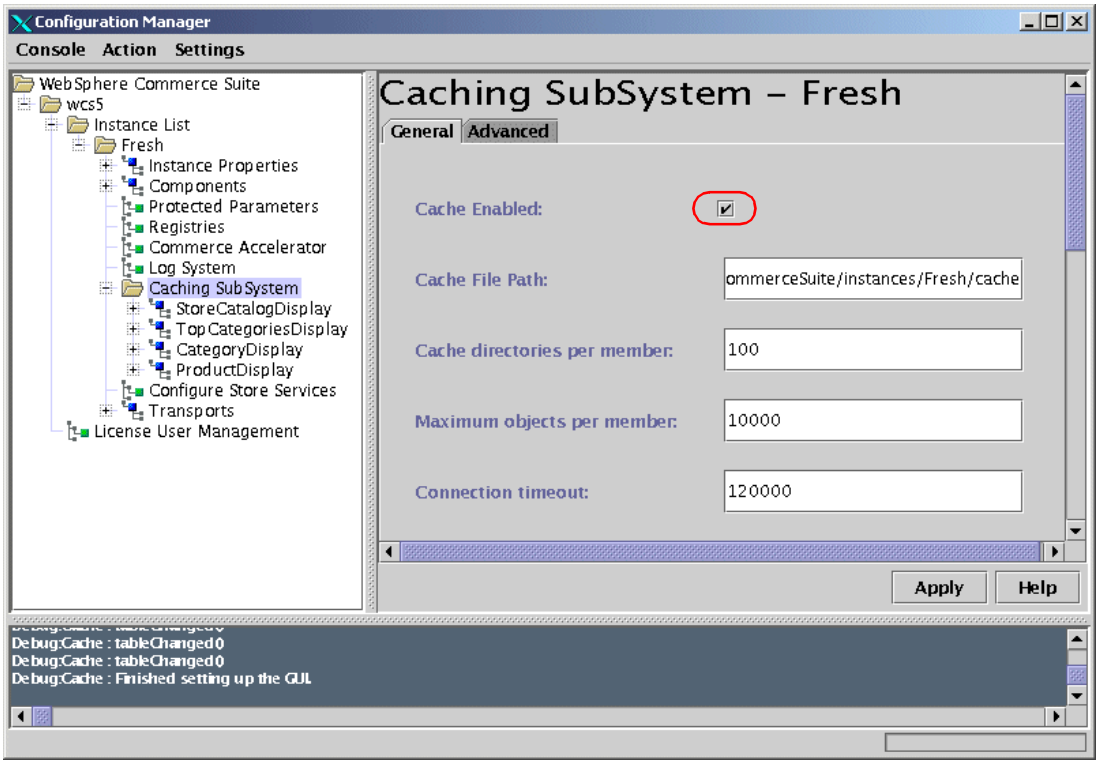


Figure 3-1 Enabling caching with WCS configuration manager

Or edit <*WCS_install_path*>/instances/<*instance_name*>/xml/<*instance_name*>.xml file and change **Cache enabled** to *Yes* as seen in Example 3-1.

Example 3-1 Enabling cache in *instance_name*.xml

```
<Cache MaxObjectsPerMember="10000"
         name="Caching SubSystem"
         MaxAllowedRefreshPeriod="3600"
         CacheDirsPerMember="100"
         AutoPageInvalidation="true"
         CacheConnectionTimeout="120000"
         CacheCleanupAgentHostname="localhost"
         CacheDaemonMaxThreads="64"
         Enabled="true"
```

It is recommended that you change the Maximum objects per member setting from the default value of 500 to 0 (zero) at this point if you have extra disk space for cache storage on your machine. Note that setting the value to 0 (zero) means you have an unlimited number of cache files.

3.1.2 CacheDirsPerMember

This parameter defines the number of subdirectories in the cache directory. If you have more directories, it reduces I/O contention for larger caches. Generally speaking, it is best to have less than 1000 files per directory. If you have a large number of categories and products (total greater than 100,000), then you will need to increase this parameter. Please take into account the number of cached pages that are going to be generated by custom commands you have configured when calculating a value for this parameter. The procedure of configuring cache for a custom command is explained in Section 3.3.1, “Adding custom pages to WebSphere Commerce Suite cache” on page 35.

3.1.3 AutoPageInvalidation

The AutoPageInvalidation parameter enables or disables the cache cleanup worker. This is needed if you are using cache invalidation triggers, or **`CacheDelete`** command. Disabling AutoPageInvalidation will reduce the system overhead slightly, but you are then responsible for cleaning up the cache manually. We recommend that you leave this parameter set to true in most cases, especially if you have a product base that changes frequently.

However, if your product catalog changes infrequently, then you may prefer to delete the cached files manually instead of using cache cleanup worker. If you choose to delete the cache files manually, then you need to identify which file corresponds with the changed product item. You can easily figure out what the file is caching from its naming convention. The following is an example of cached files stored under the WCS cache directory.

```
ProductDisplay.storeId.25.productId.17140.---.htm
```

The **`CacheDelete`** command allows you to force the cache manager to invalidate and delete cached files on request. The command is flexible, as you can use it to delete all the files for an entire store, down to specific product and category pages. For more information on the **`CacheDelete`** command, refer to the WebSphere Commerce Suite online help.

3.1.4 Cache invalidation triggers

WCS maintains a CACHLOG table that records when product or category information is changed in the database. Several tables populate the CACHLOG table using DB2 triggers. The cache uses triggers as a notification mechanism to indicate when an object is invalidated.

The cache invalidation triggers work with the cache cleanup worker. These triggers are activated when changes are made to the tables that manage products and categories. When a change is detected, the triggers add a row to the CACHLOG table, which causes the cache manager to invalidate the relevant page. This makes cache management a lot simpler. The triggers are enabled and disabled in the WebSphere Commerce Suite configuration manager.

However, updating the CACHLOG table could be sometimes consume considerable amount of system resources. If your product and category information is changing constantly, enabling triggers will increase database activity, which can result in performance problems. If this is the case, you should consider either tuning the parameters related with cache tables in the database to achieve better performance, or disabling cache invalidation triggers then deleting cache files from the file system when the product and category data is changed. To disable the triggers, uncheck the Auto Page Invalidation box in WCS configuration manager. Deleting files directly from the cache, either with a custom script or the **`CacheDelete`** command, would eliminate the extra database activity caused by updating the cache as well as updating the product and category data.

If you are going to use the WebSphere Commerce Suite loader facility to make a large update to your site, then it is worth disabling the triggers before you run the update, and then enabling them after the batch update has run. If the triggers are not disabled, then the extra database activity will slow down the update process and your site at the same time. Once the update has been completed, use the **`CacheDelete`** command to clear out the cache.

We recommend that you use the triggers if you make small or infrequent changes to your product and category data. If you are making constant updates, especially during periods of high load, or are running large batch updates to your data, then it is better to use the **`CacheDelete`** command or manually delete the cache files to refresh the cache.

3.1.5 MaxObjectsPerMember

The MaxObjectsPerMember defines the total number of files allowed in the cache. If this command is set to 0, then the cache can contain an unlimited number of files. Setting the value to 0 reduces the amount of work the cache manager has to do. This is because the cache manager does not have to keep deleting old files when it needs to store new pages. Letting the cache manager do housekeeping work may unnecessarily increase both CPU usage and the frequency of disk I/Os. We recommend to set this parameter to an unlimited number of objects in the cache to improve performance.

3.1.6 CacheFilePath

The CacheFilePath tells the cache where to store the pages on the server. By default it is set to *<WCS_install_path>*/instances/*<instance_name>*/cache. To minimize any disk I/O contention, we recommend the cache files be located on a dedicated disk. The disk should be the fastest disk available. When the system shows heavy I/O to the WCS cache directory, consider adding more disk drives to the filesystem containing the cache directory and striping the cache files. According to IBM's internal tests, it could bring a significant performance boost. It is important to make the I/O as fast as possible, as caching gives one of the largest performance gains of all the tuning options.

3.2 Session independent vs. session dependent cache

The session dependency can be set per command cached. For example, you may set the CategoryDisplay command to be session independent, but the ProductDisplay command to be session dependent. How you configure this depends on the level of personalization in your application. This in turn affects the performance gain you can achieve with WebSphere Commerce Suite caching.

To set session dependent cache per command, edit <WCS_install_directory>/instances/<instance_name>/xml/<instance_name>xml as shown in Example 3-2.

Example 3-2 Setting session dependent cache per command

```
<CacheableURL name="CategoryDisplay" sessionDependent="false">
        <KeySet name="Key Set #4" HashKey="categoryId" MemberKey="storeId" />
        <KeySet name="Key Set #5" HashKey="identifier" MemberKey="storeId" />
    </CacheableURL>
    <CacheableURL name="ProductDisplay" sessionDependent="true">
        <KeySet name="Key Set #6" HashKey="productId" MemberKey="storeId" />
        <KeySet name="Key Set #7" HashKey="partNumber" MemberKey="storeId" />
```

```
</CacheableURL>
```

You can also use WCS configuration manager to do the same job.

3.3 Caching custom WCS commands

When you customize WebSphere Commerce Suite, you may create new commands that are frequently used. If it is appropriate to do so, there are significant performance gains to be made by caching these commands.

3.3.1 Adding custom pages to WebSphere Commerce Suite cache

Once you have created your custom command, you need to configure WebSphere Commerce Suite to cache the pages generated by the command. In order for a command to be cacheable, the HTML result of that command must not vary for different users viewing the same command with the same parameters. To configure caching of a custom command, you need to do the following things:

1. Invoke the WebSphere Commerce Suite configuration manager.
2. Expand the list, and select **Caching Subsystem** for your instance.
3. From the menu, choose **Action -> Add a command to cache**.

This opens the Cache Wizard, which takes you through the steps of adding a new URL to the cache. As an exercise, we will add a fictitious URL called TrouserDisplay. This command has the same attributes as ProductDisplay, as well as a new attribute called Color.

1. First, you need to enter the URL to be cached, and the number of keys. Figure 3-2 on page 36 shows the values used for the new command.

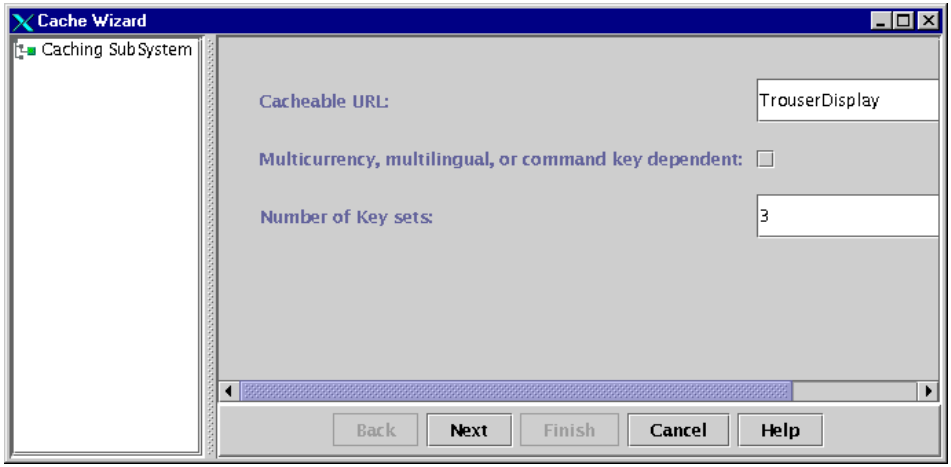


Figure 3-2 Adding a new URL to the WebSphere Commerce Suite cache

2. The next step is to define the keys for the command. These are used by the cache manager to determine if the page is unique or not. If these are not configured correctly, then the user could be shown the wrong page. Figure 3-3, Figure 3-4 on page 37, and Figure 3-5 on page 37 show the steps involved.

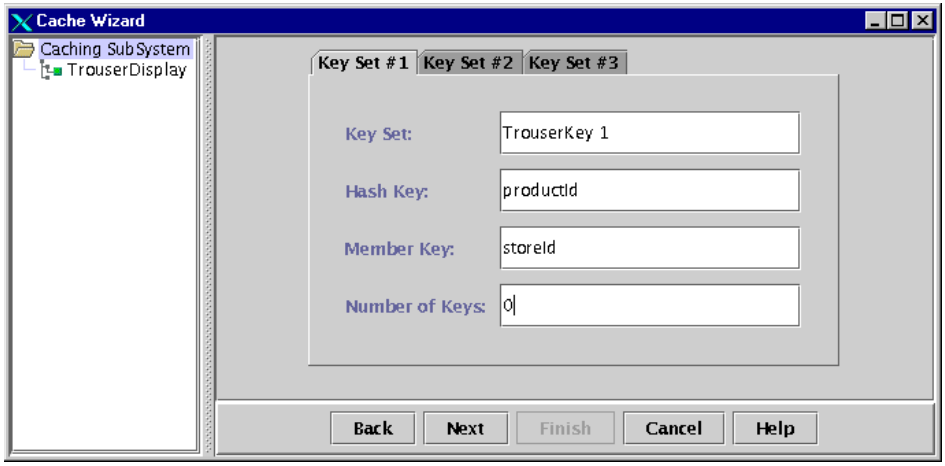


Figure 3-3 Assigning a value to Key Set #1

Clicking the Key Set #2 tab will give you the following dialog screen in Figure 3-4 on page 37.

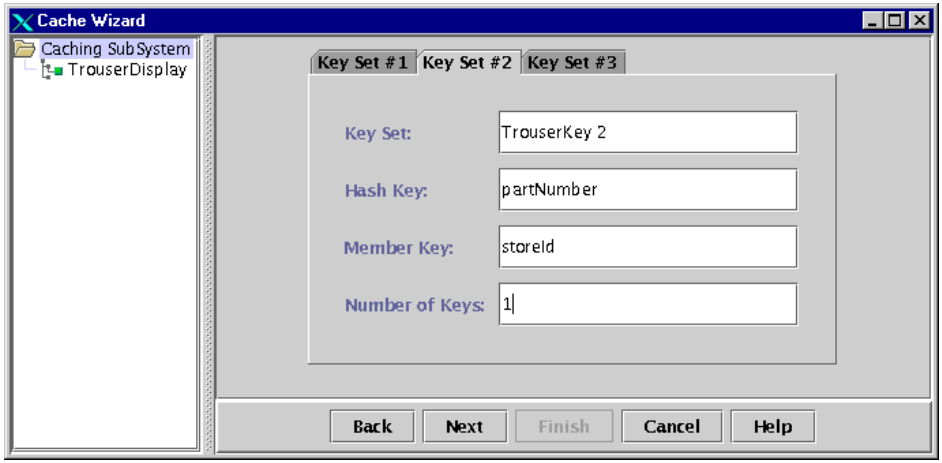


Figure 3-4 Assigning a value to Key Set #2

Finally, click the Key Set #3 tab and fill in the table as shown in Figure 3-5.

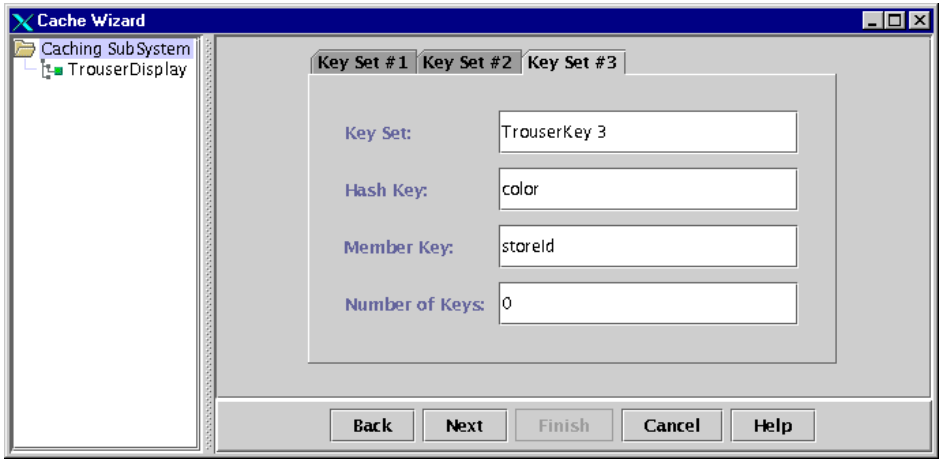


Figure 3-5 Assigning a value to Key Set #3

3. Once the parameters have been assigned, click the **Next** button. It is then necessary to define the memberId key for the partNumber parameter. Figure 3-6 on page 38 demonstrates this.

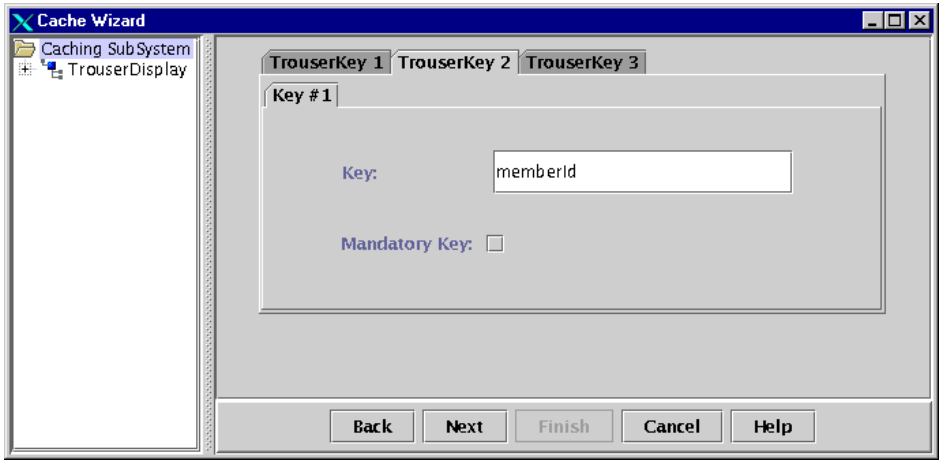


Figure 3-6 Assigning the memberId key

4. This completes the process. Click the **Finish** button. You should see a completion message.
5. Your new command should now be cached by the WebSphere Commerce Suite cache. In order for the change to become effective, you need to restart the HTTP server. You also need to restart the WCS server from the WebSphere Application Server (WAS) Administrative Console.

This data is stored in the *<instance_name>*.xml file in your instance directory. It is possible to modify this file manually, but we recommend that you use the WebSphere Commerce Suite configuration manager to make changes.

3.3.2 Checking that the cache is working with your new settings

After enabling cache for a custom command, verify whether caching works correctly. For this purpose you can use the cache trace facility provided in WebSphere Commerce Suite. This enables you to view debugging information generated by the caching system. You can use this to check whether your custom commands are being cached.

AIX systems only

To enable the cache trace, you must run the following commands:

1. Stop the HTTP server using:

   ```
   cd /usr/HTTPServer/bin
   ./apachectl stop
   ```

2. Set the WCS_CACHE_PLUGIN environment variable using:

   ```
   export WCS_CACHE_PLUGIN=/tmp/<your_filename.log>
   ```

3. Set read and write permissions on the log file using:

   chmod 777 /tmp/<your_filename.log>

4. Restart the HTTP server using:

   ./apachectl start

To stop the cache logging, stop the HTTP server and then restart it from a session where the environment variable has not been set.

Note: The cache writes to the log file each time the cache is used, which can cause the file to grow rapidly if left unchecked.

The output of the log should look something like Example 3-3. This example shows what you would expect to see if the TrouserDisplay command was being successfully cached.

Example 3-3 Output from cache trace file.

```
>nc_handle_request
iPPath /webapp/wcs/stores/servlet/TrouserDisplay
>WASInterface::handleRequest
>CachePluginControl::getUniqueInstance
<CachePluginControl::getUniqueInstance
>CachePluginControl::executeGetRequest
>CachePluginControl::isCmdCacheable
>WASInterface::getInstanceInfo
iDocumentRoot /usr/HTTPServer/htdocs/en_US
searching for /usr/HTTPServer/htdocs/en_US instance
<WASInterface::getInstanceInfo
iPPath /webapp/wcs/stores/servlet/TrouserDisplay
>WASInterface::getInstanceInfo
<WASInterface::getInstanceInfo
iPathInfo /TrouserDisplay
<CachePluginControl::isCmdCacheable
CachePluginControl - command is cacheable
iQueryString catalogId=1&storeId=1&langId=-1&color=black
iPathInfo /TrouserDisplay
>WASInterface::getInstanceInfo
<WASInterface::getInstanceInfo
>ConnectionPool::get
<ConnectionPool::get
CachePluginControl - Sending request message
CachePluginControl - Getting response message
Message - Read 931 bytes from server
Message - length of message is 931
<Message::Message(Connection)
```

```
>WASInterface::sendResponse<WASInterface::sendResponse
CachePluginControl - Returning connection to pool
CachePluginControl - CACHE HIT
<nc_handle_request
```

The output shows the checks the cache manager performs to verify that the command is cacheable, and if so, to check the cache and return the cached page. If you see **`CachePluginControl - CACHE HIT`** in the output, then your command has been served from the cache.

3.4 Optimizing cache performance

To optimize the performance of the cache, do the following after launching WCS:

- If Multi-lingual or multi-currency or member group-specific display pages are not going to be used, ensure that the field Multicurrency, multilingual, or command key dependent is not checked (Figure 3-7).

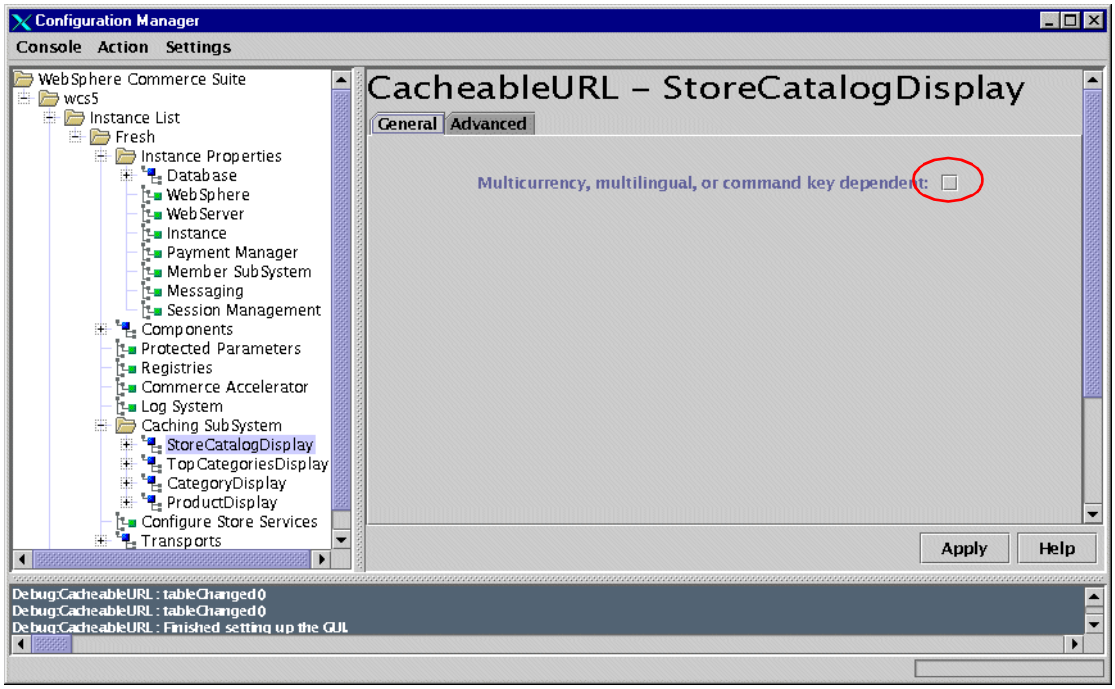


Figure 3-7 Deselecting Multicurrency, multilingual support of cache manager

- Ensure directories contain less than 1,000 files. This number has to be manually calculated by dividing Maximum objects per member by Cache directories per member. For example, if you expect one store to have 50,000 files, you will need 50 cache directories (dividing 50,000 by 1,000) to store less than 1,000 files in each directory. Set the value of the Cache directories per member field appropriately.
- If pages do not need to be automatically removed from the cache, ensure that the Auto Page Invalidation box is not checked.
- If any number of files can be in the cache, set the Max Objects Per Member field to 0. This allows unlimited number of objects in the cache.
- After making changes in the cache manager, be sure to stop and re-start the WCS instance from the WebSphere Application Server Administrative Console. Also, stop and re-start the Web server.

3.5 Setting up caching in 3-tier topology

In a 3-tier toplogy scenario, the web server is running on a machine separate from the WAS machine. To serve requests for cached data, WCS manages the cache daemon and cache client. The cache client can be run on either the WAS machine or the web server machine. Running the WCS cache client on the web server provides the advantages of serving requests for the cached files faster and consuming less system resource. The following instruction shows how to set up a Web server cache client on the web server machine.

Note: Setting up the Web server cache is only necessary if one or more of your cacheable Web addresses are session independent. If all of your cacheable Web addresses are session dependent, skip this section.

AIX only!
For NT, refer to documentation for NT

Do the following steps to set up the cache daemon so that requests are made directly from a Web server cache client on your Web server machine to a cache daemon running on your WebSphere Application Server machine.

1. Launch WCS Configuration Manager, and check whether the cache has been enabled as described in Section 3.1.1, “Enabling Cache” on page 30.
2. Change the Cleanup Agent Hostname to the remote Web server hostname (Figure 3-8 on page 42).

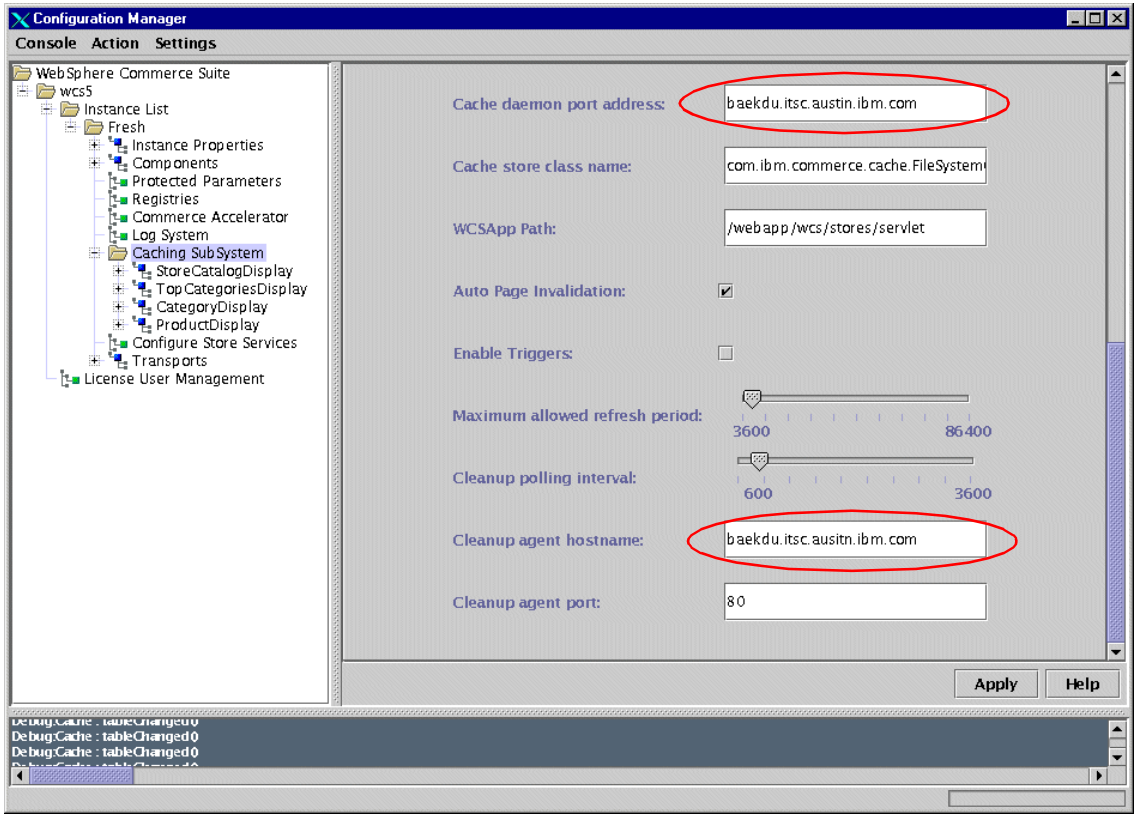


Figure 3-8 Adjusting cache manager for 3-tier configuration

3. Specify your WebSphere Application Server machine in the Cache daemon port address field.
4. Click **Advanced**. Deselect the check boxes under Session Dependent column for:
   - Store Catalog Display
   - Top Categories Display
   - Category Display
   - Product Display

   Click **Apply** (Figure 3-9 on page 43).

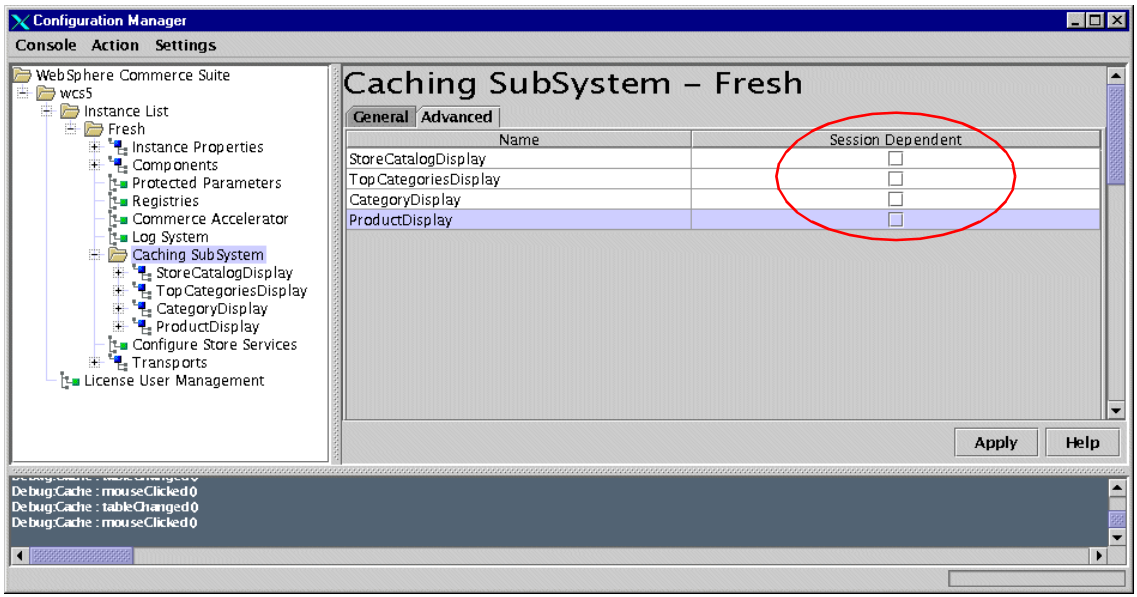


Figure 3-9 Disabling session-dependent cache

5. Copy the wcs_instances file in the <WCS_install_path>/instances directory of the WAS machine to the web server machine. You need to manually create a <WCS_install_path>/instances directory on the web server machine.
6. Copy the <instance_name>.xml file from the <WCS_install_path>/instances/<instance_name>/xml directory of the WAS machine to the same path on your remote web server machine. Note that you may need to create this directory on the web server machine.
7. Copy the following files from the<WCS_install_path>/bin directory on the WAS machine to the same directory on the web server machine. You may need to create this directory on the web server machine.
   - lib51cache.a
   - libicu-uc.a
   - libicudata.a
   - libxerces-c1_3.a
8. Create soft links from the /usr/lib directory to these files as follows:

```
# ln -s <WCS_install_path>/bin/lib51cache.a /usr/lib/lib51cache.a
# ln -s <WCS_install_path>/bin/libicu-uc.a /usr/lib/libicu-uc.a
# ln -s <WCS_install_path>/bin/libicudata.a /usr/lib/libicudata.a
# ln -s <WCS_install_path>/bin/libxerces-c1_3.a /usr/lib/libxerces-c1_3.a
```

9. Open the bootstrap.properties file in the <WAS_install_path>/properties directory on the web server machine and add the following to the end of the file:

   cache.lib=*<WCS_install_path>*/bin/lib51cache.a

   Note: In this instance, you may ignore the note at the top of the bootstrap.properties file saying that all properties in the file must start with “server.”or “ose.”

10. By default, WebSphere Application Server communicates with the Web server over port 80 and port 16999. Thus, if you have a firewall installed between these machines, you need to ensure that communication using these ports is open between WebSphere Application Server and the Web server. Cache files are stored by WebSphere Application Server in the directory or directories specified in CacheFilePath on the WAS machine.

For additional related information on caching, see the Section “Maintain the cache” in the Commerce Suite 5.1 online help.

3.6 Job scheduler

The job scheduler is a component of a WebSphere Commerce Server primarily used to schedule and launch jobs based on a timing scheme. Each scheduled job runs as a separate thread, and multiple jobs can be scheduled to run simultaneously. A job is a Commerce Suite command scheduled to run at a specified time or interval. Timing is specified in the **AddJob** command’s start and interval parameters for job execution. Job tracking information, including the job start time, end time, and results, are maintained in the database.

It is better for performance to use job scheduler on limited basis. To turn off job scheduler, change *<WCS_install_path>*/instances/*<instance_name>*/xml/*<instance_name>*.xml as follows.

```
<component compClassName="com.ibm.commerce.scheduler.SchedulerComm"
enable="false">
```

Job scheduler can be used in the following cases.

- When using IBM Payment Manager
- When running an auction

- When the site administrator is making changes to the system under a cloned environment

In other cases use of job scheduler is not recommended.

The **CleanJob** command removes jobs from the WebSphere Commerce Suite job scheduler status table based on time stamp or job reference number. Under heavy scheduler use, the scheduler status table grows tremendously large, so you can use this command to trim its size. The following example cleans all jobs that are scheduled to be completed before a given time by deleting specified entries from the job scheduler status table.

```
http://<myhostname>/webapp/wcs/stores/servlet/CleanJobendTime=2001:10:0
5:15:29:06&URL=basemall.jsp
```

You can add the auto clean job to the scheduler, which cleans up jobs from the job scheduler status table based on time stamp or job reference number.

For more information on job scheduler commands, refer to WCS on-line help or the *IBM WebSphere CommerceSuite Site Administrator Guide*.

4

Database tuning

The heart of an enterprise application is encapsulated in the business information stored within the enterprises database servers. In general, the speed at which the process of data storage and retrieval occurs invariably impacts the overall performance of the enterprise application.

In this chapter we will discuss a variety of recommendations designed to improve the performance of a database subsystem, with specific focus on enhancements to IBM WebSphere CommerceSuite Version 5.1 running against IBM DB2 UDB Version 7.1.

The topics discussed in this chapter are:

- Distribution of WebSphere database components in order to avoid issues of resource contention
- The effect of disk striping in the I/O performance of the database
- How changes in the database manager and database configurations can be used to improve performance
- The use of database maintenance utilities to improve database access times
- Use of database tools for determining the processing expense of application SQL queries against the database

4.1 WebSphere Database Distribution

WebSphere Commerce Suite maintains several databases; both WAS and WCS have their own databases, and if session information needs to be persisted to database, another database is added to the scheme. Two big questions can be raised here. First, should all the databases reside on a single machine or should they be on separate machines? Second, what are the tuning points unique to each database? In this section, we will study the characteristics of each database required by WCS and the corresponding tuning techniques.

4.1.1 WAS Administration Database

WebSphere Application Server maintains configuration data in a persistent store whose contents are loaded into memory when WebSphere Application Server is initially invoked, and when updates are made to the configuration data by an administrator.

The database can be located either on the same host where WebSphere Application Server is currently executing, or on a remote dedicated database server. In most cases, we recommend you install the WAS Administrative database on the same machine where the WebSphere Commerce Suite database has been installed. This makes it easier to maintain and backup processing tasks for site administrators, and makes the server fail-over feature possible.

The WebSphere Application Server Administrative database does not participate in the storage of enterprise data, and therefore does not experience the same volume of intensive database query activity as seen with the WebSphere Commerce Suite database. The only caveat to this statement is the extent of event logging configured by an administrator, as discussed in Section “Serious Event reporting” on page 116. The level of additionally administrative transaction queries against the database will increase significantly relative to trace level settings, which range from normal to full debug.

The majority of the techniques used to tune the WebSphere Application Server Administrative database are not specifically pertinent to tuning DB2, but rather reflect configuration changes within the WebSphere Application Server itself such as DataSource, connection pool, creation of models and clones, adding virtual hosts, and so on.

To use the clustering feature of WebSphere Application Server, the administrator has to enable multiple clones to share a common administrative repository database. This allows access to the same application configuration information for all application server clones participating in the cluster. Therefore, if horizontal cloning is used as a scaling option with clones distributed across a number of physical hosts, you need to relocate the administrative server to a centralized database server.

The distribution of the WebSphere Application Server Administration database also becomes an important consideration if CPU resource contention on the WebSphere Application Server host becomes an issue.

Creating a Remote WAS Administration Database

When creating a remotely distributed WebSphere Application Server Administration database, the list of information in Table 4-1 should be collected prior to the start of installation process.

Only on AIX. For NT, see DB2 for NT documentation

Table 4-1 Database client/server configuration checkpoints

Collected Data	Collection Procedure
DB2 server hostname	Execute **hostname** on the DB2 server
TCP service port for remote DB2 clients to connect to a DB2 instance on the server	Use a text editor to open the /etc/services file on the DB2 server and look for the entry of the form: `db2cdb2inst1 50000/tcp # Connection port for DB2 instance db2inst1` where port 5000 is the default DB2 installation value
DB2 Instance owner user id that is needed to log on to the server and connect to the database instance.	Provided by DBA or site administrator
DB2 Instance owner password is the password that the database owner needs to log on to the server and connect to the instance.	Provided by DBA or site administrator

On a server hosting a WebSphere Application Server installation, follow the installation guide for installing DB2 client services, IBM HTTP Server (IHS), and the WebSphere Application Server application with the applicable FixPaks and efixes.

Important: If installing IHS as your websever, ensure that the directives for SSL are enabled (as discussed in the Chapter 22: Enabling SSL for Production with HTTP Server, in the AIX version of the WCS Version 5.1 installation guide) **prior** to installation of WebSphere Application Server.

This avoids various manual updates of the IHS configuration after installation of WebSphere Application Server.

In the creation of a remote WAS administration database, the DB2 client application records the identity of the remote server and the service port required to connect to the DB2 instance running on a remote machine. This is done by creating a catalog entry on the client machine with the following command:

```
db2 catalog tcpip node <node name> remote <hostname> server <port value>
```

The client must then attach itself to the remote server by the host alias node name assigned in the above command, by executing the command:

```
db2 attach to <node name> user <db2 instance userid> using <db2 instance
password>
```

Once the client has successfully attached to the remote server, the WAS administration database is created by executing the following commands on the DB2 client workstation:

```
db2 create database was user <db2 instance userid> using <db2 instance
password>

db2 update db cfg for was using applheapsz 256

db2stop
```

```
db2start
```

Important:

Ensure the database has been created successfully by connecting to the database from the client by issuing the following command, and ensure the response looks similar to the following. Be sure to enter the db2 instance owner id and its password.

```
$ db2 connect to was using <db2_instatnce_owner_id> using
<db2_instance_owner_password>

Database Connection Information


 Database server = DB2/6000 7.1.0

 SQL authorization ID   = DB2INST1

 Local database alias   = WAS
```

The user ID and password for the database are also stored in the admin.config file with the settings shown in Example 4-1.

Example 4-1 Setting database user and password

```
com.ibm.ejs.sm.adminServer.dbUser=db2inst1
com.ibm.ejs.sm.adminServer.dbPassword=wcs5test
```

If you want to move the administrative database to a new database, simply create the new database, change the install.initial.config to true (so the tables will be created), and change the URL of the database in the admin.config file to point to the new database. This is especially useful in test environments where you can try different configurations by switching the administrative repository. The value for dbUrl should match the database alias name defined on the database server machine. The database alias name can be found by:

```
$ db list db directory
Database 1 entry:
Database alias          = WAS_REMOTE
Database name           = WAS_REMOTE
Node name               = ITSODB
Database release level  = 9.00
Comment                 =
Directory entry type    = Remote
```

```
Catalog node number     = -1
```

Example 4-2 shows how to do this in admin.config file.

Example 4-2 Setting WAS admin database name

```
com.ibm.ejs.sm.adminServer.dbUrl=jdbc:db2:was_remote
```

Tuning serious event reporting interval

The WAS Administrative database stores event logs as well as configurations. The same event log appears in the bottom pane of the Administrative Console. Event information is also stored in standard output files such as default_server_stdout.log and default_server_stderr.log under the <was_root>/logs directory.

The Serious Event listener is a lightweight background thread that runs every 10 seconds by default, polling the administrative database for changes in the configuration or runtime state. The listener executes select statements and stores them in the administrative database. By default, a database select is issued to the administrative database for each type of event (fatal, warning audit). Any events returned as a result are reported in the Console Messages section of the Administration Client. While it is not a significant use of resources, the Serious Event listener thread can be tuned to execute at a desired time interval. For details on how to adjust this parameter, refer to Section “Serious Event reporting” on page 116.

4.1.2 WAS persistent session management database

WebSphere Commerce Suite has the option of persisting user ID or object information by use of its own cookies, or alternatively using the session persistence services provided by WebSphere Application Server.

WebSphere Application Server persistent objects are serializable and are written to a dedicated database located on the same machine where WebSphere Application Server is running, or on a dedicated remote database server.

As discussed in Section 4.1.1, “WAS Administration Database” on page 48, the number of queries executed against the WebSphere Application Server session database is small relative to the number of queries against the WebSphere Commerce Suite database. This is because database access is only necessary if the WebSphere Application Server receives a request that has an associated session identifier not already held in memory.

The requirement for session persistence in the deployment of application server clustering dictates the deployment to a centralized database server. This is required because all application servers must have access to the same state information relative to servicing user transaction requests entering the domain. This subsequently facilitates the ability of all application server clones to actively participate in the cluster.

In light of the above observations, the primary tuning area for the WebSphere Application Server session database is the configuration of the DataSource and database connection pools as illustrated in Section 5.1.2, “Queue settings in WebSphere” on page 79.

In creating a linkage to the database to store session persistence information, the persistence type should be set to **directtodb**. This option instructs WebSphere to use container managed database access methods (provided by the WebSphere Web Container), as opposed to database access methods from a non-IBM custom EJB. This option utilizes the techniques developed to optimize the performance WebSphere Application Server’s database access methods.

Figure 4-1 on page 54 shows the console screen for enabling persistent sessions.

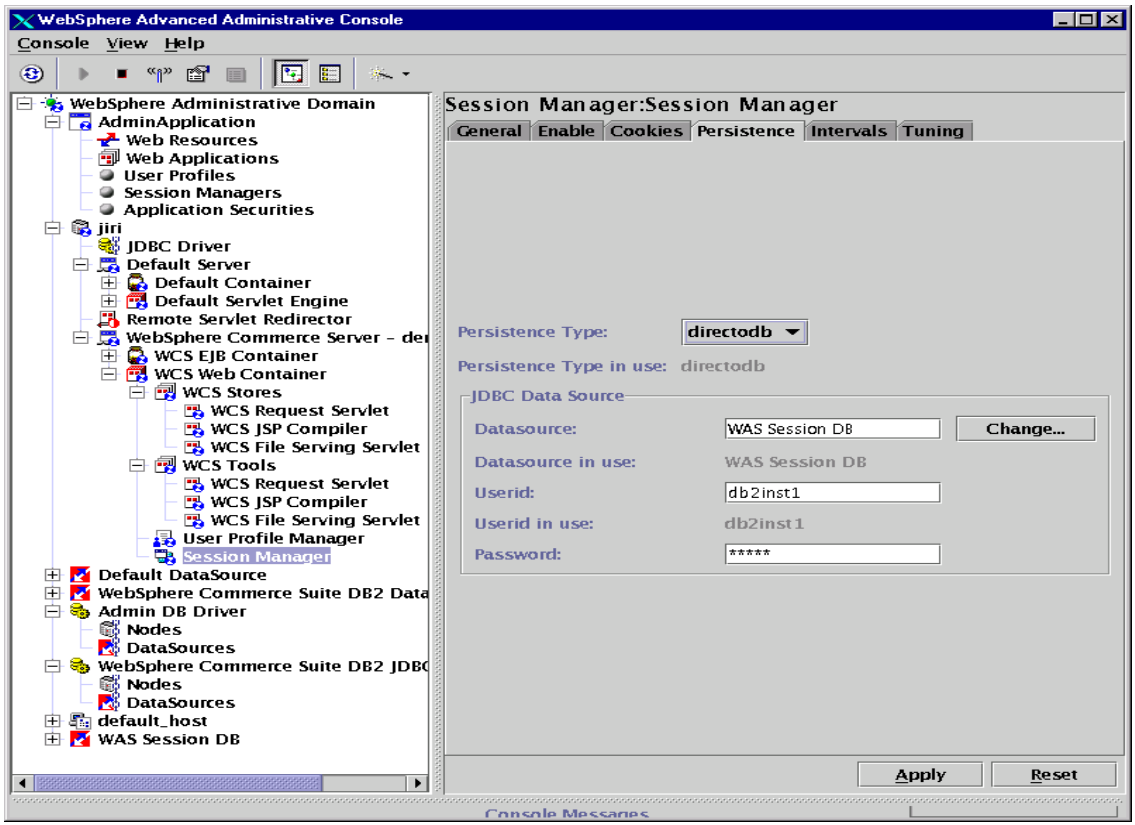


Figure 4-1 Enabling container managed session persistence

4.1.3 WebSphere Commerce Suite Database

The WebSphere Commerce Suite database encapsulates both the enterprise model data and the commerce application command set used in processing the information stored. Because the bulk of transaction activities occurring within a WebSphere Commerce Suite will be against this particular database, appropriate consideration is required in deciding where and how the database will be deployed.

WebSphere Commerce Suite provides the option of using a database located on the same host as the WebSphere Commerce Suite engine, or on a dedicated remote server. Multi-tier topology with a dedicated database server machine is usually preferred because it provides better scalability.

In terms of how the database is deployed, the important criteria to note is that WebSphere Commerce Suite is an online transaction processing (OLTP) application (that is, a high percentage of small transaction queries as opposed to a DSS application executing a number of long running transaction queries), and therefore tuning recommendations for a DB2 database deployment are based on that characteristic.

4.2 Planning for database layout

The WebSphere Commerce Suite database server requires sufficient raw processing power to handle many concurrent transaction requests and responses for a variety of workloads in a timely manner.

The following minimum hardware recommendations for a database server are presented as base reference points for medium to large site installations supporting WebSphere Commerce Suite sites:

- Processor - 4-way, 500 Mhz
- Memory - 512 MB per processor
- Disk - 4-10 drives per processor
- A separate set of dedicated (mirrored) disks (which should be as fast as possible) for storing database logs

As indicated in the above list, a minimum of a 4-way processor configuration that is at least 500 MHz per processor is highly recommended. Testing of single and two tier configurations with smaller processor speeds has resulted in very high CPU utilization rates (greater than 80%) in a number of internal test scenarios.

A high degree of contention for processor cycles amongst DB2 and other running processes on the server invariably impacts the ability of DB2 to turn around transaction requests quickly. This ultimately degrades the perceived response time performance by shoppers to the web site(s) supported by the WebSphere Commerce Suite database server.

Attention: The addition of more processors should be implemented with respect to determining the current CPU utilization (see Section “CPU tuning” on page 157 for CPU monitoring tools), ensuring that additional CPU resources will not be underutilized based on current resource consumption.

The memory requirement for a WebSphere Commerce Suite database server is relative to the current size and estimates on the growth rate of the enterprise data. As an initial recommendation, a minimum of 512 MB per processor should be allocated, with the value scaled up accordingly relative to the database sizing for buffer pool allocation (see Section 4.5, “Adjusting database bufferpool size” on page 63).

DB2 uses allocated areas of memory, termed a bufferpool, as storage for a portion or all of the contents of a database table in memory. If a request for row data is received that cannot be found in the bufferpool, the requested information is read from disk into memory as a data block at a pre-determined size defined for the table space.

A container is a physical storage entity that can either be associated with a collection of files and directories (that is, System Managed Space (SMS)), or be composed of files or raw devices under the control of DB2 (that is, Database Managed System (DMS)) (see Figure 4-2). The handling of I/O for table space requests is based on the type of container the database tables are stored in.

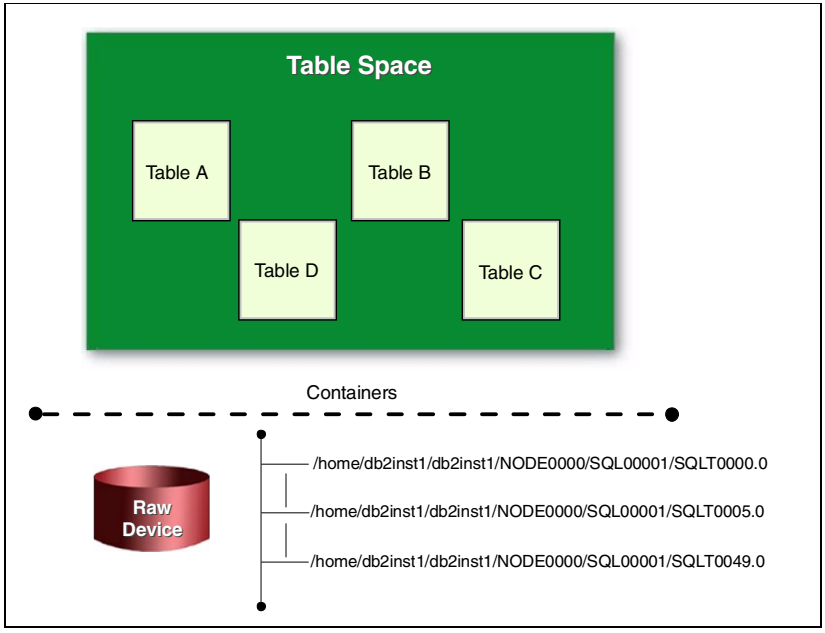


Figure 4-2 Table spaces, tables, and containers

When DB2 creates a database instance using '**db2 create database <dbname>**' command, a default set of SMS table spaces are created as follows:

- **SYSCATSPACE**: Holds information on all system catalog tables that provide details on tables and indexes created in addition to tables statistics, etc.
- **TEMPSPACE1**: Used as a 'scratch pad' area for temporary tables created during the execution of queries and may involve joins of different tables in creating the result set to be sent back in the query response.
- **USERSPACE1**: Contains the user application tables, which in the case of WebSphere Commerce Suite is where both the Commerce engine specific and user custom tables reside.

Using SMS tablespaces makes database administration work easier, as table space creation is far easier in SMS configuration than in DMS configuration. Furthermore, allocation of additional container spaces is also more easily handled by the file system.The downside is that all direct access to tables are handle by the file system, introducing an additional overhead to DB2 read/write requests.

The overhead is removed using DMS table spaces that use raw devices as opposed to file system based containers (see the following note). DB2 directly handles I/O with the raw devices, which can realize significant performance gains relative to a similarly configured SMS container-based system for large volume of transactions.

Internal testing on a 2-tiered system for small workload sizes (i.e. approximately 750 transactions/min), the performance for SMS and DMS configuration was comparable in terms of throughput. However, as workload increase, having better I/O distribution and faster access to data from DMS supported tablespaces will allow greater performance gains to be realized.

Note: File based DMS containers have file system caching controlled by the operating system and not DB2. This incurs a performance penalty relative to the same container composed of raw devices where caching is handled by DB2 directly.

It should be pointed out that SMS provides easier manageability because it allows you to use available Operating System tools for container management. The decision point relative to performance against manageability is left up to the reader to decide.

4.2.1 Recommendations for tablespace layout

The following recommendations for a tablespace layout of a WCS database are based on performance recommendations from *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012, and information gathered from IBM's engagement experience in WCS projects. By default, all the WCS tables are created in USER tablespace and we will take the default in the following example. If you want to change the location where the WCS tables will be placed, edit the script in Example 4-3 on page 60.

- Leave the SYSCATSPACE and TEMPSPACE1 as SMS tablespaces.
- Create TEMPSPACE1 container(s) on separate dedicated physical drives.
- Layout USERSPACE1 across as many DMS raw containers as are available (i.e. 4-10 physical drives per CPU), with one container per raw device.
- Database logging is extremely write intensive and database logs should reside on a separate set of dedicated disks that should be the fastest disks available instead of residing on the same disk as the db2 instance. This reduces I/O contention with other DB2 processes and improves the overall spread of I/O traffic within the system.
- The database logs should also be mirrored on separate dedicated disks to ensure the database is able to recover from system failure or corruption of the primary database logs.

The following section provides a practical look on the creation of a database based a number of the aforementioned recommendations. The steps below shows how to stripe the tablespaces over multiple disk drives.

XYZ Corporation have purchased an IBM RS/6000 pSeries 680 as a database server, with 4 x 600Mhz processors, 4GB of memory, and a 16 x 9.1 GB disk array as part of their infrastructure deployment of a WebSphere Commerce Suite site.

The assignment of physical volumes (PV's) and volume groups (VG's) and logical volumes (LV's) proposed are shown in Table 4-2.

Table 4-2 Design of physical layout

Physical Volume	Volume Group	Logical Volume	Tablespace	Type	Contents
/dev/hdisk0	rootvg		N/A		Swap space
/dev/hdisk1	rootvg	db2instlv	SYSCATSPACE1	SMS	- '/home' directory for db2 instance - '/xyzsys' directory to store SYSCATSPACE1

Physical Volume	Volume Group	Logical Volume	Tablespace	Type	Contents
/dev/hdisk2 /dev/hdisk3	rootvg	db2loglv	N/A		- '/xyzlog' directory to store logfiles
/dev/hdisk4 /dev/hdisk5	rootvg	db2templv	TEMPSPACE1	SMS	- '/xyztemp' directory to store TEMPSPACE1
/dev/hdisk6 ~ /dev/hdisk15	datavg	db2data(x)lv x = 6 ~ 15	USERSPACE1	DMS	XYZ Corporate WCS Commerce data

The following instruction gives an example of the procedure required to create logical volumes on an AIX system that correspond with the physical layout plan of the database in Table 4-2.

The following steps are applicable only to AIX.

- Login as **root** user.
- Add hdisk1 to hdisk5 to rootvg volume group:

```
extendvg -f roovg hdisk1 hdisk2 hdisk3 hdisk4 hdisk5
```

- Create the logical volume for the database instance on hdisk1:

```
mklv -L 'db2instlv' -S4K -t 'jfs' rootvg 1 hdisk1
```

- Create a filesystem to hold the database on the logical volume. Allocate 9GB as the initial size of filesystem. Note 9GB = 17578125 (512 byte-blocks).

```
crfs -d 'db2instlv' -m '/xyzsys' -a size=2250000 -g 'rootvg'
```

- Create the logical volume for the database logs on hdisk2 and hdisk3:

```
mklv -a e -u 2 -L 'db2loglv' -S4K -t 'jfs' rootvg 1 hdisk2 hdisk3
```

Important: The logical volume for db2 logging has been created with physical partitions on the outer edge of the physical disk drives with '-a e' parameter, as DB2 logging is both sequential and write intensive, and this area provides the highest track density for this type of data transfer.

- Create a filesystem that will hold the database logs on the logical volume, allocating 18 GB as the initial size of filesystem. Note 2 x 9.1 GB = 35546875 (512 byte-blocks):

```
crfs -d 'db2loglv' -m '/xyzlog' -a size=35546875 -g 'rootvg'
```

- Create the logical volume for the db2 TEMPSPACE1 tablespace on hdisk4 and hdisk5:

```
mklv -L 'db2templv' -t 'jfs' u 2 -S4K rootvg 1 hdisk4 hdisk5
```

- Create a filesystem to hold TEMPSPACE1 on the logical volume, allocating 18GB as the initial size of filesystem:

  crfs -d ‘db2templv’ -m ‘/xyztemp’ -a size=35546875 -g ‘rootvg’

- Before the 10 ‘unstructured’ logical volumes for XYZ Corp. data on hdisk6 to hdisk15 can be created, a new volume group, which will be called ‘db2datavg’, needs to be created:

  **mkvg -d 10 -y ‘db2datavg’ hdisk6 hdisk7 hisk8 hdisk9 hdisk10 **
  hdisk11 hdisk12 hdisk13 hdisk14 hdisk15

- The command for creation of logical volumes for each disk containing original data will be of the form:

  mklv -L <LV name> -t <LV type> datavg

  For example, **mklv -L ‘db2data8lv’ -t ‘raw’ db2datavg**

- Change ownership for the logical volumes to the db2 instance owner; that is, user id ‘db2inst1’, group id ‘db2iadm1’:

  chown -r db2inst1.db2iadm1 /dev/db2*lv

  chown -r db2inst1.db2iadm1 /dev/rdb2*lv

- Install DB2. /home/db2inst1 as the database home directory.
- Login as the db2 instance owner and run the Korn shell script shown in Example 4-3 to create the following tablespaces:
  - SYSCATSPACE on ‘/xyzsys’
  - TEMPSPACE1 on ‘/xyztemp’
  - DB2 logging on ‘/xyzlog’
  - USERSPACE1 on the raw logical volumes rdata6lv - rdata15lv

Example 4-3 Sample shell script to create database

```
#! /bin/ksh

        database="XYZDB"

        echo "Creating db ${database}..."
        db2 "create database ${database} \
            catalog tablespace managed by system using (‘/xyzsys’) \
            temporary tablespace managed by system using (‘/xyztemp’) \
            user tablespace managed by database using \
                (DEVICE '/dev/rdb2data6lv' 2250000, \
                 DEVICE '/dev/rdb2data7lv' 2250000, \
                 DEVICE '/dev/rdb2data8lv ' 2250000, \
                 DEVICE '/dev/rdb2data9lv' 2250000, \
                 DEVICE '/dev/rdb2data10lv' 2250000, \
```

```
        DEVICE '/dev/rdb2data11lv' 2250000, \
        DEVICE '/dev/rdb2data12lv' 2250000, \
        DEVICE '/dev/rdb2data13lv' 2250000, \
        DEVICE '/dev/rdb2data14lv' 2250000, \
        DEVICE '/dev/rdb2data15lv' 2250000)"
echo "Database ${database} tablespaces created successfully"
echo
echo "Changing the DB2 logging path"
db2 "update db cfg for ${database} using NEWLOGPATH '/xyzlog'"
echo "Disconnecting from database ${database}"
db2 "disconnect ${database}"
db2stop
db2start
echo "Done ...
```

4.3 Improving performance by data striping

The distribution of data across a collection of disks greatly reduces the possibility of I/O bottleneck occurring on a given disk. A high volume of I/O requests for data on a specific disk on a non-distributed system will invariably lower the overall performance of the database server.

Data striping can be implemented at the operating system level, hardware level using RAID technology, or by manually distributing the data across disks. As discussed in "Recommendations for tablespace layout" on page 58, DMS tablespaces using raw devices as containers provide a performance gain relative to similar sized SMS tablespaces, as disk I/O is handled directly by DB2 and not the file system.

DB2 can stripe data across a set of containers for a DMS tablespace and provide 'off-by-N' mirroring scheme (see note below), using the AIX's LVM.

Note: The ‘off-by-N’ mirroring scheme locates the mirrored copy of the original data for a disk on a portion of another data disk in the array. Each disk in the array will have its own mirrored copy on another disk, ensuring that all original versions of data have a mirrored copy residing on another disk in the array.

The N factor is the offset value for disk location of the mirrored copy relative to the disk where the original data resides. As an example, if N = 2, then for a 4 disk array the arrangement would be as follows:

- disk1: Data + Mirror for disk3
- disk2: Data + Mirror copy for disk4
- disk3: Data + Mirror copy for disk1
- disk4: Data + Mirror copy for disk2

Deployment of ‘off-by-N’ mirroring requires use of map files to specify the exact mapping of physical volumes areas to logical volumes, in addition to the **`mklvcopy`** command for placement of mirrored copies.

‘off-by-N’ mirroring provides a less expensive alternative to having a separate dedicated disk for each mirrored copy.

What is Map File?
Ascii text file that allows the specification of exactly which PV and PP’s are mapped to an LV.

In addition, implementing RAID on top of a DB2 database disk array introduces the following drawbacks:

- RAID takes away the database server’s ability to parallelize its I/O across multiple containers, resulting in poorer I/O performance.
- DB2 uses direct I/O for writes and RAID creates another layer, which increases the processing time required for each write.
- Having access to multiple containers provides the DB2 with the option of reading from the fastest path.

4.4 Separate tablespace for indexes

Under heavy workload, it is often preferable to assign a separate tablespace to indexes. One of the main reasons for separating your indexes from the data by placing them in a separate table space is that you may wish to place them on faster devices or allocate them their own buffer pool. Note that using DMS does not separate data and indexes into different table spaces unless you do so specifically to assign them to different buffer pools. If you need to ensure that

certain indexes always remain in memory, then that would be a valid reason to move the indexes to their own table space and assign that table space its own buffer pool. Make sure you have sufficient disk I/O bandwidth to cope with this setup.

4.5 Adjusting database bufferpool size

The database bufferpool provides memory space for holding a portion of the database in memory. It allows faster data access to the same data obtained from disk because it does not require I/O to physical disks.

However, allocating a sizeable portion of memory to the database bufferpool has to be weighed against the amount of memory available for both database and other processes running on the server. If insufficient memory is available for executing processes, the operating system has to compensate by swapping the required pages to and from disk, with the I/O involved offsetting any performance gain by the increased bufferpool size.

You can check whether the current bufferpool size is adequate by measuring the 'bufferpool hit ratio', which is the ratio of accesses to the bufferpool relative to accesses to data on disks. For OLTP applications such as WebSphere Commerce Suite, this percentage should be fairly high - in the region of 95 to 99 per cent.

How is the bufferpool hit ratio determined? The following steps describe the procedure needed for extracting this value for a database system:

- Obtain snapshots of buffer pool activity as transactions are executed against the database. Example 4-4 is a Korn shell script that could be used in testing by modifying the highlighted entries as necessary (ensure that the db2 instance owner has permissions to execute this script):

Valid only for AIX!

Example 4-4 Shell Script for Capturing Bufferpool Snapshot Data

```
#!/bin/ksh
#-------------------------------------
# db2 snapshot monitor script - 04/10/01
#-------------------------------------

# Change to the name of your database - this example uses 'mall'
database="mall"

# Change total monitoring time and snapshot intervals to
accommodate your test
# requirements -
#
```

```
# This example tests every 5 secs for 15 minutes of \
transactionactivity
# 15 x 60 = 900
typeset -i snapshot_count=0
typeset -i snapshot_interval=0

while (( $snapshot_interval < 900 ))
do
   echo "Connecting to the ${database} database"
   db2 connect to ${database}

   # Setup the database monitor switches
   echo "Creating the database monitor switches"
   db2 "update monitor switches using bufferpool on"

   # Now collect the database snapshot and put into timestamp \
file
   echo "Collecting snaphot data"
   echo
   db2 "get snapshot for all on ${database}" > snapshot-`date | \
awk '{print $4}'`

   # turn off database  monitoring
   echo "Switching off monitor switches"
   echo
   db2 "update monitor switches using bufferpool off"

   # Disconnecting from database
   echo "Disconnecting from the database"
   echo
   db2 disconnect ${database}

  ((snapshot_interval += 5))
 ((snapshot_count += 1))
  echo "Number of database snapshots taken = ${snapshot_count}"
 echo "Snaphot interval = ${snaphot_interval}"


# Wait for an interval of five seconds before collecting the next
snapshot
   echo
   echo "Sleeping for 5 seconds......"
   sleep 5

done
echo
echo "Done....."
exit 0
```

- If the above script is used, time-stamped data collection files are created for each 15 second interval. The script should be invoked as the db2 instance owner by the command **./dbsnapshot.sh**
- From the list of snapshot files, determine the values for both buffer pool logical and physical reads and writes by issuing the following command:

  ```
  grep “Buffer pool” snap* | grep read > buffer_pool.out
  ```

  with the generated output of the form shown in the example snapshot entry:

  snapshot-Tue Apr 10 10:49:20 CDT 2001:**Buffer pool data logical reads** = ...

  snapshot-Tue Apr 10 10:49:20 CDT 2001:**Buffer pool data physical reads** = ...

  snapshot-Tue Apr 10 10:49:20 CDT 2001:**Buffer pool index logical reads** = ...

  snapshot-Tue Apr 10 10:49:20 CDT 2001:**Buffer pool index physical reads** = ...

- From the listings in the buffer_pool.out file (see above highlighted entries), calculate the bufferpool hit ratio for a snapshot interval using the formula:

  1 - {(pool_data_physical_reads + pool_index_physical_reads)
  /(pool_data_logical_reads + pool_index_logical_reads)} * 100%

If the bufferpool hit ratio is less than 95%, then calculate the current size of the database using the following commands:

- **db2 list tablespaces show detail**

  note the Total Pages for SYSCATSPACE, let us say, Ns
- **db2 "select sum(npages) from syscat.tables"** = Nt
- **db2 "select sum(nleaf) from syscat.indexes"** = Ni

and calculate the database size as:

```
Total Size = ((Nt + Ni) + Ns) x 4096 / (1024)**2 = ... MB (approx).
```

Based on the memory available on the server and the total size of the database, the general rule of thumb is to allocate a *minimum* buffer pool size equal to the total database size.

Attention: The bufferpool allocation should allow for growth of the Commerce database and should be revised if new custom tables and indexes are added to the base schema.

Otherwise, size to the maximum amount of shared memory, keeping the following in mind:

- UNIX limits are on total shared segments, which include the utility heap (20MB), dbheap (20MB each), locklist (16MB), pckcache (26MB), sortheapthresh, and so on. Inspection of the database and database manager configurations will assist in determining the current values of these settings.
- The size of the shared memory segment size (shmmax) parameter may need to be increased if it is too low. Setting the segment size to 3 - 3.5GB provides a sufficiently generous ceiling level for the majority of installations.

4.6 Running reorg and runstats

DB2 has two commands for cleaning up and improving database performance, called **runstats** and **reorg**. **reorg** has an additional command called **reorgchk**.

We recommend that **runstats** be run regularly on all the database tables to optimize database access. The state of the tables should be monitored with **reorgchk** regularly. Actual reorganization of tables should not be performed unless it is perceived to be causing a performance problem. For more detailed information about these commands, see *DB2 UDB Command Reference*, SC01-2951.

While performing our tests, we were unable to record any significant performance gain between measurements taken before and after the changes. However, if we had a much larger database, running **reorg** would have helped performance to a larger degree.

4.6.1 runstats

runstats updates statistics about physical characteristics of a table and the associated indexes. These statistics are used by the database optimizer when calculating access paths to data. This should be performed on tables that have many updates, and tables that have just been reorganized. To run **runstats** on a table, use:

```
db2 runstats on table <schema.table name> with distribution and
detailed indexes all shrlevel change
```

This command calculates the most detailed statistics on the table and indexes, and allows applications to read and write to the table while the command is running.

We recommend that this command is used regularly on tables with a lot of updates, as well as tables after large data loads, such as tables associated with product data. When planning batch jobs to perform **runstats**, it is important to research the most commonly updated tables for your WebSphere Commerce Suite instance. From our testing, we produced Figure 4-3 on page 71, which shows the most active tables in our test database.

4.6.2 reorg

reorg is used to reorganize tables by eliminating fragmented data, and compacting data. You have the option of physically ordering the data in the table by a named index, or simply compacting the data without any reordering. **Reorg** requires a temporary space to store the data being reconstructed before it replaces the existing table. The default is to use the same tablespace as the table being reconstructed, or you can specify a specific tablespace for the temporary storage. This is especially important for large tables, as the command will fail if the temporary table fills the available space. Also, distributing the I/O over two or more physical devices will accelerate the process.

To find out if your tables need reorganizing, use the **reorgchk** command. **reorgchk** runs several tests against the database tables and their indexes. Tables failing to meet any one of the test criteria are candidates for reorganization. **reorgchk** produces a detailed report showing this information. To run **reorgchk** on all the user and system tables:

```
db2 reorgchk current statistics on table all
```

Alternatively, you can specify the *update* flag. This causes **runstats** to be run on the database tables before **reorgchk** determines if tables need reorganizing or not. To do this you must use this command:

```
db2 reorgchk update statistics on table all
```

The advantage of updating the statistics is that the database manager will have the most up-to-date information for calculating if tables need reorganizing.

Note: Running statistics on many tables can take time, especially if the tables are large.

Example 4-5 on page 68 is an example of part of the output from **reorgchk**. In this case, the results show that the CATENTDESC, CATGPENREL and SUBORDERS tables should be reorganized. This is indicated by the asterisks in the right hand column.

Example 4-5 Sample output of reorgchk

```
F1: 100 * OVERFLOW / CARD < 5
F2: 100 * TSIZE / ((FPAGES-1) * (TABLEPAGESIZE-76)) > 70
F3: 100 * NPAGES / FPAGES > 80

CREATOR   NAME              CARD    OV    NP    FP    TSIZE  F1  F2 F3 REORG
-------------------------------------------------------------------------------
DB2RT     ACCCUSTEXC           -     -     -     -        -  -   -  - ---
DB2RT     ADDRBOOK         15061     0   185   185   737989   0  99 100 ---
DB2RT     ADDRESS          25066    43  2441  2441  9424816   0  96 100 ---
DB2RT     CATENTDESC       11126  4586   655  1072  4405896  41 100  61 *-*
DB2RT     CATENTREL           53     0     2     2     5671   0 100 100 ---
DB2RT     CATENTRY         11064     0   460   460  1803432   0  97 100 ---
DB2RT     WTAXINFO             -     -     -     -        -  -   -  - ---
DB2RT     ZIPCODE              -     -     -     -        -  -   -  - ---
-------------------------------------------------------------------------------

Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) / (NLEAF * INDEXPAGESIZE) > 50
F6: (100-PCTFREE) * (INDEXPAGESIZE-96) / (ISIZE+12) ** (NLEVELS-2) *
(INDEXPAGESIZE-96) / (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) < 100

CREATOR  NAME              CARD  LEAF  LVLS ISIZE   KEYS   F4   F5  F6 REORG
-------------------------------------------------------------------------------
Table: DB2RT.ACCCMDGRP
DB2RT    P_ACCCMDGRP        881     3     2     4    881  100   86   34 ---
DB2RT    UI_ACCCMDGRP       881    22     2    86    881   84   91    4 ---
Table: DB2RT.CATENTDESC
DB2RT    P_CATENTRYDESC   11126    62     2    12  11126   99   87    1 ---
able: DB2RT.CATGPENREL
DB2RT    P_CATGPENREL     11063    99     3    24  11063    5   87  112 *-*
Table: DB2RT.CATGROUP
DB2RT    P_CATGROUP        1013     5     2     8   1013  100   79   22 ---
DB2RT    UI_CATGROUP       1013    13     2    38   1013   98   87    7 ---
Table: DB2RT.SUBORDERS
DB2RT    I_SUBORDERS1      5583    17     2    13   2205   48   85    6 *--
DB2RT    I_SUBORDERS2      5583    26     2     8   5583   98   83    4 ---
DB2RT    P_SUBORDERS       5583    25     2     8   5583   99   87    4 ---
-------------------------------------------------------------------------------
```

CLUSTERRATIO or normalized CLUSTERFACTOR (F4) will indicate REORG is necessary for indexes that are not in the same sequence as the base table. When multiple indexes are defined on a table, one or more indexes may be flagged as needing REORG. Specify the most important index for REORG sequencing.

To reorganize a table, use the following commands:

```
db2 reorg table <schema.table name> index <schema.index name> -use
<tablespace name>
```

When you have a table with multiple indexes defined, you should use the primary key index. Once you have reorganized a table, you should use **runstats** to update the table statistics.

Reorganizing tables is something that should need to be done infrequently. We recommend that you use **reorgchk** regularly to monitor your databases, and take action if database performance drops as a result of fragmentation. With WebSphere Commerce Suite, the most likely time you may have to use **reorg** is after you have used the database cleanup utility to delete a proportion of some tables. The more active a table is, the more likely it will become fragmented and disordered. Figure 4-3 on page 71 provides information on table usage.

4.7 Effect of the database cleanup utility

The database cleanup utility, or **dbclean**, is provided by WebSphere Commerce Suite as a flexible way of clearing old data from the database. Removing old and unused data from the database can improve the performance of the overall system, especially if there is a large amount of redundant data. Generally speaking, the more commands and transactions your site performs, the more often **dbclean** should be run.

Important: If dbclean is run infrequently, the dbclean operation may take a long time to complete and consume a large amount of system resources on the database server, causing disruption to your service.

4.7.1 Running dbclean

dbclean is a flexible and extensible tool for cleaning up your database. It allows you to clean up tables selectively, while maintaining referential integrity of the database. **dbclean** can clean in two ways:

- *Top down* deletes child table rows with a delete cascade.
- If a delete restriction is specified in the referential integrity, then you have to use the *bottom up* method. This is where the rows in the child tables are deleted first, followed by the parent table rows.

The top down method is faster to run. The bottom up method is used when the referential integrity is set to *on delete restrict.* This is usually used when the child data is relied on by more than one parent table.

dbclean comes with more than 20 preset cleanup options. These are aimed at the most commonly used tables. You can define others, either for other WebSphere Commerce Suite tables, or your own custom application tables.

You need to change user to the DB2 instance owner id to run the command. An example of running the **dbclean** application is:

```
dbclean.sh -table member -type guest_shopper -db mall -days 30
-loglevel 0
```

This command deletes non-registered shopper records that are not associated with any orders, and have not been updated for 30 days or more. The member table grows quickly, as a record is created for each guest shopper when they add the first item to their shopping basket, and when a user registers.

Note: Is is important to consider carefully which data to delete using **dbclean**. This can depend on how long you want to keep data for marketing or fulfilment purposes that are not directly related to the day to day running of your site.

4.7.2 Identifying most frequently accessed tables

WebSphere Commerce Suite uses some tables more than others. These tables are the most important to clean up. If there are excessive old rows, the extra data will reduce the speed of queries against that table. Figure 4-3 on page 71 is the results of collecting a DB2 snapshot of the WebSphere Commerce Suite database tables for a 20 minute load test. This test was performed with WebSphere Commerce Suite caching turned off, which gives us the most accurate view of the table activity.

This information can be used to help define the most efficient tablespace layout for your WebSphere Commerce Suite database. Refer to Section 4.2.1, “Recommendations for tablespace layout” on page 58 for information on how to define custom tablespaces.

Should you activate other functionalities of WebSphere Commerce Suite such as the auction component, this will cause more activity on the auction specific tables.

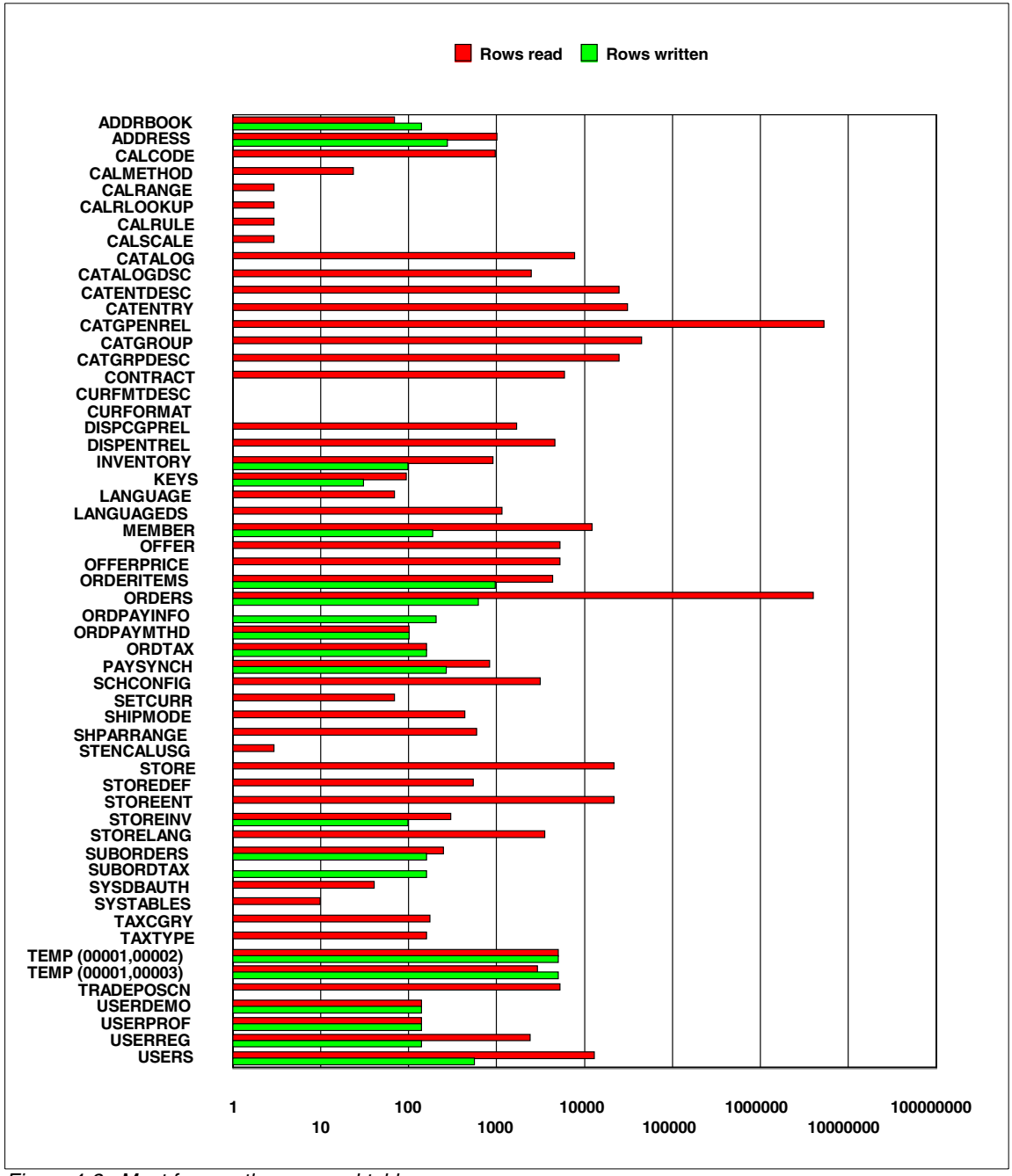


Figure 4-3 Most frequently accessed tables

4.8 Tuning other database parameters

This section covers some general database parameters to tune WebSphere Commerce Suite. All DB2 configuration parameters can apply to the WebSphere Commerce Suite's database, WebSphere Application Server's administrative database, and the persistent session database if used. Several of the parameters may not improve performance significantly unless changed in conjunction with others. This is especially important for large WebSphere Commerce Suite database configurations, where there are many concurrent applications connecting to the database from multiple application servers. Please note that if you tune the WebSphere Application Server queues to optimize throughput, you will need to increase some database parameters in order for the changes to be effective.

These parameters have a lower effect on the overall system performance than other tuning methods. This is because with WebSphere Commerce Suite 5.1, the application server consumes more system resources to run than the database server.

When choosing your own database parameters, it is good practice to load test your system while collecting database snapshot data. You can use the snapshots to look for bottlenecks, such as if the database is creating temporary tables that generate a lot of reads and writes. This implies that there is not enough sort space available for the database sort. By increasing the sortheap parameter, and then testing the database again, you can see if this has reduced or eliminated the temporary table creation.

Table 4-3 shows the parameters we worked with while tuning our application. Each change was applied one by one to see their incremental effects, and were changed in the order presented.

Table 4-3 Key parameters used for database tuning

Parameter	Default value	Tested value
APPLHEAPSZ	128 (4KB pages) changed to 256 during install	512
PCKCACHESZ	8 * MAXAPPLS. If the number is less than 32, then 32 will be the default value	640
MAXAPPLS	40	80
LOCKLIST	100 (4 KB)	200 (4 KB)
MAXLOCKS	10%	20%

Parameter	Default value	Tested value
MAXAGENTS	200	200

4.8.1 applheapsz

Application heap size defines the number of memory blocks available to the database manager for processing requests. To update the applheapsz parameter, use the command:

```
db2 update db cfg for <database name> using applheapsz <block amount>
```

The default value is 128, but the WebSphere Commerce Suite install guide tells you to set the value to 256.

4.8.2 pckcachesz

The package cache is a cache area allocated in database global memory (*applheapsz*). The cache stores database access plans in packages. This reduces the database manager overhead, as it does not have to access the system catalogs or recompile dynamic SQL. This is especially effective for queries that are used often. As WebSphere Commerce Suite performs a lot of repetitive queries, this parameter is important when tuning the application.

To change the package cache size parameter, use the command:

```
db2 update db cfg for <database name> using pckcachesz <cache size>
```

The cache size is allocated in 4K segments. By default, the value is set to -1. The database manager interprets -1 to be 8 times the maximum number of active applications (*maxappls*) parameter.

4.8.3 maxappls

This database parameter defines the maximum number of concurrent applications that can connect to a database. This includes both remote and local applications. When choosing this parameter, it is important to take the following into consideration:

a. The application data source maximum connection pool size

b. The session data source maximum connection pool size

c. The number of clone servers

d. Other database connections, for example from the DB2 command line

A rough guide for choosing the maxappls value is (a + b) *c + d. Please remember that if the DataSource Maximum connection pool size is greater than the value of maxappls, then the database will return SQL Error SQL1040N, and your application will fail. We recommend to increase maxappls value to 100 or larger.

To change the maxappls parameter, use the command:

```
db2 update db cfg for <database name> using maxapplsz <new value>
```

4.8.4 locklist

locklist defines the memory allocated to the lock list. Locking is the mechanism that the database manager uses to control concurrent access to data in the database by multiple applications. Both rows and tables can be locked. When the percentage of the lock list used by one application reaches *maxlocks*, the database manager will perform lock escalation, from row to table, for the locks held by that application. Locking tables reduces concurrency, and will reduce database performance for accesses to the affected tables. In addition, lock escalation occurs when the locklist is full. This could lead to deadlocking, as the applications are waiting on a limited number of table locks.

To change the locklist parameter, use the command:

```
db2 update db cfg for <database name> using locklist <new value>
```

As WebSphere Commerce Suite relies on several key tables, a heavily loaded system could generate many locks. Therefore, it is worth increasing the locklist and maxlocks values to avoid potential table locks or deadlocks. It is more likely that WebSphere Commerce Suite will fill the lock list, rather than an individual application generate enough locks to cause lock escalation.

4.8.5 maxlocks

The maxlocks parameter defines the *percentage* of the locklist held by an application. When the number of locks held by any one application reaches this value, the database manager will perform lock escalation for the locks held by that application.

To change the maxlocks parameter, use the command:

```
db2 update db cfg for <database name> using maxlocks <new value>
```

Tip: Monitoring Lock Escalation

Use the **snapshot** command to see if lock escalation has occurred. By default, locking related information is not monitored. There are two ways to turn on monitoring for locking information:

Set the DFT_MON_LOCK parameter to ON with the following command:

```
db2 update dbm cfg using DFT_MON_LOCK ON
```

Use the update monitor switches command:

```
db2 update monitor switches using LOCK on
```

To reset snapshot data, use the following command:

```
db2 reset monitor all
```

Use the following commands to connect to the database and take a snapshot of the lock related information:

```
db2 connect to <database name>
db2 get snapshot for locks on <database name>
```

4.8.6 maxagents

maxagents is a database manager parameter. It defines the maximum number of database agents that are available to handle application requests. This parameter can be useful in memory constrained environments to limit the total memory usage of the database manager, because each additional agent requires additional memory. The value of maxagents should be at least the sum of the values for maxappls in each database allowed to be accessed concurrently. Remember at least two or more databases are created by default in WCS. Be aware that each agent takes up some resource overhead.

To change the maxagents parameter, use the command:

```
db2 update dbm cfg using maxagents <new value>
```

We did not change this parameter for our tests, as we never reached a level of system load where more than 200 database connections, and therefore agents, were required. This parameter becomes more important when you have multiple application servers, which result in more connections to the database.

4.9 Reducing deadlocks

Changing the DB2 registry setting DB2_RR_TO_RS can significantly reduce deadlocks in your WebSphere Commerce Suite database. Enabling this parameter forces DB2 to convert all queries using RR (repeatable read) to RS (read stability). The parameters are two of four possible isolation levels. The advantage of using read stability is that it generates far fewer locks than repeatable read. At the same time, it ensures the rows required for the transaction remain stable by locking them. This improves concurrent database access, especially on systems where there is a large load on the database.

To change the DB2_RR_TO_RS setting, use the command:

```
db2set DB2_RR_TO_RS=YES
```

Once this command has been run, you need to stop and start the database manager for the changes to take effect.

Important: This setting may not be good for other, non WebSphere Commerce Suite applications. The DB2_RR_TO_RS parameter is granular only to the instance level. Should you have application databases that require repeatable read access to the database, you should use a different DB2 instance for the database.

5

Tuning WebSphere Application Server

In this chapter we introduce the concept of the WebSphere Queuing model and other performance tuning tips for WebSphere Application Server that you can follow to enhance the overall performance of the WebSphere Commerce Suite system. Topics in this chapter include:

- Adjusting queue sizes
- Tuning the Java Virtual Machine (JVM)
- Relaxing auto reloads
- Optimizing logging systems
- Using call-by-reference
- Tuning Enterprise JavaBeans (EJB) container cache
- Effect of using persistent session management vs. WebSphere Commerce Suite (WCS) sessions

5.1 Adjusting queue sizes

WebSphere Application Server maintains a pipeline of queues to manage WCS sessions. Those queues represent resources defined in the network, web server, servlet engine, EJB container, data source and possibly a connection manager to a custom backend system. Each of these WCS resources manages a queue of requests waiting for their requested resource(s). The WebSphere queues are load-dependent resources. The average service time of a request depends on the number of concurrent clients. The queuing concept is shown in Figure 5-1.

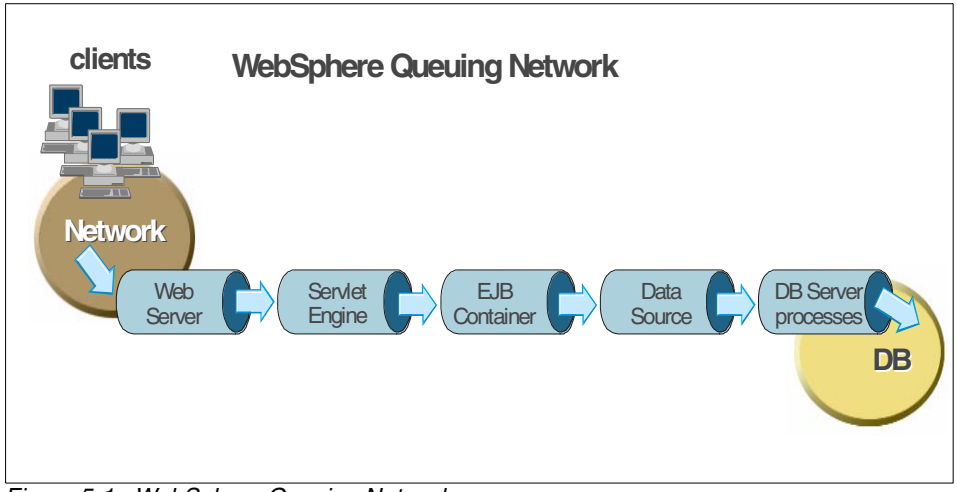


Figure 5-1 WebSphere Queuing Network

5.1.1 Closed queues versus open queues

Most of the queues comprising the WebSphere queueing network are *closed queues*. A closed queue places a limit on the maximum number of requests active in the queue. Conversely, an *open queue* places no such restrictions on the maximum number of requests active in a queue. A closed queue allows system resources to be tightly managed. For example, the WebSphere servlet engine's *Max Connections* setting controls the size of the servlet engine queue. Hence, closed queues allow system administrators to manage their applications more effectively and robustly.

In a closed queue, a request can be in one of two states: active or waiting. An active request is either doing work or is waiting for a response from a downstream queue. For example, an active request in the web server is either doing work (such as retrieving static HTML) or waiting for a request to complete in the servlet engine. In waiting state, the request is waiting to become active. The request will remain in waiting state until one of the active requests leaves the queue.

All web servers supported by WebSphere Application Server are closed queues. The WebSphere servlet engine and data source are also closed queues in that they allow you to specify the maximum concurrency at the resource. The EJB container inherits its queue behavior from built-in Java ORB. Hence, the EJB container, like the Java ORB, is an open queue. Given this fact, it is important to maintain the number of concurrent EJB calls by WebSphere Commerce Suite at an appropriate level. If enterprise beans are being called by servlets, the servlet engine will limit the number of total concurrent requests into an EJB container.

5.1.2 Queue settings in WebSphere

Because each of the server resources manages a queue of requests waiting to be processed, each of these queues has a way of adjusting the default settings, providing a way of fine tuning the performance of your WCS system.

In Web Server

MaxClients is one of the determining factors for WebSphere Commerce Suite queue. This parameter limits the total number of simultaneous HTTP requests that IHS can process. Chapter 6, “Tuning Web Server” on page 127 provides further information on changing this value and other IHS values. Table 5-1 shows the *MaxClient* setting available for the IBM HTTP Server (IHS).

For Apache on NT, use ThreadsPerChild in httpd.cnf.

Table 5-1 Adjust queue size in web server

Server name	Where to set?	Parameter	Default value	Comments
IBM HTTP Server	<IHS_install_directory>/conf/httpd.conf	MaxClients	150	A number of related parameters such as KeepAlive can also be found in this file.

In servlet engine

Just as the web server has an in-bound queuing parameter to fine tune the number of connections, so does the WebSphere Application Server servlet engine. *Max Connections* specifies the number of connections to use for the communication channel between the web server and a servlet engine. Each connection represents a request for a servlet. Table 5-2 shows the *Max Connections* parameter available within the servlet engine.

Table 5-2 Adjusting queue size in Servlet Engine

Server name	Where to set?	Parameter	Default value	Comments
WebSphere Application Server	WAS Administrative Client	Max Connections	10	Located on the Advanced Tab of the Servlet Engine property sheet.

To change the value of MaxConnections, expand **WebSphere Administrative Domain** -> expand **node <your hostname>** -> **WebSphere Commerce Server - <your instance name>** -> Click on the servlet engine **WCS Web Container** -> select **Advanced** tab as shown in Figure 5-2 on page 81. After updating the value, click **Apply** and restart the application server.

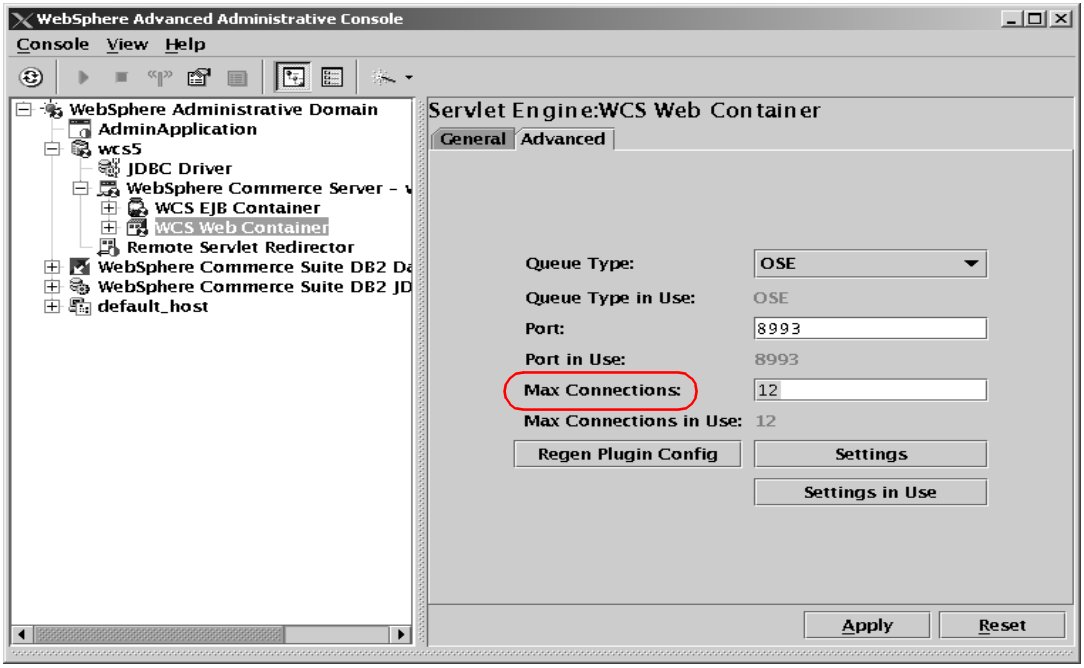


Figure 5-2 Adjust queue size in Servlet Engine

In DataSource definition

Most Java applications use Java Database Connectivity (JDBC) to access database. In any access to a database, the initial database connection is a very expensive operation. WebSphere Application Server provides database connection pooling and connection reuse. The connection pool is used to direct JDBC calls. Connection pooling gives significant performance improvement. Table 5-3 shows the database *connection pool* parameter that is available within the application server's database connection manager.

Table 5-3 Adjust queue size of data source

Server name	Where to set?	Parameter	Default values	Comments
WebSphere Application Server - DB Connection Manager	WAS Administrative Client	Minimum / Maximum connection pool size	1 / 11	Located on the Advanced Tab of the DataSource property sheet.

To change the value of minimum and maximum connections, expand **WebSphere Administrative Domain** -> **<your node hostname>** -> **WebSphere Commerce Suite DB2 DataSource** (Figure 5-3). After updating the values, click **Apply** and restart the application server.

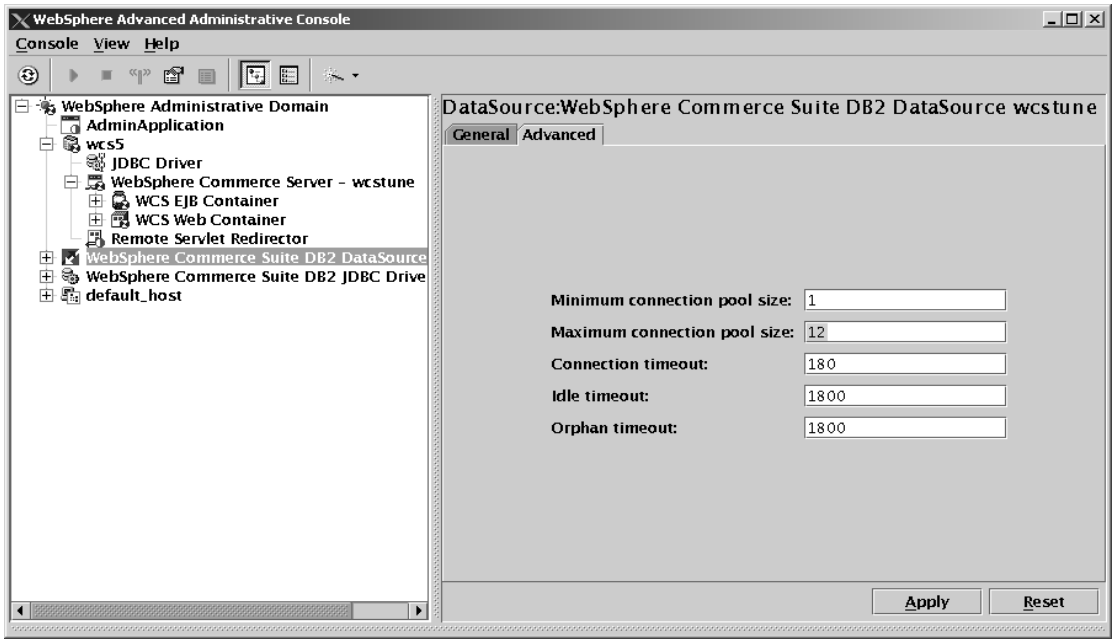


Figure 5-3 Adjust queue size of data source

In database

DB2 also has parameters that contribute to the WAS queuing pipeline. Those are:

- maxagents

  DB2 facilitates communication between the database manager and local applications through database agents. These agents handle requests from servlet engines and worker threads. When idle, agents wait in an agent pool to be assigned. The number of available agents is dependent on the database manager configuration parameters maxagents and num_poolagents. Once the number of agents reaches the value of maxagents, all subsequent requests that require a new agent are denied until the number of agents falls below that value.

- maxappls

  This parameter specifies the maximum number of concurrent applications that can be connected (both local and remote) to a database. An application can only connect to the database if there is an available connection (maxappls) as well as an available agent (maxagents). When an application attempts to connect to a database, but maxappls has already been reached, an error is returned to the application indicating that the maximum number of applications have been connected to the database.

  The value of this parameter must be equal to or greater than the sum of the connected applications, plus the number of these same applications that may be concurrently in the process of completing a two-phase commit or rollback.

5.1.3 Determining queue setting

The following section describes a methodology for adjusting the settings of WebSphere Application Server queues.Please keep in mind before starting your tests that tuning should always be done using a reasonable replica of your production environment.

Queuing before WebSphere Application Server

To begin with, you must understand that the queuing model for WebSphere Commerce Suite V5.1 is similar to that of WebSphere Application Server, but not exactly the same. The first rule of WebSphere Application Server's queue tuning is to minimize the number of requests in the WebSphere queues. In general, it is better for requests to wait in the network (in front of the web server) than to wait in the WebSphere Application Server. This configuration will result in only allowing requests into the WebSphere Application Server queueing network that are ready to be processed. To effectively configure the application server for WebSphere Application Server in this fashion, the queues farthest upstream (closest to client) should by the largest. Queues further downstream should be progressively smaller.

However, the situation is a little bit different in case of WebSphere Commerce Suite. Performance would be degraded if you make the size of DataSource queue smaller than that of servlet engine queue because WCS internally manages a number of worker threads serving resources other than servlet requests. Many of the threads require access to the database. Theoretically, the size of database connection pool is determined by adding up the number of servlet requests and database connection requests from other threads. As a rule of thumb, we recommend that the *database connection pool* size be bigger than *MaxConnections* value at least by 20 or more. If your WAS system has enough memory, make the *database connection pool* size much bigger than the *MaxConnections* value. As a result, you start with a big queue on the web server,

then define a smaller queue on the servlet engine side and another big queue on the DataSource side. A word of caution at this point; increasing the *database connection pool* is good only if the size of memory is able to sustain the number of database connections. On average, each DB2 connection uses 1-2MB of memory. Also, you need to make sure DB2 has enough connection to support your current connection pool size by adjusting the maxappls parameter.

Figure 5-4 illustrates a sample configuration that reflects this queuing model. Please keep in mind that this is just an example to illustrate the different queue settings and their relationship. You should analyze your workload requirement closely and set the parameters appropriately.

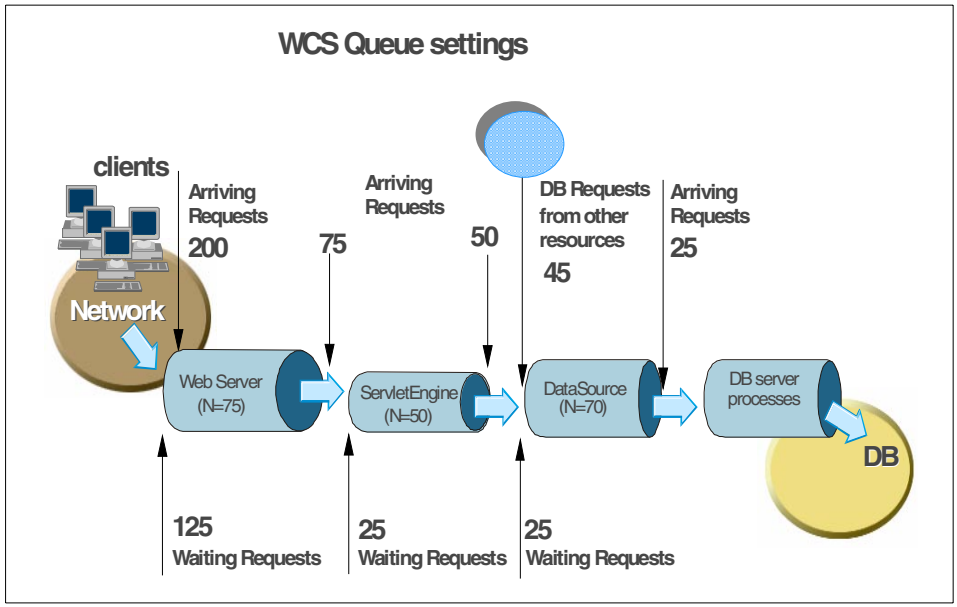


Figure 5-4 Example for WCS queue settings

The example shows 200 clients arriving at the web server. Because the web server is set to handle 75 concurrent clients, 125 requests will remain queued in the network. As the 75 requests pass from the web server to the servlet engine, 25 will remain queued in the web server and the remaining 50 will be handled by the servlet engine. This process will progress through the data source. The database connection requests from the servlet engine are processed simultaneously with database connection requests, let us say 45, from other sources. The DataSource driver manages these 70 (25 + 45) requests altogether and establishes connection with the database. No component in this system will

have to wait for work to arrive because at each point upstream, there is some work waiting to enter that component. The bulk of the requests will be waiting outside of WebSphere, that is, on the network. This will also add stability to WebSphere because no one component is overloaded.

As mentioned above, this is just an example. In the following sections we will show you how to figure out the proper queue settings for your environment, which depend very much on your hardware capacity.

Drawing a throughput curve

Using a test case that represents the full scale production environment (in terms of hardware capacity, network environment, customized WebSphere Commerce Suite, and possibly backend systems), you need to run a set of experiments to determine the saturation point of your system's full capacity. The typical goal of these tests is to drive your CPUs to near 100 percent utilization.

Start your initial baseline experiment with large queues. This will allow maximum concurrency through your system. We suggest you to start your first experiment with queue size of 100 for each component in the queuing network; web server, servlet engine, and data source. You also might need to increase the maximum connection allowed to your database instance (for example, DB2 parameter MAXAPPLS).

Now begin a series of experiments to plot a throughput curve, increasing the concurrent user load after each experiment. For example, perform experiments with 1, 2, 5, 10, 15, 25, 50, 75, and 100 users. After each test run, record the throughput (requests/second) and the response time (seconds/request). Monitoring the usage of your CPU(s) is also very important to find out the correct saturation point. Plot a graph of throughput numbers versus number of concurrent users. The curve resulting from your baseline experiments should resemble the typical throughput curve shown in Figure 5-5 on page 86.

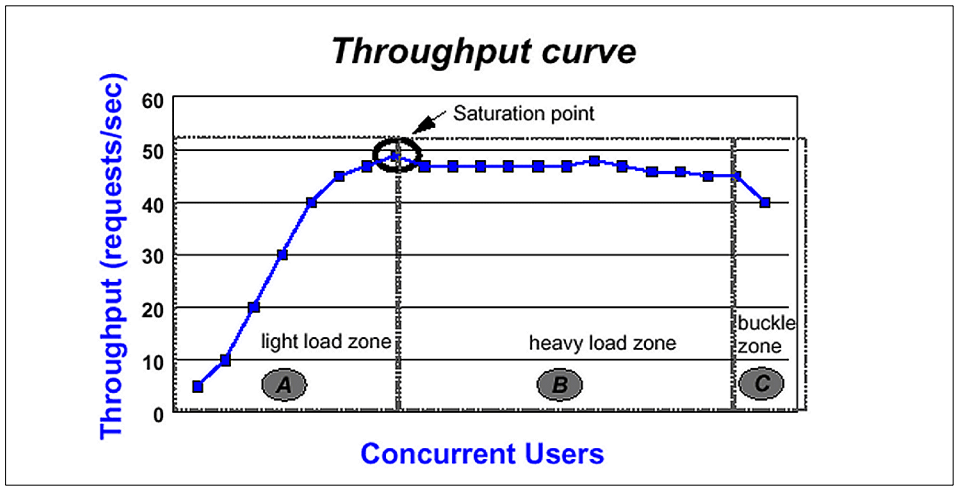


Figure 5-5 Throughput curve example

The throughput of WebSphere Commerce Suite servers is a function of the number of concurrent requests present on the total system.

Section A, “the light load zone,” shows that as the number of concurrent user requests increase, the throughput increases almost linearly with the number of requests. This reflects the fact that, at light loads, concurrent requests face very little congestion within WebSphere’s system queues. After some point, congestion starts to build up and throughput increases at a much lower rate until it hits a saturation point that represents the maximum throughput value, as determined by some bottleneck in the WebSphere system. The best type of bottleneck is when the CPUs of the WebSphere Application Server become saturated. This is desirable because you can easily remedy a CPU bottleneck by adding additional or more powerful CPUs.

Section B, in the above figure is the “heavy load zone.” As you increase the concurrent client load in this zone, throughput will remain relatively constant. However, your response time will increase proportionally to your user load. That is, if you double the user load in the “heavy load zone,” the response time will double. At some point, represented by Section C (the “buckle zone”), one of the system components becomes exhausted. At this point, throughput will start to degrade. For example, the system might enter the “buckle zone” when the network connections at the web server exhaust the limits of the network adapter, or if you exceed the limit in the operating system for the number of open files.

If you have reached the saturation point by driving the system CPUs close to 90-100 percent, you are ready to move on to the next step. If your CPU was not driven to 90-100 percent, then you may need to conduct a deeper analysis of your test system to look for the cause of the bottleneck. For example, your application might be creating excessive Java objects, causing garbage collection bottlenecks (as discussed further in Section 5.2, “Tuning JVM” on page 89). Cloning is another way to deal with application bottlenecks.

5.1.4 Queue adjustments

The number of the concurrent users at the saturation point represents the maximum concurrency of your application. It also defines the boundary between the light and heavy zones. Select a concurrent user value in the light load zone, that has a desirable response time and throughput combination. For example if your WebSphere Commerce Suite 5.1 server is saturated at 50 users, you might find that 48 users gave the best throughput and response time combination. We will call this the **Max Application Concurrency** value.

Max Application Concurrency becomes the value to use as the basis for adjusting your WebSphere Commerce Suite 5.1 system queues.

Remember, we want most users to wait on the network, so the size of the servlet engine queue should be smaller than the size of the web server queue. Also, the size of the DataSource queue should be bigger than that of the servlet engine queue to accommodate the extra database requests. However, making the servlet engine queue too small is not recommended. According to internal benchmark tests, the value of *Max connections* should be greater than 25 to achieve better performance. All these considerations work out as follows. For example, given Max Application Concurrency value of 48, you might want to set the values of WebSphere queues with the following values; *MaxClients*=75, *Max Connections*=50, *ConnectionPool*=(48 + 20)=68. Perform a set of additional experiments adjusting these values slightly higher and lower to find the best throughput/second and response time/second combination.

In many cases, only a fraction of the requests passing through one queue enter the next queue downstream. For example, in a site with many static pages and graphics, many requests will be turned around at the web server and will not be passed to the servlet engine. So a higher value (150 or more) for the web server queue (MaxClients in UNIX version of Apache) will result in a much higher throughput in case of static contents.

We cannot provide standard queue size values valid for every hardware platforms because these values heavily depend on your system environment (hardware, network and so on).

5.1.5 Adjusting transport queue type

To route requests from the web server plug-in to the servlet engine of WebSphere Commerce Suite 5.1, a transport queue has to be defined. The WebSphere Application Server provides two different kind of queues types as well as three different transport types.

To modify and view the settings for the transport queues, use the admin console as follows:

Expand **WebSphere Administrative Domain** -> expand **<your node hostname>** -> expand **WebSphere Commerce Server - <your instance name>** -> Click on the servlet engine **WCS Web Container** -> select the **Advanced** tab to display the WCS Web Container as shown in Figure 5-6. Use drop down box of **Queue Type** to change a selection.

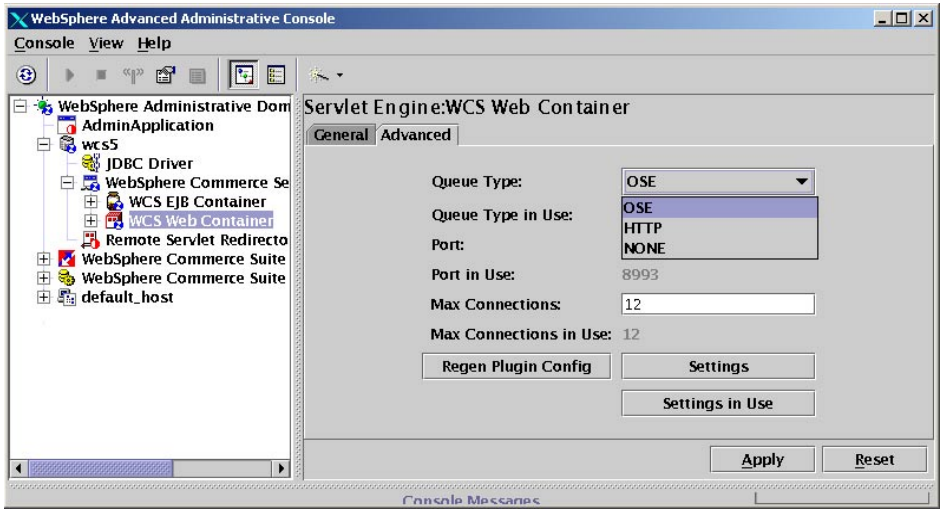


Figure 5-6 Adjusting queue type

To define your transport type, select **Settings** as shown in Figure 5-6. A small screen is displayed similar to Figure 5-7 on page 89 where you can change the transport type and other values.

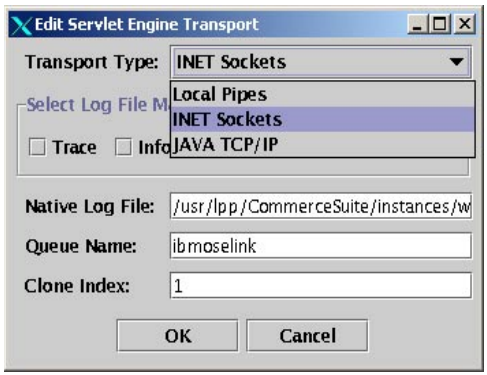


Figure 5-7 Adjusting transport type

Queue types

- OSE will provide the best performance for the transport queue between the Webserver plug-in and WebSphere servlet engine.
- HTTP should not be used at this time.
- NONE deactivates any communication between servlet engine and web server plug-in, and should only be used with servlet redirector configurations. (See *WebSphere Scalability: WLM and Clustering using WebSphere Application Server*, SG24-6153)

OSE is the default queue type defined for Websphere Commerce Suite. We recommend to stay with this configuration to achieve maximum performance.

Transport types

- Local Pipes should perform faster on AIX. In our test runs, we have observed about a 30 percent increase in total throughput.
- INET Sockets is the default for WebSphere Commerce Suite 5.1, and typically performs better on heavy loads.
- Java TCP/IP is only provided for debugging in special cases, and is not recommended for production environments.

5.2 Tuning JVM

There are two different kinds of JVMs running on a WebSphere Commerce Suite 5.1 system. One kind is used by the WebSphere Application Server itself, and the other is used exclusively for WebSphere Commerce Suite 5.1. At the operating system level you will see four Java processes running if WebSphere

Commerce Suite 5.1 is started (Example 5-1). Each of these Java processes represents a JVM. The first two of these processes belong to the WebSphere Application Server; one for AdminServer process of WebSphere Application Server and the other for the Nanny process. The third java process is WAS Administrative Console and the fourth is the application server process running WebSphere Commerce Suite.

Example 5-1 Four JVM processes

```
#ps -e|grep java
  6880  pts/0  2:54 java
 16138  pts/0  0:09 java
 16908  pts/0  1:00 java
 17204  pts/0  2:01 java
#
#ps -ef|grep java
 root 6880 16138 0 14:07:24 pts/0 2:54 /usr/IBMWebAS/java/jre/sh/../bin/java
  <----------------------Some texts have been deleted here------------------->
js.jar com.ibm.ejs.sm.server.AdminServer -bootFile admin.config -nodeRestart

root 16138 15838 0 14:07:20 pts/0 0:09 /usr/IBMWebAS/java/jre/sh/../bin/java
  <----------------------Some texts have been deleted here------------------->
com.ibm.ejs.sm.util.process.Nanny admin.config

root 16908 17676 0 14:07:49 pts/0 1:01 /usr/IBMWebAS/java/jre/sh/../bin/java
  <----------------------Some texts have been deleted here------------------->
-Dserver.root=/usr/IBMWebAS com.ibm.ejs.sm.client.ui.EJSConsole

root 17204 6880 0 14:15:55 pts/0 2:01 /usr/IBMWebAS/java/jre/sh/../bin/java
  <----------------------Some texts have been deleted here------------------->
/lib/wcsruntime.jar:./lib/wcssfc.jar:./lib/wcsuser.jar:./lib/wcsutilities.jar:
```

A JVM offers several tuning parameters that will impact the performance of the runtime environment of the WebSphere Application Server itself as well as for the WebSphere Commerce Suite 5.1 engine. In this part of the chapter we focus on tuning the JVM settings for the application server of WebSphere Commerce Suite 5.1. Tuning the JVM parameters of the WebSphere Application Server Node (AdminServer process and Nanny process) has minimal impact on the performance of the WebSphere Commerce Suite 5.1 runtime. Try to tune the JVM settings of the AdminServer and Nanny processes only when you need to shorten the time required to start WebSphere Application Server.

The JVM parameters of each WebSphere Application Server process are set via the command line arguments defined in WebSphere Administration Console. The field to enter command line arguments is located in the general tab for each application server in the WebSphere Administration Console.

To view and modify the JVM settings of WebSphere Commerce Suite instance, you just need to highlight your WebSphere Commerce Server in the Administration Console as shown in Figure 5-8. Expand **WebSphere Administrative Domain** -> Click on **WebSphere Commerce Server** -> open the **General** tab to display the WCS Web Container as shown in Figure 5-8.

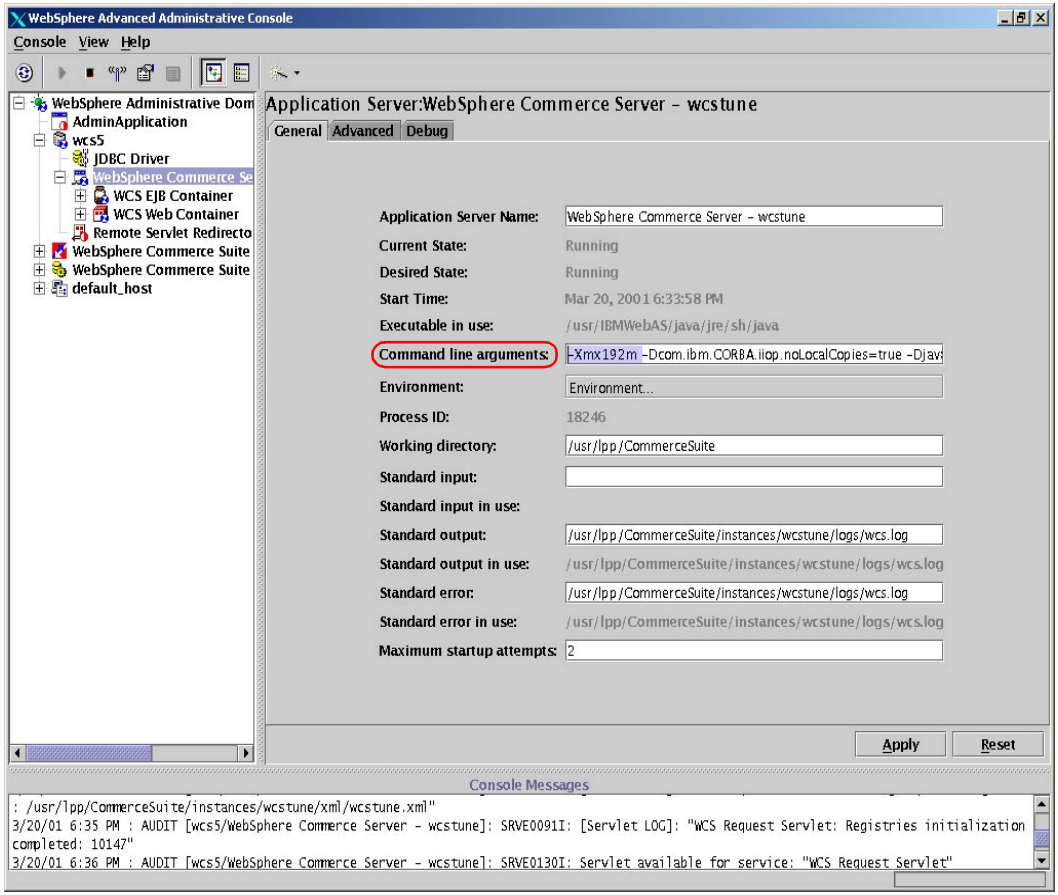


Figure 5-8 Tuning JVM settings with the Administrative Console

There are two main areas you need to pay attention to when you tune JVM:

- Heap Size
- Garbage collection

5.2.1 JVM heap size

The JVM offers two parameters for tuning of the heap size:

- *Xmx* specifies the maximum heap size of JVM
- *Xms* specifies the starting (minimum) heap size of JVM

Bigger heap size does not always result in better performance. In general, increasing the size of Java heap improves throughput to the point where the heap no longer resides in physical memory. Once the heap begins swapping to disk, Java performance drastically suffers. Therefore, the Xmx heap setting should be set small enough to contain the heap within physical memory. This will depend on your particular hardware configuration and usage pattern because physical memory usage will be shared between JVMs and other applications.

The other issue to consider when increasing the heap size is that throughput will be improved as heap size is increased, but the pause time period which is caused by JVM's garbage collection activity will also increase. The larger heap size becomes, the longer time it takes to fill up the heap memory. On the other hand, garbage collection occurs less frequently with larger heap size.

You can use -verbosegc option of JVM to see if the current heap size is adequate. Some sample output from -verbosegc is shown below in Example 5-2, which was recorded during the startup phases of the webSphere Commerce Server. The information collected with -verbosegc option is appended to the standard error output of the application server.

Example 5-2 Sample output to Admin Console with -verbosegc option

```
<AF[32]: Allocation Failure. need 24 bytes, 10890 ms since last AF>
<AF[32]: managing allocation failure, action=1 (0/66363384) (3145728/3145728)>
<GC: freeing class sun.security.provider.SHA(83118518)>
<GC: freeing class java.lang.IllegalAccessException(831a8c98)>
<GC: freeing class java.lang.InstantiationException(831a8c00)>
<GC: unloaded and freed 3 classes>
<GC(32): freed 16135336 bytes in 482 ms, 27% free (19281064/69509112)>
<GC(32): mark: 418 ms, sweep: 64 ms, compact: 0 ms>
<GC(32): refs: soft 18 (age >= 2), weak 1, final 96, phantom 0>
<AF[32]: managing allocation failure, action=3 (19281064/69509112)>
<GC(32): need to expand mark bits for 81674232-byte heap>
<GC(32): expanded mark bits by 188416 to 1277952 bytes>
<GC(32): need to expand alloc bits for 81674232-byte heap>
<GC(32): expanded alloc bits by 188416 to 1277952 bytes>
<GC(32): expanded heap by 12165120 to 81674232 bytes, 38% free>
<AF[32]: completed in 699 ms>
```

Example 5-2 on page 92 shows in the highlighted line that the heap size has been increased to just over 77MB (81674232-bytes) during the startup of WebSphere Commerce Server JVM process. Therefore, the first thing you need to consider is increasing the minimum heap size by using -Xms option. You will need to further examine the output in order to estimate how much the heap size should be increased. Making the heap size a little bigger than the point where JVM heap size was automatically increased will be a good starting point. Taking this into account, we decided to increase the minimum heap size (-Xms) to 85MB in our tests. It turned out that we could avoid expansion of the heap area with this size, but it was still not sufficient to avoid garbage collection during startup. So we then increased the minimum heap size to 96MB. This setting did prevent expansion as well as garbage collection from occurring.

5.2.2 Monitoring garbage collection

It is normal for garbage collection to consume anywhere from 5% to 20% of the total execution time of a well-behaved WebSphere application. If not kept in check, garbage collection can be your application's biggest bottleneck, especially when running on Symetric MultiProcessor (SMP) server machines.

The problem with garbage collection is simple; during garbage collection, all application work stops. This is because modern JVMs support a single-threaded garbage collector. During garbage collection, not only are freed objects collected, but memory is also compacted to avoid fragmentation. It is this compacting that forces Java to stop all other activity in the JVM. Figure 5-9 shows how garbage collection can impact performance on a 2-way SMP computer.

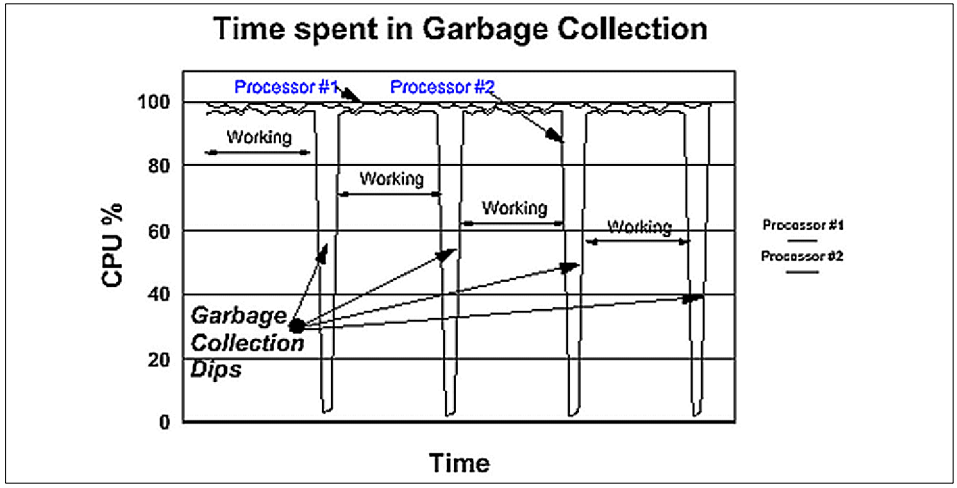


Figure 5-9 JVM garbage collection

To analyze the behavior of garbage collection, you can use a small java program named GCStats. In order to use it, you must first obtain its source code and compile the source. We have included the source code of GCStat in Appendix D, “GCStats.java” on page 199. GCStats tabulates statistics using the output of the -verbosegc flag of the JVM. Therefore, you must start WCS server with -verbosegc option before running GCStats. GCStats takes two arguments. The first argument is the location of output file contain verbosegc data. The second argument is the duration of the workload in which garbage collection took place. It is specified in milliseconds. For example:

```
java GCStats /usr/lpp/CommerceSuite/instances/myinstance/logs/wcs.log 3600000
```

This tool summarizes the collected information about garbage collection activity and is stored in the standard output log file. Example 5-3 below shows a sample output of GCStats captured during the startup phase of WebSphere Commerce Server. Minimum heap size was not specified in this case.

Example 5-3 Sample output of GCStats

```
FreeMem: 705720
TotalMem: 1048567
UsedMem: 343360
-------------------------------------------------
- GC Statistics for file - wcs.log
-------------------------------------------------
-**** Totals ***
- 41 Total number of GCs
- 14098 ms. Total time in GCs
- 523523 Kbytes. Total memory collected during GCs
-
-**** Averages ***
- 343 ms. Average time per GC. (stddev=229 ms.)
- 12768 Kbytes. Average memory collected per GC. (stddev=11450 Kbytes)
- 29%. Free memory after each GC. (stddev=7%)
- 0.4699333333333333% of total time (3000000ms.) spent in GC.
_____________________________ Wed May 09 17:04:22 CDT 2001
```

GCStats can provide clues as to whether your application is over utilizing objects. The primary statistic to look at is “total time spent in GC” (the last statistic presented in the output). As a rule of thumb, this number should not be much larger than 15%.

If the number leads you to believe that over-utilizing objects is leading to garbage collection bottleneck, there are two possible actions. The most cost effective but limited remedy is to optimize your application by implementing object caches and pools. Use a Java profiler to determine which objects in your application should be targeted. But this method is applicable only when you have access to source

codes. The second way is to use a combination of server cloning and additional memory and processors (in an SMP computer). The additional memory will allow each clone to maintain a reasonable heap size. The additional processors will allow the clones to distribute the workload among multiple processors.

So far we have showed you how to figure out the minimum heap size for the startup of the WebSphere Commerce Server. The next step is to find out a suitable amount of heap space during workload.

To get this value, you can either collect and analyze the output of garbage collection (by looking for entries of “expanded heap by ..” in the log file as shown in Example 5-2 on page 92), or to use Resource Analyzer. The more convenient and powerful way to do analysis is definitely to use Resource Analyzer (Figure 5-10) because it also provides automatically generated graphical output; average, minimum, and maximum values; and even has a replay function for logged data.

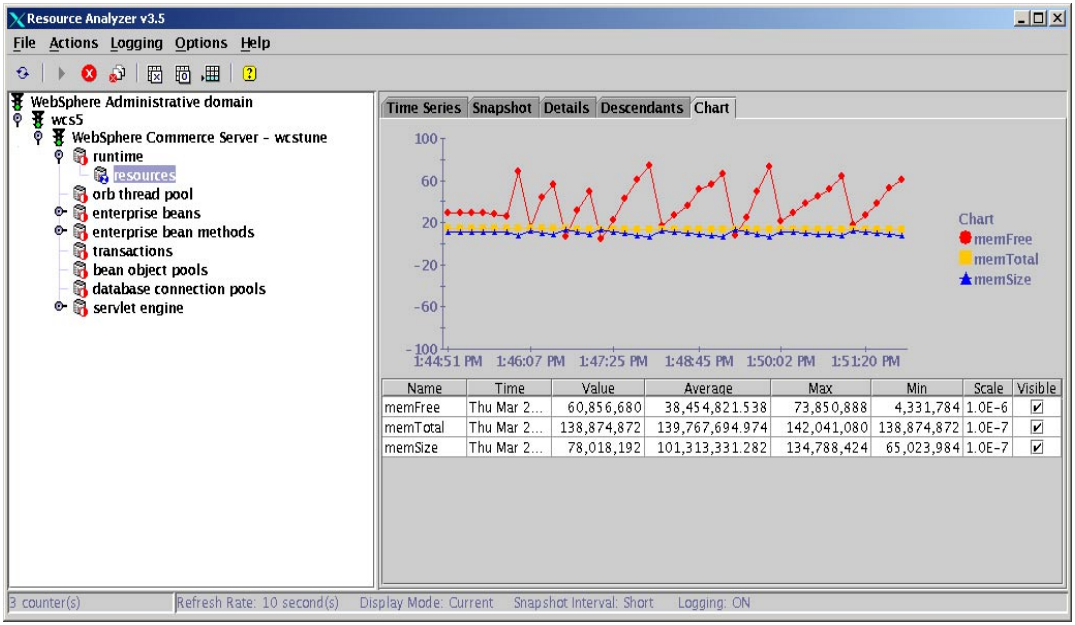


Figure 5-10 Using Resource Analyzer for JVM monitoring

To monitor the behavior of the JVM heap size with Resource Analyzer you need to start the **resources** monitor in the GUI of Resource Analyzer (see Figure 5-10 above):

Expand **WebSphere Administrative Domain** -> **<your Node-name>** -> **WebSphere Commerce Server** - **<your instance name>** -> **runtime** -> highlight **resources** -> click the **start** button icon in the menu bar.

If you want to log the data collected by Resource Analyzer, you must manually activate the logging option (this also allows you to use the playback function of Resource Analyzer):

Select **Logging** in menu bar -> click **start** -> specify your log file name -> click on **OK**.

The next step is to figure out what the best maximum heap size is under a given workload. To find out a suitable value for the maximum heap size, you should run multiple tests with different combinations for the maximum and minimum heap size. During these tests the workload must be maintained at a constant level to get comparable results for throughput/second and response time/request.

As a good starting point of tuning JVM heap size, consider setting the maximum heap size to one quarter of the total physical memory of your server and setting the minimum to one half of the maximum. For example, for a server with 1GB of memory:

```
-Xms128m -Xmx256m
```

Then you should try some higher and lower values for both of maximum and minimum heap size, setting both values equal or running only with maximum heap size defined in different values.

During our tests, with different combinations of maximum and minimum heap sizes values, the average heap size value stayed very stable at about 140-160MB (value obtained from average memSize in Resource Analyzer). Sometimes we hit the maximum heap size of 192MB (in one of our test cases with -Xmx192m), but the garbage collection reduced the used memory very fast and kept the memory utilization ratio of the system stable.

Set the minimum heap size to be as big as maximum heap size. This technique prevents java heap from resizing and will probably make the system more stable. In our testing we found out that the best throughput number could be achieved with settings of minimum heap size to -Xms192m and maximum heap size to -Xmx192m - so both to 192MB. These maximum and minimum values could vary machine by machine. Our test was done on a two-way F50 AIX server with 1GB of RAM and another two-way F50 as a remote database server.

5.3 Relaxing auto reloads

WebSphere Application Server provides reloading of servlet class files as well as reloading of JSP files. These reload settings are set for each web application.

5.3.1 Servlet auto reload

WebSphere Application Server has an auto reload capability that specifies whether to automatically reload servlets in a Web application when its class files change. This capability may simplify procedures required for testing and managing your web site because you don't need to restart the changed web application after changing servlet class files. However, this dynamic ability to reload servlets requires polling at a regular interval. The overhead for doing polling could have a negative impact on performance. Once you are in production mode, we recommend turning off servlet auto reload for WebSphere Commerce Suite 5.1 application server.

WebSphere Commerce Suite 5.1 uses two web applications in which you should turn off servlet auto reload option. To modify the settings of the web applications WCS Stores and WCS Tools, use the admin client of WebSphere Application Server as follows:

Expand **WebSphere Administrative Domain** -> expand **<your node hostname>** -> expand application server **WebSphere Commerce Server - <your instance name>** -> servlet engine **WCS Web Container** -> click on web application **WCS Stores** or **WCS Tools** -> select the **Advanced** tab to display the Web Application screen as shown in Figure 5-11 on page 98. Scroll down to the **Auto Reload** setting and switch True (default value) to False in the drop down box. Click **Apply** to save your new setting.

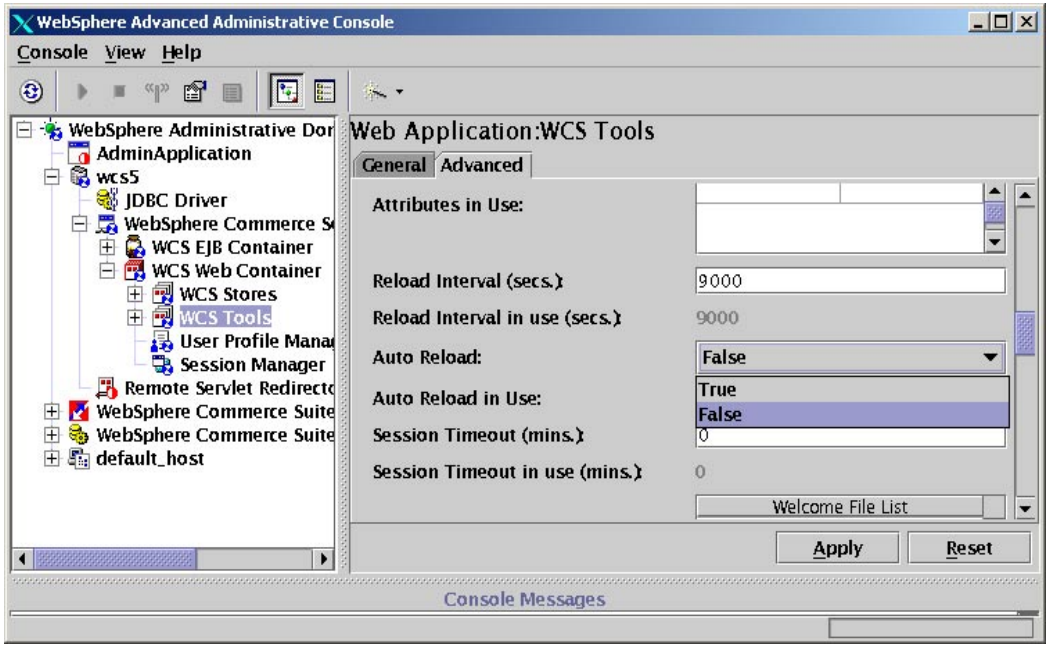


Figure 5-11 Adjusting servlet auto reload

There is also a reload interval property, as shown in Figure 5-11, that can be used to control the number of seconds between class reloading. An alternative to changing auto reload to off would be to change the value for reload interval to a very long interval value.

During our tests we did not find any significant performance improvement (less than 1 percent) by turning off servlet auto reload or setting it to a very high value. We suppose the workload given to the test system was not big enough to observe the effect of this option. However, turning off auto servlet reload makes sense on a production environment and should decrease the amount of overhead in the runtime environment of WebSphere Commerce Suite 5.1.

5.3.2 JSP reload interval

In WebSphere Application Server 3.5 and above, the JSP processing engine is implemented as a servlet. The JSP servlet is invoked to handle requests destined to all JSP files matching the assigned URL filter, such as /webapp/*.jsp. Each time a JSP file is requested, the corresponding disk file is checked to see if a newer version is available. This can be an expensive operation. As mentioned in Section 5.3.1, “Servlet auto reload” on page 97, WebSphere Commerce Suite

5.1 has two web applications named WCS Stores and WCS Tools. Change the rechecking interval by adding an init parameter, *minTimeBetweenStat,* to the JSP servlet named “WCS JSP Compiler” in each of the two web applications. To do this you can use the WebSphere Application Server Administrative Console:

Expand **WebSphere Administrative Domain** -> expand **<your node hostname>** -> expand **WebSphere Commerce Server - <your instance name>** -> expand servlet engine **WCS Web Container** -> expand both web applications named **WCS Stores** and **WCS Tools** -> click on each JSP servlet named **WCS JSP Compiler** -> open the **Advanced** tab to display the WCS JSP Complier screen as shown in Figure 5-12. Enter a new parameter name as minTimeBetweenStat in Init Parameters table and enter a value for it. Its default value is 1000(ms). 100,000 is recommended. Then click **Apply** to save your changes.

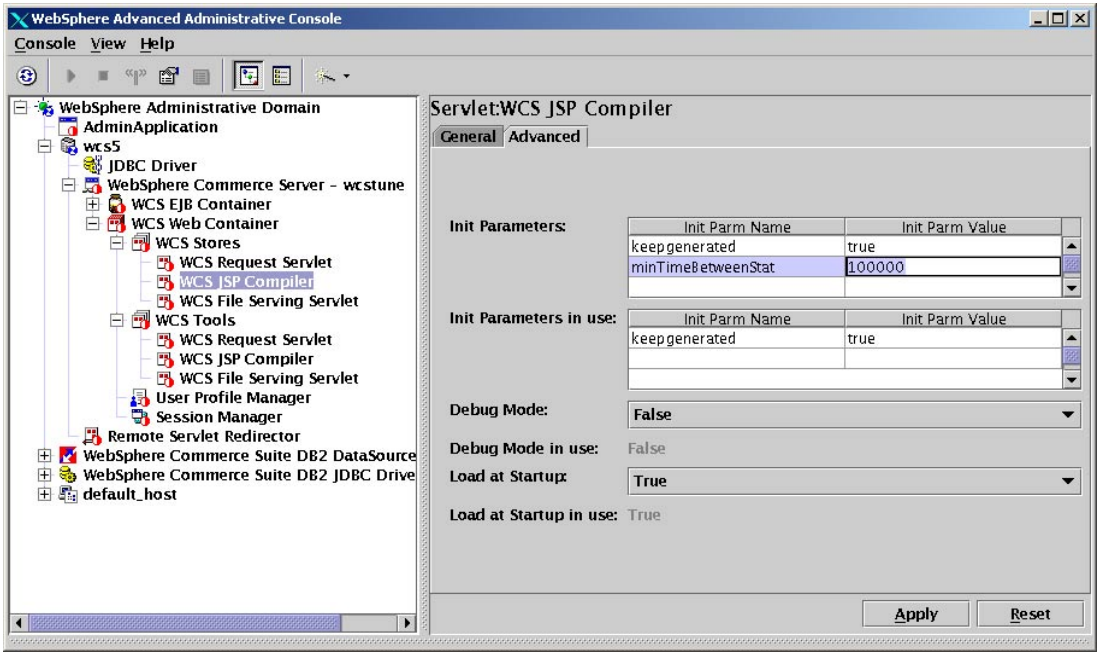


Figure 5-12 Adjusting JSP reload interval

5.4 Tuning EJB performance

WebSphere Commerce Suite uses Enterprise JavaBeans (EJB) as its default method to access database information.Those Java beans are mostly entity beans (221 entity beans out of 238 Java beans), plus a few stateless session beans performing intensive database operations (17 session beans). Those

EJBs constitute a persistent object layer for WebSphere Commerce Suite and WebSphere Commerce Suite data should be accessed only through those EJB objects. There are two points to consider which are discussed in the following sections:

- EJB container cache
- EJB pool

5.4.1 Tuning EJB container cache

EJBs reside in the Enterprise JavaBean container of the WebSphere Application Server. WebSphere Application Server has a few parameters associated with the EJB container. Figure 5-13 on page 101 presents the parameters you can modify. To access those values in the WAS Administration Console, expand the domain information, the server node, and the **WebSphere Commerce Server** instance, and click on **WCS EJB Container**. Then click on the **Advanced** tab.

The default WebSphere Commerce Suite values are the same as the default values set in any standard WebSphere Application Server installation. They are:

- Cache size : 2047
- Cache preferred limit : 2000
- Cache absolute limit : 2047
- Cache clean-up interval : 1000
- Passivation directory : null

Prevent the number of concurrent active EJBs from exceeding the limits specified in WebSphere Application Server. According to our tests, the default values of EJB container cache parameters are big enough to sustain the EJBs deployed in WebSphere Commerce Suite V5.1. However, we recommend you check whether those values are set to appropriate values before you transition to a production stage. We will discuss how to determine the maximum number of active EJBs in more detail later in this section.

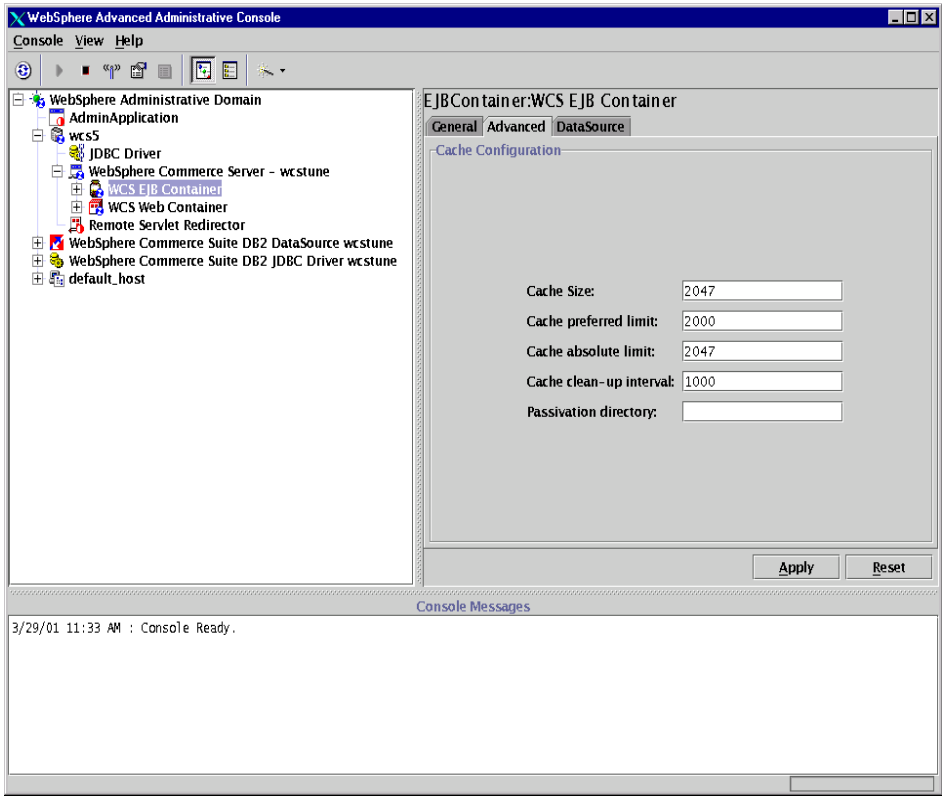


Figure 5-13 EJB Container cache parameters

Cache size

This parameter specifies the number of buckets in the cache's hash table, not the size of the container cache. If you change this value, change the Cache absolute limit property to **correspond**. For example, if you change the cache size to 3000, change the cache absolute limit to 3000, unless for some reason you do not want all of the available cache to be used. However, we recommend you not change the default value.

Cache preferred limit

This parameter specifies the number of bean instances the container attempts to retain in the cache. The actual number of bean instances cached can still grow up to the cache absolute limit value, but WebSphere Application Server will remove them the next time it cleans up the cache. Once the number of active beans reaches the cache absolute limit, the container will not create any new bean instances.

Cache absolute limit

This is the absolute limit for entries that will be maintained in cache by the container cache manager. The container will fail to allocate new bean instances when the total number of active beans reaches this limit.

Cache clean-up interval

This parameter is used to specify the time in seconds separating two attempts by the system to reduce the number of cached EJBs to the cache preferred limit. The larger the value of the cache preferred limit is, the longer this time should be set.

Passivation directory

Specifies the name of a directory where the container saves the persistent state of passivated session beans. Session beans are passivated when the container needs to reclaim space in the bean cache. At passivation time, the container serializes the bean instance to a file in the passivation directory and discards the instance from the bean cache. If, at a later time, a request arrives for the passivated bean instance, the container retrieves it from the passivation directory, deserializes it, returns it to the cache, and dispatches the request to it. If any of these steps fail (for example, if the bean instance is no longer in the passivation directory), then the method invocation fails.

This parameter is not set by default. It is not useful for WebSphere Commerce Suite because WebSphere Commerce Suite uses only stateless session beans.

In case that stateful session beans are going to be used, we recommend you limit activation and passivation of EJBs to a minimum level because these events invoke disk I/O activities. To minimize frequency of passivation and activation, adjust the cache parameters described above so that EJB container will not run out of resources.

In order to determine appropriate values for the Cache preferred limit and Cache absolute limit, you need to find out the maximum number of EJBs simultaneously in use.

A rule of thumb to get a rough estimate of this value is to multiply the maximum number of simultaneous transactions supported by the site by the average number of entity beans used in the transactions, and then add the maximum number of session beans.

The cache absolute limit value v is shown in Figure 5-14 on page 103.

$$v \leq s_{max} + n \cdot e_{max}$$

Figure 5-14 Cache absolute limit

with s_{max} the maximum number of session beans, e_{max} the maximum number of entity beans, and n the maximum number of transactions executed at the moment.

Remember that transactional behavior of your site will change from time to time, and the pattern you are seeing might be very different from the one you have already seen. The fluctuation in usage pattern could be quite significant from time to time. To get to a more accurate assessment of v value, use several time measurements of s_{max} and e_{max}. Using average values based on sample statistics collected over a period of time is recommended.

We used Resource Analyzer to monitor EJB activity during our tests, but we never observed more than 300 EJBs active at a time. As you will notice, this is far less than 2000, which is the default value of Cache preferred limit. It was not surprising, considering all EJBs in WebSphere Commerce Suite are deployed with WLM enabled. For WLM-enabled EJBs, their database access mode is set to shared mode, which is Option C caching. Option C caching means the EJBs are cached only for the length of a transaction, removing the bean upon completion of the transaction. A following transaction requiring the same EJB need to reload the bean to the cache. Therefore the EJB cache parameters will have very little effect on performance in this case. Throughout our test runs, we were unable to see any need to change the default EJB parameters defined in WebSphere Application Server.

Restriction: You cannot use Option A caching with WLM enabled.

To summarize our findings, tuning EJB cache parameters has little effect because of the way EJBs in WCS have been deployed. We recommend to change the default values only if your application clearly requires more EJBs to be cached and Option A caching can be applied.

5.4.2 Tuning EJB pools

Another way to tune your system is to limit the number of EJBs available in the pool. Each instance of EJB available in the pool consumes a certain amount of system power. Therefore, if the system actually needs far less EJBs than what are retained in the pool, the system power is wasted to keep those unused EJB instances.

To find the correct value for your pools, you first have to set the pool size to a large number and see what is required by your store. You can find this value by monitoring EJB utilization with Resource Analyzer. This is not an easy task, because in order to trace EJB activities, you have to capture various trace information from WebSphere Application Server. Turning on trace will impact the overall performance of your system. Of course, you do not need to turn on tracing for long, but to be able to determine appropriate value for EJB pool size, you need to capture trace data under a workload similar to the workload generated in the production environment. Another factor to consider is that WebSphere Commerce Suite has almost 300 EJBs. Keeping track of all of them might require a lot of work. To be practical, you had better concentrate on several most frequently used EJBs.

The values of EJB pools can be set in the advanced tab of the EJB parameters in the WAS Administrative Console as shown in Figure 5-15 on page 104. The default value is set to 100 for each EJB.

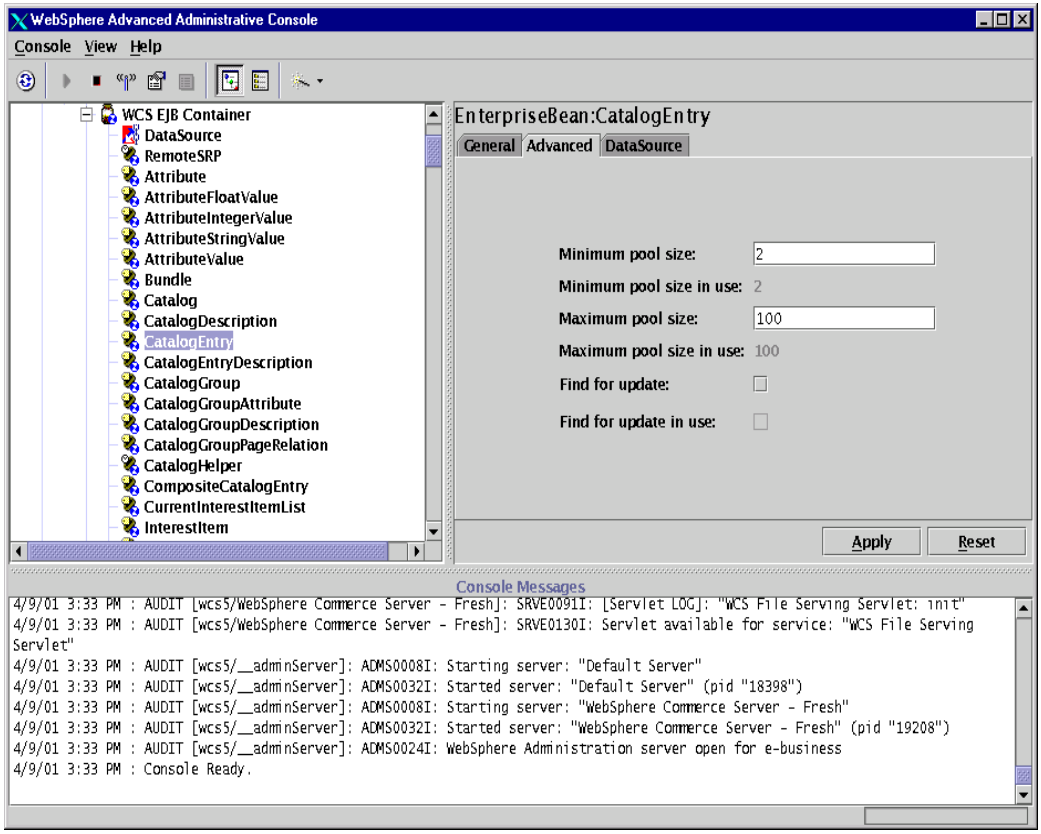


Figure 5-15 Setting EJB pool size

5.5 Effect of enabling WAS session management

WebSphere Commerce Suite supports two types of session management; cookie-based and URL rewriting. The administrator can choose to support either only cookie-based session management or both cookie-based and URL rewriting session management. If Commerce Suite only supports cookie-based management, shoppers' browsers must be able to accept cookies. If both cookie-based and URL rewriting are selected, Commerce Suite first attempts to use cookies to manage sessions. If the shopper's browser is set to not accept cookies, URL rewriting is used.

Commerce Suite session cookies are internal to Commerce Suite and are not persisted to the database, whereas WebSphere Application Server cookies provide an option to persist either to memory or to the database. There are two cases when WAS session management turns out to be more useful than WCS session management. First, if you plan to install multiple WCS machines and want to let them share session information, use WebSphere Application Server's session management to persist the session information to WebSphere Application Server's database. Second, if users wish to store and maintain their own session information, it may not be appropriate to use WCS sessions, especially if the users' session information is large. Under these conditions, the users may decide to use WAS Session Management.

To turn on WAS session management, go to WebSphere Commerce Suite Configuration Manager. Develop the tree by first selecting **WebSphere Commerce Suite -> node name -> Instance List -> instance name -> Instance properties -> Session Management**.

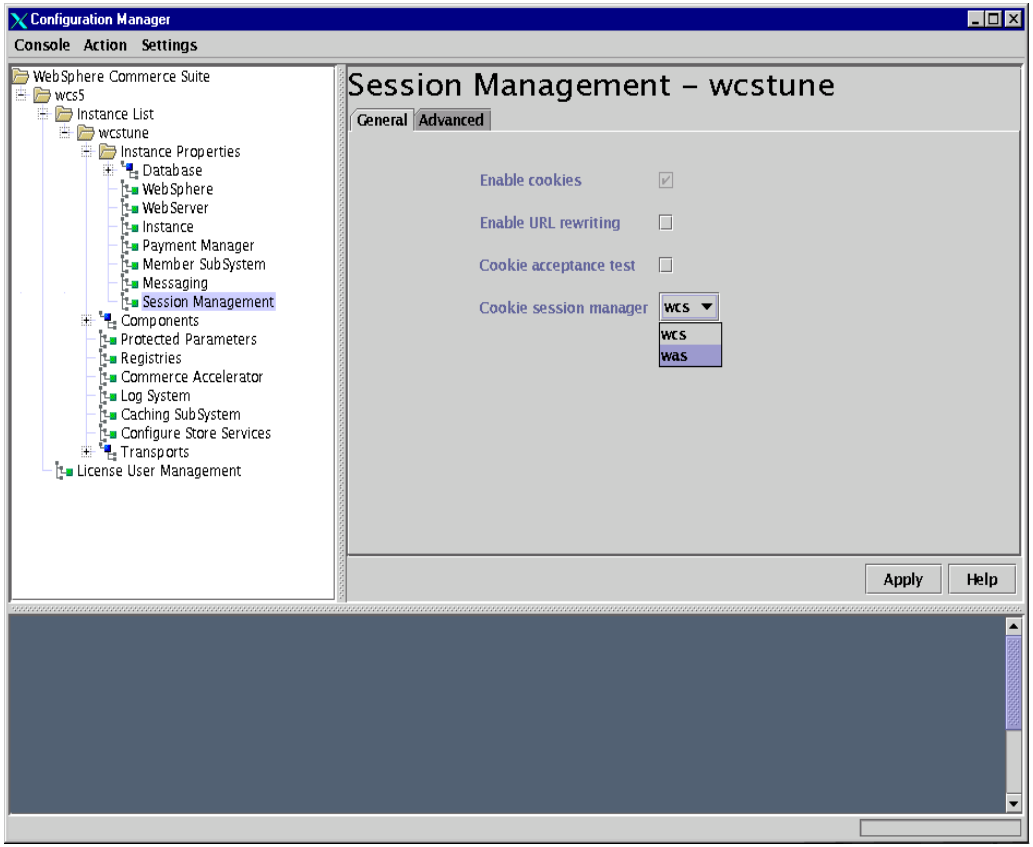


Figure 5-16 Switching from WCS to WAS session management

As you can see in Figure 5-16, the Cookie session manager parameter in the General tab has a drop down list that allows you to select the management system required by your application.

Note that WebSphere Commerce Suite session manager automatically enables session persistence in case of multi-tier installation. You do not have to do it yourself. If you are using WAS session management, however, you will have to take care of the persistence issue. To enable WAS session management, simply switch the value of Cookie session manager from “WCS” to “WAS” .

WebSphere Application Server manages sessions in two different ways. You can either set it to keep session information in its memory, or to save it in a database. The second option is accomplished by enabling persistent session management.

The benefits of using persistent sessions are:

- To keep session information from disappearing in case of a failure of the application server node.
- To share this information between different application server instances.
- To reduce the quantity of information loaded in memory when you store very large objects in your sessions.

You need to enable persistent session management in case that you want to establish workload management among multiple WCS clones. You can get more information on the procedure required to create clones in Chapter 4 of *WebSphere Scalability: WLM and Clustering using WebSphere Application Server*, SG24-6153.

To activate persistent session management in WAS, you first need to create a database that will hold all the session information. Log on as your db2 administrator ID, then enter the following command on your database server:

```
db2 create db <name_of_db>
```

where you will replace <name_of_db> with the name you want to give to this database. For our tests, we simply called it “session”. If the database is on a remote machine, you need to create a database alias on the WAS machine.

Create a new DataSource in WebSphere, referencing the new table. To do that, right-click on **WebSphere Administrative Domain**, and then select **Create -> DataSource.** Give a name to the DataSource. We called our new DataSource “WCS Sessions DB2 Datasource session”. In case that the database is on a remote machine, you need to provide database user id and its password. Enter the name of your database, which was named as “session” in our case. Next select the correct driver for your database.

Once your DataSource is created, you can activate the persistence sessions in WebSphere Application Server. Expand **WebSphere Administrative Domain -> node name -> WebSphere Commerce Server -> WCS Web Container -> Session Manager**.

Select the **Enable** tab, as shown on Figure 5-17 on page 108, and then set Enable Persistent Sessions to **Yes**.

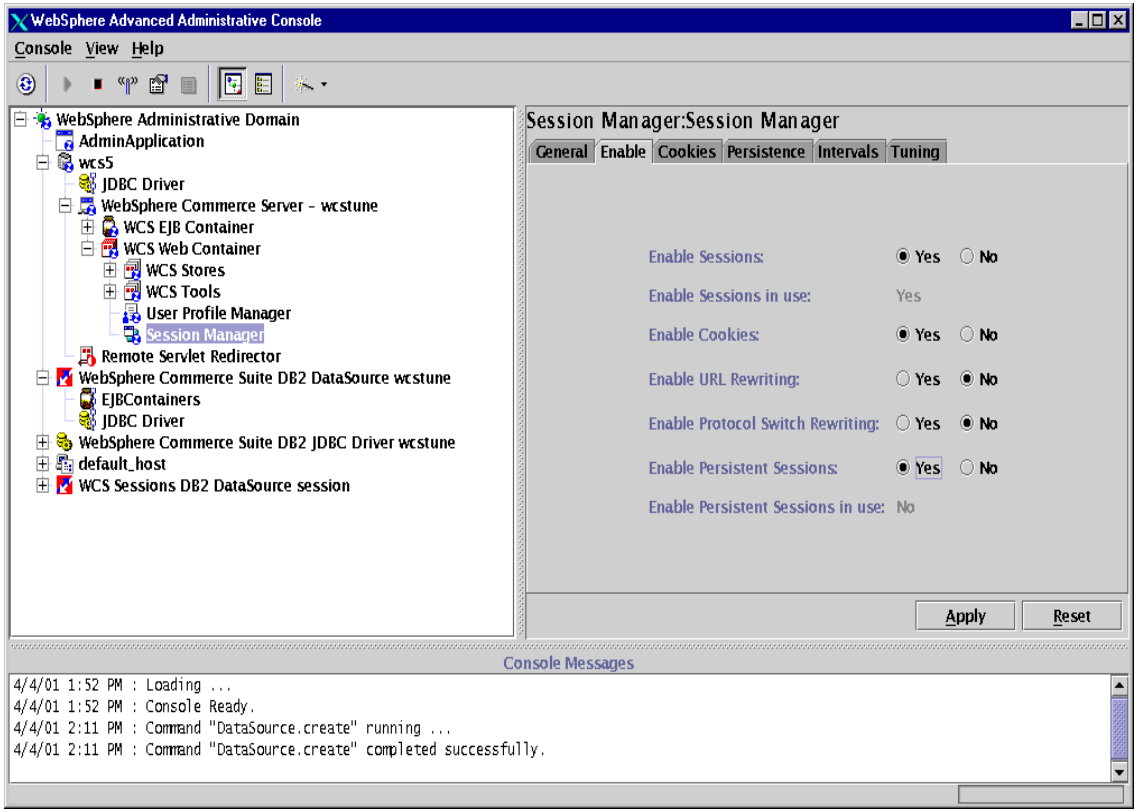


Figure 5-17 Enabling persistent session management

Next switch to the Persistence tab and enter the DataSource name you have defined, as shown in Figure 5-18 on page 109.

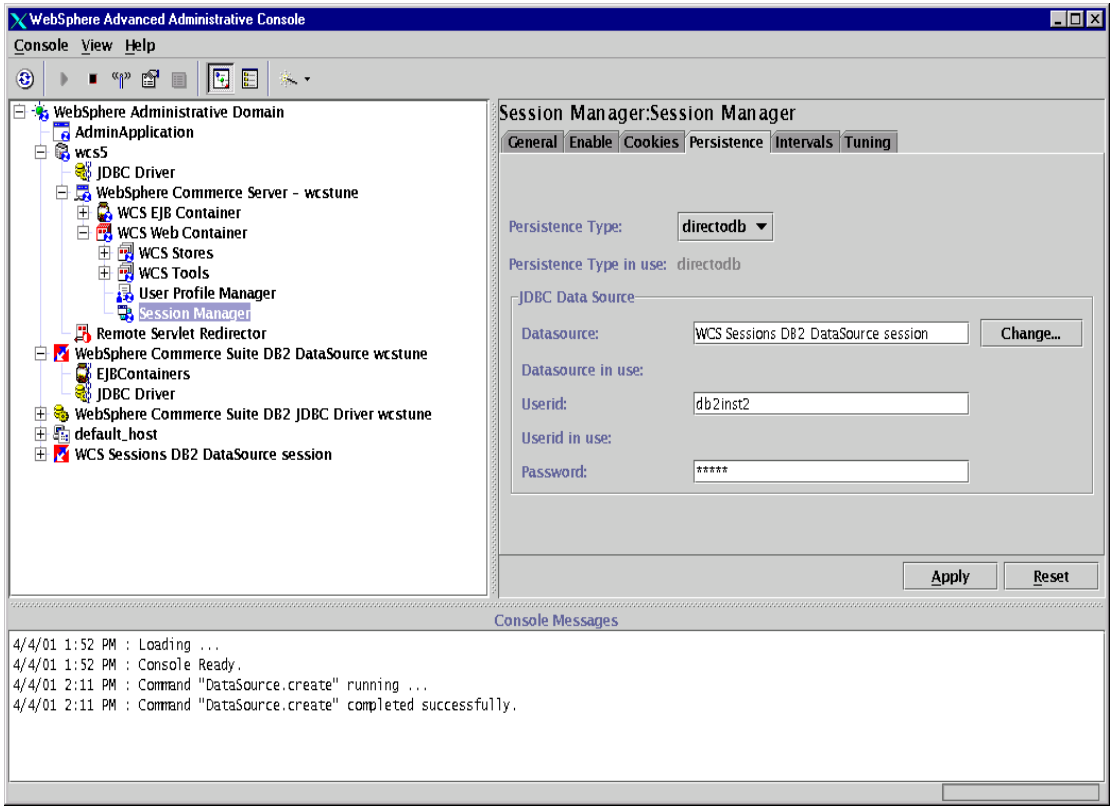


Figure 5-18 Configuring persistent session management

The tests we performed in a 2 tier configuration showed that WCS Session Management provided the best throughput. Switching to WAS Session Management resulted in a significant decrease in throughput. In our benchmark test, the throughput was decreased by eight percent. As you can suppose, switching from persistent session management to a database was even worse. We observed a ten percent decrease in throughput when compared with WCS session management.

The values given here are examples. They depend on your architecture and machine configurations, the behavior of the customers, and many other factors. If you have done a lot of customization and the amount of each session information is large, you may actually experience better performance while using WAS persistent session management.

Note: Using WLM with multiple clones of WCS instances

We have already explained before that WebSphere Commerce Suite is now a WebSphere Application Server application. As such, it can benefit from the cloning possibilities of WebSphere.Application Server.

For more information about cloning and Workload management, See *WebSphere Scalibility: WLM and Clustering using WebSphere Application Server*, SG24-6153.

5.6 Prepared statement cache

Each time your application submits a query to a database, the database manager creates a query plan. The query is then executed with this plan. In WebSphere Commerce Suite, these queries are part of the entity EJBs. WebSphere Commerce Suite uses the *PreparedStatement* class to define query objects. PreparedStatement is different from the Statement class, as PreparedStatement creates an object that precompiles the SQL associated with the query plan. This is a big improvement over the Statement class, which requires the SQL to be compiled each time the SQL statement is run.

On top of this, WebSphere Application Server provides a PreparedStatement cache. The cache reduces the initial overhead on the database manager. Where possible, WebSphere provides the database manager with an existing execution plan for the prepared statement.

If you write your own entity beans while customizing your WebSphere Commerce Suite instance, please remember to use the PreparedStatement class.

5.6.1 Choosing a value for the prepared statement cache

The cache is shared across all data sources. We used this formula to calculate the cache size:

```
Cache = (number of unique prepared statements) * (number of
concurrent clients)
```

5.6.2 Enabling and changing the prepared statement cache

The size of the prepared statement cache is determined by a command line argument to the WebSphere Commerce Suite application server. As shown in Figure 5-19, in the command line arguments field specify:

```
-Dcom.ibm.ejs.dbm.PrepStmtCacheSize
```

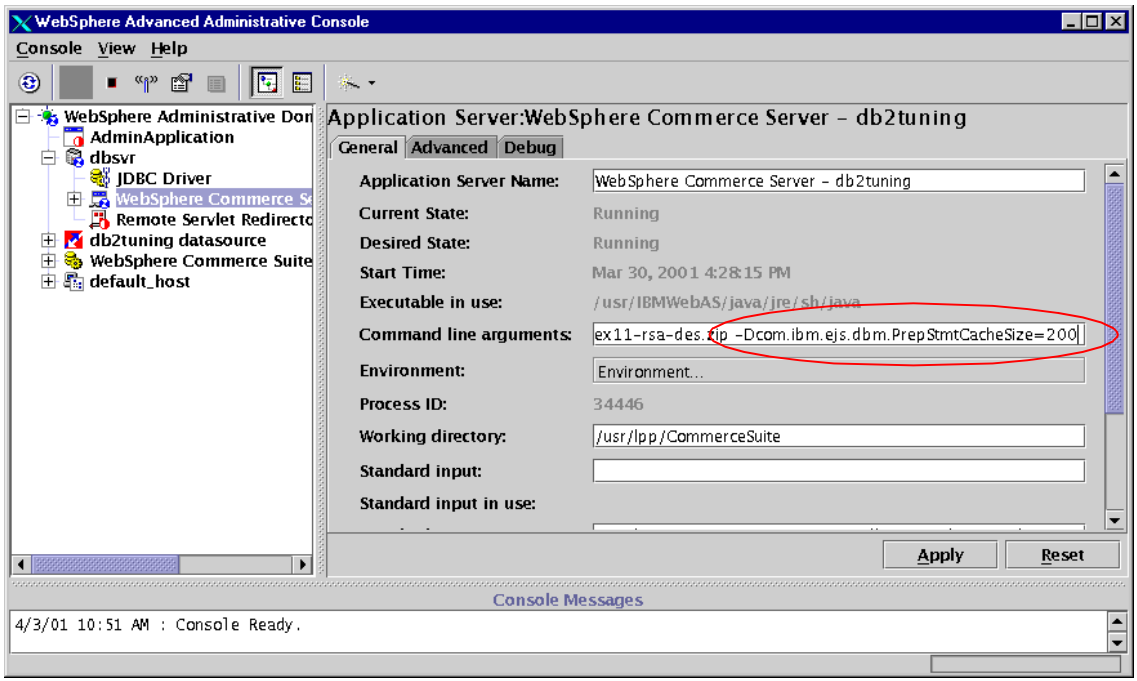


Figure 5-19 Adjusting prepared statement cache size

5.6.3 Prepared statement key cache

This parameter controls the size of the intermediate cache used to map between SQL statements and entries in the prepared statement cache. Unlike the prepared statement cache, which is shared among all DataSources, this parameter specifies the number of entries in the cache for each Data Source. The suggested value is the number of prepared statements in the application. Figure 5-20 on page 112 shows how to specify the prepared statement key cache in the Command line arguments field:

```
Dcom.ibm.ejs.dbm.PrepStmtKeyCacheSize
```

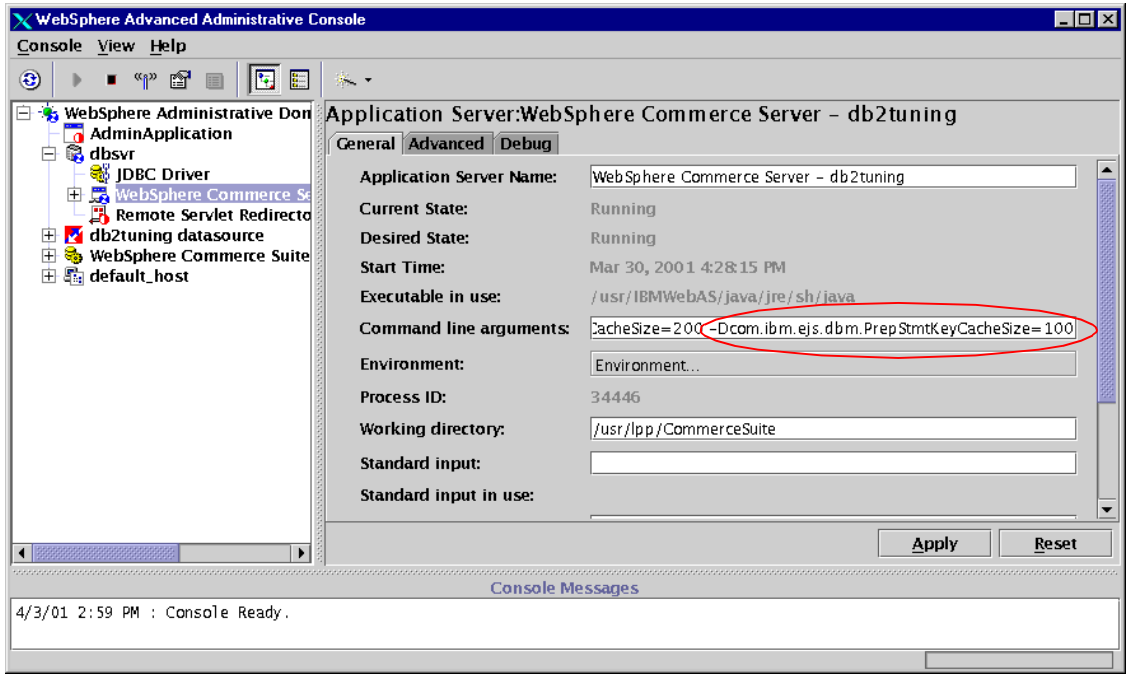


Figure 5-20 Adjusting prepared statement key cache

5.7 Call-by-reference

The EJB 1.0 specifications states that two types of method calls can be made when invoking remote methods; call-by-reference and call-by-value. The call-by-value method must first make a copy of all parameters that might be used by the remote method before the actual call is made. It is this copying process that adversely effects system performance. It is possible to configure WAS to pass a reference to the object without first having to make a copy of first.

Significant performance gains of up to 50% can be achieved by configuring WAS to use call-by-reference when the EJB client (e.g. a servlet) and the EJB server are installed on the same WAS instance. However, call-by-reference only helps performance when non-primitive Objects types are being passed as parameters. Meaning, int, floats, etc., are always copied regardless of the call model. Call-by-reference can have side effects if the remote method attempts to modify the referenced object resulting in unexpected side effects.

The default installation behavior for WCS 5.1 is to use call-by-reference because of the performance gains that are inherent with this calling model, and the following two lines are added to the command line arguments of the WCS Application Server:

```
-Djavax.rmi.CORBA.UtilClass=com.ibm.CORBA.iiop.Util
-Dcom.ibm.CORBA.iiop.noLocalCopies=true
```

Figure 5-21 shows how this setting was done in the Administrative Console.

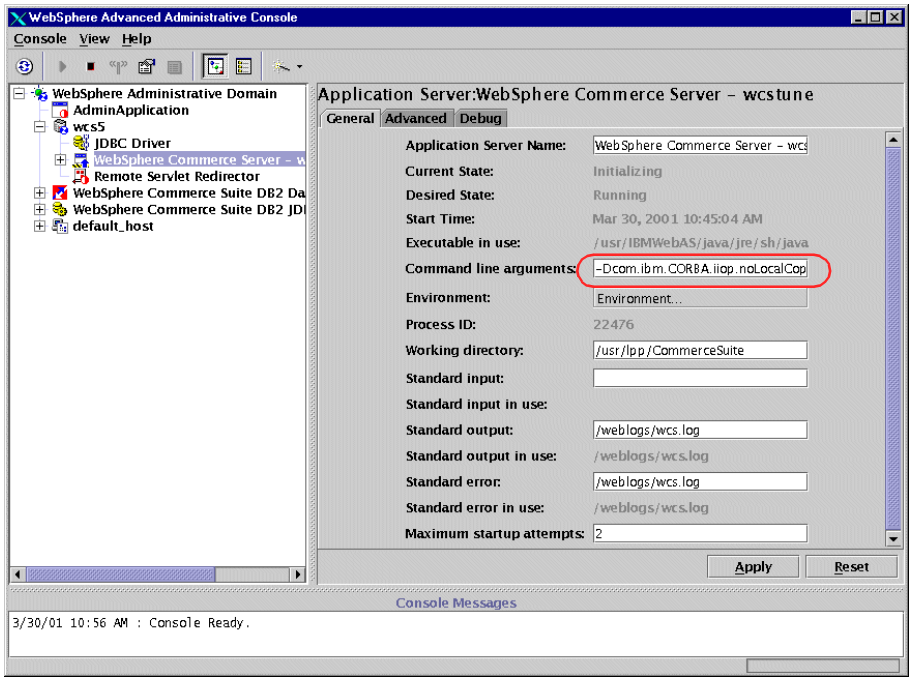


Figure 5-21 WebSphere Commerce Server command line arguments

5.8 Optimizing logging systems

A WCS system is dependant on many subsystems and components all working together in harmony. A variety of logs are produced by the WAS, WCS, and IHS servers that can provide valuable performance and diagnostic data about the system operation. This section describes what logs files are created by the WAS, WCS, and IHS servers, and how to change their settings and default locations.

Because each of the subsystems generate one or more log files apiece, the following sections will categorize them as follows:

- IHS logs
- WAS logs
- WCS logs

5.8.1 IHS logs

IHS has the ability to log all client access and errors to a file for reporting purposes. On a busy site, the access.log file can grow over 1MB or more for each 10,000 requests received. Keeping this in mind, all IHS logs should be kept on a separate set of drives within their own volume group. By doing this you will reduce the amount of disk I/O on the root volume group. If your installation does not provide enough drives to allow for a separate log volume group, then a new logical volume should be created using a size which will handle the expected size of your log files.

The following IBM redbook is an excellent reference when configuring volume groups and logical volumes.See *AIX Logical Volume Manager, From A to Z: Introduction and Concepts*, SG24-5432

Note: Keep in mind that the amount of space allocated for the IHS logs should be large enough to accommodate several days worth of logs.

access.log

It will probably be necessary to move or delete the access.log log file on a regular basis. This cannot be done while the server is still running because IHS will continue writing to the old log file. Instead, the server must be restarted after the log file is moved or deleted so that it will open a new log. On a production system this is not practical, so one way around this is to use the **rotatelogs** command included with IHS. **rotatelogs** will automatically move and create a log file at a pre-set interval, such as once per day, eliminating the need to restart the IHS server. Example 5-4 shows an example of how **rotatelogs** is used.

Example 5-4 Rotatelogs usage for TransferLog in the httpd.conf file

```
#
TransferLog "|rotatelogs <path_to_logs>/access.log 86400"
#
```

where "| rotatelogs <path_to_logs>/access.log 86400" instructs IHS to send all access log data to the rotatelogs program, which will be written to the /path/to/logs/access.log.*nnnn* file every 86400 seconds or 24 hours. The *nnnn* extension is added by the **rotatelogs** program and denotes system time when the log was created.

Note: CustomLog and TransferLog work in the same fashion for logging, with the only exception being that TransferLog does not allow for the log format to be specified explicitly or for conditional logging of requests.

In Section 5.1, "Adjusting queue sizes" on page 78, we discuss the various queues that are inherent in WCS and the methods of adjusting them. It is worth noting that access.log file is the file that records all client activity with the greatest amount of detail. This file should be reviewed using a product such as the IBM WebSphere Site Analyzer to measure overall site performance as you make changes to various system settings.

error.log

The error.log file behaves in the same way as the access.log file, although there should be far fewer entries in the error.log file. It is a good practice to check the contents of the error.log file from time to time to identify errors and determine their severity.

The **rotatelogs** command can also be used to copy the error.log on a daily basis in a similar fashion to the access.log. Example 5-5 shows an example of how **rotatelogs** is used.

Example 5-5 Rotatelogs usage for error.log file in the httpd.conf file

```
#
ErrorLog "|rotatelogs <path_to_logs>/error.log 86400"
#
```

5.8.2 WAS logs

Just as IHS creates a number logs showing the operation of the system, WAS also has several log files that can be utilized to check whether the system is functioning properly. They have a low-to-medium impact on performance, and should be left enabled, allowing them to be reviewed should an error or problem condition occur. If possible, logs should be stored on a separate logical volume.

The log files specific to the WebSphere Application Server Administrative Server will be listed below. Section 5.8.3, "WCS logs" on page 120 will detail more specific information specific to WCS logging.

Serious Event reporting

The administrative repository stores event logs as well as configurations. Event information will appear in the bottom pane of the Administrative Console. Event information is also stored in standard output files such as <application server name>stdout.log and <application server name>stderr.log under the <WAS_install_path>/logs directory.

Serious Event Pool interval

The Serious Event Listener is a lightweight background thread that runs every 10 seconds by default, polling the administrative repository for changes in the configuration or runtime state. The listener executes select statements and stores them in the administrative repository. By default, a database select is issued to the administrative repository for each type of event (fatal, warning, audit). Any events returned as a result are reported in the Console Messages section of the Administration Client. Although it does use significant amount of system resources, the Serious Event listener thread can be tuned to execute at a desired time interval.

You can change the polling interval and the types of messages reported by selecting the node in the Administration Client then selecting **Console-> Trace-> Event Viewer...**, A new window will pop up, presenting Events Viewer dialog. The pool interval field can be accessed under the **Preferences** tab shown in Figure 5-22 on page 117 and adjusted by changing **Serious Event Pool Interval** from the default of 10 seconds.

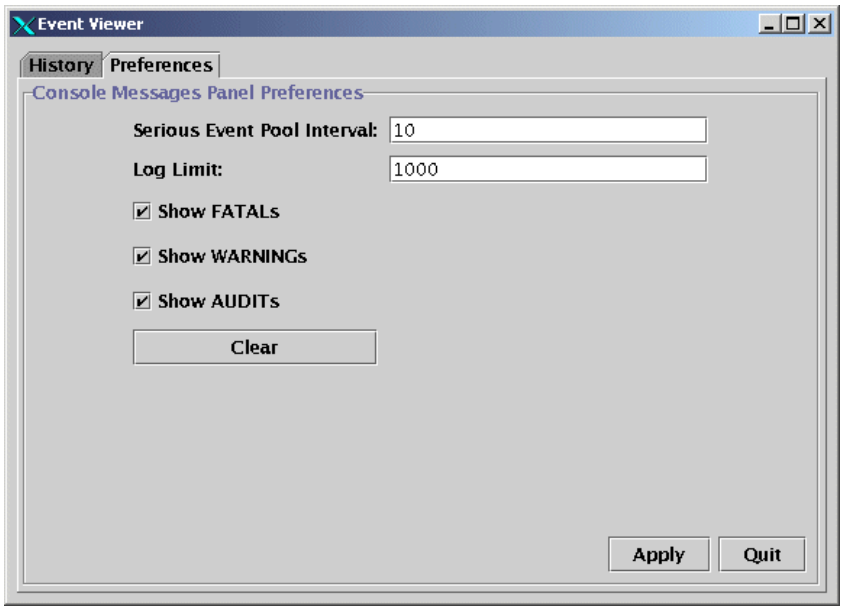


Figure 5-22 Serious Event preferences screen

A value of 0 turns this off and is not recommended. Once you go into production you can set this interval period to be longer. The check boxes allow you to choose the event categories to be selected.

Log Limit

A more important setting for performance is the Log Limit for the Serious Events table in the WebSphere configuration repository. The default setting for the Log Limit is 0 (zero) for versions of WAS prior to 3.5. This setting is misleading because this actually maps to no limit being set for the entries in the Serious Events table. Over time this will degrade database performance as the table grows, and could result in a database error if sufficient space on the file is not allocated. Setting this to a value of between 1000 and 5000 should provide adequate logging while not adversely impacting system performance.

tracefile

WAS displays system messages in the administration console messages area and also writes these messages to a file called “tracefile” located in the <WAS_install_path>/logs directory by default. When WAS is first started, all startup and initialization messages are written the tracefile log including the message “WebSphere Administration server open for e-business”, indicating that

the WAS initialization is complete. If desired, you can specify an alternative file name and directory by updating the entry for the “com.ibm.ejs.sm.adminServer.traceFile” property in the <WAS_install_path>/bin/admin.config file.

nanny.trace

The nanny.trace file records administrative events for WAS. This file is located in the /usr/IBMWebAS/logs directory by default. If desired, you can specify an alternative file name and directory by updating the entry for the “com.ibm.ejs.sm.util.process.Nanny.tracefile” property in the <WAS_install_path>/bin/admin.config file.

This file records events associated with starting and restarting WAS and application servers, and as such its size should not grow significantly.

<application server name>_stdout.log | stderr.log

Each application server created within WAS has two standard logs files that are associated with it; stdout.log and stderr.log. The default setting of standard output and standard error are stdout.log and stderr.log. We suggest that you specify the file name as a unique name, such as <application server name>_stdout.log or stderr.log.

The stdout.log contains System.out messages from the application server or Servlet Redirector, and stderr.log contains System.err messages from it. The amount of information logged to these files is application dependant, so their size should be monitored and truncated as required.

To change the log file names or the directory location using the WAS administrative client, expand the **WebSphere Administrative Domain** -> expand **<your node hostname>** -> click on **WebSphere Commerce Server** - <**your instance name>**, and enter a new directory and filename for each of the logs. Then click **Apply** as shown in Figure 5-23 on page 119.

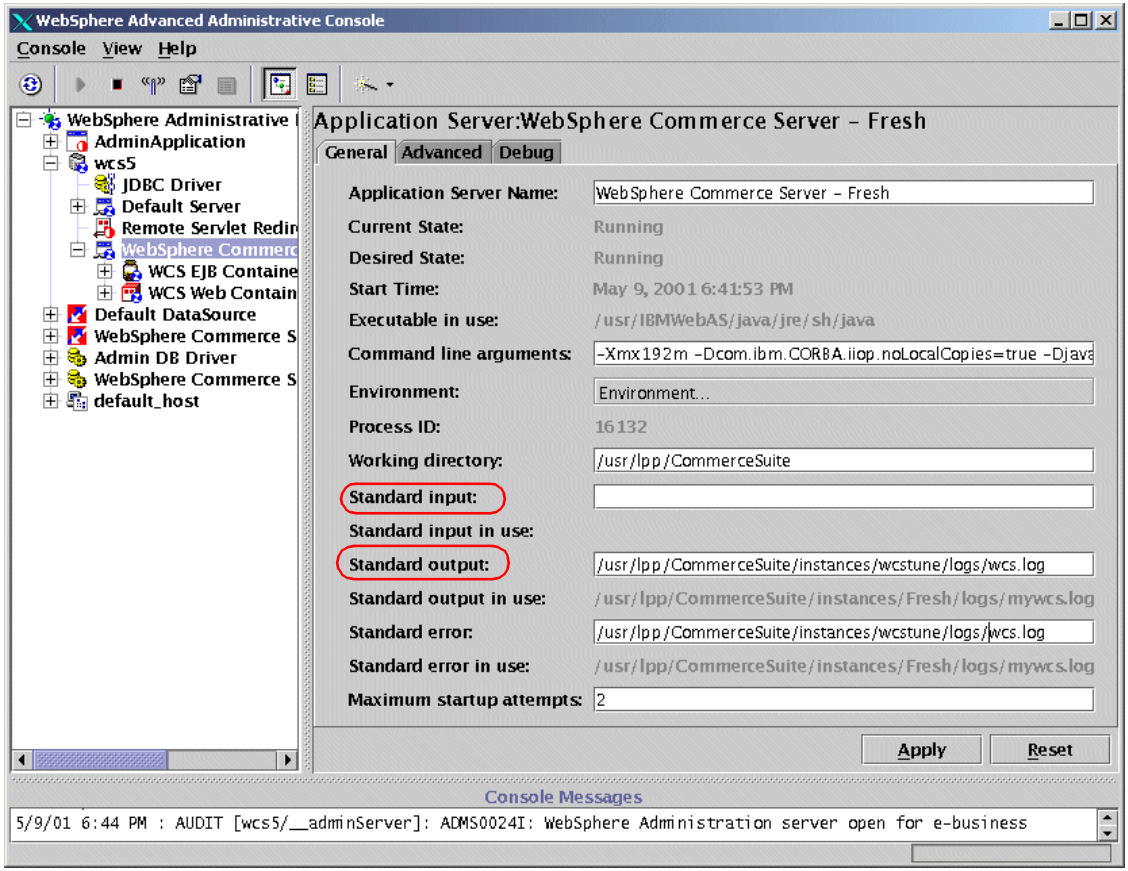


Figure 5-23 WAS Application Server stdout and stderr log entries

There is a debug feature that can also be enabled for each application server that will increase the amount of logged information by adding debugging information. In a production environment this setting should be turned off because of the performance impact it causes on the system. The default setting for debug mode is off. To verify the setting of debug mode expand the **WebSphere Administrative Domain** -> expand **<your node hostname>** -> click on **WebSphere Commerce Server** - **<your instance name>**, then click on the **Debug** tab and verify that the check box next to “Debug enabled” is unchecked as shown in Figure 5-24 on page 120.

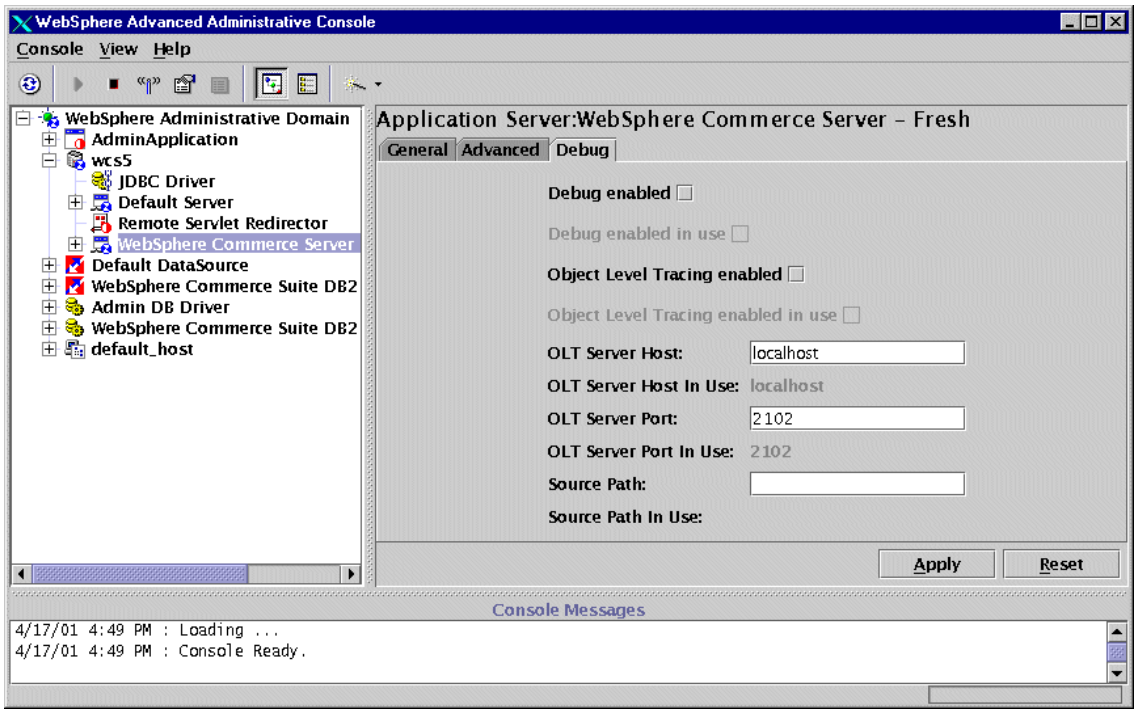


Figure 5-24 Turning off Debug option

5.8.3 WCS logs

There are two types of logs found in the WebSphere Commerce Server; diagnostic and activity. Diagnostic logs are used for problem determination and are stored in log files. Activity logs record events that are stored in the Commerce Suite database. WCS allows you to choose which components you want to appear in the trace file, as well as the level of defect tracking that you want the trace file to contain. The more items you trace with a higher defect level, the more performance impact you will experience on the overall system. Unless it is needed, WCS logging should be kept to the minimum number of components required, with the defect level set either to *normal* or *none*.

The following displays the general settings for the WCS logging facility. To make changes to the WCS log system, start the configuration manager and then expand **WebSphere Commerce Suite** -> expand **<your node hostname>** -> expand **Instance Lis**t -> expand **<your instance name>** -> expand **Instance Properties** -> then click on **Log System**. Figure 5-25 on page 121 will be displayed showing the parameters that were contained in the Instance Creation wizard.

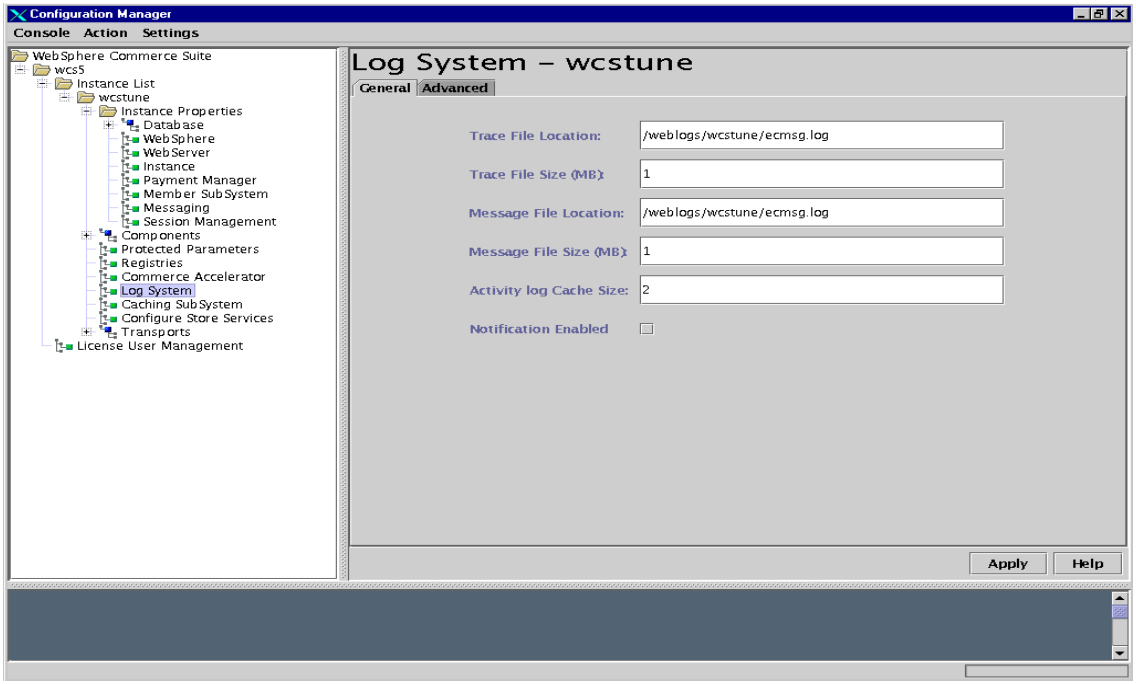


Figure 5-25 WCS Log System General settings

Selecting the **Advanced** tab will display Figure 5-26 on page 122, which allows you to select the components you want traced and the trace level. Specify only those components that are necessary as each selection will have an effect on overall performance. Once your selections are complete, click **Apply**. Then stop and restart WCS. For more information on individual components, please see the Commerce Suite online help.

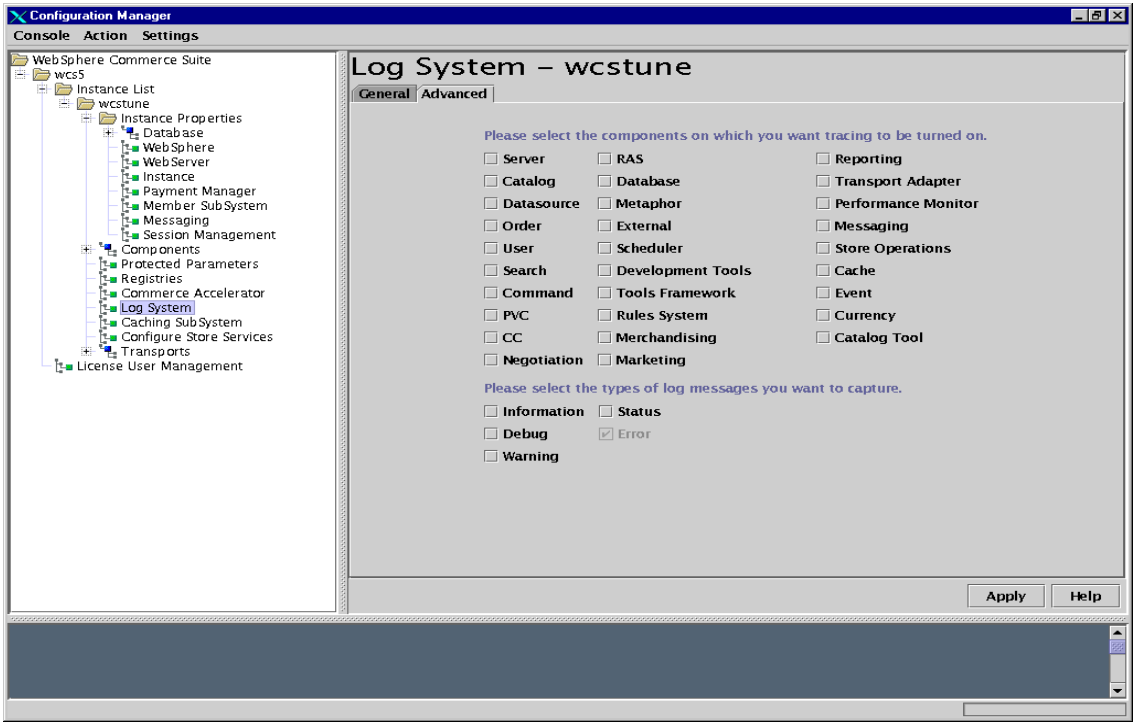


Figure 5-26 WCS Log System Advanced settings

WASConfig.log

Found in your /usr/lpp/CommerceSuite/instances/<*instance_name*>/logs directory. This log describes the WebSphere Application Server actions such as importing Commerce Suite entity beans and creating the DataSource during the initial installation, and is not used afterwards.

createdbschema.log

Found in your /usr/lpp/CommerceSuite/instances/<*instance_name*>/logs directory. This logs the steps during the initial Commerce Suite schema creation process, and is not used afterwards.

populatedb.log

Found in your /usr/lpp/CommerceSuite/<*instance_name*>/logs directory. This log describes the population history of the database that is created during the initial instance creation, and is not used afterwards.

wcs.log

Found in your /usr/lpp/CommerceSuite/ instances/<*instance_name*>/logs directory. This log describes the WebSphere Commerce Server.

WCSconfig.log

Found in your /usr/lpp/CommerceSuite/instances/<*instance_name*>/logs directory. This log describes what the Configuration Manager is doing. You can modify the level of detail in this log through the menu options shown in Figure 5-27. To get this menu, click **Settings** -> **Log Settings...** in the WCS Configuration Manager. The log level should be normal in most cases, with the check box for “Append to existing log” left unchecked. This will recreate the wcsconfig.log file each time the Configuration Manager is started.

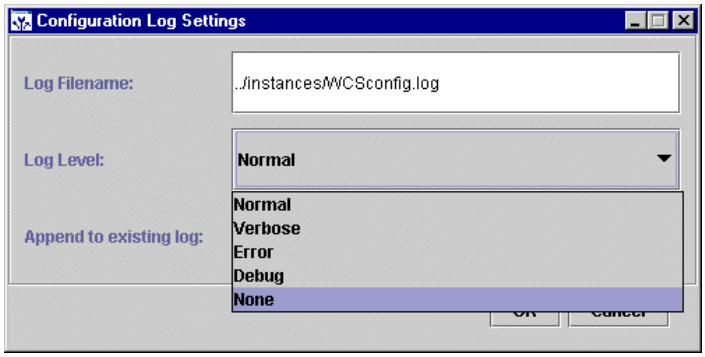


Figure 5-27 WCS Configuration Log Settings

5.9 Avoiding file serving servlet

When placing web resources such as HTML files, JavaScript scripts, image files, JSP files, and so on, you can choose to have the pages be served by WAS or just be served by the web server. To have the files served by the web server, you need to place them in the web server's document root. If you choose to have the files be served by WAS, you need to use file servlet. The file servlet (sometimes called file-serving servlet or file serving enabler) can serve HTML files or other files located in the web application’s document root, which is defined in WAS, without the help of extra pass rule definitions in the web server. Remember that by default two web applications, WCS Stores and WCS Tools, exist in WCS V5.1. The file servlets defined in the web applications handle files whose URLs are not covered by the configured servlet URLs.

For the case where HTML pages are served by the web server, as opposed to being served by the WAS, there may be an increase in performance because the web server is serving the pages directly without the help of file serving servlet. Therefore, we recommend you avoid using the file serving servlet whenever possible.

There is a caveat you should be aware of in applying this technique. The procedure requires changes in the source codes of your web resources as well as changes in the web server configuration. For example, suppose a GIF file is referenced with a relative path of 'images/picture.gif' in a JSP file, let's say /usr/lpp/CommerceSuite/stores/web/<*store_name*>/banner.jsp. When the JSP file is called from a browser, because it is served by a servlet the image file is expected to be found in a path relative to the WCS Stores's document root. The default web path in this Web Application is <WCS_install_directory>/stores/web. The problem here is that the web path cannot be understood by the web server if you disable the file serving servlet for performance reasons. Remember that we have used a relative path to point to picture.gif in the JSP file. Relative paths are relative to the original request. Because we have disabled the file servlet, the relative path is now pointing to the web server's document root directory, not the WCS Stores's document root. To correct this problem, you also need to modify the original source of programs.

Do the following steps to have the web server serve web file resources:

1. The two file servlets are enabled by default in WCS V5.1. Disable them by **WebSphere Administrative Domain** -> expand **<node your hostname>** -> expand **WebSphere Commerce Server** - **<your instance name>** -> Click on the servlet engine **WCS Web Container** -> **WCS Stores** (or **WCS Tools**)-> Right click **WCS File Serving Servlet**. See Figure 5-28 on page 125.

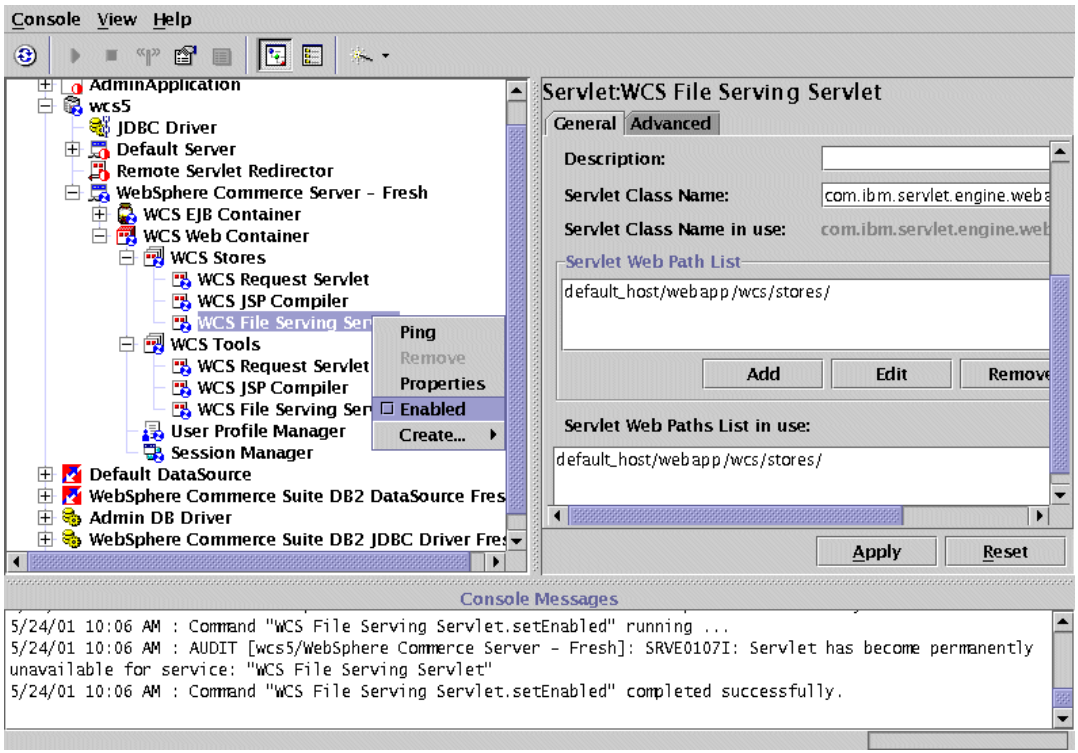


Figure 5-28 Disabling File Servlet

2. Create a directory, for example, /production/images, and copy the picture.gif file into it.
3. Edit the IHS configuration file and add an alias entry for the directory you have created. For example:

   Alias /images /production/images

   You need to stop and restart the IHS server processes.

   You could repeat the step 2 and 3 for other web files (HTML files, JSP files and so on).
4. Edit the banner.jsp and change the path to the picture.gif file from ‘images/picture.gif’ to ‘/images/picture.gif’.

5.10 HttpSessions in JSP

This section gives a little JSP programing tip that will help performance. By default, JSP files create HttpSessions. This is in compliance with J2EE to facilitate the use of JSP implicit objects, which can be referenced in JSP source and tags without explicit declaration. HttpSession is one of those objects. If you do not use HttpSession in your JSP files, you can save some performance overhead with the following JSP page directive:

<%@ page session="false"%>

6

Tuning Web Server

The purpose of this chapter is to describe some of the configuration settings that should be used when implementing the IBM HTTP Server V1.3.12.1 (IHS), which is based on the popular open standards-based Apache Web server.

The following provides a methodology to be used while tuning the performance of IHS and the actual settings that work best under WebSphere Commerce Suite (WCS) environment. However, as configuration choices may differ case by case, your settings should be tested with each individual application that will be run on the server. Keep in mind that changing the value of one directive can have a negative effect on other directives, so we recommend you make changes one by one and test the results in your system implementation.

Windows: Check httpd.cnf file.

There are several directives that we can specify in the httpd.conf file, which is located in the in the /usr/HTTPServer/conf directory on AIX. You can use a text editor to modify the parameters in the httpd.conf file. We have divided the directives into five different areas:

- Process handling
- Connection
- Resource usage
- Name resolution
- Fast Response Cache Accelerator (FRCA)

6.1 Process handling

Parameters belonging to this category are primarily related to the httpd processes.

6.1.1 MaxClients

Windows
ThreadPerChild

This parameter limits the total number of simultaneous HTTP requests that IHS can process. Because IHS uses one child server process for each HTTP request, this limits the number of child server processes that are able to run simultaneously. The default value is 150 and maximum value is 2048.

The value of this directive can significantly impact the performance of your applications, particularly if it is too high. We have seen that the speed, number of CPU's, and the amount of memory can have an effect on the overall performance of the system. The optimum value depends on your application and your exact machine configuration. In general we suggest the following:

- For simple CPU intensive applications, use a lower MaxClients value.
- For more complex database intensive application with longer wait times, use a slightly higher MaxClients value.
- For static HTML-only applications, use an even higher MaxClients value.

For example, exceptional performance on simple servlets such as HelloWorld and Snoop have been achieved using values as low as 25.

Because WebSphere Commerce Suite is a database-intensive application, the default setting of 150 may be too large. Use a number around 50 as a starting point and adjust up or down using small increments and decrements. The goal with adjusting this value is to always have a moderate number of HTTP requests in the wait queue ready to be passed to the WCS's servlet engine, but not many waiting requests that are held for a long time.

During your tests, be sure to watch the server CPU utilization. Do not increase the MaxClients setting if the CPU utilization reaches 100% busy and in doing so causes the server response time to exceed your response time criteria.

Note: Setting the value of MaxClients(or ThreadPerChild) only controls the maximum number of HTTP servers that can be started, not how many will be started. Set the StartServers directive equal to MaxClients when attempting to determine the appropriate value of MaxClients for your system.

Example 6-1 shows how to set its value.

Example 6-1 MaxClients directive in httpd.conf file

```
#
MaxClients 50
#
```

6.1.2 StartServers

The number of child server processes that are created when IHS is initially started. Increase this value to equal the number of MaxClients once that number is known. This will insure that all HTTP processes are started and available. The default value is 5. Example 6-2 shows how to set its value.

Example 6-2 StartServers directive in httpd.conf file

```
#
StartServers 50
#
```

6.1.3 MaxSpareServers

Sets the maximum number of idle httpd child processes to keep running. If there are more idle httpd child processes running then what MaxSpareServers is currently defined, then the extra httpd process will be killed off.

This value should also be changed to the same value of MaxClients, keeping all processes available. The default value of MaxSpareServers is 10. Example 6-3 shows how to set its value.

Example 6-3 MaxSpareServers directive in httpd.conf file

```
#
MaxSpareServers 50
#
```

6.1.4 MinSpareServers

Sets the minimum number of idle httpd child processes. If there are fewer than MinSpareServers, then additional httpd child processes will be created at a maximum rate of 1 per second. If there are more than MaxSpareServers, then the parent process kills off the excess child processes. The default value of MinSpareServers is 5. Example 6-4 on page 130 shows how to set a value for MinSpareServers.

Example 6-4 MinSpareServers directive in httpd.conf file

```
#
MinSpareServers 5
#
```

The three directives above can also impact your application performance. For optimum performance, first determine the appropriate value of MaxClients, then keep the MaxClients, the StartServers and the MaxSpareServers directives equal so that CPU is not expended creating and destroying httpd child server processes.

6.1.5 MaxRequestsPerChild

This parameter sets the maximum number of requests that can be handled by each child httpd process. Once this value is reached, the httpd child process terminates. One of the intentions of this parameter is to limit the lifetime of an httpd client process in order to prevent it from using too much memory resource in case of memory leaks. Set this value to a fairly high number to avoid abrupt termination of transactions. The default value is 10000. Setting this parameter is shown in Example 6-5.

Example 6-5 MaxRequestsPerChild directive in httpd.conf file

```
#
MaxRequestsPerChild 10000
#
```

6.1.6 ListenBacklog

This parameter sets the maximum length of the queue of pending connections from the clients. Generally no tuning is needed or desired. However, on some systems it is desirable to increase this when under a TCP SYN flood attack.

This will often be limited to a smaller number by the operating system. This varies from operating system to operating system. Also note that many operating systems do not use exactly what is specified as the backlog, but use a number based on (but normally larger than) what is set. The default value is 511. Example 6-6 shows how to set its value.

Example 6-6 ListenBacklog directive in httpd.conf file

```
#
ListenBacklog 511
#
```

6.2 Connection

These directives deal with the persistent connection feature of the HTTP/1.1 specification. With HTTP/1.0, each HTTP session establishes a new TCP connection. If your home page has a lot of images, you will need to establish TCP connections many times to send all data for one page. The persistent connection feature is designed to avoid this behavior. After one session is finished, the connection still remains and the next request can re-use the connection. If IHS gets an HTTP/1.1 request, IHS can re-use the connection until it receives the connection close request.

6.2.1 KeepAlive

Whether or not to allow persistent connections (more than one request per connection). Set to "off" to deactivate. Default is on. Example 6-7 shows how to set KeepAlive.

Example 6-7 KeepAlive directive in httpd.conf file

```
#
KeepAlive ON
#
```

6.2.2 KeepAliveTimeout

Number of seconds to wait for the next request. Default value is 15. To avoid waiting too long for the next request, you can specify the number of seconds to wait (Example 6-8). Once the request has been received, the TimeOut directive will apply.

Example 6-8 KeepAliveTimeout directive in httpd.conf file

```
#
KeepAliveTimeout 15
#
```

6.2.3 MaxKeepAliveRequests

The maximum number of requests to allow during a persistent connection. Set to 0 to allow an unlimited number. See Example 6-9.

Example 6-9 MaxKeepAliveRequests directive in httpd.conf file

```
#
MaxKeepAliveRequests 0
#
```

Note: If your Web site is busy, you should set a very small KeepAliveTimeout such as 2 or 3 because if a browser does not send a connection close request, IHS keeps the connection open until the period of time specified in the KeepAliveTimeout directive. If you specify a large number, it blocks the system resources if no requests are submitted.

6.2.4 TimeOut

Sets the number of seconds that IHS waits for these three events:

- Time taken to receive a GET request
- Time taken between receipt of TCP packets on a POST or PUT request
- Time taken between acknowledgments on transmissions of TCP packets in responses

Default value is 300 (Example 6-10).

Example 6-10 TimeOut directive in httpd.conf file

```
#
TimeOut 300
#
```

6.3 Resource Usage

The following directives restrict the amount of system resource usage by httpd child processes. The directives explained in this section are not included in the default httpd.conf configuration file, as they are normally not used because they can be specified at the operating system level, if required.

Note: These directives should only be used when the values need to be set lower than what the operating system permits.

6.3.1 RLimitCPU

This parameter controls the number of seconds per process. This directive takes one or two parameters. The first parameter sets the soft resource limit for all processes and can be specified as a number or “max.” The second parameter can be specified only as “max.” The “max” means the maximum resource limit allowed by the operating system.

6.3.2 RLimitMEM

This parameter sets the number of bytes per process. This directive also takes one or two parameters. The first parameter sets the soft resource limit for all processes and can be set to a number or “max”. The “max” indicates to the server that the limit should be set to the maximum resource limit allowed by the operating system. The second parameter specifies the limit allowed by the operating system.

6.3.3 RLimitNPROC

This parameter controls the maximum number of simultaneous processes per use. This directive also takes one or two parameters. The first parameter sets the soft resource limit for all processes, and the second parameter sets the maximum resource limit similar to the above two directives. For the case of CGI processes running under the same User ID, (UID) as the Web server, which is the normal case, the limitation set with this directive restricts the number of processes the server itself can create by forking. Thus, it might limit a server’s ability to create new httpd processes.

6.4 Name resolution

The following directive controls whether DNS lookups should be performed for each HTTP client connection. The default value is set to off in order to save the network traffic for those sites that don't truly need the reverse lookups done. It is also better for the end users because they don't have to suffer the extra latency that a lookup entails. Heavily loaded sites should leave this directive off because DNS lookups can take considerable amounts of time and adversely effect system performance.

6.4.1 HostnameLookups

Enables or disables DNS lookups to be performed such that host names (rather than IP addresses) can be logged. The values allowed are *On*, *Off*, or *Double*. The value Double refers to doing a double-reverse DNS lookup. That is, after a reverse lookup is performed, a forward lookup is then performed on that result where at least one of the IP addresses in the forward lookup must match the original address. To increase performance, you should set the HostnameLookups directive to Off, which is the default setting (Example 6-11).

Example 6-11 HostnameLookups directive in httpd.conf file

```
#
HostnameLookups Off
#
```

6.5 Effect of using Fast Response Cache Accelerator

Fast Response Cache Accelerator (FRCA) provides a kernel level cache to store static HTML documents and images. However, there are limitations when using FRCA. FRCA can cache only static contents. In other words, dynamic contents generated by servlets, JSPs, and EJBs are not cached by FRCA. FRCA does not support protected pages, POST method, or any pages over SSL connections.

The degree of performance gain brought by FRCA depends on the amount of static HTML pages on your site. If your web site uses a large amount of static contents, the advantages of using FRCA will be substantial. In that case, not only does FRCA help performance by caching the static contents in memory, but also provides a way to avoid using resource-consuming file serving servlets. When most of the pages returned by WCS are dynamic, the use of the FRCA will have little or negligible effect.

By default FRCA is not configured during WCS installation. The following are the basic steps for setting up FRCA for AIX for use with static content.

To use FRCA, you need to install the http_server.frca fileset. Then configure AIX for FRCA. During the AIX FRCA configuration, the Network Buffer Cache options will be requested, which can be obtained by using the **no** command.

Setting Network Buffer Cache options:

- *nbc_limit*

  Sets the maximum size of the network buffer cache in KB. This value should not be set higher than 1/2 the value of **thewall**, which you can get with **no -a** command.

- *nbc_max_cache*

  Sets the maximum size of a cache object that will be allowed in the Network Buffer Cache.

- *nbc_min_cache*

  Sets the minimum size of a cache object that will be allowed in the Network Buffer Cache.

After you set the Network Buffer Cache options, you can load the Fast Response Cache Accelerator into the AIX kernel. The command **frcactrl load** will activate the FRCA kernel. You have to load it *before* IHS is started. By default, IHS is started automatically when AIX is booted. If you do not want to start IHS automatically, you can comment out the line that starts with **ihshttpd** in the /etc/inittab.

To activate the FRCA module, the following lines in Example 6-12 should be added to the IHS configuration file (/usr/HTTPServer/conf/httpd.conf by default):

Example 6-12 LoadModule ibm_afpa_module directive in httpd.conf file

```
#
LoadModule ibm_afpa_module libexec/mod_ibm_afpa.so
AddModule mod_ibm_afpa.c
#
```

Note: The LoadModule and AddModule directives should be the first dynamic modules listed in the configuration.

There are several directives that are related to FRCA:

AfpaBindLogger [-1, 0, 1, ..., n]: allows you to bind the FRCA logging thread to a specific CPU on a multiple processor machine.

AfpaCache on | off: allows you to turn FRCA on or off for a particular scope such as a directory.

AfpaEnable enables the FRCA to listen on the TCP port specified by the Port directive, or the default port 80.

AfpaLogFile file_path_and _name log_format: sets the FRCA log file name, location, and format. The supported log formats are:

- CLF: Command Log Format
- ECLF: Extended Common Log Format
- V-CLF: Common Log Format with virtual host information
- V-ECLF: Extended Common Log Format with virtual host information
- BINARY: Binary Log Format with virtual host information

AfpaSendServerHeader true | false: specifies whether or not FRCA will send the HTTP Server header in the response.

AfpaLogging on | off: turns FRCA logging on or off.

AfpaMaxCache [size]: specifies the maximum file size in bytes that can be added to the FRCA cache.

AfpaMinCache [size]: specifies the minimum file size in bytes which can be added to the FRCA cache.

AfpaRevalidationTimeout [seconds]: sets the time interval for files cached to be revalidated.

In our test environment, we specified the AFPA configuration directives in the /usr/HTTPServer/conf/httpd.conf file. See Example 6-13.

Example 6-13 FRCA configuration settings in httpd.conf file

```
#
LoadModule ibm_afpa_module
AddModule mod_ibm_afpa.c
AfpaEnable
AfpaCache on
AfpaLogFile /usr/HTTPServer/logs/afpa-log V-ECLF
AfpaMinCache 0
AfpaMaxCache 100000
AfpaLogging on
AfpaBindLogger -1
AfpaSendServerHeader true
#
```

Then, start IHS with the **/usr/HTTPServer/bin/apachectl start** command. There are three ways to monitor FRCA:

- Fast Response Cache Accelerator Log:

  FRCA log can be used to observe files being served out of the cache.

- netstat

  By using the command **netstat -c**, you can observe the current status of the Network Buffer Cache.

- frcactrl

  using the command **frcactrl stats** you can observe statistics from the FRCA kernel such as the total number of requests handled and the total number of successful cache hits.

We tested FRCA with two scenarios. One is for HTML static documents and the other is for a dynamic document that is created by a servlet. For the HTML static document test case, we accessed the welcome page, which includes several GIF files. We saw significant performance improvements while running FRCA. In our environment, using FRCA allowed twice as many HTTP requests than without FRCA. In addition the CPU utilization of the Web server machine was lower than the non-FRCA case. With the **frcactrl stats** command, we noticed that our HTTP requests hit the cached data.

The other test case for dynamic content showed that using FRCA did not improve performance. From this test result, we can tell that FRCA does not cause any performance increase for dynamic content. We also observed that FRCA would not negatively affect the performance of dynamic contents.

A

Performance Monitoring Tools

This appendix provides summary descriptions of a set of tools that provide varying degrees of assistance in determining the performance of WCS, in addition to highlighting possible issues related to the resources used by WCS that have some impact on runtime performance.

All the tools described have already been documented in other IBM publications, and references to these publications are provided accordingly as part of each tool's description.

The tools discussed are as follows:

- WebSphere Commerce Suite Performance Monitor
- WAS Resource Analyzer
- WebSphere Site Analyzer
- Page Detailer
- AIX Tools
- DB2 Tools
- Silk Preview

WCS Performance Monitor

The WCS Performance Monitor is installed as part of the WCS Administration subsystem. The information analyzed is a subset of the performance data generated by WAS as part of the internal instrumentation of WAS components, and includes the following data points:

- The number of occurrences of a given task activity
- The duration of the executing task activity
- The maximum and minimum times for task execution
- The average time measured by the sum of the squares for the values returned
- A measure of the standard deviation of the values returned for task measurements
- Identification of access counts by store
- The last access time
- The last response time

The WCS Performance Monitor provides the option of printing the generated results or saving the information in an HTML formatted file for viewing in a web browser.

The WCS Performance Monitor is not enabled by default. To use it, follow the below steps.

1. Invoke WCS configuration manager. Then select **`Instance List`** -> **<instance name>** -> **`Components`** -> **`PerfMonitor`**, select **`Enable Component`**, and select **`Apply`** as shown in Figure A-1 on page 139.

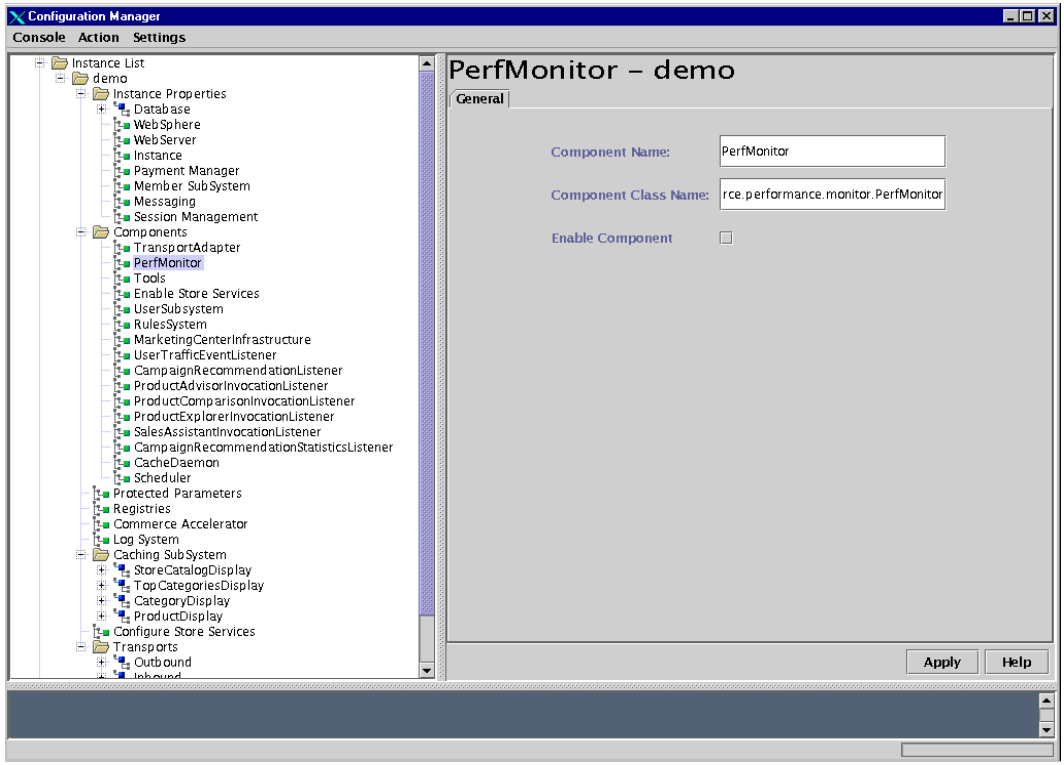


Figure A-1 Enabling WCS PerfMonitor Component

2. From the WebSphere Application Server Administration Console, stop and re-start the instance.
3. Open the Commerce Suite Administration Console by pointing your browser to `http://<hostname>/adminconsole` as shown in Figure A-2 on page 140. The default logon id and password is set to wcsadmin. You may need to change the password when you first log on.

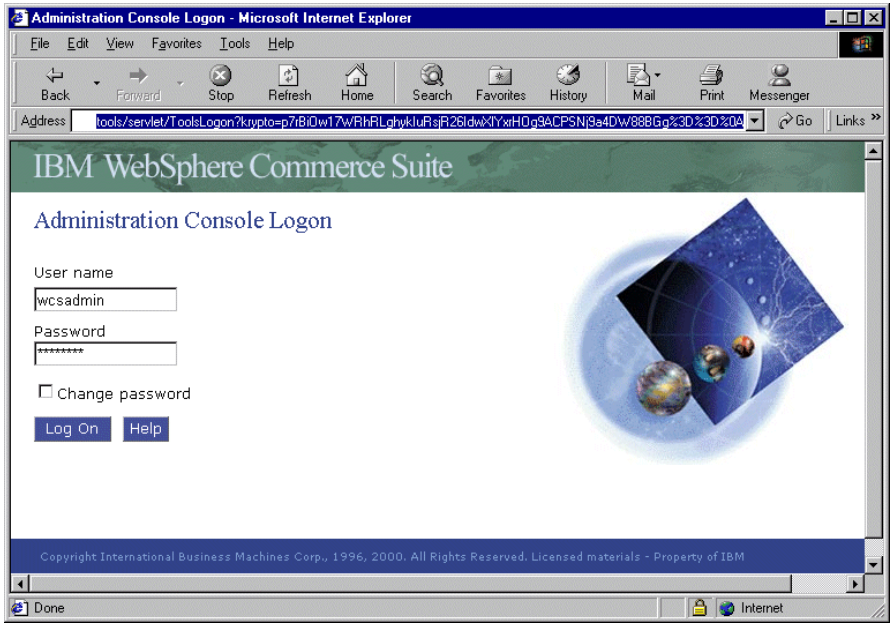


Figure A-2 Logging on to WCS admin console

4. Click **Site** -> **Performance** -> **Statistics** in the menu bar. The Performance Monitor window will pop up (Figure A-3 on page 141). Click **Start monitor**.

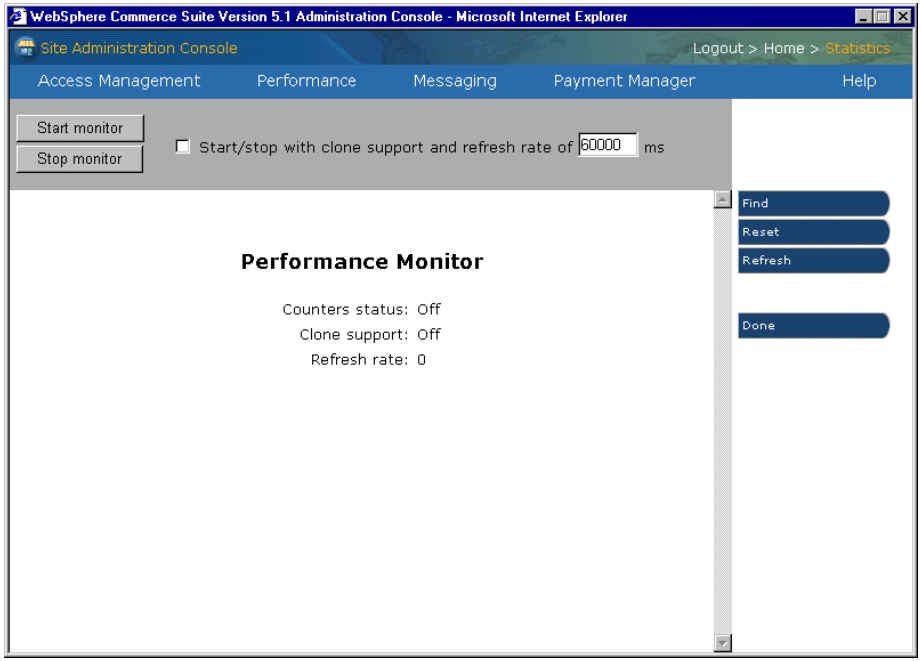


Figure A-3 Starting WCS Performance Monitor

5. The performance values will be collected during your test. An example screen is shown in Figure A-4 on page 142. Click **Refresh** to view updated data. You can also specify refresh interval.

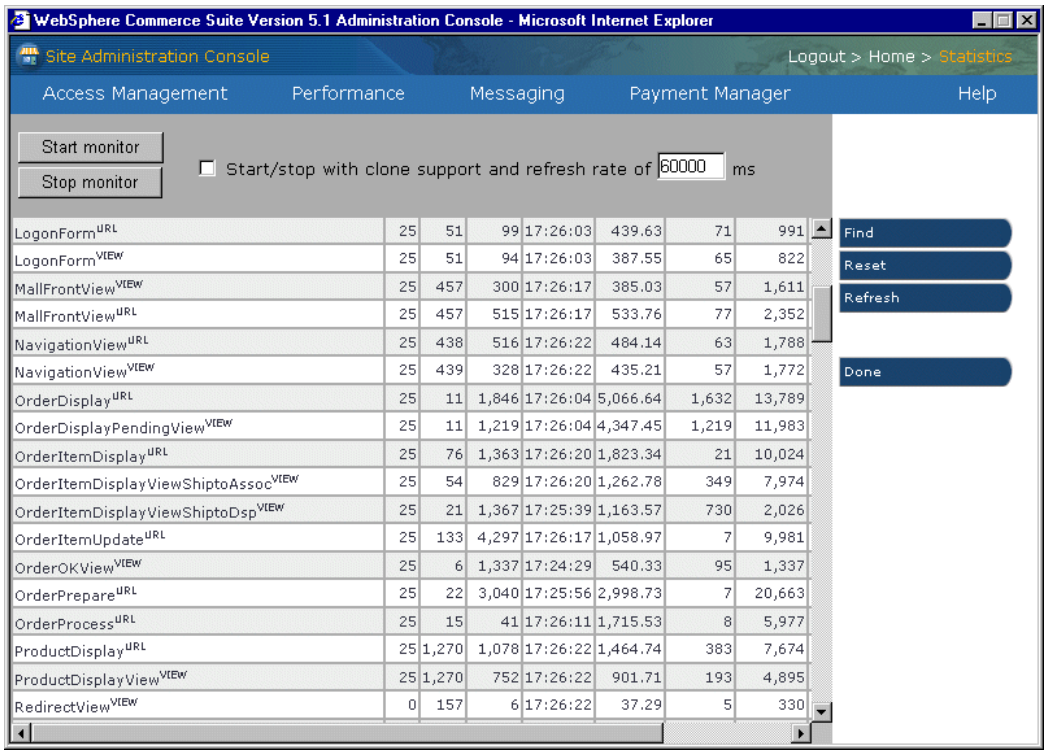


Figure A-4 Sample output of WCS Performance Monitor

For further information on the WCS Performance Monitor, refer to chapter 8 of *IBM WebSphere CommerceSuite Site Administrator Version 5.1*.

WAS Resource Analyzer

The Resource Analyzer is a stand-alone performance monitor for WebSphere Application Server Advanced Edition. It is available as a technology preview product, and is not included in the base WebSphere Application Server product CD. The download link URL for obtaining the WAS Resource Analyzer is:

```
http://www-4.ibm.com/software/webservers/appserv/download_ra.html
```

Because Resource Analyzer is a pure-Java based client, the same code can be run on NT as well as Unix platforms. And it can connect across platforms to the WebSphere Administrative Server running locally or on a remote machine. The Resource Analyzer retrieves performance data by periodically polling the WebSphere Administrative Server. The collected information provides the basis for the statistics analysis used by the WCS Performance Monitor.

The Analyzer collects and reports performance data for the following resources of WebSphere Application Server:

- WebSphere runtime

  Reports memory used by a process as reported by the JVM. Examples are the total memory available and the amount of free memory for the JVM.

- Object Request Broker (ORB) thread pools

  Reports information about the pool of threads an application server uses to process remote methods. Examples are the number of threads created and destroyed, the maximum number of pooled threads allowed, and the average number of active threads in the pool.

- Database connection pools

  Reports usage information about connection pools for a database. Examples are the average size of the connection pool (number of connections), the average number of threads waiting for a connection, the average wait time in milliseconds for a connection to be granted, and the average time the connection was in use.

- Enterprise beans

  Reports load values, response times, and lifecycle activities for enterprise beans. Examples include the average number of active beans and the average number of methods being processed concurrently.

- Enterprise bean methods

  Reports information about an enterprise bean's remote interfaces. Examples include the number of times a method was called and the average response time for the method.

- Enterprise bean object pools

  Reports information on the size and usage of a cache of bean objects. Examples include the number of calls attempting to retrieve an object from a pool and the number of times an object was found available in the pool.

- Transactions

  Reports transaction information for the container. Examples include the average number of active transactions, the average duration of transactions, and the average number of methods per transaction.

- Servlet engines

  Reports usage information for Web applications, servlets, JavaServer Pages (JSPs), and HTTP sessions. Examples include the average number of concurrent requests for a servlet, the amount of time it takes for a servlet to perform a request, the number of loaded servlets in a Web application, and the average number of concurrently active HTTP sessions.

Figure A-5 shows the monitoring screen for Resource Analyzer.

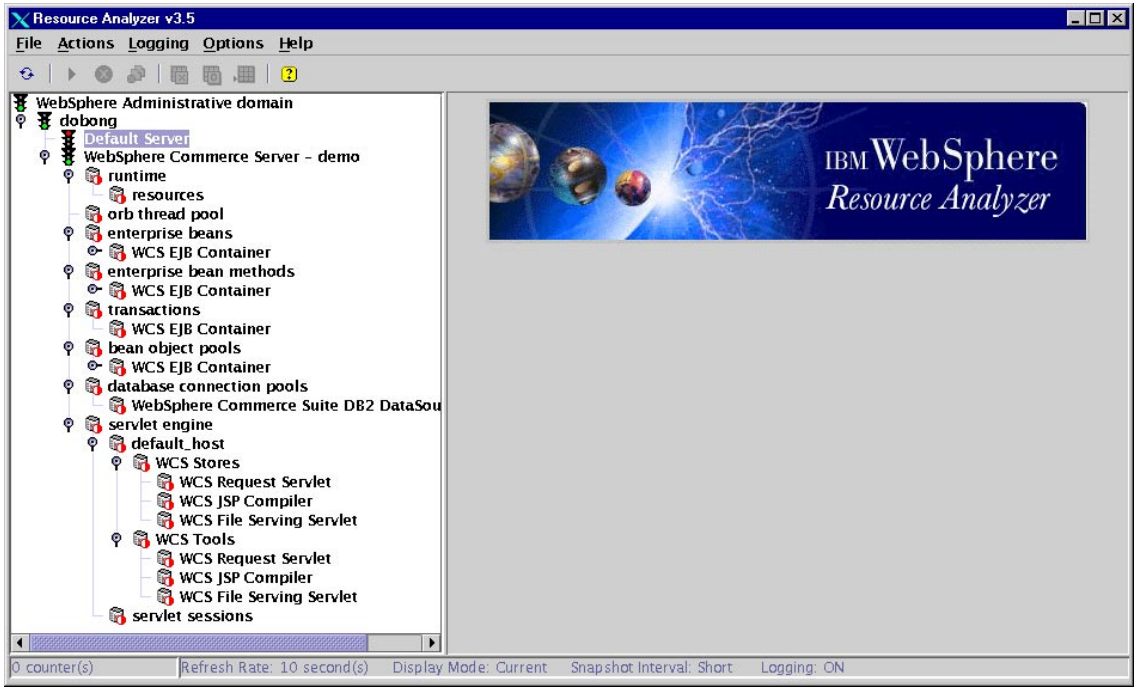


Figure A-5 Resource Analyzer component measurements

For further information on the Resource Analyzer, refer to chapter 25 of *WebSphere Commerce Suite V5.1 Handbook,* SG24-6167.

WebSphere Site Analyzer

WebSphere Site Analyzer uses Web server logs to analyze site activity. From these logs it can determine who the users were, what pages and resources were accessed, error codes received, and other useful information. There are several log formats being used by various web servers, but NCSA Combined log format should be used in case IBM HTTP Server is being used in conjunction with Site Analyzer.

Site Analyzer provides the following features:

- Commerce analysis features

  Site Analyzer provides commerce related analysis by reading the commerce data from the commerce server database. Using Site Analyzer, you can respond to the following questions:

  - Which products did users most often add, or least often add to their shopping cart?
  - Which products were most often viewed, and which products were least often viewed?
  - Which products were looked at most by a single user?

- Operational analysis features

  Site Analyzer will scan your web site and report which links are broken and why they are broken. It can also report the response time required to retrieve each resource, (e.g. page, image, etc.). Site Analyzer will visually show you your web site's hierarchical structure through a graphic tree view representation.

- Business analysis features

  Site Analyzer can show you the top IP address accessing your site, or given an IP address, what pages the visitor accessed. Site Analyzer can categorize visitor sessions by domain or subdomain, allowing you to see, for example, if your users are coming from .edu, .com, .org, etc. Site Analyzer can also tell you what browsers and what platforms your visitors are using. Site Analyzer also keeps track of the referring web site for each of your visitors so you can see which advertisements are generating the most traffic to your site.

- Reporting features

  Within Site Analyzer, you can establish categories for data based on patterns. For example, you may categorize a subset of your site's pages into a 'new products' category, then report on the number of hits on your 'new products'. Secondly, if you want to report on page return codes by browser or platform type, you can aggregate return code, browser, and platform together by using Site Analyzer. Thirdly, the tool provides various forms of graphs to display your data, such as pies, bars, lines, in 2D or 3D.

- Traffic analysis features

Because we are interested in how Site Analyzer can be used for performance tuning, we will limit the discussion to Traffic analysis features.

Traffic analysis

This function processes the HTTP logs of all major web servers, providing measurements such as the number of visits, number of pages viewed, and total hits to the site. Traffic analysis also provides significant measurements in the area of customer analysis such as where visitors come from (top domains and referrals), what they do while on the site, and what browser they are running. Using the default set of data options, administrators can collect information such as number of page views, most commonly used browsers, most commonly used platforms, top referring sites, and number of visits.

- Visitor / Session Analysis

  Site Analyzer employs a sophisticated algorithm to group hits together to calculate visitor sessions. Criteria include date/time, IP address, cookie, referral URL, user agent, etc. As a result, Site Analyzer's session analysis is more accurate than common standards because it uses more criteria.

- Dynamic Web Site Support

  Site Analyzer allows you to track activity to your dynamic web pages based on different parameter values to pages served dynamically by server side programs such as servlets.

- Trending Analysis

  With Site Analyzer, you can accumulate web traffic data over time. You can then report on traffic measurement trends, such as by hour of the day or by day of the week.

- Advanced Filtering

  During processing of a web site's traffic log, you may wish to exclude specific records, or include only certain records based on criteria such as IP address, URL pattern, user agent (browser, platform), status code, referral URL pattern, cookies, etc. Site Analyzer contains advanced filtering support allowing lists of multiple wild card pattern matches to exclude or include records.

- Server Cluster Support

  Site Analyzer supports server clusters by analyzing the servers' multiple log files at once, or one at a time by merging the data together to create a complete usage history of your web site.

Integration with WCS

Site Analyzer provides a tool that can extract data from the Commerce Suite database and create a log that can then be analyzed. Currently this tool only supports WebSphere Commerce Suite V4.1, but a fix code for WCS V5 is in test and will be available soon on `http://www-4.ibm.com/software/webservers/siteanalyzer/efix.html`. There is also a limitation in terms of supported database. Even though WCS can be installed with either DB2 or Oracle, Site Analyzer only supports analyzing data extracted from DB2.

For more information about WebSphere Site Analyzer, refer to *Up and Running with WebSphere Site Analyzer*, SG24-6169.

Page Detailer

Page Detailer monitors the end user's interactions as pages are requested and retrieved, reporting:

- Overall response time
- Detailed timing broken down into individual page components
- Size of individual page components
- Identity of each item on a page

The details revealed can be used to identify areas where performance could be improved to enhance the end user experience.

Page Detailer is shipped as a part of IBM WebSphere Studio Advanced Edition. WebSphere Studio can be installed on Windows 95, Windows 98, Windows NT Workstation or Server V4.0, or Windows 2000 Professional or Server or Advanced Server. But Page Detailer can be run independently of the other components of WebSphere Studio.

The design of the communication protocols used by browsers for Web access creates times when either the Web browser or the Web server must wait for responses from other components. The more time spent in these protocol waits, the longer the delay site visitors may experience while waiting for page content. The 'farther' the browser is from the server, the greater the likelihood of delays due to intermediate links or devices in the path between the browser and the server. A delay could be added by any hardware or software component, including components or subsystems of the browser or the server itself. Page Detailer is an excellent tool for revealing these delays.

The following shows a brief walkthrough of Page Detailer. For more information about Page Detailer, refer to `http://www-4.ibm.com/software/webservers/studio/doc/v35/pagedetailer/EN/HTML/index.html`

To start the Page Detailer, click **Programs** -> **IBM WebSphere** -> **Studio 3.5** on the system that has WebSphere Studio. You will be presented with a Page Detailer window as shown in Figure A-6.

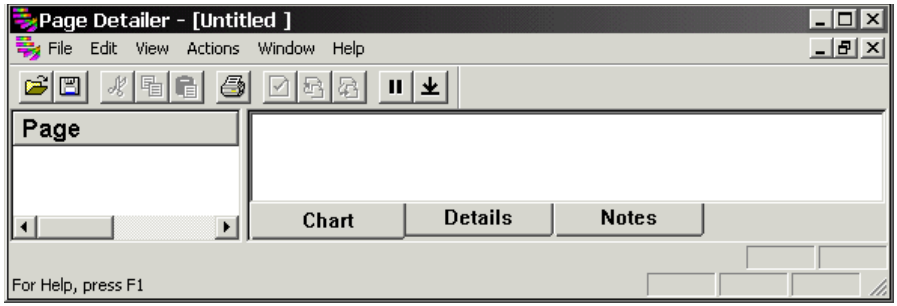


Figure A-6 Page Detailer window

Start a browser and point to any Web page, for example `http://www.ibm.com`. The Page Detailer automatically captures the timing of all the downloaded files and produces a graphical picture of the operation as shown in Figure A-7 on page 149.

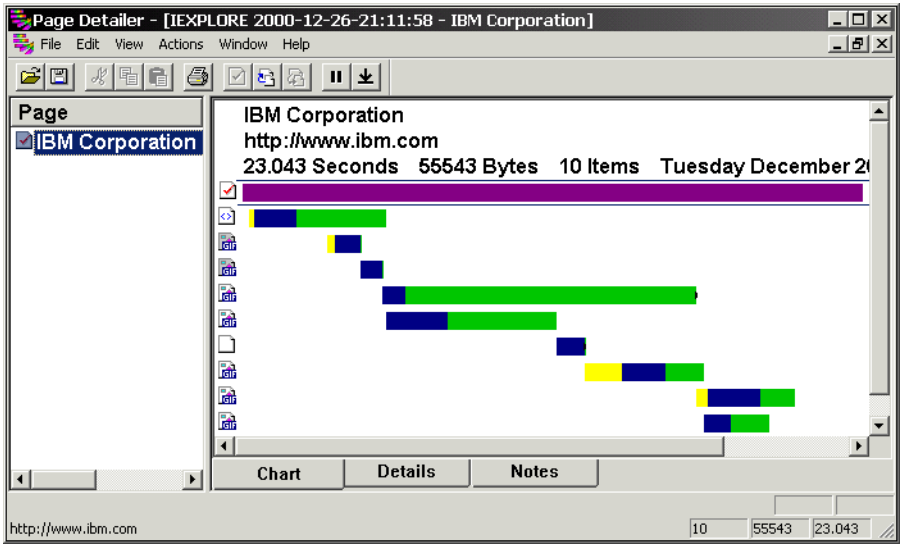


Figure A-7 Page Detailer result for http://www.ibm.com

Legend

Select **View** -> **Legend** to see an explanation of the color coding and the icons for each measured metric. A sample legend is shown in Figure A-8.

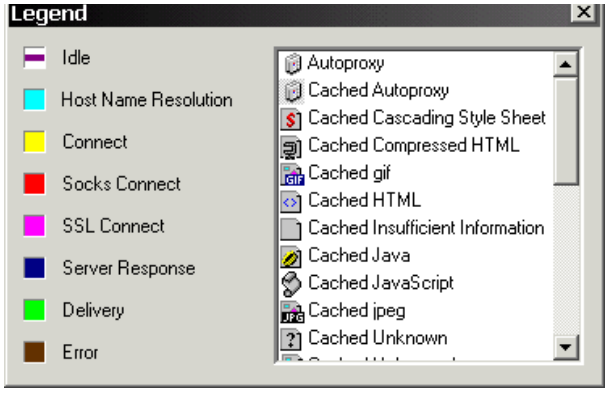


Figure A-8 Page Detailer Legend

Detailed analysis

Click on **Details** to see the download time and size of each file of the Web page. A sample output is shown in Figure A-9 on page 150.

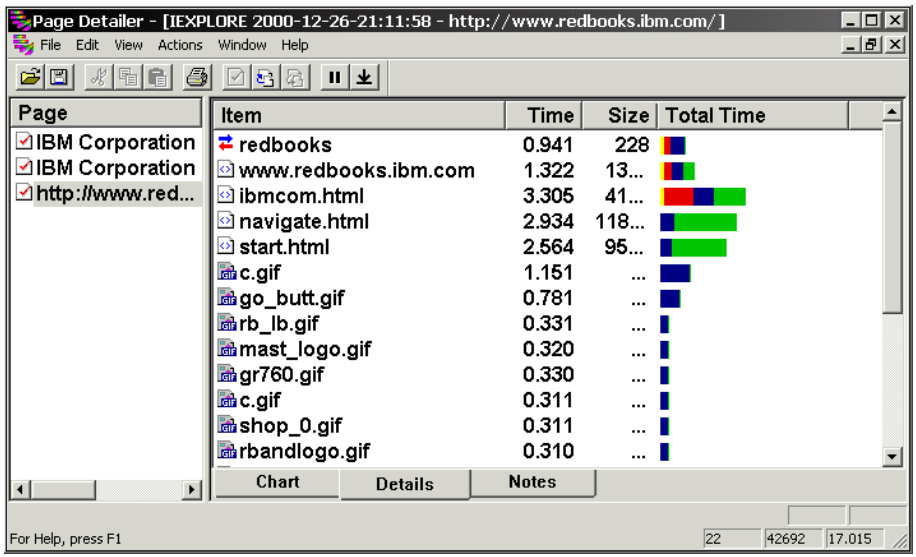


Figure A-9 Page Detailer details

You can see a summary, the totals, and the events using **View** -> **Properties**. You can save the result of the analysis as a .PDE file.

Export

The chart data can be exported as an XML file, a text file, or a comma separated value (CSV) file. CSV files can be imported into spreadsheet programs. The result files are stored under the D:\WebSphere\Studio35\PageDetailer with names:

pd_export.xml

pd_export.txt

pd_page.csv, pd_item.csv, pd_header.csv, pd_event.csv

The export directory can be tailored using **Edit** -> **Preferences**.

AIX monitoring tools

Restriction: The performance monitoring tools and techniques are applicable only to AIX.

The majority of performance problems related to application execution within a given platform system environment falls into four main categories:

- Network
- CPU
- Disk I/O
- Memory

Tools to monitor general system performance metrics

There are two utilities that could be used to display overall system performance metrics.

topas

The **topas** command reports selected statistics about the activity on the local system. topas is a utility provided by IBM, and is available on the AIX CD.

Important: Install the perfagent.tools fileset on your AIX system to use topas.

The top two lines at the left of the output show the name of the system the topas program runs on, the date and time of the last observation, and the monitoring interval. Following this is a section that lists the CPU utilization in both numeric and block-graph format. The second part contains five subsections of statistics, EVENTS/QUEUES, FILE/TTY, PAGING, MEMORY, and PAGING SPACE. The variable part of the topas display can have one, two, or three subsections. If more than one appears, the subsections are always shown in the following order:

- Network interfaces
- Physical disks
- Processes

While the topas program is running, it accepts one-character subcommands. Each time the monitoring interval elapses, the program checks for one of the following subcommands and responds to the action requested.

a Show all of the variable sections (network, disk, and process) if space allows.

d Show disk information.

h Show the same help screen as displayed by the **-h** command line argument.

n Show network interfaces information.

p Show process information.

q Quit the program.

A sample output of **topas** is shown in Example A-6 on page 159.

monitor

The **monitor** utility provides an ASCII display of a collection of system performance indicators that is a composite of information provided from other tools such as **vmstat**, **iostat**, and **netstat**. It is provided as a freeware on the Internet. It can be downloaded from `ftp://aixpdslib.seas.ucla.edu/pub/monitor/RISC/4.3/exec/monitor.2.1.5.tar.Z`.

The monitor also includes the top utility, which provides a rolling display of top CPU-using processes on a Unix system. It also displays other information about the overall health of the system, including load averages and memory utilization.

Example: A-1 Sample output of monitor

```
AIX System monitor v2.1.5 13aug1998: wcs5.itsc.austin.iSun Jun  3 17:04:17 2001
Uptime:  30 days, 22:23   Users:   0 of   0 active 0 remote 00:00 sleep time
CPU: User  0.1% Sys  1.2% Wait  0.0% Idle 98.7%   Refresh:  6.40 s
0%             25%              50%               75%              100%


Runnable (Swap-in) processes  0.00 (0.94)  load average:  0.02,  0.02,  0.02

Memory    Real     Virtual    Paging (4kB)    Process events     File/TTY-IO
free     2048 MB    204 MB       3.6 pgfaults      32 pswitch       0 iget
procs  16774165 MB      3 MB     0.0 pgin          36 syscall       1 namei
files    2026 MB                 0.0 pgout          2 read          0 dirblk
total    1024 MB    208 MB       0.0 pgsin          0 write       916 readch
IO (kB/s) read  write busy%      0.0 pgsout         0 fork         48 writech
hdisk0     0.0    0.0    0                         0 exec          0 ttyrawch
hdisk1     0.0    0.0    0                         0 rcvint        0 ttycanch
```

```
cd0        0.0    0.0    0                          0 xmtint       48 ttyoutch
                                                     0 mdmint

                                                       Netw   read  write kB/s
                                                        tr0     0.1     0.1
                                                        lo0     0.0     0.0
   PID USER     PRI NICE SIZE   RES STAT     TIME  CPU%     COMMAND
   774 root     127  21    8k 7796k  run 24+22:01 49.9/80.6 Kernel (wait)
```

Tools to monitor network

An easy way to tell if the network is affecting overall performance is to compare those operations that involve the network with those that do not. If you are running a program that does a considerable amount of remote reads and writes and it is running slowly, but everything else seems to be running normally, then it is probably a network problem.

netstat -m

netstat -m displays the statistics recorded by the mbuf memory-management routines. Example A-2 shows sample output.

- What to watch

  Check if there is any non-zero values in the *requests for mbufs denied* section and in the *failed* column for each CPU. These values should be zero.

- Action

  Increase the value of thewall parameter by:

  # no -o thewall=New_value

  If the requests for mbufs denied is not displayed, enable the statistics by:

  # no -o extendednetstats=1

Example: A-2 Sample output of netstat -m

```
# netstat -m
29 mbufs in use:
16 mbuf cluster pages in use
71 Kbytes allocated to mbufs
0 requests for mbufs denied
0 calls to protocol drain routines

Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed   delayed    free   hiwat   freed
```

```
32              245    182806      0         0      523    1226       0
64               95      7970      0         0       33     613       0
128             210     15974      0         0       78     306       0
256             241  12306567      0         0      447     736       0
512             100    934557      0         0       36      76       5
1024             83    398269      0         0      185     191    3201
2048              0    242008      0         0      190     191     487
4096             67   1847877      0         0      102     230       0
8192              5     68407      0         0        0      19       0
16384             1     70479      0         0       34      46       0
65536             1         1      0         0        0    1535       0

By type       inuse   calls failed delayed  memuse  memmax  mapb

Streams mblk statistic failures:
0 high priority mblk failures
0 medium priority mblk failures
0 low priority mblk failures
```

nestat -i

Use the **netstat -i** command to monitor the status of your network interface during the experiment. You may find that the system is dropping packets during a send by looking at the Oerrs output.

- What to watch:

1. If the number of errors during input packets is greater than 1 percent of the total number of input packets (from the command **netstat -i**); that is,

   Ierrs > 0.01 x Ipkts

   Then run the **netstat -m** command to check for a lack of memory. Verify the check points described in Section "netstat -m" on page 153. Increase the size of thewall when necessary.

2. If the number of errors during output packets is greater than 1 percent of the total number of output packets (from the command netstat -i); that is,

   Oerrs > 0.01 x Opkts

   Then increase the transmit queue size (xmt_que_size) for that interface. The size of the xmt_que_size could be checked with the following command:

   lsattr -El ent*n* (or tok*n* where n=0,1,2, and so on)

   Use the following command to alter the queue size:

   chdev -l ent0 -a xmt_que_size=<new_value>

Note: The changed network interface must be down and detached with TCPIP services stopped. Alternatively, add the -P flag to update the database only and reboot for the changes to take effect.

3. If the collision rate is greater than 10 percent, that is,

   Coll / Opkts > 0.1

   Then there is a high network utilization, and a reorganization or partitioning of the network may be necessary. Use the **`netstat -v`** command to determine the collision rate.

netstat -v

The netstat -v command displays the statistics for each network interface-specific device driver that is in operation. Every interface has its own specific information and some general information.

- What to watch:

  Check for non-zero values in any error counts in the netstat -v output.

- Action:
  - Max Packets on S/W Transmit Queue

    Maximum number of outgoing packets ever queued to the software transmit queue. An indication of an inadequate queue size is if the maximum transmits queued equals the current queue size (xmt_que_size). Check the current size of the queue, using **`lsattr -El <adapter_name>`** command, then increase the size of xmt_que-size using **`chdev -l <adapter_name> -a xmt_que_size=<new_value>`** command.

  - No mbuf Errors

    Number of times that mbufs were not available to the device driver. If the mbuf pool for the requested size is empty, the packet will be discarded. Increase the parameter thewall by **`no -o thewall = <new_value>`**.

Example A-3 shows the Token-Ring part of the **`netstat -v`** command. The most important output fields are highlighted.

Example: A-3 Sample output of netstat -v

```
TOKEN-RING STATISTICS (tok0) :
Device Type: IBM PCI Tokenring Adapter (14103e00)
Hardware Address: 00:20:35:7a:12:8a
Elapsed Time: 29 days 18 hours 3 minutes 47 seconds
Transmit Statistics:                          Receive Statistics:
```

```
--------------------                              -------------------
Packets: 1355364                                  Packets: 55782254
Bytes: 791555422                                  Bytes: 6679991641
Interrupts: 902315                                Interrupts: 55782192
Transmit Errors: 0                                Receive Errors: 1
Packets Dropped: 0                                Packets Dropped: 0
                                                  Bad Packets: 0
Max Packets on S/W Transmit Queue: 182
S/W Transmit Queue Overflow: 42
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 18878                          Broadcast Packets: 54615793
Multicast Packets: 0                              Multicast Packets: 569
Timeout Errors: 0                                 Receive Congestion Errors: 0
Current SW Transmit Queue Length: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0                                 Lobe Wire Faults: 0
Abort Errors: 2                                   AC Errors: 0
Burst Errors: 0                                   Frame Copy Errors: 0
Frequency Errors: 0                               Hard Errors: 0
Internal Errors: 0                                Line Errors: 0
Lost Frame Errors: 0                              Only Station: 0
Token Errors: 0                                   Remove Received: 0
Ring Recovered: 0                                 Signal Loss Errors: 0
Soft Errors: 2                                    Transmit Beacon Errors: 0
Driver Flags: Up Broadcast Running
   AlternateAddress 64BitSupport ReceiveFunctionalAddr
   16 Mbps
```

TCP/UDP Message Buffers

AIX has four parameters that specify the system default socket buffer sizes for sending or receiving data. They are tcp_sendspace, tcp_recvspace, udp_sendspace, and udp_recvspace. They affect the window sizes used by TCP or UDP. In general, increasing those socket buffer sizes improves performance over Standard Ethernet and Token-Ring networks. However, lower bandwidth networks, such as Serial Line Internet Protocol (SLIP), or higher bandwidth networks, such as Serial Optical Link, should have different optimum buffer sizes. The optimum buffer size is the product of the media bandwidth and the average round-trip time of a packet.

- What to watch

If data transfers appear sluggish, determine which protocol is being used and increase either the TCP or UDP (broadcast) message buffers accordingly. The defaults for these settings are shown in Example A-4 (derived from an AIX 4.3.3 system).

Example: A-4 Using no -a

```
no -a | grep space

tcp_sendspace = 16384
tcp_recvspace = 16384
udp_sendspace = 9216
udp_recvspace = 41920
no -a | grep space
```

- Action

  Increase the appropriate buffer size by:

  no -o <option_name> = <new_value>

CPU tuning

ps

The **ps** command is used to locate the processes dominating CPU usage. The command writes the current status of active processes and (if the -m flag is given) associated kernel threads to standard output. The command **ps aux** provides a set of resource utilization metrics including the utilization ratio of CPU and memory.

- What to watch

%CPU	The percentage of time the process has used the CPU since the process started. The value is computed by dividing the time the process uses the CPU by the elapsed time of the process.
%MEM	The percentage of real memory used by this process
COMMAND	name of the command (process)

- Action

  Identify which process is using most of CPU time. Determine whether this usage pattern is valid. For WAS and DB2, consider decreasing the number of agent processes.

A sample output from the ps utility is shown in Example A-5.

Example: A-5 Output from **ps aux** command

```
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
root       516 49.0  1.0    8 9952 - A    18:11:13 831:12 kproc
root       774 48.8  1.0    8 9952      - A    18:11:13 826:36 kproc
root     15506  1.3 22.0 167584 166176  pts/0 A    18:25:52 22:21
/usr/IBMWebAS/jav
root     21682  0.2  5.0 41460 41640  pts/1 A    18:29:10  2:46
/usr/IBMWebAS/jav
nobody   11166  0.2  0.0 3588 1216      - A    07:48:23  0:06 /usr/HTTPServer/b
db2inst1 20640  0.1  0.0 2852 2712      - A    18:26:42  1:09 db2agent (RMALL)
db2inst1 18062  0.1  0.0 2872 2748      - A    18:25:07  1:06 db2agent (RMALL)
db2inst1 20900  0.1  0.0 2900 2752      - A    18:28:17  0:57 db2agent (RMALL)
root         0  0.0  1.0   12 9956      - A    18:11:13  0:21 swapper
root      1548  0.0  1.0   64 10008      - A    18:11:13  0:21 kproc
root      3396  0.0  0.0  164  188      - A    18:12:48  0:17 /usr/sbin/syncd 6
nobody    7998  0.0  0.0 3468 1216      - A    18:13:20  0:11 /usr/HTTPServer/b
nobody    8514  0.0  0.0 3508 1216      - A    18:13:20  0:11 /usr/HTTPServer/b
nobody    7740  0.0  0.0 3452 1220      - A    18:13:20  0:11 /usr/HTTPServer/b
nobody   18588  0.0  0.0 3568 1216      - A    18:28:09  0:10 /usr/HTTPServer/b
nobody    7486  0.0  0.0 3428 1216      - A    18:13:20  0:10 /usr/HTTPServer/b
nobody    8256  0.0  0.0 3488 1216      - A    18:13:20  0:10 /usr/HTTPServer/b
```

Note that the sequence shown in the listing generated is not presented in any particular order, so examination of the entire output is recommended to ensure all process entries of a given application are accounted for.

Note: It is normal to see a process named kproc (PID of 516 in operating system version 4) using CPU time. When there is no task to run during a time slice, the scheduler of AIX assigns the CPU for that time slice to this kernel process (kproc), which is known as the idle or wait kproc. SMP systems will have an idle kproc for each processor.

topas -p

topas can be used to list currently running processes in the order of %CPU utilization. For example, **topas -p 10** will list the top 10 processes in the order of high CPU utilization as well as other performance metrics. A sample output of this example is shown in Example A-6 on page 159.

- What to watch

  The boldfaced column in Example A-6 shows %CPU value and the list is arranged the order of highest %CPU value.

Example: A-6 Sample output of **topas -p**

```
Topas Monitor for host:    wcs5                 EVENTS/QUEUES    FILE/TTY
Sun Jun  3 16:42:0 2001   Interval:  2         Cswitch      3  Readch     1505
                                                Syscall      4  Writech
Kernel    0.2   |                           |  Reads          4  Rawin          0
User      0.0   |                           |  Writes         0  Ttyout
Wait      0.0   |                           |  Forks          0  Igets          0
Idle     99.7   |###########################|  Execs          0  Namei          0
                                                Runqueue    0.0  Dirblk         0
Interf   KBPS   I-Pack  O-Pack   KB-In  KB-Out  Waitqueue   1.0
tr0       0.0     0.4     0.4     0.0     0.0
lo0       0.0     0.0     0.0     0.0     0.0  PAGING           MEMORY
                                                Faults          0  Real,MB    1023
                                                Steals          0  % Comp     57.8
topas    (17820)  0.5% PgSp: 0.4mb root         PgspIn          0  % Noncomp   9.1
syncd    (2886)   0.0% PgSp: 0.1mb root         PgspOut         0  % Client    0.5
java     (23778)  0.0% PgSp:227.0m: root        PageIn          0
httpd    (2772)   0.0% PgSp: 3.7mb root         PageOut         0  PAGING SPACE
sendmail (5578)   0.0% PgSp: 0.7mb root         Sios            0  Size,MB    2048
gil      (1548)   0.0% PgSp: 0.0mb root                            % Used      1.5
httpd    (35382)  0.0% PgSp: 4.5mb nobody                          % Free     98.4
java     (18278)  0.0% PgSp:55.0mb root
cron     (6234)   0.0% PgSp: 0.3mb root
ksh      (11304)  0.0% PgSp: 0.3mb root            Press "h" for help screen.
                                                   Press "q" to quit program.
```

sar

sar (System Activity Report) command can be used in two ways to collect data: one is to view system data in real time, and the other is to view data previously captured. The **sar** utility is helpful in determining whether the system is constrained by CPU or disk I/O. It also provides whether the system is balancing the workload evenly across all CPUs on an SMP system. Usage of **sar** is also recommended when you need to collect history of system usage over a period of time.

- What to watch

  The following metrics are important to determine whether the workload is CPU-bound or I/O-bound. If the sum of %usr and %sys is large, say greater than 70%, the workload is CPU-bound. If %wio is high while %usr and %sys remain low, the workload is I/O-bound.

  %idle — Reports the percentage of time the CPU or CPUs were idle with no outstanding disk I/O requests.

%sys	Reports the percentage of time the CPU or CPUs spent in execution at the system (or kernel) level.
%usr	Reports the percentage of time the CPU or CPUs spent in execution at the user (or application) level.
%wio	Reports the percentage of time the CPU or CPUs were idle waiting for disk I/O to complete. For system-wide statistics, this value may be slightly inflated if several processors are idling at the same time, an unusual occurrence.

Example A-7 shows **sar** output for a machine with four CPUs, with a sampling interval of 2 seconds for 3 intervals.

Example: A-7 Sample output of sar

```
# sar -P 0,1,2,3  2  3

AIX wcs5 3 4 000376384C00    03/14/01

11:58:30     cpu    %usr    %sys    %wio   %idle
11:58:32     0      0       0       0      100
             1      0       0       0      100
             2      0       0       0      100
             3      0       0       0      100
11:58:34     0      0       0       0      100
             1      0       0       0      100
             2      0       0       0      100
             3      0       0       0      100
11:58:36     0      0       0       0      100
             1      0       0       0      100
             2      0       0       0      100
             3      0       0       0      100

Average      0      0       0       0      100
             1      0       0       0      100
             2      0       0       0      100
             3      0       0       0      100
```

It should be noted that **sar** does introduce additional load on the system that could compound an existing performance problem. Therefore, the **sar** results should not be used as the only reference point in determining CPU utilization.

Disk I/O

iostat

When you suspect a disk I/O performance problem, use the **iostat** command. The **iostat** command will break down which disks are being utilized most based on the time the physical disks are active relative to the average transfer rates.

- What to watch

  % tm_act — Indicates the percentage of time that the physical disk was active. The "disk active time" percentage is directly proportional to resource contention and inversely proportional to performance. In general, when the utilization exceeds 70 percent, processes are waiting longer than necessary for I/O to complete because most UNIX processes block (or sleep) while waiting for their I/O requests to complete.

  Kbps — Indicates the amount of data transferred (read or written) to the drive in KB per second. This is the sum of Kb_read plus Kb_wrtn, divided by the seconds in the reporting interval.

  tps — Indicates the number of I/O transfers per second that were issued to the physical disk. A transfer is of indeterminate size.

  Kb_read — Reports the total data (in KB) read from the physical volume during the measured interval.

  Kb_wrtn — Reports the total data (in KB) written from the physical volume during the measured interval.

  Taken alone, there is no unacceptable value for any of the above fields. Therefore, when you are evaluating data, look for patterns and relationships. The most common relationship is between disk utilization (%tm_act) and data transfer rate (tps).

- Action

  Identify the busiest disk drives. Then analyze which filesystem or file contained on the disk was most active. The filemon utility explained in Section “filemon” on page 162 can help to find out this information. **lspv -l <hdisk_name>** also displays the list of logical volumes on the drive. Look for busy versus idle drives. Moving data from busy to idle drives can help alleviate a disk bottleneck.

An example of the output is shown in Example A-8.

Example: A-8 Sample output of iostat

```
# iostat 3 3

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.2         75.7                5.2      0.7       93.6       0.5

Disks:       % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk1          0.0       0.0       0.0          0         0
hdisk0          1.1      12.2       1.7      62897   2334816
cd0             0.0       0.0       0.0       6544         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0         39.0                0.0      0.0      100.0       0.0

Disks:       % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk1          0.0       0.0       0.0          0         0
hdisk0          0.0       0.0       0.0          0         0
cd0             0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0         39.0                0.0      0.0      100.0       0.0

Disks:       % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk1          0.0       0.0       0.0          0         0
hdisk0          0.0       0.0       0.0          0         0
cd0             0.0       0.0       0.0          0         0
```

In reading the information report generated, ignore the first collection of statistics (highlighted in the above example), as this reflects the statistics since the system was last rebooted.

filemon

Once the disks experiencing maximum utilization have been identified, then determine the most active files and most active disks. The filemon utility can help you break down which logical volume is most active as well as which files are most active. Invoke it from the command line as follows:

```
filemon -o filemon.out
```

where `filemon.out` will contain the output trace information consisting of a summary of the most active physical and logical volumes. To use filemon, install perfagent.tools fileset on your system.

After waiting for an interval of time, the trace should be stopped by issuing the **trcstop** command.

- What to watch

  Check the list of most active files, logical volumes, and physical volumes reported in the output report, and evaluate whether they are reasonable.

- Action

  Based on the output report, figure out how to re-distribute disk I/O requests by changing the filesystem layout.

Example A-9 shows the sample output listing generated by **filemon** command. In our tests, we observed that WCS cache files were among the most active files on the system.

Example: A-9 Sample output of filemon

```
Fri Apr  6 11:26:15 2001
System: AIX wcs5 Node: 4 Machine: 000BC6CD4C00

Cpu utilization:  7.2%

Most Active Files
------------------------------------------------------------------------
  #MBs  #opns   #rds   #wrs  file                 volume:inode
------------------------------------------------------------------------
   0.1     11     19      0  libc.cat             /dev/hd2:78239
   0.1     11     19      0  hosts                /dev/hd4:4634
   0.1     23     23     46  activity.log         /dev/hd2:325642
   0.1      7      6      0  StoreCatalogDisplay.storeId.25.storeId.25.---.htm
/dev/hd2:18710

   0.0      5      5      0
TopCategoriesDisplay.storeId.25.catalogId.1000.---.htm  /dev/hd2:29426
   0.0      2      2      0  ProductDisplay.storeId.25.productId.11194.---.htm
/dev/hd2:397670
   0.0      2      2      0
CategoryDisplay.storeId.25.categoryId.10003.catalogId.1000.---.htm
/dev/hd2:411974
Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume               description
------------------------------------------------------------------------
  0.03   1352      0   40.9  /dev/hd2             /usr
  0.02    232      0    7.0  /dev/hd6             paging
  0.01      0    312    9.4  /dev/lv_weblogs      /weblogs
  0.00      0      8    0.2  /dev/hd8             jfslog
  0.00      8      0    0.2  /dev/hd4             /
```

```
Most Active Physical Volumes
------------------------------------------------------------------------
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume               description
------------------------------------------------------------------------
  0.06   1592    320   57.8  /dev/hdisk0          N/A
```

Memory

vmstat

The **vmstat** command is one of the most helpful tools in determining if the system is short of physical memory. It summarizes the total active virtual memory used by all of the processes in the system. It also reports the statistics regarding kernel threads, disk, and CPU activity. The output format for the command is shown below (Example A-10).

Example: A-10 Sample output of vmstat

```
# vmstat 2

kthr  memory             page            faults             cpu
----- -----------  ------------------- ----------------- -----------
r  b  avm   fre    re pi po fr sr cy   in  sy   cs       us sy  id wa
0  0  16998 14612  0  0  0  0  0  0    101 10   8        55 0   44 0
0  1  16998 14611  0  0  0  0  0  0    411 2199 54       0  0   99 0
0  1  16784 14850  0  0  0  0  0  0    412 120  51       0  0   99 0
0  1  16784 14850  0  0  0  0  0  0    412 88   50       0  0   99 0
```

The columns highlighted in the following list provide an indication where most CPU cycles are being spent:

r Average # of kernel threads queued for execution in the process run queue per second.

b Average # of kernel threads awaiting I/O resources or for a page thread to be paged in.

avm The average number of 4 K pages that are allocated to paging space. The number in the avm field divided by 256 will yield the approximate number of megabytes (MB) allocated to paging space system-wide.

fre The average number of free memory pages. A page is a 4 KB area of real memory.

Important: When an application terminates, all of its working pages are immediately returned to the free list. Its persistent pages (files), however, remain in RAM and are not added back to the free list until they are stolen by the VMM (Virtual Memory Manager) for other programs. For this reason, the fre value may not indicate all the real memory that can be readily available for use by processes. If a page frame is needed, then persistent pages related to terminated applications are among the first to be handed over to another program.

pi Details the number (rate) of pages paged in from paging space. Paging space is the part of virtual memory that resides on disk. It is used as an overflow when memory is overcommitted.

po shows the number (rate) of pages paged out to paging space. If the system is reading in a significant number of persistent pages (files), you might see an increase in po without corresponding increases in pi.

us % of CPU time spent in user mode. When in user mode a process does not require kernel resources for execution.

sy % of CPU time spent in system mode i.e. processes that utilize kernel resources must use do so via execution of kernel system calls.

id % of CPU time spent idle, or waiting, without pending local I/O.

wa % of CPU time spent idle with pending local disk I/O.

- What to watch

  When determining if a system might be short on memory or if some memory tuning needs to be done, run the **`vmstat`** command over a set interval and examine the *pi* and *po* columns on the resulting report. These columns indicate the number of paging space page-ins per second and the number of paging space page-outs per second. For a system experiencing memory contention, the count of disk reads of data into memory (indicated by the *pi* column) and the amount written out to disk from memory (indicted by the *po* column) will be consistently non-zero. Having occasional non-zero values is not a concern because paging is the main principle of virtual memory. If insufficient memory exists to hold all the data needed by an application, then the system will need to page-in the required data from disk into memory, but may also need to page-out data to make room for the new page. This invariably results in slower system performance based on the disk I/O overhead for satisfying successive paging requests for processes.

Although no absolute value can be proposed based on the varieties of hardware configurations, a *pi* value greater than 5 per second generally provides an initial indication of memory resource contention. If you observe high I/O wait in the wa column and also a non-negligible number in the b (blocked queue of threads) column as well as high numbers in pi and po, you can suspect memory shortage.

- Action

  When you find a memory shortage, the most effective remedy is to add more memory to your system. In some cases, your application has a memory leak problem. If this is the case, the application should be fixed.

svmon

The **svmon** command provides a more in-depth analysis of memory usage than **vmstat**. Because WebSphere Application Server consumes a good deal of memory, this command will provide a better insight on performance problems related with WAS.

To determine whether **svmon** is installed and available, run the following command:

```
# lslpp -lI perfagent.tools
```

topas

AIX memory management subsystem classifies memory pages into the following two categories:

Computational pages	Pages that belong to working-storage segments or program text segments. A segment is considered to be a program text segment if an instruction cache miss occurs on any of its pages.
File pages	The remaining pages of total memory. These are usually pages from permanent data files on disks.

When an application terminates, all of its working storage pages are immediately returned to the free list. Its persistent pages (files), however, remain in RAM and are not added back to the free list until they are stolen by the VMM for other programs. For this reason, the fre value may not indicate all the real memory that can be readily available for use by processes.

topas is an AIX utility that can address this issue.

- What to watch

  Compare the following values in MEMORY section.

  Real,MB The size of real memory in megabytes.

% Comp	The percentage of real memory currently allocated to computational page frames. Computational page frames are generally those that are backed by paging space.
% Noncomp	The percentage of real memory currently allocated to non-computational frames. Non-computational page frames are generally those that are backed by file space, either data files, executable files, or shared library files.
% Client	The percentage of real memory currently allocated to cache remotely mounted files.

If the memory utilization rate is 100%, your system is short of memory.

svmon -P

svmon -P command sorts processes by memory usage and displays the memory usage statistics for the top Count processes. It is useful for watching which process is using the most memory.

- What to watch

  The **svmon -Put 10** lists the top 10 processes with detailed reports on their memory usage, identifying exactly how much memory each process on the system is using. In our tests, we observed the most memory-intensive processes are java processes invoked by WAS. Our example is shown in Example A-11.

- Action

  No particular system tuning tips are applicable.

Example: A-11 Sample output of svmon

```
# svmon -Put10
-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   23778 java           74977     1397      895    60289        N        Y

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
 18839        3 work shmat/mmap            58025     0    0 58002   0..60952
 18018        d work shared library text    5243     0   29    38   0..65535
     0        0 work kernel seg             1944  1388  866  1979   0..21611 :

65474..65535
 1c41d        - pers /dev/hd2:82003         1857     0    -     -   0..2828
  d4ac        - pers /dev/hd2:153679        1307     0    -     -   0..1347
  68a7        - pers /dev/hd2:565297        1094     0    -     -   0..1093
  f4ae        - pers /dev/hd2:153681         604     0    -     -   0..1742
 13432        - pers /dev/hd2:168292         516     0    -     -   0..760
```

DB2 Monitoring Tools

IBM DB2 provides a number of internal monitoring tools that are shipped as part of the base product. The tools in question provide various viewpoints of database activity. The following sections provide an introduction of to the tools in question with further implementation details available from the IBM Redbook *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012.

Snapshot Monitor

DB2 provides a utility called **snapshot** for capturing component activity measurements in the form of counts for specific actions within the database at a specific point in time. However, before attempting to extract what is termed a monitored 'snapshot' of database system activity, the applicable switches for the components to be included in the snapshot report must be enabled.

To determine the status of current monitor switch settings, issue the following command as the db2 instance owner:

```
db2 get monitor switches
```

This command provides output of the form shown in Example A-12.

Example: A-12 Output of monitor switches

```
Monitor Recording Switches

Switch list for node 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
Unit of Work Information                  (UOW) = OFF
```

Note: Monitoring at the database manager level will impact the performance of all databases created under that instance.

Depending on the which aspect of DB2 activity is of interest for analysis, the appropriate switch is activated by executing the following DB2 command format:

```
db2 update monitor switches using { switch-name {ON | OFF} ...}

[AT NODE node-num | GLOBAL]
```

```
switch-name:

BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, UOW
```

For example, in order to examine the bufferpool and statement activities on a local database, the following switches are activated by the command:

```
db2 update monitor switches using BUFFERPOOL ON STATEMENT ON
```

Once monitor switches are enabled, snapshots can be obtained at any point for either the database manager:

```
db2 get snapshot for database manager
```

or for a specific database (that is, a database with at least one active connection):

```
db2 get snapshot for {all|<specific component>} on <db alias name>
```

The output report will be of the form shown in the sample output in Section “DB2 snapshot output” on page 180, with the above command piped to a file for later review. We have used this command to study database buffer pool utilization under a stress test.

The points of interest differ according to the component(s) being monitored and further details on the data gathering and analysis process is available in the DB2 product documentation, *DB2 UDB System Monitor Guide and Reference V7,* SC09-2956.

Event monitor

DB2 event monitor differs in action from a snapshot monitor in that data collected and reported for components occurs over a period of time. This creates a useful trace of the state of events occurring in the database over the monitored period, and provides a preliminary indicator on possible problematic areas at the database or application level.

Before data can be recorded, an event monitor must be created that indicates the components to be measured. The command used to create an event monitor is of the format:

```
db2 create event monitor <monitor name> for [database | tables |
deadlocks | bufferpools | statements | transactions] write to file
<pathname to file>
```

Note: Do not add a filename in the path statement, as the event monitor generates log file names automatically at runtime.

For example, if deadlock problems need to be analyzed, the following command can be used to create an applicable event monitor:

```
db2 “create event monitor STTRC for deadlocks write to file
‘/itso/db2inst1’”
```

which will create an event monitor that will log the predefined metrics relative to dead lock analysis.

In order to minimize the impact of monitoring activities on the database performance, it is usually worthwhile to create a ‘non-blocking’ event monitor that will discard DB2 agent data if the monitors data collection buffers become full. Otherwise, the monitor will suspend the DB2 agents from sending further data, which incurs an additional runtime overhead as agents will need to check the monitor status for sending data each time the particular event occurs.

For creating a non-blocking event monitor, the parameter **NONBLOCKED** is added e.g. for the above example, the command would be changed to:

```
db2 “create event monitor STTRC for statements write to file ‘/itso/db2inst1’
NONBLOCKED”
```

The event monitor is then activated by the following command:

```
db2 “set event monitor STTRC state = 1”
```

Important: The monitor will stop logging if disk space becomes exhausted, therefore sufficient free space should be made available prior to enabling monitor logging.

To dump the logged information and close the event monitor file, the event monitor should be deactivated as:

```
db2 “set event monitor STTRC state = 0”
```

To analyze collected data, you need to re-format binary trace information to ASCII text format using the db2evmon utility. For example,

```
db2evmon -db <database name> -evm STTRC > sttrace.txt
```

This step must be done *before* resetting or dropping the event monitor. Otherwise, db2evmon will not be able to generate the trace log.

A sample of the output for statement monitoring is shown in Section “DB2 event monitor sample output” on page 192.

Once testing has been completed, the event monitor can either be reset or removed by issuing the respective commands:

```
db2 reset monitor for database <database name>
```

or

```
db2 drop event monitor STTRC
```

Important: On a production system, it is strongly recommended that once event monitoring data has be collected for analysis, all enabled event monitors are disabled to avoid impacting system performance.

The Explain Facility

If you want to know how a query will be executed by DB2, you must analyze its access plan, which is the method for retrieving data from a specific table. The Explain Facility will provide information about how DB2 will access the data in order to resolve the SQL statements. If you have identified a particular application as the possible source of a performance problem, you need to obtain the SQL statements that the application issues and analyze the access plan for those SQL statements.The DB2 Explain Facility provides a mechanism for extracting information regarding the data access methods used in execution of SQL queries by the DB2 optimizer. Further details on how to create the necessary tables and analyze the information reported by the Explain tool is provided in *DB2 UDB V7.1 Performance Tuning Guide,* SG24-6012.

CLI/ODBC/JDBC Trace Facility

The CLI/ODBC/JDBC Trace Facility is used to capture information specific to the communication between a CLI client and DB2. All function calls executed are recorded in a text file for later analysis. In addition to functional information, however, the trace file contains elapsed time information that can be extremely useful for application and database tuning.This facility can help pinpoint long running statements and analyze the time spent in the client application, DB2, or the network. Further details can be found in Appendix K of *DB2 UDB CLI Guide and Reference V7*, SC09-2950.

To obtain a CLI trace, run the application after activating the trace using the command:

```
db2 update cli cfg for section common using trace 1 tracepathname <fully
qualified pathname>

db2 update cli cfg for section common using tracecomm 1
```

Silk Preview

Silk Preview by Segue Software, Inc. is a free introduction to load and scalability testing. Silk Preview monitors the access times of Web pages, and gives you valuable insight into whether or not your application is suffering performance degradation.

- Real-world load testing is very complex, and covers a lot more territory.
- Silk Preview is only a sample of the functionality and user interface of SilkPerformer, Segue's high-end load testing tool.
- The system requirements for SilkPreview include Windows NT 4.0 with Service Pack 5 or higher or Windows 2000, and a Windows installer service (redistributed on SilkPerformer media).

Note: Although SilkPreview only runs on Windows NT or Windows 2000, you can use it to monitor the access times of any WebSphere Commerce Suite Web pages, regardless of the operating system on which you are running Commerce Suite.

B

Oracle tuning tips

This appendix provides the brief summary of Oracle database tuning tips pertinent to WCS 5.1. The tips given here were tested on WCS 4.1 but have not been tested on WCS 5.1. Therefore, the information in this appendix should be used at the user's discretion.

For more detailed information on Oracle database tuning, refer to *Database Performance on AIX in DB2 UDB and Oracle Environments*, SG24-5511.

Top 10s

1. Set CURSOR_SHSRING in init$ORACLE_SID.ora to "force" in order for Oracle to build up bind variables out of dynamic SQLs.
2. Run analyze as frequently as possible on all WCS tables. Running it once a week might be a good start.
3. Rebuild your indexes frequently. Although the frequency for this depends on the amount of update activities, the generally recommended best practice is to make it a weekly or biweekly activity.
4. Avoid setting the shared pool size of EJSADMIN too large. A size of 50-70 MB is a good start.
5. Setting WCS database's shared pool size to 100MB is a good start.
6. Keep the KEYS table in the database buffer cache.
7. Set checkpoint_process to TRUE. The checkpoint process should always be running if you are running Oracle database on an SMP machine and the database has more than 5 or 10 data files.
8. Set redo log size to 5 ~ 20M in general. For heavy update workload, a size of 50M or more is a good start. A redo log switch should occur every 20 to 30 minutes. The redo log switch interval can be controlled by adjusting redo logs size and setting LOG_CHECKPOINT_INTERVAL equal to the number of database blocks that should be kept in database buffer cache before writing to disk.
9. Note that the default value for log_buffer (8K) is usually way too small. Setting it to 512k to 1Mb is a good start.
10. The number of rollback segment should approximately be equal to (number of concurrent update transactions) / 4.

Recommended values

Table 6-1 Recommended values for tunable parameters

Parameter Name	Recommended value	Remark
db_block_size	8,192	
sort_area_size	2,097,152	
sort_area_retained_size	2,097,152	
db_block_buffers	20,000	db_block_buffers (in bytes) + shared_pool_size should not exceed 2/3 of physical memory.
shared_pool_size	102,400,000	Tune this parameter to maximize the hit ratio on shared pool area and to accommodate non-sharable SQL. If there are contentions on the shared memory, increase this number.
log_buffer	1,024,000	
pre_page_sga	Yes	Keep SGA in memory

Tips for physical layout design

- Assign a separate tablespace for rollback segments.
- Active tables should be placed on separate disks. Be sure to apply this rule to system tablespace.
- Redo log files should be placed on separate disks.
- No user should have system tablespace as his default tablespace. Check the current setting with the following SQL statement:

  ```
  select username from dba_users where default_tablespace = 'SYSTEM';
  ```

- Share Temp tablespace as temporary tablespace for normal users. Check the current setting with the following SQL statement:

  ```
  select username from dba_users where temporary_tablespace = 'TEMP';
  ```

- The Temp tablespace should be placed on a separate disk.
- Spread the following segments across as many disk drives as possible.
  - RBS (especially if there are heavy update transactions)
  - TEMP(especially if there are many sorts to disks)
  - SYSTEM
  - DATA
  - INDEX
- Check the values of physical reads and physical writes in v$filestat view.
- Spread control files & redo logs across available disk drives.
- To see if there is I/O contention, check the value of ‘db file sequential read’ column in v$system_event view.

Optimizing sorts

- Try to do most sorts in memory.
- The common size for sort_area_size is 1MB
- Watch out for page swaps at OS level and parallelism.
- Set sort_write_buffers to 2 ~ 8.
- Set sort_write_buffer_size to 32K ~ 64K.

C

Sample Outputs

DB2 snapshot output

The following is typical output resulting from a request for database manager information:

```
                Database Manager Snapshot

Node type                                      = Database Server with
local and remote clients
Instance name                                  = user1
Database manager status                        = Active

Product name                                   =
Product identification                         =
Service level                                  =

Sort heap allocated                            = 0
Post threshold sorts                           = 0
Piped sorts requested                          = 3
Piped sorts accepted                           = 3

Start Database Manager timestamp               = 04-04-1997
10:54:32.357375
Last reset timestamp                           = 04-04-1997
14:28:54.799017
Snapshot timestamp                             = 04-06-1997
14:27:19.996209

Remote connections to db manager               = 0
Remote connections executing in db manager     = 0
Local connections                              = 1
Local connections executing in db manager      = 0
Active local databases                         = 1

High water mark for agents registered          = 2
High water mark for agents waiting for a token = 1
Agents registered                              = 2
Agents waiting for a token                     = 0
Idle agents                                    = 0

Committed private Memory (Bytes)               = 12435456

Buffer Pool Activity Information  (BUFFERPOOL) = ON  04-04-1997 14:29:42
Lock Information                         (LOCK) = ON  04-04-1997 14:29:42
Sorting Information                      (SORT) = ON  04-04-1997 14:29:42
SQL Statement Information          (STATEMENT) = ON  04-04-1997 14:29:42
Table Activity Information             (TABLE) = ON  04-04-1997 14:29:42
Unit of Work Information                 (UOW) = ON  04-04-1997 14:29:42
```

```
Agents assigned from pool                      = 5
Agents created from empty pool                 = 2
Agents stolen from another application         = 0
High water mark for coordinating agents        = 2
Max agents overflow                            = 0
```

The following is typical output resulting from a request for database information:

```
             Database Snapshot

Database name                               = SAMPLE
Database path                               =
/home/user1/user1/NODE0000/SQL00011/
Input database alias                        = SAMPLE
Database status                             = Active
Catalog node number                         = 0

Catalog network node name                   =
Operating system running at database server= AIX
Location of the database                    = Remote

Locks held currently                        = 7
Lock waits                                  = 0
Time database waited on locks (ms)          = 0
Lock list memory in use (Bytes)             = 1400
Deadlocks detected                          = 0
Lock escalations                            = 0
Exclusive lock escalations                  = 0
Current applications waiting on locks       = 0
Lock Timeouts                               = 0

Total sort heap allocated                   = 0
Total sorts                                 = 3
Total sort time (ms)                        = 1
Sort overflows                              = 0
Active sorts                                = 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

Buffer pool data logical reads              = 32
Buffer pool data physical reads             = 13
Asynchronous pool data page reads           = 0
Buffer pool data writes                     = 0
Asynchronous pool data page writes          = 0
```

```
Buffer pool index logical reads            = 55
Buffer pool index physical reads           = 23
Asynchronous pool index page reads         = 0
Buffer pool index writes                   = 0
Asynchronous pool index page writes        = 0
Total buffer pool read time (ms)           = 364
Total buffer pool write time (ms)          = 0
Total elapsed asynchronous read time       = 0
Total elapsed asynchronous write time      = 0
Asynchronous 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)              = 0

Direct reads                               = 34
Direct writes                              = 0
Direct read requests                       = 4
Direct write requests                      = 0
Direct reads elapsed time (ms)             = 1
Direct write elapsed time (ms)             = 0

Database files closed                      = 0

Commit statements attempted                = 1
Rollback statements attempted              = 0
Dynamic statements attempted               = 13
Static statements attempted                = 1
Failed statement operations                = 0
Select SQL statements executed             = 1
Update/Insert/Delete statements executed   = 0

DDL statements executed                    = 0
Internal automatic rebinds                 = 0
Internal rows deleted                      = 0
Internal rows inserted                     = 0
Internal rows updated                      = 0
Internal commits                           = 1
Internal rollbacks                         = 0
Internal rollbacks due to deadlock         = 0

Rows deleted                               = 0
Rows inserted                              = 0
Rows updated                               = 0
Rows selected                              = 8

Binds/precompiles attempted                = 0

First database connect timestamp           = 04-04-1997 14:29:55.197659
```

```
Last reset timestamp                         =
Last backup timestamp                        =
Snapshot timestamp                           = 04-04-1997 14:32:14.151875

High water mark for connections              = 1
High water mark for database heap            = 652811
Application connects                         = 1
Secondary connects total                     = 0
Applications connected currently             = 1
Appls. executing in db manager currently     = 0

Maximum secondary log space used (Bytes)     = 0
Maximum total log space used (Bytes)         = 0
Secondary logs allocated currently           = 0
Log pages read                               = 0
Log pages written                            = 0

Package cache lookups                        = 1
Package cache inserts                        = 1
Application section lookups                  = 13
Application section inserts                  = 1

Catalog cache lookups                        = 2
Catalog cache inserts                        = 2
Catalog cache overflows                      = 0
Catalog cache heap full                      = 0

Agents associated with appls.                = 1
Maximum agents associated with appls.        = 1
Maximum coordinating agents                  = 0
Agents waiting on locks                      = 0
```

The following is typical output resulting from a request for application information (by specifying either an application ID, an agent ID, all applications, or all applications on a database):

```
            Application Snapshot

Application handle                           = 5
Application status                           = UOW Waiting
Status change time                           = 04-04-1997 14:31:46.930243
Application code page                        = 850
Application country code                     = 1
DUOW correlation token                       = *LOCAL.user1.970404192955
Application name                             = db2bp_32
Application ID                               = *LOCAL.user1.970404192956
Sequence number                              = 0001
```

```
Connection request start timestamp         = 04-04-1997 14:29:55.197659
Connect request completion timestamp       = 04-04-1997 14:29:55.726359
Application idle time                      = 28 seconds
Authorization ID                           = USER1
Execution ID                               = user1
Configuration NNAME of client              =
Client database manager product ID         = SQL03010
Process ID of client application           = 18660
Platform of client application             = AIX
Communication protocol of client           = Local Client
Database name                              = SAMPLE
Database path                              =
/home/user1/user1/NODE0000/SQL00011/
Client database alias                      = sample
Input database alias                       = SAMPLE
Last reset timestamp                       =
Snapshot timestamp                         = 04-04-1997 14:32:14.151875
The highest authority level granted        =
        Direct DBADM authority
        Direct CREATETAB authority
        Direct BINDADD authority
        Direct CONNECT authority
        Direct CREATE_NOT_FENC authority
        Direct IMPLICIT_SCHEMA authority
        Indirect SYSADM authority
        Indirect CREATETAB authority
        Indirect BINDADD authority
        Indirect CONNECT authority
        Indirect IMPLICIT_SCHEMA authority
Coordinating node number                   = 0
Current node number                        = 0
Coordinator agent process or thread ID     = 75340
Agents working for the application         = 1
Agents stolen                              = 0
Agents waiting on locks                    = 0
Maximum associated agents                  = 1
Priority at which application agents work  = 0
Priority type                              = Static

Locks held by application                  = 7
Lock waits since connect                   = 0
Time application waited on locks (ms)      = 0
Deadlocks detected                         = 0
Lock escalations                           = 0
Exclusive lock escalations                 = 0
Number of Lock Timeouts since connected    = 0
Total time UOW waited on locks (ms)        = 0

Total sorts                                = 3
```

```
Total sort time (ms)                       = 1
Total sort overflows                       = 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
Buffer pool data logical reads             = 32
Buffer pool data physical reads            = 13
Buffer pool data writes                    = 0
Buffer pool index logical reads            = 55
Buffer pool index physical reads           = 23
Buffer pool index writes                   = 0
Total buffer pool read time (ms)           = 364
Total buffer pool write time (ms)          = 0
Time waited for prefetch (ms)              = 0
Direct reads                               = 34
Direct writes                              = 0
Direct read requests                       = 4
Direct write requests                      = 0
Direct reads elapsed time (ms)             = 1
Direct write elapsed time (ms)             = 0

Number of SQL requests since last commit   = 13
Commit statements                          = 1
Rollback statements                        = 0
Dynamic SQL statements attempted           = 13
Static SQL statements attempted            = 1
Failed statement operations                = 0
Select SQL statements executed             = 1
Update/Insert/Delete statements executed   = 0
DDL statements executed                    = 0
Internal automatic rebinds                 = 0
Internal rows deleted                      = 0
Internal rows inserted                     = 0
Internal rows updated                      = 0
Internal commits                           = 1
Internal rollbacks                         = 0
Internal rollbacks due to deadlock         = 0
Binds/precompiles attempted                = 0
Rows deleted                               = 0
Rows inserted                              = 0
Rows updated                               = 0
Rows selected                              = 8
Rows read                                  = 53
Rows written                               = 0

UOW log space used (Bytes)                 = 0
Previous UOW completion timestamp          = 04-04-1997 14:29:55.728025
```

```
UOW start timestamp                         = 04-04-1997 14:31:46.580691
UOW stop timestamp                          =
UOW completion status                       =
Open remote cursors                         = 0
Open remote cursors with blocking           = 0
Rejected Block Remote Cursor requests       = 0
Accepted Block Remote Cursor requests       = 1
Open local cursors                          = 0
Open local cursors with blocking            = 0

Total User CPU Time used by agent (s)       = 0.070000
Total System CPU Time used by agent (s)     = 0.020000
Package cache lookups                       = 1
Package cache inserts                       = 1
Application section lookups                 = 13
Application section inserts                 = 1
Catalog cache lookups                       = 0
Catalog cache inserts                       = 0
Catalog cache overflows                     = 0
Catalog cache heap full                     = 0

Most recent operation                       = Close
Cursor name                                 = SQLCUR201
Most recent operation start timestamp       = 04-04-1997 14:31:46.859493
Most recent operation stop timestamp        = 04-04-1997 14:31:46.930287

Statement type                              = Dynamic SQL Statement
Statement                                   = Select
Section number                              =
Application creator                         =
Package name                                =
Cursor name                                 = SQLCUR201
Statement node number                       = 0
Statement start timestamp                   = 04-04-1997 14:31:46.859493
Statement stop timestamp                    = 04-04-1997 14:31:46.930287
Total user CPU time                         = 0.000000
Total system CPU time                       = 0.000000
SQL compiler cost estimate in timerons      = 9355
SQL compiler cardinality estimate           = 1600
Degree of parallelism requested             = 1
Number of agents working on statement       = 1
Number of subagents created for statement   = 1
Statement sorts                             = 3
Total sort time                             = 1
Sort overflows                              = 0
Rows read                                   = 0
Rows written                                = 0
Rows deleted                                = 0
Rows updated                                = 0
```

```
Rows inserted                               = 0
Rows fetched                                = 0
Number of subsections                       = 1
Dynamic SQL statement text                  =
SELECT DEPTNAME, DEPTNUMB, MANAGER, NAME FROM ORG, STAFF
  WHERE DEPTNUMB = DEPT AND MANAGER = ID ORDER BY DEPTNAME
```

The following is typical output resulting from a request for buffer pool information:

```
             Bufferpool Snapshot

Bufferpool name                             = IBMDEFAULTBP
Database name                               = SAMPLE
Database path                               =
/home/user1/user1/NODE0000/SQL00011/
Input database alias                        = SAMPLE
Buffer pool data logical reads              = 32
Buffer pool data physical reads             = 13
Buffer pool data writes                     = 0
Buffer pool index logical reads             = 55
Buffer pool index physical reads            = 23
Total buffer pool read time (ms)            = 364
Total buffer pool write time (ms)           = 0
Database files closed                       = 0

Asynchronous pool data page reads           = 0
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        = 0
Total elapsed asynchronous write time       = 0
Asynchronous read requests                  = 0
Direct reads                                = 34
Direct writes                               = 0
Direct read requests                        = 4
Direct write requests                       = 0
Direct reads elapsed time (ms)              = 1
Direct write elapsed time (ms)              = 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
```

The following is typical output resulting from a request for table information:

```
            Table Snapshot

First database connect timestamp     = 04-04-1997 14:29:55.197659

Last reset timestamp                 =
Snapshot timestamp                   = 04-04-1997 14:32:14.151875
Database name                        = SAMPLE
Database path                        =
/home/user1/user1/NODE0000/SQL00011/
Input database alias                 = SAMPLE
Number of accessed tables            = 6

Table Schema         Table Name           Table Type           Rows
Written  Rows Read  Overflows
-------------------- -------------------- --------------------
------------ ---------- ----------
USER1                STAFF                User
0          35           0
USER1                ORG                  User
0           8           0
SYSIBM               SYSTABLES            Catalog
0           2           0
SYSIBM               SYSTABLESPACES       Catalog
0           3           0
SYSIBM               SYSPLAN              Catalog
0           1           0
SYSIBM               SYSDBAUTH            Catalog
0           3           0
```

The following is typical output resulting from a request for table space information:

```
             Tablespace Snapshot

First database connect timestamp          = 04-04-1997 14:29:55.197659
Last reset timestamp                      =
Snapshot timestamp                        = 04-04-1997 14:32:14.151875
Database name                             = SAMPLE
Database path                             =
/home/user1/user1/NODE0000/SQL00011/
Input database alias                      = SAMPLE
Number of accessed tablespaces            = 3

Tablespace name                           = SYSCATSPACE

  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

  Buffer pool data logical reads           = 26
  Buffer pool data physical reads          = 11
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 55
  Buffer pool index physical reads         = 23
  Asynchronous pool index page reads       = 0
  Buffer pool index writes                 = 0
  Asynchronous pool index page writes      = 0
  Total buffer pool read time (ms)         = 342
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
  Asynchronous read requests               = 0

  Direct reads                             = 34
  Direct writes                            = 0
  Direct read requests                     = 4
  Direct write requests                    = 0
  Direct reads elapsed time (ms)           = 1
  Direct write elapsed time (ms)           = 0

  Number of files closed                   = 0

Tablespace name                            = TEMPSPACE1

  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

  Buffer pool data logical reads           = 0
  Buffer pool data physical reads          = 0
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 0
  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)         = 0
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
```

```
  Asynchronous read requests               = 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

  Number of files closed                   = 0

 Tablespace name                           = USERSPACE1

  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

  Buffer pool data logical reads           = 6
  Buffer pool data physical reads          = 2
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 0
  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)         = 22
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
  Asynchronous read requests               = 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

  Number of files closed                   = 0
```

The following is typical output resulting from a request for lock information:

```
                Database Lock Snapshot
```

```
Database name                               = SAMPLE
Database path                               =
/home/user1/user1/NODE0000/SQL00011/
Input database alias                        = SAMPLE
Locks held                                  = 7
Applications currently connected            = 1
Applications currently waiting on locks     = 0
Snapshot timestamp                          = 04-04-1997 14:32:14.151875

Application handle                          = 5
Application ID                              = *LOCAL.user1.970404192956
Sequence number                             = 0001
Application name                            = db2bp_32
Authorization ID                            = USER1
Application status                          = UOW Waiting
Status change time                          =
Application code page                       = 850
Locks held                                  = 7
Total wait time (ms)                        = 0

Object Name Object Type     Tablespace Name      Table Schema
Table Name           Mode Status
----------- -------------- -------------------- --------------------
------------------- ---- ----------
1545        Row              SYSCATSPACE          SYSIBM
SYSTABLES            NS   Granted
1544        Row              SYSCATSPACE          SYSIBM
SYSTABLES            NS   Granted
2           Table            SYSCATSPACE          SYSIBM
SYSTABLES            IS   Granted
27          Table            SYSCATSPACE          SYSIBM
SYSTABLESPACES       S    Granted
257         Row              SYSCATSPACE          SYSIBM
SYSPLAN              S    Granted
7           Table            SYSCATSPACE          SYSIBM
SYSPLAN              IS   Granted
0         Internal
S    Granted
```

DB2 event monitor sample output

The following sample was generated from a event monitor created for statement event logging.

```
------------------------------------------------------------------------
                            EVENT LOG HEADER
  Event Monitor name: STTRC
  Server Product ID: SQL07010
  Version of event monitor data: 6
  Byte order: BIG ENDIAN
  Number of nodes in db2 instance: 1
  Codepage of database: 1208
  Country code of database: 1
  Server instance name: db2inst1
------------------------------------------------------------------------
--

------------------------------------------------------------------------
--
  Database Name: MALL
  Database Path: /itso/db2inst1/db2inst1/NODE0000/SQL00002/
  First connection timestamp: 03-28-2001 10:43:44.771625
  Event Monitor Start time:   03-28-2001 12:21:29.892155
------------------------------------------------------------------------
--

3) Connection Header Event ...
  Appl Handle: 38
  Appl Id: 0903BB88.AAC7.010328164344
  Appl Seq number: 0001
  DRDA AS Correlation Token: 0903BB88.AAC7.010328164344
  Program Name    : java
  Authorization Id: DB2INST1
  Execution Id    : root
  Codepage Id: 819
  Country code: 1
  Client Process Id: 20158
  Client Database Alias: MALL
  Client Product Id: SQL07010
  Client Platform: AIX
  Client Communication Protocol: TCPIP
  Client Network Name: dobong
  Connect timestamp: 03-28-2001 10:43:44.771625

4) Connection Header Event ...
  Appl Handle: 39
  Appl Id: 0903BB88.AAE0.010328164640
  Appl Seq number: 0001
```

```
  DRDA AS Correlation Token: 0903BB88.AAE0.010328164640
  Program Name    : java
  Authorization Id: DB2INST1
  Execution Id    : root
  Codepage Id: 819
  Country code: 1
  Client Process Id: 20158
  Client Database Alias: MALL
  Client Product Id: SQL07010
  Client Platform: AIX
  Client Communication Protocol: TCPIP
  Client Network Name: dobong
  Connect timestamp: 03-28-2001 10:46:40.592150

5) Connection Header Event ...
  Appl Handle: 40
  Appl Id: *LOCAL.db2inst1.010328173938
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.db2inst1.010328173938
  Program Name    : db2bp
  Authorization Id: DB2INST1
  Execution Id    : db2inst1
  Codepage Id: 819
  Country code: 1
  Client Process Id: 19180
  Client Database Alias: mall
  Client Product Id: SQL07010
  Client Platform: AIX
  Client Communication Protocol: Local
  Client Network Name:
  Connect timestamp: 03-28-2001 11:39:38.477171

6) Statement Event ...
  Appl Handle: 40
  Appl Id: *LOCAL.db2inst1.010328173938
  Appl Seq number: 0001

  Record is the result of a flush: FALSE
  ------------------------------------------
  Operation: Static Commit
  Package  : SQLC2D01
  Cursor   :
  Cursor was blocking: FALSE
  ------------------------------------------
  Start Time: 03-28-2001 12:21:29.944614
  Stop Time:  03-28-2001 12:21:29.944683
  Exec Time:  0.000069 seconds
  Number of Agents created: 1
  User CPU:   0.000000 seconds
```

```
  System CPU: 0.000000 seconds
  Fetch Count: 0
  Sorts: 0
  Total sort time: 0
  Sort overflows: 0
  Rows read: 0
  Rows written: 0
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
  SQLCA:
   sqlcode: 0
   sqlstate: 00000

7) Statement Event ...
  Appl Handle: 40
  Appl Id: *LOCAL.db2inst1.010328173938
  Appl Seq number: 0001

  Record is the result of a flush: FALSE
  ------------------------------------------
  Type     : Dynamic
  Operation: Prepare
  Section  : 201
  Creator  : NULLID
  Package  : SQLC2D01
  Cursor   : SQLCUR201
  Cursor was blocking: FALSE
  Text     : select count(*) from users
  ------------------------------------------
  Start Time: 03-28-2001 12:21:58.729329
  Stop Time:  03-28-2001 12:21:58.738108
  Exec Time:  0.008779 seconds
  Number of Agents created: 1
  User CPU:   0.000000 seconds
  System CPU: 0.000000 seconds
  Fetch Count: 0
  Sorts: 0
  Total sort time: 0
  Sort overflows: 0
  Rows read: 0
  Rows written: 0
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
  SQLCA:
   sqlcode: 0
   sqlstate: 00000
```

```
8) Statement Event ...
  Appl Handle: 40
  Appl Id: *LOCAL.db2inst1.010328173938
  Appl Seq number: 0001

  Record is the result of a flush: FALSE
  ------------------------------------------
  Type     : Dynamic
  Operation: Open
  Section  : 201
  Creator  : NULLID
  Package  : SQLC2D01
  Cursor   : SQLCUR201
  Cursor was blocking: FALSE
  Text     : select count(*) from users
  ------------------------------------------
  Start Time: 03-28-2001 12:21:58.738538
  Stop Time:  03-28-2001 12:21:58.738596
  Exec Time:  0.000058 seconds
  Number of Agents created: 1
  User CPU:   0.000000 seconds
  System CPU: 0.000000 seconds
  Fetch Count: 0
  Sorts: 0
  Total sort time: 0
  Sort overflows: 0
  Rows read: 0
  Rows written: 0
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
  SQLCA:
   sqlcode: 0
   sqlstate: 00000

9) Statement Event ...
  Appl Handle: 40
  Appl Id: *LOCAL.db2inst1.010328173938
  Appl Seq number: 0001

  Record is the result of a flush: FALSE
  ------------------------------------------
  Type     : Dynamic
  Operation: Describe
  Section  : 201
  Creator  : NULLID
  Package  : SQLC2D01
  Cursor   : SQLCUR201
  Cursor was blocking: FALSE
```

```
  Text     : select count(*) from users
  ------------------------------------------
  Start Time: 03-28-2001 12:21:58.738538
  Stop Time:  03-28-2001 12:21:58.739028
  Exec Time:  0.000490 seconds
  Number of Agents created: 1
  User CPU:   0.000000 seconds
  System CPU: 0.000000 seconds
  Fetch Count: 1
  Sorts: 0
  Total sort time: 0
  Sort overflows: 0
  Rows read: 0
  Rows written: 0
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
  SQLCA:
   sqlcode: 0
   sqlstate: 00000

10) Statement Event ...
  Appl Handle: 40
  Appl Id: *LOCAL.db2inst1.010328173938
  Appl Seq number: 0001

  Record is the result of a flush: FALSE
  ------------------------------------------
  Type     : Dynamic
  Operation: Close
  Section  : 201
  Creator  : NULLID
  Package  : SQLC2D01
  Cursor   : SQLCUR201
  Cursor was blocking: FALSE
  Text     : select count(*) from users
  ------------------------------------------
  Start Time: 03-28-2001 12:21:58.738538
  Stop Time:  03-28-2001 12:21:58.741030
  Exec Time:  0.002492 seconds
  Number of Agents created: 1
  User CPU:   0.000000 seconds
  System CPU: 0.000000 seconds
  Fetch Count: 1
  Sorts: 0
  Total sort time: 0
  Sort overflows: 0
  Rows read: 0
  Rows written: 0
```

```
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
  SQLCA:
   sqlcode: 0
   sqlstate: 00000

11) Statement Event ...
  Appl Handle: 40
  Appl Id: *LOCAL.db2inst1.010328173938
  Appl Seq number: 0001

  Record is the result of a flush: FALSE
  ------------------------------------------
  Operation: Static Commit
  Package  : SQLC2D01
  Cursor   :
  Cursor was blocking: FALSE
  ------------------------------------------
  Start Time: 03-28-2001 12:21:58.742004
  Stop Time:  03-28-2001 12:21:58.742064
  Exec Time:  0.000060 seconds
  Number of Agents created: 1
  User CPU:   0.000000 seconds
  System CPU: 0.000000 seconds
  Fetch Count: 0
  Sorts: 0
  Total sort time: 0
  Sort overflows: 0
  Rows read: 0
  Rows written: 0
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
  SQLCA:
   sqlcode: 0
   sqlstate: 00000
```

D

GCStats.java

This appendix provides the source code for the tool GCStats, which collects and summarizes information about garbage collection.

GCStats.java

```
// GCStats.java
// This utility tabulates data generated from a verbose garbage collection
trace.
// To run this utility type:
//     java GCStats inputfile [total_time]
//
// Gennaro (Jerry) Cuomo - IBM Corp.  03/2000
// Carmine F. Greco 3/17/00 - JDK1.2.2 compatibility
//
import java.io.*;
import java.util.*;

public class GCStats {

   static int   total_time=-1;                    // total time of run in ms
   static long  total_gctime=0, total_gctime1=0;  // total time spent in GCs
   static long  total_bytes=0, total_bytes1=0;    // total bytes collected
   static long  total_free=0, total_free1=0;      // total
   static int   total_gc=0;                       // total number of GCs

   static boolean verbose=false; // debug trace on/off

   public static void parseLine(String line) {
      //  parsing a string that looks like this...
      //  <GC(31): freed 16407744 bytes in 107 ms, 97% free
(16417112/16777208)>

      if (isGCStatsLine(line)) {  // First test if line starts with "<GC..."

         if (verbose) System.out.println("GOT a GC - "+line);
         long temp=numberBefore(line, " bytes")/1024;     // get total memory
collected
         total_bytes+=temp; total_bytes1+=(temp*temp);
         temp=numberBefore(line, " ms");             // get time in GC
         total_gctime+=temp; total_gctime1+=(temp*temp);
         temp=numberBefore(line, "% free");          // get time % free
         total_free+=temp; total_free1+=(temp*temp);
         if (temp!=0) {
            total_gc++;                              // total number of GCs
         }
      }
   }

   public static int numberBefore( String line, String s) {
      int    ret = 0;
      int    idx = line.indexOf(s);
```

```
      int    idx1= idx-1;
      if (idx>0) {

         // the string was found, now walk backwards until we find the blank
         while (idx1!=0 && line.charAt(idx1)!=' ') idx1--;
         if (idx1>0) {
            String temp=line.substring(idx1+1,idx);
            if (temp!=null) {
               ret=Integer.parseInt(temp);    // convert from string to number
            }
         } else {
            if (verbose) System.out.println("ERROR: numberBefore() - Parse
Error looking for "+s);
         }
      }
      return ret;
   }

   public static boolean isGCStatsLine(String line) {
      return ( (line.indexOf("<GC") > -1) && (line.indexOf(" freed")>0) &&
(line.indexOf(" bytes")>0));
   }

   public static void main (String args[]) {
      String filename=null;
      BufferedReader foS=null;
      boolean keepgoing=true;

      if (args.length==0) {
         System.out.println("GCStats - ");
         System.out.println("     - ");
         System.out.println("     - Syntax: GCStats filename
[run_duration(ms)]");
         System.out.println("     -  filename = file containing -verbosegc
data");
         System.out.println("     -  run_duration(ms) = duration of fixed work
run in which GCs took place");
         return;
      }
      if (args.length>0) {
         filename=args[0];
      }
      if (args.length>1) {
         total_time=Integer.parseInt(args[1]);
      }
      if (verbose) System.out.println("Filename="+filename);

      try {
         foS = new BufferedReader(new FileReader(filename));
```

```
      } catch (Throwable e) {
         System.out.println("Error opening file="+filename);
         return;
      }

      while (keepgoing) {
         String nextLine;
         try {
            nextLine=foS.readLine();
         } catch (Throwable e) {
            System.out.println("Cannot read file="+filename);
            return;
         }
         if (nextLine!=null) {
            parseLine(nextLine);
         } else {
            keepgoing=false;
         }
      }
      try {
         foS.close();
      } catch (Throwable e) {
         System.out.println("Cannot close file="+filename);
         return;
      }

      System.out.println("-------------------------------------------------");
      System.out.println("- GC Statistics for file - "+filename);
      System.out.println("-------------------------------------------------");
      System.out.println("-**** Totals ***");
      System.out.println("- "+total_gc+" Total number of GCs");
      System.out.println("- "+total_gctime+" ms. Total time in GCs");
      System.out.println("- "+total_bytes+" Kbytes. Total memory collected
during GCs");
      System.out.println("- ");
      System.out.println("-**** Averages ***");

      double mean=total_gctime/total_gc,
stddev=Math.sqrt((total_gctime1-2*mean*total_gctime+total_gc*mean*mean)/total_g
c);
      int    imean=new Double(mean).intValue(), istddev=new
Double(stddev).intValue();
      System.out.println("- "+imean+" ms. Average time per GC.
(stddev="+istddev+" ms.)");

      mean=total_bytes/total_gc;
stddev=Math.sqrt((total_bytes1-2*mean*total_bytes+total_gc*mean*mean)/total_gc)
;
      imean=new Double(mean).intValue(); istddev=new Double(stddev).intValue();
```

```
      System.out.println("- "+imean+" Kbytes. Average memory collected per GC.
(stddev="+istddev+" Kbytes)");

      mean=total_free/total_gc;
stddev=Math.sqrt((total_free1-2*mean*total_free+total_gc*mean*mean)/total_gc);
      imean=new Double(mean).intValue(); istddev=new Double(stddev).intValue();
      System.out.println("- "+imean+"%. Free memory after each GC.
(stddev="+istddev+"%)");

      if (total_time>0 && total_gctime>0) {
        System.out.println("- "+((total_gctime*1.0)/(total_time*1.0))*100.0+"%
of total time ("+total_time+"ms.) spent in GC.");
      }
      System.out.println("_____________________________ "+new Date());
      System.out.println("");
   }
```

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 on ordering these publications see , “How to get IBM Redbooks” on page 206.

- *WebSphere V3 Performance Tuning Guide,* SG24-5657
- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167
- *WebSphere Scalability: WLM and Clustering using WebSphere Application Server*, SG24-6153
- *Up and Running with WebSphere Site Analyzer*, SG24-6169
- *AIX Logical Volume Manager: From A to Z: Introduction and Concepts*, SG24-5432
- *IBM Certification Study Guide: AIX Performance and System Tuning*, SG24-6184
- *Database Performance on AIX in DB2 UDB and Oracle Environments*, SG24-5511
- *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012

Other resources

These publications are also relevant as further information sources:

- *A Methodology for Production Performance Tuning,* An IBM whitepaper for WebSphere Application Server Standard and Advanced Editions*,*
- *IBM WebSphere CommerceSuite What’s new in Version 5.1,* shipped on product CD.
- *IBM WebSphere CommerceSuite Site Administrator Version 5.1,* shipped on product CD.
- *DB2 UDB Command Reference,* SC09-2951
- *DB2 UDB System Monitor Guide and Reference V7,* SC09-2956
- *DB2 UDB CLI Guide and Reference V7*, SC09-295*0*

Referenced Web sites

These Web sites are also relevant as further information sources:

- http://www-4.ibm.com/software/webservers/commerce/wcs_pro/WhatsNew.pdf
- http://www-4.ibm.com/software/webservers/appserv/download_ra.html - Resource Analyzer installation and usage guide
- http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/toc.htm - Excellent guide for AIX tuning and available tools
- http://www-4.ibm.com/software/webservers/commerce/wcs_pro/ax51adv.pdf - A detailed implementation guide for 3 tier configuration
- http://www-4.ibm.com/software/webservers/edgeserver/
- http://www.ibm.com/software/webservers/commerce/paymentmanager - Introduction to IBM Payment Manager
- http://www-4.ibm.com/software/webservers/studio/doc/v35/pagedetailer/EN/HTML/index.html - Page Detailer guide
- ftp://aixpdslib.seas.ucla.edu/pub/monitor/RISC/4.3/exec/monitor.2.1.5.tar.z

How to get IBM Redbooks

Search for additional Redbooks or redpieces, view, download, or order hardcopy from the Redbooks Web Site

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web Site for information about all the CD-ROMs offered, updates and formats.

Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.
Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of other companies:

Index

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

WCS V5.1 Performance Tuning

Performance guideline for your e-commerce site

Understand the hot spots in WCS

Explained with implementation examples

WCS V5.1 Performance Tuning describes tuning techniques for WebSphere Commerce Suite V5.1. WCS V5 is built on a new framework based on IBM WebSphere Application Server. One of the main subjects discussed in this book is studying the effects of the newly introduced Java technology.

This redbook covers the implications of WCS V5.1 in terms of performance, and covers the tuning techniques you will need for use with the new environment. e-Commerce administrators and developers will find it a useful addition to their technical library.

The contents of this redbook have been broken down by individual component of WebSphere Commerce Suite to allow quick access to the information you will need. WCS V5.1 Performance Tuning provides a single source of the information you will need to tune your e-commerce site, and is enhanced with examples, easy-to-follow instructions, and ample illustrations.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6258-00 ISBN 073842255X