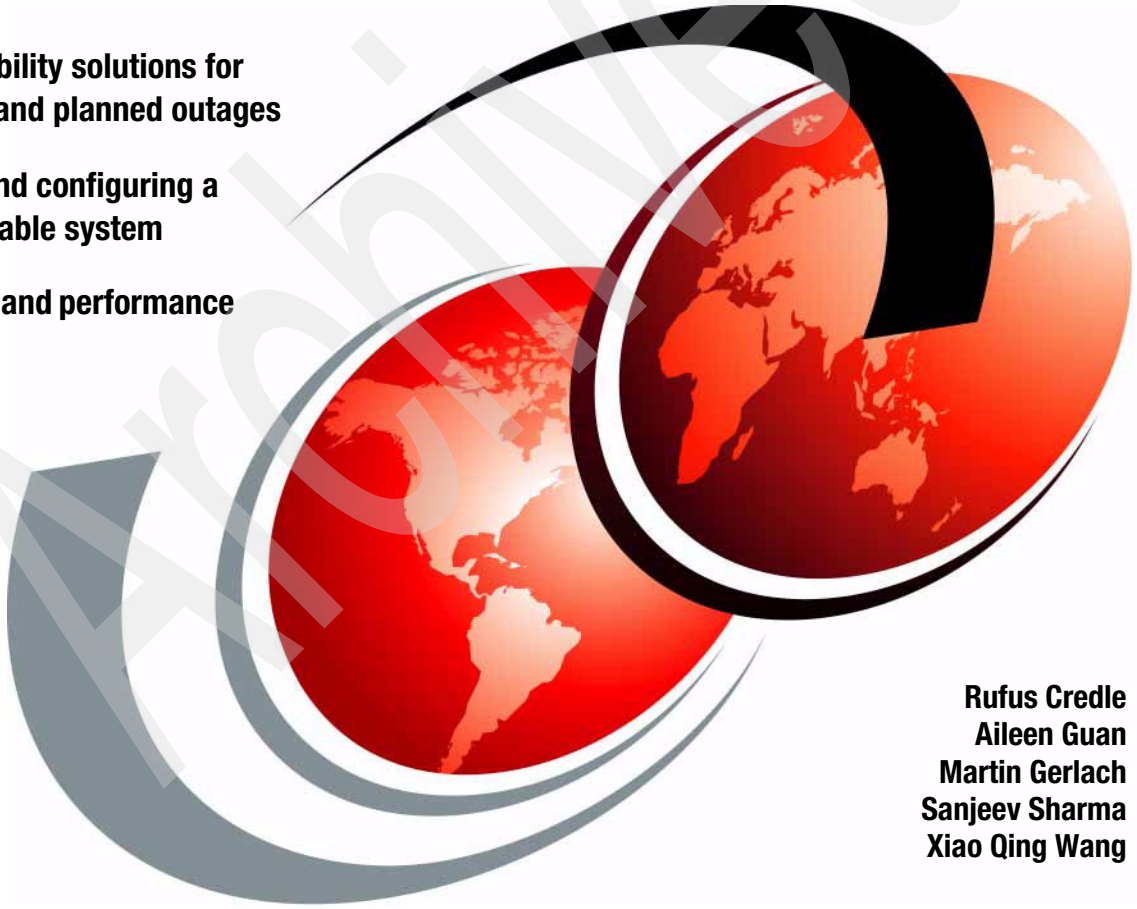
WebSphere® software

IBM

WebSphere Commerce High Availability and Performance Solutions

- High Availability solutions for unplanned and planned outages
- Installing and configuring a highly available system
- Monitoring and performance tuning



Rufus Credle
Aileen Guan
Martin Gerlach
Sanjeev Sharma
Xiao Qing Wang

Archive

Redbooks

ibm.com/redbooks

Archived

International Technical Support Organization

WebSphere Commerce High Availability and Performance Solutions

August 2008

Archived

SG24-7512-00

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

Archived

First Edition (August 2008)

This edition applies to IBM WebSphere Application Server Network Deployment 6.0 and WebSphere Commerce Version 6.0.

© Copyright International Business Machines Corporation 2008. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Archived

© Copyright IBM Corp. 2008. All rights reserved.

Archived

Archive

Archived

Archived

Archived

Archived

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

© Copyright IBM Corp. 2008. All rights reserved.

Trademarks

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

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

AIX®	HACMP™	Redbooks®
alphaWorks®	i5/OS®	Redbooks (logo)  ®
BladeCenter®	IBM®	S/390®
DB2 Connect™	Informix®	System i™
DB2 Universal Database™	iSeries®	System x™
DB2®	OpenPower®	Tivoli®
developerWorks®	POWER5™	WebSphere®
eServer™	pSeries®	zSeries®
Express Portfolio™	Rational®	

The following terms are trademarks of other companies:

Juniper, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

AMD, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

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

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

EJB, J2EE, Java, JavaScript, JavaServer, JDBC, JDK, JMX, JNI, JRE, JSP, JVM, Solaris, Sun, Sun Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

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

Archived

Preface

Building a high performance and high availability Commerce site is not a trivial task, from having the correct capacity hardware to handle the workload to properly test the code change before deploying in production site. This IBM® Redbooks® publication covers several major areas that need to be considered when using WebSphere® Commerce and provide solutions for how to address them. Here are some of the topics discussed:

- How to build a Commerce site to deal with various kinds of unplanned outage. This include utilizing IBM WebSphere Application Server Network Deployment 6.0 and IBM DB2® High Availability Disaster Recovery (HADR) in the Commerce environment.
- How to build a Commerce site to deal with planned outages such as software fixes and operation updates. This includes use of the WebSphere Application Server Rollout Update feature and the use of Commerce Staging Server and Content Management.
- How to proactively monitor the Commerce site and prevent potential problems from occurring. Various tools are discussed, such as WebSphere Application Server build-in tools and Tivoli®'s Performance Viewer.
- How to utilize Dynacache to future enhance your Commerce site's performance. This includes additional Commerce command caching introduced in Commerce fix pack and e-spot caching.
- The methodology of doing performance and scalability testing on a Commerce site.

The team that wrote this book

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

Rufus Credle is a Certified Consulting IT Specialist at the ITSO, Raleigh Center. In his role as Project Leader, he conducts residencies and develops IBM Redbooks about network operating systems, ERP solutions, voice technology, high availability and clustering solutions, Web application servers, pervasive computing, IBM and OEM e-business applications, IBM System x™, System x, and IBM BladeCenter®. Rufus' various positions during his IBM career have included assignments in administration and asset management, systems engineering, sales and marketing, and IT services. He holds a BS degree in

© Copyright IBM Corp. 2008. All rights reserved.

business management from Saint Augustine's College. Rufus has been employed at IBM for 28 years.

Aileen Guan is a Software Services Specialist based in Toronto, Canada. She is a part of WebSphere Commerce Advanced Technical Services affiliated with IBM Software Group, Application and Integration Middleware Software division. She is her team subject matter expert on high availability and also a subject matter expert on Commerce migration. Aileen co-authored a whitepaper on Migrating WebSphere Commerce in a clustered environment. Furthermore, she specializes in performance tuning in a large-scale Commerce environment, having done numerous performance tuning engagements for Commerce customers.

Martin Gerlach is a Software Engineer working for the IBM Centers for Solution Innovation in Hamburg, Germany (CSI::Hamburg), which is part of IBM Global Business Services. He has seven years of experience in developing end-to-end Internet applications and business integration software in research as well as in service delivery teams, using WebSphere Application Server, WebSphere Commerce, WebSphere Portal, IBM DB2, Oracle®, and various open source software. He holds an MSc degree in Computer Science/Distributed Systems from the Hamburg University of Applied Sciences. His areas of expertise include WebSphere Commerce high availability, J(2)EE and WebSphere Commerce application development, WebSphere Portal application development, Web services, and SOA concepts.

Sanjeev Sharma is a Team Lead of the WebSphere Commerce development organization in IBM Canada's software development lab in Toronto. He has nine years of experience in the WebSphere Commerce and database administration fields. He holds a computer engineering degree from McGill University in Canada. His areas of expertise include solution design, development, installation, performance, integration, and testing. He has written extensively about WebSphere Commerce best practices, WebSphere Commerce migration, and test methodologies.

Xiao Qing Wang joined IBM in 2005 and now works in the WebSphere Commerce System Verification Test team in IBM China Software Development Lab. He holds a master's degree in software engineering from BeiJing JiaoTong University in China. His areas of interest include automation test, test tools, unified development process, and performance analysis. He has written extensively on the WebSphere Commerce database high availability solution. In his spare time, he enjoys sports, collects stamps, and travels.

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

Tamikia Barrow, Margaret Ticknor
International Technical Support Organization, Raleigh Center

Thanks to the authors of the previous editions of this book and other contributing specialists.

The *WebSphere CommerceWebSphere Commerce V5.5 Capacity Planning*, ZG24-6733, Redbooks team of experts from the IBM Software (SWG) and Systems and Technology (STG) groups: Zamil Janmohamed, Chris Moss, Jose Antonio Roa, Sean Holden, Leny Veliyathuparambil, Charek Chen, and Joseph Fung.

The *WebSphere Application ServerWebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392, Redbooks team: Gang Chen, Andre de Oliveira Fernandes, Cristiane Ferreira, Rodney Krick, Denis Ley, and Robert Peterson

The *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0* whitepaper team: Jennifer Allan, Sidy Doumbia, Meng Fu, Polina Gohshtein, Aileen Guan, and Mark Kershaw

Experts from the IBM Software (SWG) and Systems and Technology (STG) groups: Michael Smith, Andrew Jones, and Sandra K Johnson.

Venkat Venkataraman, Program Director, IBM Power Systems
IBM Austin

Vickie Hessenius, SWG WW Midmarket Program Manager
IBM Rochester

Thomas Pack, Executive IT Architect, SMB Solutions, IBM Solution Builder Express Portfolio™ Development
IBM Research Triangle Park

Ingrid Moulckers, STSM, CTO ISV and Developer Relations Technical Enablement, Master Inventor
IBM Austin

Marcela Adan, IBM STG Small and Medium Enterprise (SME) programs, Certified Executive IT Specialist, iTC
IBM Rochester

Kelly Schmotzer, Senior Marketing Manager, GB MM High Volume Program Manager
IBM Cleveland

Steffen Eckardt, Advisory IT Architect (WebSphere Commerce, RFID)
IBM Germany

Archived

Jennifer Allan, Sidy Doumbia, Meng Fu, Polina Gohshtein, Mark Kershaw, John Hasty, and Sammy Chow, WebSphere Commerce Support and Services
IBM Canada

Daniel Owusu-Afari, Anson Y. Chan, WebSphere Advanced Technical Services
IBM Canada

Andres Voldman, Ali Asghar, WebSphere Commerce Advanced Technical Services
IBM Canada

Tatiana Jimenez, SWG Services WebSphere Consultant
IBM Boca Raton

Joseph Spano, Consulting Architect - ISSW Performance Practice
IBM Poughkeepsie

Mark Ho, Keri-Anne Lounsbury, James Tang, Kevin Yu, Software Services for WebSphere
IBM Canada

Stacy Joines, Distinguished Engineer, ISSW Performance and Enablement
IBM Research Triangle Park

Darl Crick, STSM, ISSW Performance
IBM Canada

John Hasty, WebSphere Services
IBM Austin

Scott Guminy, WebSphere Commerce Architect
IBM Canada

Robert Dunn, WebSphere Commerce Performance
IBM Canada

Jacob Vandergoot, Kevin Kam, David Yuan, WebSphere Commerce Development
IBM Canada

Rohit D Kelapure, Andy Chow, WebSphere Application Server Development
IBM Research Triangle Park

Nadir Anwer, Integrated Technology Delivery, Server Systems Operations
IBM Canada

Archived

Merla Black, DB2 Data Management
IBM Lexington

Robert Wilson, DB2 Development
IBM San Jose

Steve Rosengren, Performance Technology Practice
IBM Austin

Non-IBMers: Yung Wu, former WebSphere Commerce Support and Services and Bryan Einwalter, Sears Performance Team Manager

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

Archived

- Mail your comments to:

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

Archived

Part 1

Getting started

High Availability of eCommerce sites and of performance are closely related. In the case of eCommerce sites, High Availability must be spoken of in terms of the performance of the site.

A site that is online but not performing to give any indication of it being available can, of course, not be considered available. Thus, the performance metrics such as response time and throughput of the site have to be factored into the very definition of High Availability.

In this part of the book, we discuss the definition of High Availability and introduce a number of performance terms and other WebSphere Commerce terms that are used throughout the reminder of this book.

© Copyright IBM Corp. 2008. All rights reserved.

Archived

1

Introduction

As the reliance of businesses on eCommerce increases so does the need for an eCommerce site to provide uninterrupted and steady services. High Availability sites are able to withstand faults by either detecting them or by tolerating any failures caused by them.

Archived

© Copyright IBM Corp. 2008. All rights reserved.

1.1 Introduction to some key High Availability terms

As mentioned above, High Availability, in the context of eCommerce sites, has additional requirements above and beyond the site being merely online. In this section, we define High Availability and some terms associated with it.

1.1.1 High Availability

High Availability refers to reliably providing services to all the users of the system within a reasonable response time for a long duration of time.

Let us expand on the key, underlined attributes that define High Availability.

A system that is operational but not reliable, or a system that is operational but not acceptably responsive, is not considered to be available. The response times are defined by the business requirements for the site.

The number of users refers to the peak number of users, again, as defined by the business requirements.

Two implementations of High Availability can be distinguished from one another by the duration for which they remain available. Obviously, a system that is unavailable for few minutes a year is better than a system that is unavailable for a few minutes every month. Systems implemented for High Availability are usually assigned a certain percentage number indicating the level of their High Availability. For example, a 99.999% availability refers to unavailability (or outage) of a system for only up to 5.26 minutes per year, whereas 99.9% availability refers to unavailability of system for up to 526 minutes (7 hours 46 minutes) per year. A 100% availability is also called *continuous availability*. Such a solution is virtually impossible, but still a valid ideal goal to aspire to. Any solution that attempts to have continuous availability tends to be very expensive and very complex to implement as well as manage.

Generally, High Availability refers to 99.9% or higher availability. Also, many products and businesses do not count any planned outage towards High Availability calculation, as any planned outage may be managed in such a way that it does impact the business operations.

It is not always an easy task to assign a specific availability number to a given system. The complexity of this task increases as the number of components of a system and dependencies amongst them increase. Furthermore, each component or sub-system may have different availabilities under different states. And, even if a High Availability number can be assigned, it is very difficult to validate it since such a system should rarely fail.

Archived

A WebSphere Commerce site may be an aggregation of many components such as database servers, LDAP servers, IBM WebSphere MQ servers, WebSphere Application Servers, Web servers, Load Balancers, fire walls, and so on. It is a very complex task to ascribe a High Availability number to such a site. Instead of focusing on a specific availability number, our focus in this book is to have the ideal goal of ensuring continuous availability. To reach this ideal goal, we need to constantly watch the performance and workload of the site, and we need to keep reinforcing the components according to their strengths and weaknesses to ensure that no unplanned outage may occur. The reinforcement can be managing redundancy of the component, their configuration, or their performance.

1.1.2 Failover and mutual failover

Failover refers to the process of diverting services to a secondary system when the primary system fails. If the two systems are mutually secondary systems for one another, then such a process is called a mutual failover.

1.1.3 Switchover

A switchover can be considered as a safe, deliberate failover.

1.1.4 Fail back of fallback

Fail back or fallback refers to bringing the failed system back to handle services.

1.1.5 Nodes, cells, and clusters

WebSphere Commerce is an WebSphere Application Server application. WebSphere Application Server provides clustering capability that is exploited fully in this book.

Node

Node has two different meanings in this book, depending on the context in which it is used.

A node is a single machine or machine partition with a unique IP host address on which you install one or more WebSphere Commerce components.

When discussing federation, however, a node is a single occurrence of WebSphere Application Server and the applications that run inside the occurrence of WebSphere Application Server.

Archive

Cell

A node in a cell might or might not be running the same enterprise application as other nodes in the same cell. Cells are arbitrary, logical groupings of one or more nodes in a WebSphere Application Server distributed network that are managed together. In this definition, a node is a single WebSphere Application Server profile. One or more cells managed by a single occurrence of IBM WebSphere Application Server Network Deployment deployment manager are called a deployment manager cell.

Cluster

Clusters are groups of servers that are managed together and participate in workload management. Clusters are responsible for balancing workload among servers. Servers that are a part of a cluster are called cluster members. When you install an application on a cluster, the application is automatically installed on each cluster member.

Clusters were known in previous releases as server groups or clones. The act of creating clusters is called clustering. Clustering was known as cloning in previous releases.

Profile

A profile is a runtime environment for J2EE™ applications. Each profile is made up of configuration settings that are customized and specific to the profile itself. All profiles also share the same JAR files and runtime code from the WebSphere Application Server installation.

A WebSphere Commerce instance is a J2EE application that is run by a server in an WebSphere Application Server profile.

Federated or managed environment

If you have several environments with several applications on several machines, you will by default have several administrative consoles. To simplify management, you may want to use a single administrative console. To do this you federate to a deployment manager. The federation process aggregates the configuration from all of the nodes to a single point of management at the deployment manager. The deployment manager provides the ability to manage multiple servers from a single point. The deployment manager is responsible for distributing the configuration across all of the machines that it manages. To do this there is a network of servers that communicate with each other. They share information about the state of the configuration on each node. This grouping of servers is called a cell. The cell encompasses the DMGR and all of the nodes and processes that it is managing.

Archived

The DMGR contains the master configuration. You must make changes in the master configuration by connecting to the deployment manager.

Clustering

Clustering is where you run a single application on many servers. This allows you to take advantage of extra hardware to get better performance. It also gives you the ability to have failover support. If one server goes down, there is another one that will handle the processing. Think of a cluster as a group of servers that all act together as one. A cell can have any number of clusters within its configuration. You must be federated to create and manage clusters because they span multiple nodes.

Vertical - all members of the cluster on the same machine

You would do this if your Java™ Virtual Machine (JVM™) cannot use all of the memory on the machine. For example, the max heap size for a 32-bit JVM is 4 GB, while in practice you would want to stay less than 1.5 GB. If your machine has 8 GB, one JVM cannot use all of the memory. In this scenario you could create a vertical cluster member so that you can make effective use of all of the memory on that machine. Of course, you need to ensure that your system has enough processing power for the two JVMs.

Horizontal - cluster members on several different machines

This gives you hardware failover and allows you to spread the workload over many machines to increase throughput by using more CPUs and memory.

1.2 Introduction to different performance metrics and terminology

An eCommerce site's performance is defined in terms of its ability to handle workload, which is defined in terms of throughput, response time, capacity, and reliability.

Performance testing, also known as load testing, is used to validate these various performance metrics.

1.2.1 Workload

Workload is the amount of work that a site either handles or needs to handle. For example, this work can be due to shoppers browsing or placing orders, or business analysts administering the content offered by the site.

In performance testing or load testing, this refers to the simulated workload that mimics the anticipated production load. That is, it is supposed to mimic both the actions of multiple shoppers or the type of incoming user traffic accessing the site, as well as a distribution of various operations possible. Inherently, performance testing implies testing concurrency of all these operations being done by various shoppers or users.

The workload defines how the performance of a system is evaluated. A workload should have the following characteristics:

- Measurable: A metric that can be quantified, such as throughput and response time.
- Reproducible: The same results can be reproduced when the same test is executed multiple times.
- Static: The same results can be achieved no matter for how long you execute the run.
- Representative: The workload realistically represents the stress to the system under normal operating considerations.

1.2.2 Transaction

In the context of eCommerce, we define a transaction as a request-response pair from the context of user. For example, an HTTP transaction is the occurrence of the following sequence of events:

1. An HTTP request sent by a browser to the eCommerce server
2. An HTTP response, for the request sent in step1, from the server to the browser

A response may be a simple HTML response or it may be a composite response in which the content is aggregated from various components, such as JSPs, images, Java Scripts, and so on, and sources, which may be local or remote.

1.2.3 Scenario

A scenario is a sequence of interactions that are logically grouped together because they represent a user task. This scenario includes the series of actions that virtual users execute while interacting with the WebSphere Commerce site. Inherently, this also includes different interfaces or utilities that might be used to interact with the WebSphere Commerce site. This scenario is decided by the usecases for your site design. Usecases should be available in your site design documents.

Archived

1.2.4 Throughput

Throughput means number of customer requests relative to some unit of time. For example, if a Commerce server can handle 10 customer requests simultaneously and each request takes one second to process, this site can have a potential throughput of 10 requests per second. Let us say that each customer on average submits 60 requests per visit. Then we can also represent throughput by estimating six visits/minute or 360 visits/hour.

Two common units for throughput are *page hits/second* and *scenarios/hour*:

- Page hits/second can have different meanings. For example, a page hit could refer to the single click that a customer does to download a composite Web page made up of many fragments. Alternatively, it could mean the total number of pages that actually get downloaded as result of that single click. Both of these interpretations have their significance.

  However, most of the time we are interested in the first interpretation. That is, each page hit is a Web page from a shopper or user's point of view. So, for example, if the page that a customer requested was redirected to another page then the page hit would still be counted as one page. Similarly, if a JSP™ aggregates static and dynamic content from many other files the page hit would still be considered for a single page.

- Scenarios/hour is another very useful unit of throughput. A scenario is your test scenario approved by your test plan approvers (including your business analysts) to mimic your real business scenarios. In our testing we prefer this unit, as page hits/second can be misleading since it changes for each scenario, whereas scenario/hour is inherently relative to a scenario.

1.2.5 Response time

Keeping in mind the discussion we had for the throughput, the page hit response time is different from the http interaction response time, as one page hit may result in multiple http interactions.

1.2.6 Capacity

Capacity of a site is the maximum throughput that can be provided within required response time with high reliability. In this book, we call this the maximum business capacity available.

The maximum business capacity available is to be distinguished from the peak capacity expected and the maximum system capacity.

The peak capacity expected is the system capacity at the peak expected workload. This should always be lower than the maximum business capacity available.

1.2.7 Failover

Failover is a concept that can be used to evaluate how fast can a site be recovered from any planned or unplanned outage, which has a closed relationship with the High Availability site. Recovery can be achieved by automation management tools or manually, which is up to which High Availability strategy you are applying in your site.

Generally, the measurement to failover can be classified by two aspects:

- System's point of view

  The system view is much narrower than the customers' view, so we should only concentrate on one specific tier. That is, if it is a database outage, from the system's point of view, how long can a database successfully restore from disaster should be a key measurement for failover from the system's point of view.

- Customers' point of view

  This is much more complex than evaluating failover time from the system's view, since WebSphere Commerce builds above a complex infrastructure, where the application server, Web server, database, and WebSphere Commerce should work together to deliver sufficient utilities. So that the information that the users (customers) can see should be the data that have been processed by the Web server, application server, and database. If there is an outage in one tier, it should impact tiers in the entire environment, while the major impact should be to customers. From the beginning of system failure untill the functionality becomes workable again to customers is the measurement with which to evaluate the failover capability from the customers' point of view.

1.3 Introduction to different WebSphere Commerce environments

Different WebSphere Commerce (customer) sites use a variety of names when referring to various WebSphere Commerce environments within their site.

1.3.1 Development environment

A setup of WebSphere Commerce Developer, based on the IBM Rational Application Developer, is the WebSphere Commerce development environment.

1.3.2 Runtime environment

Runtime environment refers to any setup that has any edition of WebSphere Commerce set up other than the WebSphere Commerce Developer. This includes WebSphere Commerce Express, WebSphere Commerce Professionall, and WebSphere Commerce Enterprise Edition. Any of these setups can be for the production environment or for an environment that is used for testing or aggregating the assets before they are pushed or published on the production environment.

Development environment

The development environment refers to the machines that you use to develop custom code. For development environment migration, use the in-place migration approach, where all the migration steps are done on your existing machine. We recommend that you have a second development machine available, to maintain your existing WebSphere Commerce site.

Lightweight test environment

WebSphere Commerce Developer has a fully functional lightweight test environment that emulates the WebSphere Commerce production environment but with a greatly reduced memory footprint and startup time.

Also available is the full WebSphere Commerce test environment, which provides support for more advanced development tasks, additional configuration options, and end-to-end testing on a local WebSphere Commerce Server.

Runtime environment

The runtime environment refers to your test environment as well as your production environment where you deploy your custom code and serve your site.

Production environment

A production environment refers to your production site that is open to your customer, shoppers, or user access. If you have a for-profit business then this is your revenue-generating site.

Staging and authoring environments

A staging environment allows the site administrators to update the data and test the changes, and then propagate the change to the production server. This is

Archive

useful for testing updates to the product catalog, but it is also important for testing new shopping process commands.

Test and stress environments

In addition to the staging environment, as discussed above, many sites, especially large sites, may have decided to have one or more test environments to test or experiment a variety of updates or scenarios.

An environment for stress testing your system should mimic your production environment closely. If the stress environment does not mimic your production environment closely then you need to figure a correlation between the results of your performance test cases and how those results translate to the impact on the performance of your production site, as discussed in Chapter 6, “WebSphere Application Server High Availability” on page 55.

1.4 Considerations for implementing High Availability solution

Important: High Availability of eCommerce sites does not simply mean that they remain online, but that they must continue to provide a high level of performance.

The level of High Availability that a business requires is dictated by the business requirements. So, in this sense the level of High Availability is a relative term. High Availability for eCommerce sites has some specific requirements above and beyond the site simply being *online*. Here we define what High Availability entails in context of eCommerce sites.

Many High Availability implementations focus on failover and disaster recovery. However, a system must also ensure continuous capacity and performance as defined by its business requirements. This is especially true for eCommerce sites.

You also need to monitor your system proactively so that any change in workload, say due to aging data or a change in the distribution and type of incoming user traffic, and so on, does not lead to degradation in performance or risk to High Availability. Monitoring capability and performance tuning is thus an integral part of implementing a High Availability solution.

Archive

1.4.1 Continuous business capacity and performance

Ensuring that a system is able to handle a certain peak capacity and performance, as required by its business, requires eliminating single points of failure and building redundant capacity above and beyond the peak capacity. If part of the system goes out then the redundancy in the system ensures that 100% of the business capacity requirements can still be met by the system without any impact to the business functionality or performance.

Redundancy may need to be implemented at various levels, including:

- Hardware redundancy: In a clustered environment, such as the one employing WebSphere Application Server, this may refer to setting up horizontal cluster members, which are set up on different physical machines.
- Software redundancy: In a clustered environment, such as the one employing WebSphere Application Server, this may refer to setting up vertical cluster members, which are set up on same physical machine.
- Data redundancy: In a system that relies on a source of data, this refers to having more than one identical source of data. If one source of data goes down, the other sources of data continue to handle service requests. For example, DB2 HADR can provide one such solution (see Chapter 5, “Database tier High Availability” on page 39).
- Communications redundancy: This refers to the retry logic within an application whereby if the application is not successful in an operation that required communication outside of the application, then it retries the operation a certain number of times before signaling a failure. Retry logic may be implemented to identify faults in a certain hardware, software, or data source and to try the alternate, redundant, or standby hardware, software, or data source.
- Network redundancy: This refers to the hardware, such as bridges, access points, transmission cables, and so on (and the associated software), redundancy to ensure that the any network failures do not adversely impact a site’s performance.

1.4.2 Failover support and disaster recovery

Faults are distinguished from failures by the fact that the faults of the system may or may not cause failures. If faults exist in a system and, in a given scenario, no action is performed with the faulty component that may expose the fault, then the failure will not happen.

Failures are distinguished from disasters by the fact that failures may or may not cause disasters. While failures tend to be specific and localized, the disasters

tend to be widespread in their scope, usually sudden and highly disruptive. Disasters can be natural, man-made, or a combination of the two. Disasters can be earthquakes, volcanic eruptions, fire, power failure, theft, terrorist attack, and so on.

In failover solutions the tasks of failed components are assumed by the operational components. Disaster recovery solutions are implemented to localize the scope of the failures and to minimize or eliminate any disruptions that the disaster might otherwise have caused.

Active-passive failover support

This is a system that includes one or more redundant components passively waiting, in a stand-by mode, for failures to occur, and then they become active by assuming the role of the failed components. The active-passive failover support of a system is frequently defined as an m:n ratio of m active components (which are actively processing workload) to n passive components (which are passively waiting in stand-by mode).

Thus, a 1:1 active-passive system has 100% redundancy with one passive component per active component, whereas an n:1 system has one passive component for n active components.

Active-manual failover support

The failed component or system is manually swapped for the passive component. For example, in case of fire, when one system is fully destroyed, its backup image can be restored onto another system and it could replace the damaged system. Usually, an active-manual failover support cannot provide high availability, especially if the failover detection is also manual.

Active-active failover support

A system with active-active failover support has duplicate components that are all simultaneously active and processing workload. In case of a failure, the overall capacity of the system is lowered until the failure is corrected and the failed component is active once again. Depending on the business capacity requirements, this failover support may also require building redundant capacity in the system.

1.4.3 System monitoring and performance tuning

To ensure that your system or site is highly available you must monitor it not only for the failover support but also for the acceptable performance as the workload or site traffic changes with time. Refer to Part 5, “Monitoring and performance tuning” on page 321.

1.4.4 Performance testing

Performance tuning and performance testing go together. You must not adjust or experiment with performance tuning on your production environment directly. Any changes to the workload distribution on your production environment should be mimicked on your test or staging environments, and performance tuning should be tested or validated on them first.

1.5 Types of system outages

System outages can be due to different causes and can have varied manifestations and extent.

1.5.1 Different scopes of system outage

System outage can be complete or partial outage.

Complete outage

This is when the outage is complete the system has 0% capacity to respond within a required response time.

Partial outage

A partially unavailable system has reduced capacity to respond or process its workload within a required response time. The required response time is defined by the business requirements.

1.5.2 Different causes of system outage

Ideally, system outage should occur only in a planned way during a service window in which you will not impact your users or your revenue generation. However, it is possible that an outage may happen when it is not expected. You should have a High Availability solution that can tolerate system outages, but, at the same time, you should ensure that you have an established process, as a last resort, for when that is not possible. For example, who will handle what aspect of the system outage and how the communications will be sent and so on?

Planned outage

A planned outage can be either a complete or a partial outage. It is possible that some businesses may accept lower capacity and performance of their systems

Archived

during planned outage, say, during off-peak seasons, days, or hours. In such cases the planned, partial outages are much more palatable even though they may last longer than complete planned outage. Such planned, partial outages may not drop the service availability by deploying *switchover* of services from one component (for example, one server) to another. A switchover can be considered as a safe, deliberate failover. WebSphere Application Server provides *rolling upgrade* functionality that employs this scheme. For more information about rollout upgrades refer to Part 7, “Maintenance” on page 561.

Unplanned outage

Unplanned outage is an unscheduled outage that happens if the system was not set up to handle it by implementing redundancy or failover support, and so on.

Hardware

Unplanned outages can happen due to hardware failures resulting from hardware faults, power failure, natural disaster, and so on.

Software product

Software crashes can lead to unplanned outages as well. Although such a crash should be investigated so that the associated outage can be avoided in the future, most outages can be recovered by just restarting the software application. As such, setting up an automated recovery approach or just restarting the application can be your first line of defense.

Capacity

Even though all the system components may be active and operational it is possible that the system is not operating at the capacity required by your business specifications. For example, the response time has increased past your acceptable time. This could happen due to an increase in workload, but the possibility increases significantly if some of the redundant components fail.

1.6 High Availability solution for WebSphere Commerce

This book is divided into eight parts along the life cycle events of a site development, for example, install, development, testing, performance tuning, maintenance, and so on.

The central theme of High Availability and performance permeates throughout the book.

Archived

In Table 1-1, we list some of the potential single points of failure in a WebSphere Commercesite and links to some of the key, relevant sections that contain the High Availability and performance considerations.

Table 1-1 Single points of failure in a WebSphere Commerce site

Points of failure	References
Load Balancer	Refer to 7.2, “Introduction to Load Balancer High Availability” on page 85.
Web server	Refer to 7.1, “Introduction to Web server High Availability” on page 81.
WebSphere Application Server	Refer to Chapter 6, “WebSphere Application Server High Availability” on page 55.
WebSphere Application Server node agent	Visit http://www.redbooks.ibm.com/abstracts/SG246451.html?Open and http://www.redbooks.ibm.com/abstracts/SG247304.html?Open.
WebSphere Application Server ND	Visit http://www-306.ibm.com/software/webservers/appserv/was/network/features/?S_CMP=rnav.
Database	Refer to Chapter 5, “Database tier High Availability” on page 39.

Archived

Archived

2

Project planning for High Availability and performance

In addition to ensuring the High Availability of system resources, it is also imperative to manage the human resources to support the High Availability of the system. Project management, skill availability, and accountability of site assets are a few of the key aspects.

Although project management and the associated activities are beyond the scope of this text, we would like to provide a checklist that can help project managers better manage their team that supports High Availability systems.

Archived

© Copyright IBM Corp. 2008. All rights reserved.

2.1 Identify your scenario

If you are planning on implementing a High Availability solution then there are two possible scenarios that have different project requirements and milestones. In one scenario, you may be an existing customer who wants to convert your existing single point of failure to a High Availability component. In the other scenario, you may be a new customer building a new High Availability site. The two scenarios require different implementation steps.

It is usually easiest to build a site with High Availability. If, however, you are converting your existing single point of failure to a High Availability component then you must transition the architecture in such a fashion such that there is minimum impact to the site's operation. For example, converting a standalone Web server to a cluster of Web servers and a Load Balancer would require you to set up the Load Balancer with the same virtual IP that the standalone Web server used to have. The site operation would be impacted for the duration of the switchover of the virtual IP address.

2.2 Identify your resources and skills requirements

It is also imperative to be able to account for all the assets had by your site. For example, you should have all the source code for your custom code in a code repository. Building an inventory of assets would enable you to track (or identify the need for) High Availability implementations for the various components, as well as, and perhaps more importantly, High Availability processes. For example, do you have enough build, test, or staging environments between your development machines and production environment? You must test the assets that you develop before you deploy them on to your production environment. Insufficient testing prior to deployment puts your production environment at risk and may impact its availability.

2.2.1 Inventory of site assets

Take an inventory of your existing WebSphere Commerce site assets, which include:

- Topology information

  Topology information is required to figure out any single points of failure, as well as any performance bottlenecks that might be there. For example, after you have ensured that you do not have a single point in failure, then what level of redundancy do you have in your site? How many cluster members are allowed to go offline before it impacts your site's performance or availability?

Is the number acceptable? Along the same lines, is the failover strategy or detection time, such as in case of IBM DB2 HADR, acceptable to your business? Gathering information such as the following is required to build a site's topology:

- Machines
- Instances
- WebSphere Application Server Cluster members
- Web servers
- Databases
- High-level architecture of the site
  - Development environments
  - Build environment
  - Test environment
  - Staging/Authoring environment
  - Production environment

- Custom code developed

  Ensure that you have all your custom source code available in a code repository. There may be situations when you need to diagnose or recompile the code and having the binary will not be sufficient. Also, use the following list to identify whether you have appropriate test environments to properly test for any changes in the following that may be required:

  - Enterprise Java Beans and commands
  - Custom Java code
  - Java Server Pages
  - Database schema customizations
  - Code deployment scripts
  - Data loading scripts
  - WebSphere Commerce tools, and so on

- Existing business processes

  To ensure that existing processes continue to work flawlessly, you first need to identify them. You also need to identify the priority and best times of execution of various maintenance or repetitive tasks. For example, what is the best time to mass load data or publish catalog content into your database? Or, when should the scheduled jobs run and with what priority, and so on? The priority and schedule of the jobs have significant impact on a site's performance.

Archived

- Integration points

  Integration points require the same considerations as the core WebSphere Commerce assets that you may have. These include:

  - WebSphere family of products, for example, MQ, LDAP, messaging, and so on
  - Third-party software, for example, payments, taxation, and so on
  - Back-end integration

2.2.2 Inventory skilled resources required

Inventory the skills of the people who will be involved in implementing, maintaining, and supporting your site. Consider whether you have the appropriate database, Java, coding, testing, and IT systems administration skills on your team:

- *Database administration skills* to help set up a High Availability database system, to performance tune, and to help troubleshoot any database performance issues, such as poorly performing SQL queries. Database performance is one of the most significant, if not the most significant, contributors to your site's performance and availability.
- *WebSphere Commerce development skills* to write any custom Java code. The developers should understand the basic concepts of how Java constructs and structures use JVM heap and how they may impact the performance.
- *Administration skills*, to minimize the impact to your production site's availability and performance during installation, upgrade, and configuration operations involving WebSphere Commerce and co-required software products, for example:
  - The operating system
  - WebSphere Application Server administration
  - IBM HTTP Server administration
  - Understanding of TCP/IP, HTTP, and Secure Sockets Layer (SSL) protocols

Archived

2.3 Plan your activities

Keeping with the theme of this chapter, here we provide a high level sample project plan, without delving deep into the details. The diagram in Figure 2-1 should be self-explanatory, but we review some of the key aspects of project planning below.

Attention: Performance and High Availability considerations must be an integral part of all design, development, test, maintenance, and even business discussions.

Putting performance considerations off until later in the project (until the performance test phase and, even at that stage, compromising the scope of performance testing) is a common, dangerous, and very expensive mistake.

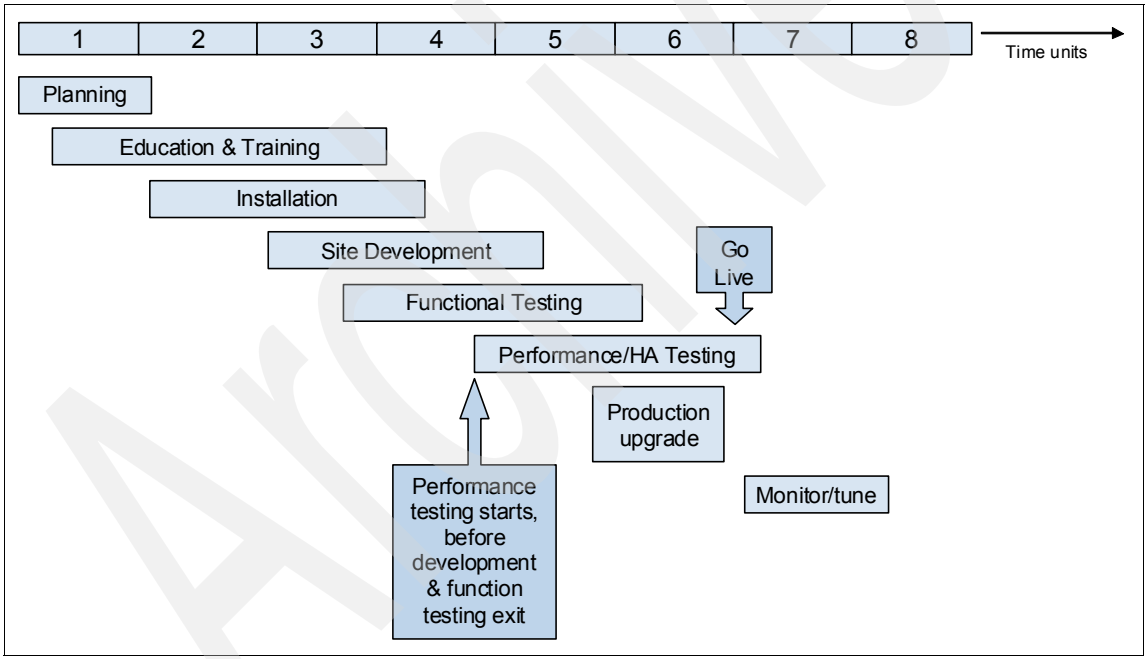


Figure 2-1 Planning your site development with High Availability and performance considerations

2.3.1 Education and training

Consider training your team in performance aspects administration and development. For example, in addition to being familiar with WebSphere

Commerce features, your developers should be familiar with the performance considerations of development and code profiling techniques. For more information refer to Part 4, “Design with performance in mind” on page 247.

2.3.2 Getting skilled help

Depending on the complexity of your site, your schedule, the skilled resources available on your team, and the critical nature of your business, you may consider getting help from an IBM services team or an IBM Business Partner.

You can outsource your complete project. Alternately, you can seek assistance from one or more experts to help with your project while you provide the rest of the resources for the project. The latter approach may be useful if you have skilled resources to assist the external experts. This approach has the benefit of having skilled resources helping you with the actual implementation tasks. It also helps the rest of your team in answering their questions and assisting with problem determination, if required. Either way, in your plan, ensure that you account for skills transfer from the experts to your team so that your team can function efficiently on the site when the expert leaves.

You may also consider outsourcing performance testing to an IBM services team, as they have the automated tools and processes to provide assistance with quick turn-around, and as performance testing is non-trivial in nature. We discuss this further in 20.1, “Why is it complex” on page 455.

Finally, if you identify skill shortfalls in your team and you require external assistance then bring help in sooner rather than later.

2.3.3 Site development life cycle

Figure 2-1 on page 23 is useful from a scheduling perspective, as you can see how some aspects of the site development life cycle can be executed in parallel to one another.

Figure 20-1 on page 457 goes into further details of the processes involved in the site development life cycle.

2.3.4 Scaling hardware versus performance tuning

Scaling hardware may be an easier solution from an implementation perspective than performance tuning, but it may not be easy to translate that into a business case for it.

Sometimes, however, the opposite is also true. Instead of a person spending months investigating how the performance of a well-tuned system can be improved further, it may be cheaper to add more hardware to your site.

Thus, you should consider the pros and cons of each approach.

2.3.5 Performance testing is critical

Important: Start performance testing as early as possible. The later you do performance testing the more expensive performance defects become.

In many projects testing is left for the end of the project and performance testing is usually last. As such, any delay in project execution impacts performance testing significantly. Any problems (such as performance issues) that happen during runtime usually have a much serious impact than any roadblocks you may encounter prior to going live. Also, many times performance issues are not trivial to address, so you do not want to discover them at runtime. So, it is imperative to plan for performance testing and tuning of your system sooner in your project, as well as allocating sufficient time to it.

2.3.6 Failover support for launch

You must also plan some sort of failover plan or roll-back plan should you encounter any catastrophic problem during the site launch or go-live.

Recall the two scenarios discussed in the beginning of this chapter. If you are building a new site then any catastrophe will delay your site launch. However, if you are upgrading your site (for High Availability or performance reasons) then this would have a significantly larger impact to your on-going business.

Throughout this book we discuss various techniques to address the latter. One common theme is to decide on a service window of your site that will least impact your user base, and then partially upgrade your site. If your upgrade is not successful for any reason then one alternative is to bring your site up with the remaining capacity and then work in parallel to either troubleshoot the problem or to revert the upgraded capacity back to the previous version or topology of your site.

Archive

Archived

3

Scenario for this book

In this chapter, we describe the scenario that is used throughout this book to demonstrate and test the discussed High Availability and high performance concepts. We also give an overview of the following parts:

- Setting up High Availability for WebSphere Commerce
- Developing high-performance WebSphere Commerce custom applications
- Monitoring and tuning WebSphere Commerce components
- Conducting performance tests for WebSphere Commerce
- Maintaining WebSphere Commerce components
- Migrating WebSphere Commerce

Archived

© Copyright IBM Corp. 2008. All rights reserved.

3.1 Topology

Figure 3-1 shows the basic topology of our WebSphere Commerce test environment. The diagram reflects the typical three-tier architecture of a J2EE application.

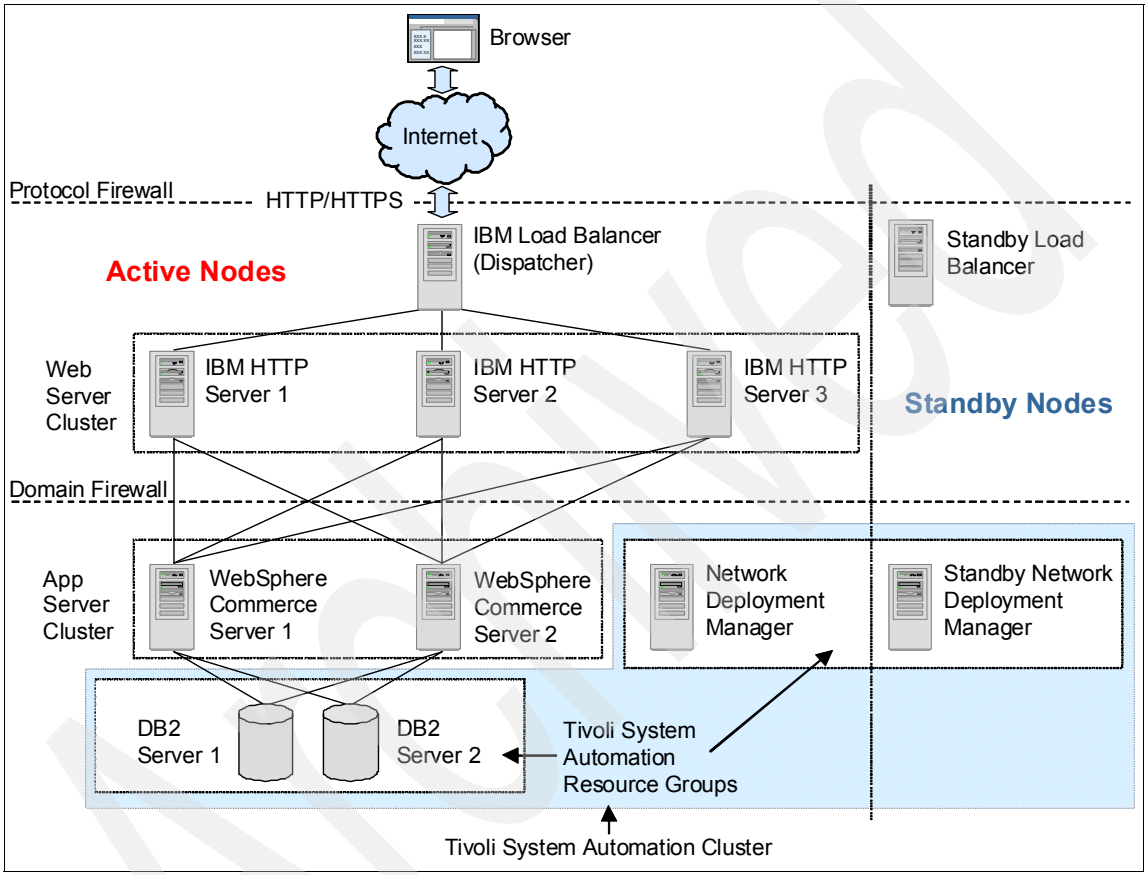


Figure 3-1 Topology diagram of our scenario

3.2 Chapters overview

The rest of the book is structured as described in the following sections.

3.2.1 High Availability

Part 3, “Install and configure a High Availability WebSphere Commerce system” on page 87, describes how to set up the WebSphere Commerce topology with High Availability.

As a special case, some of the nodes shown in Figure 3-1 on page 28 could actually be in different networks. We experimented with putting the following nodes into remote networks:

- HTTP Server 3: See 7.1.2, “IBM WebSphere Edge Components Load Balancer” on page 82 (NAT forwarding).
- 11.2.2, “NAT forwarding” on page 210.

The following Web site discusses frequently asked questions about WebSphere Application Server, including how to run it over multiple data centers:

http://www-128.ibm.com/developerworks/websphere/techjournal/0606_col_alcott/0606_col_alcott.html

Without the HA Deployment Manager capability of WebSphere Application Server XD, the most practical way to replace a failed DMgr is the cold/warm standby server, as shown in Figure 3-1 on page 28. For additional information about an alternative HA solution, go to the following link:

http://www.ibm.com/developerworks/websphere/library/techarticles/0304_alcott/alcott.html

3.2.2 Application development

Part 4, “Design with performance in mind” on page 247, describes how to develop your custom WebSphere Commerce application (for example, a shopping site) such that your site can be tuned to yield optimal performance.

3.2.3 Performance monitoring and tuning

Part 5, “Monitoring and performance tuning” on page 321, describes how to monitor the application at different tiers. We also provide hints on tuning parameters for each software component used in our test environment.

3.2.4 Performance test

Part 6, “Performance test” on page 451, describes test methodologies and tools for performance testing WebSphere Commerce sites.

Archived

3.2.5 Maintenance

Part 7, “Maintenance” on page 561, describes how to maintain and update all the different hardware and software in way that minimizes the impact of the maintenance and update operations on site High Availability and performance.

Archived

Part 2

High Availability solutions for unplanned and planned outages

Multiple High Availability solutions are introduced in this part, which mainly focuses on four areas:

- External clustering software
- High Availability solution in database tier
- High Availability solution in WebSphere Application Server node
- High Availability solution in Web server node

© Copyright IBM Corp. 2008. All rights reserved.

Archived

4

External clustering software

A typical WebSphere Commerce system spans across various tiers: Web server, application server, database server.

Each tier of product has its own specific solution to achieve High Availability. For example, at the Web server tier, Load Balancer is commonly used to distribute traffic across a Web server cluster. At the application server tier, WebSphere Application Server federation and clustering can be managed by the Network Deployment Manager.

In addition to these product-specific High Availability solutions, IBM supports several external clustering solutions. Using this external clustering software on its own, one may achieve High Availability for all servers in a WebSphere Commerce environment. This external clustering software can also be used to integrate with existing product-based High Availability solutions.

In this chapter, we introduce you to some external clustering software, such as:

- Reliability Scalable Cluster Technology (RSCT)
- Tivoli System Automation (TSA)
- HACMP™

Later in the book, we show you how to achieve High Availability with DB2 using RSCT and TSA.

© Copyright IBM Corp. 2008. All rights reserved.

4.1 Reliability Scalable Cluster Technology

Reliable Scalable Cluster Technology (RSCT) is a product that is fully integrated into TSA. RSCT is a set of software products that provides a comprehensive clustering environment for AIX® and Linux®. RSCT provides clusters with improved system availability, scalability, and ease of use. RSCT provides several basic components, or layers, of functionality:

- Resource Monitoring and Control (RMC)

  Resource Monitoring and Control provides global access for configuring, monitoring, and controlling resources in a peer domain.

  RMC is the scalable, reliable backbone of RSCT. It runs on a single system or on each node of a cluster, providing a common abstraction for the resources of the cluster. In a cluster, the RMC framework allows a process on any node to perform an operation on one or more resources on any other node inside the cluster.

  As the name implies, RMC monitors resources (disk space, CPU usage, processor status, application processes, and so on) and controls the system by performing actions in response to defined conditions.

- Resource Management (RM)

  There is a core set of resource managers provided by RSCT, and additional resource managers are provided by CSM for AIX and CSM for Linux. Resource managers provide low-level instrumentation and control, or act as a foundation for management applications. The following are the core resource managers of RSCT:

  - Audit Log resource manager
  - Configuration resource manager
  - Event resource manager
  - File System resource manager
  - Host resource manager
  - Sensor resource manager

- Cluster Security Services (CtSec)

  Cluster Security Services is a new security infrastructure that is used by RMC to authenticate a node within the cluster, verifying that the node is who it says it is. This is not to be confused with authorization (granting or denying access to resources), which is handled by RMC.

  CtSec uses credential-based authentication that enables:

  - A client process to present information to the server that owns the resource to be accessed in a way that cannot be imitated.

Archived

  - A server process to clearly identify the client and the validity of the information.
  - Credential-based authentication uses a third party that both the client and the server trust. In this release, only UNIX® host-based authentication is supported.
- High Availability Group Services (HAGS) is a distributed coordination, messaging, and synchronization service.
- High Availability Topology Services (HATS) provides a scalable heartbeat for adapter and node failure detection, and a reliable messaging service in a peer domain.

For more information about RSCT, see the IBM Cluster Information Center:

http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsctbooks.html

4.2 Tivoli System Automation

IBM Tivoli System Automation for Multiplatforms plays an important role in improving the availability and resilience of critical business processes delivered to users through self-managing and self-healing autonomic computing capabilities.

4.2.1 Introduction

IBM Tivoli System Automation manages the availability of applications running in Linux systems or clusters on System x, zSeries®, iSeries®, pSeries®, and AIX systems or clusters. It consists of the following features:

- High Availability and resource monitoring

  TSA provides a High Availability environment. It offers mainframe-like High Availability by using fast detection of outages and sophisticated knowledge about application components and their relationships.
- Policy-based automation

  TSA configures High Availability systems through the use of policies that define the relationships among the various components.
- Automatic recovery

  TSA quickly and consistently performs an automatic restart of failed resources or entire applications either in place or on another system of a Linux or AIX cluster. This greatly reduces system outages.

Archived

- Automatic movement of applications

  TSA manages the cluster-wide relationships among resources for which it is responsible.

- Resource grouping

  You can group resources together in TSA. Once grouped, all relationships among the members of the group are established, such as location relationships, start and stop relationships, and so on.

Figure 4-1 is the architecture of Tivoli System Automation.

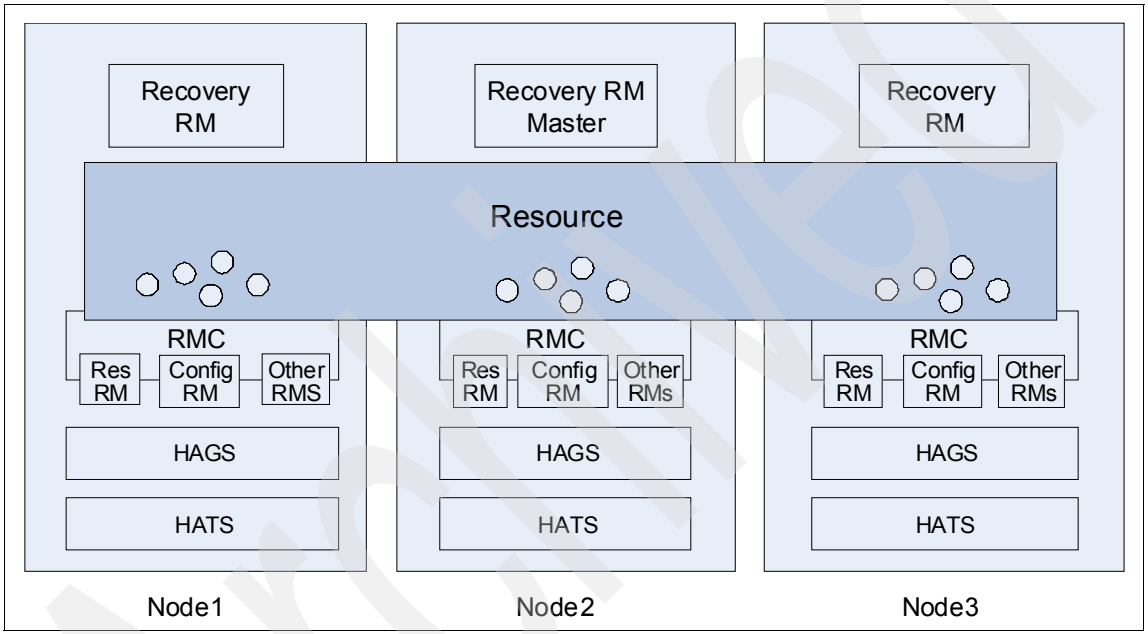


Figure 4-1 Architecture of Tivoli System Automation

TSA provides High Availability by automating resources, such as processes, applications, IP addresses, and others in Linux-based clusters. To automate an IT resource (for example, a DB2 database instance), you define the resource to TSA. Furthermore, these resources must all be contained in at least one resource group. If these resources are always required to be hosted on the same machine, they are placed in the same resource group. For more information about TSA, see the Tivoli Software Information Center at the following link:

http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.1.html

4.2.2 Terms in Tivoli System Automation

There are many terms that we need to use when we use Tivoli System Automation as our automation tooling:

- Cluster

  A cluster is a group of host systems under IBM Tivoli System Automation management.

- Resource

  A resource is any piece of hardware or software that can be defined to IBM Tivoli System Automation. These resources can be either defined manually by the administrator using the **`mkrsrc`** (make resource) command or through the *harvesting* functionality of the cluster infrastructure, whereby resources are automatically detected and prepared for use.

  - Fixed resource

    A fixed resource is a resource that has only a single instance within the cluster. It represents one entity that is defined for a single node, and this is the only node on which it runs.

  - Floating resource

    A floating resource is a resource that can run on several nodes in the cluster.

- Attribute

  A resource attribute describes some characteristics of a resource. There are two types of resource attributes: persistent attributes and dynamic attributes.

- Resource group

  Resource groups are the primary mechanism for automatic operations within TSA. A collection of resources can be included in a resource group, which can be called as a logical container. This container can help you control multiple resources as a single logical entity.

- Equivalency

  An equivalency is a collection of resources that provides the same functionality. For example, equivalencies are used for selecting network adapters that should host a same IP address.

- Relationship

  IBM Tivoli System Automation allows the definition of relationships between resources in a cluster. There are two different relationship types:

  - Start/stop relationships
  - Location relationships

Archived

- Tie breaker

  The tie breaker is used to determine which subcluster will have an operational quorum.

4.2.3 Start with Tivoli System Automation

The general steps to start Tivoli System Automation are:

1. Create and administer a cluster.
2. Define resources for the RSCT cluster defined previously.
3. Define or customize the automation policy.
4. Register the automation policy with resources.
5. Go online and validate the entire cluster.

In 9.1.3, “Installing Tivoli System Automation” on page 155, we give a detailed installation and configuration introduction about how to use TSA to manage and monitor DB2 HADR, which is strictly following the steps listed above.

4.2.4 Relationship with RSCT

Reliable Scalable Cluster Technology (RSCT) is a product fully integrated into IBM Tivoli System Automation. Since RSCT is a set of software products that together provide a comprehensive clustering environment for AIX and Linux, it is the infrastructure to provide clusters with improved system availability, scalability, and ease of use. You should make sure that the RSCT infrastructure is working before you install TSA.

4.3 HACMP

In general, High Availability is achieved by making systems redundant. The more system redundancy, the higher the level of availability that can be achieved. IBM HACMP for AIX provides a highly available computing environment by adding software and redundant hardware components. It automatically switches applications and data from one system to another in an HACMP cluster after a hardware or software failure.

In a WebSphere Commerce environment, HACMP is most commonly used for the database tier. For more detailed information about HACMP setup, refer to the HACMP documentation available at:

http://www.ibm.com/servers/eserver/pseries/library/hacmp_docs.html

5

Database tier High Availability

Currently, most of the transaction processing systems are driven by database management systems, so that more and more e-businesses are looking for an uninterrupted and highly available infrastructure in their transaction processing systems. This chapter focuses on IBM DB2, and introduces some existing solutions to help DB2 achieve 24x7 High Availability:

- DB2 HADR
- HACMP
- DB2 SQL replication

Archived

© Copyright IBM Corp. 2008. All rights reserved.

5.1 High Availability Disaster Recovery

DB2 Universal Database™ (DB2 UDB) High Availability Disaster Recovery (HADR) is a database replication feature that provides a High Availability solution for both partial and complete site failures. In this solution, data can be protected from being lost by replicating from the source database (called primary database) to the target database (called the standby database) without significant performance impact to the normal transactions.

5.1.1 Introduction

High Availability (HA) strategies enable database solutions to remain available to process client application requests despite hardware or software failure, which can ensure that:

- Transactions are processed efficiently, without appreciable performance degradation.
- Systems recover quickly when hardware or software failures occur, or when disaster strikes.
- Software that powers the enterprise databases is continuously running and available for transaction processing.

Archived

5.1.2 Architecture of HADR

Figure 5-1 illustrates the architecture of HADR.

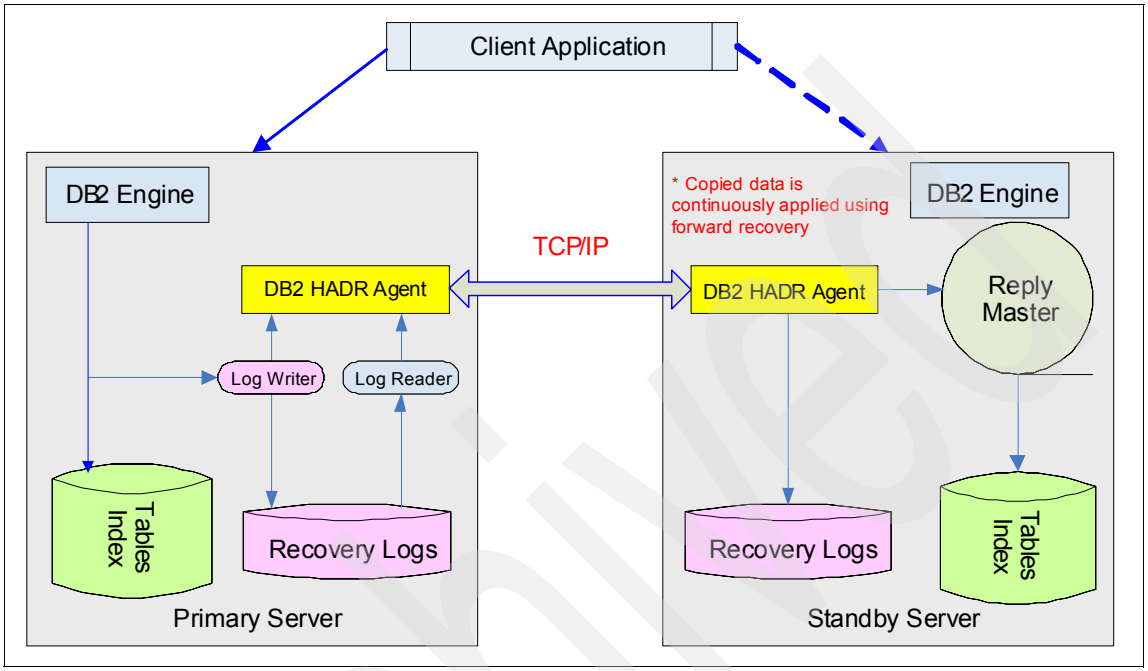


Figure 5-1 DB2 HADR architecture

Figure 5-1 illustrates the following key aspects of a HADR environment:

- Enabling HADR on a primary server starts up a process called db2hadrp that communicates with the standby server. At the same time, on the standby server, a process called db2hadrs is started, which receives log records from the primary server, writes them to the log file on the standby server, and applies those transactions to the data and index pages.
- When an application is running on the primary server, normal insert/update/delete activity results in log records being written to the log buffer. When the log buffer is full, or whenever a transaction commits, the log buffer is flushed to disk (to the log files) prior to the application receiving a successful return code to its commit request.
- When the primary database is in peer state, log pages are shipped to the standby database whenever the primary database flushes a log page to disk. The log pages are written to the local log files on the standby database to ensure that the primary and standby databases have identical log file

Archive

sequences. The log pages can then be replayed on the standby database to keep the standby database synchronized with the primary database.

Since HADR ensures that the primary and standby databases have identical log file sequences. If a disaster happens at the primary site, the standby database server can take over as the new primary database server to handle client application requests.

5.1.3 How HADR works

As described in the previous section, HADR uses the DB2 replication infrastructure to achieve data consistency and synchronization between the primary database and the standby database. Basically, we can separate the process into five steps. Figure 5-2 depicts these steps.

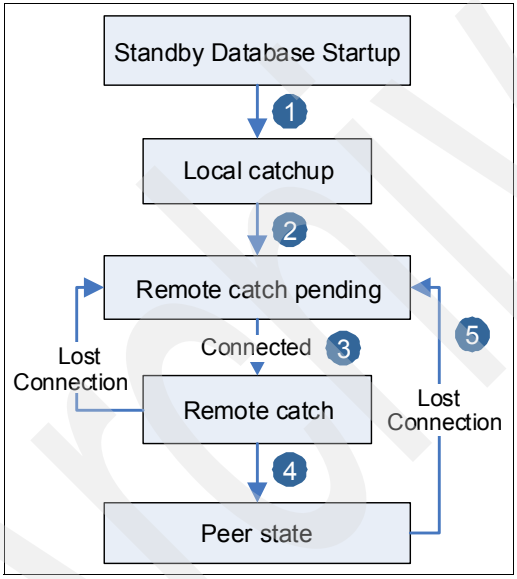


Figure 5-2 How HADR works

The steps are:

1. With the High Availability disaster recovery (HADR) feature, when the standby database is started, it enters local catchup state and attempts to read the log files in its local log path. If it does not find a log file in the local log path and a log archiving method has been specified, the log file is retrieved using the specified method. After the log files are read, they are replayed on the standby database.

Archived

2. When the end of local log files is reached, the standby database enters remote catchup pending state.
3. The standby database remains in remote catchup pending state until a connection to the primary database is established, at which time the standby database enters remote catchup state. During this time, the primary database reads log data from its log path or by way of a log archiving method and sends the log files to the standby database.
4. The standby database receives and replays the log data. When all of the log files on disk have been replayed by the standby database, the primary and standby systems enter peer state.
5. When in peer state, log pages are shipped to the standby database whenever the primary database flushes a log page to disk. The log pages are written to the local log files on the standby database to ensure that the primary and standby databases have identical log file sequences. If for any reason the standby cannot keep the data consistency, the standby database will change to remote catch pending status again.

5.1.4 Synchronization modes for HADR

With HADR, you can choose the level of protection that you want from a potential loss of data by specifying one of three synchronization modes: synchronous, near synchronous, or asynchronous. The synchronization mode indicates how log writing is managed between the primary and standby databases. These modes apply only when the primary and standby databases are in peer state.

- SYNC (synchronous)

  This mode provides the greatest protection against transaction loss, and using it results in the longest transaction response time amongst the three modes. In this mode, log writes are considered successful only when logs have been written to log files on the primary database and when the primary database has received acknowledgement from the standby database that the logs have also been written to log files on the standby database. The log data is guaranteed to be stored at both sites.

- NEARSYNC (near synchronous)

  While this mode has a shorter transaction response time than synchronous mode, it also provides slightly less protection against transaction loss. In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and when the primary database has received acknowledgement from the standby system that the logs have also been written to main memory on the standby system. Loss of data occurs only if both sites fail simultaneously and if the target site has not transferred to nonvolatile storage all of the log data that it has received.

Archive

- ASYNC (asynchronous)

  This mode has the highest chance of transaction loss if the primary system fails. It also has the shortest transaction response time among the three modes. In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and have been delivered to the TCP layer of the primary system's host machine. Because the primary system does not wait for acknowledgement from the standby system, transactions might be considered committed when they are still on their way to the standby.

Automatic Client Reroute (ACR) is another feature that was first introduced in DB2 UDB v8.2. If a database application loses communication with a DB2 database server, ACR reroutes that client application to an alternate database server so that the application can continue its work with minimal interruption. Rerouting is only possible when an alternate database location has been specified at the primary database server. ACR is only supported with the TCP/IP protocol.

5.1.5 Automatic Client Reroute

ACR is not tied to HADR. You can use it with HADR, clustering software, in a partitioned database environment, replication, and so on. ACR automatically and transparently reconnects DB2 database client applications to an alternate server without the application or user being exposed to a communications error. The alternate server information is stored on the primary database server, and loaded into the client's cache upon a successful connection to the primary database server. This means that for a client application to know the alternate database server, it must first successfully connect to the primary database server.

When ACR is configured, the built-in retry logic alternates between the original primary server and the alternate server for 10 minutes, or until a database connection is re-established to the primary database server.

Acquiescently, the retry logic built into ACR will:

- Try to re-establish a connection to the original primary server to ensure that there is no *accidental* failure.
- Alternate connection attempts between both the original primary database server and the alternate database server every 2 seconds for 30–60 seconds.
- Alternate connection attempts between both the original primary database server and the alternate database server every 5 seconds for 1–2 minutes.
- Alternate connection attempts between both the original primary database server and the alternate database server every 10 seconds for 2–5 minutes.

- Alternate connection attempts between both the original primary database server and the alternate database server every 30 seconds for 5–10 minutes.
- If no connection to the original primary database server is made after all of these attempts, the SQL30081N error code is returned to the client application.

At the same time, there are two DB2 database registry variables called DB2_MAX_CLIENT_CONNRETRIES and DB2_CONNRETRIES_INTERVAL that help to accurately configure the retry logic of ACR:

- DB2_MAX_CLIENT_CONNRETRIES defines the maximum number for a DB2 client retry to connect to the database server.
- DB2_CONNRETRIES_INTERVAL defines the interval between two different reconnection attempts.

Note: The tests in this book do not cover these two DB2 database registry variables.

For more information about ACR, see The IBM DB2 Version 8.2 Automatic Client Reroute Facility at:

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0512zikopoulos/#main

5.2 HACMP

A brief introduction to HACMP was presented in 4.3, “HACMP” on page 38. Also, configuration and implementation information about how to make IBM DB2 Universal Database highly available using HACMP is provided in the *IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES* guide, which is available at:

ftp://ftp.software.ibm.com/software/data/pubs/papers/db2ee-aixhacmp.pdf

Also refer to the DB2 Information Center at:

http://publib.boulder.ibm.com/infocenter/db2help/index.jsp

Archived

5.3 SQL replication

Replication is the copying of data from one place to another. From a business's point of view, replication plays a critical role in business transactions. The general considerations are:

- Distribution of data to other locations
- Consolidation of data from other locations
- Bidirectional exchange of data with other locations
- Some variation or combination of the above

Data can be extracted by specific programs, transported to some predefined location, and then loaded to the target location. A more efficient alternative is to extract only the changes since the last processing cycle and transport/apply those to the receiving location. From the other hand, data may be filtered and transformed during replication. At the same time, there may be other requirements for replication, such as time constraints. Replication processes need to be monitored, since replication must have minimal impact on production systems, which bring significant benefit to the software system.

5.3.1 Introduction

SQL replication allows you to replicate data from DB2 sources to targets by using two programs: *Capture and Apply.* The Capture program runs on the source system. The Capture program reads DB2 recovery logs for changed source data and saves the committed changed data to staging tables. The Apply program typically runs on the target system. The Apply program retrieves captured data from staging tables and delivers the data to targets. Both programs use a set of DB2 tables to track the information that they require to do their tasks and to store information that they generate themselves, such as information that you can use to see how well they are performing. You create these tables before you tell SQL replication what are your replication sources and targets. Figure 5-3 on page 47 shows the infrastructure for a simple configuration of DB2 SQL replication.

Redbooks

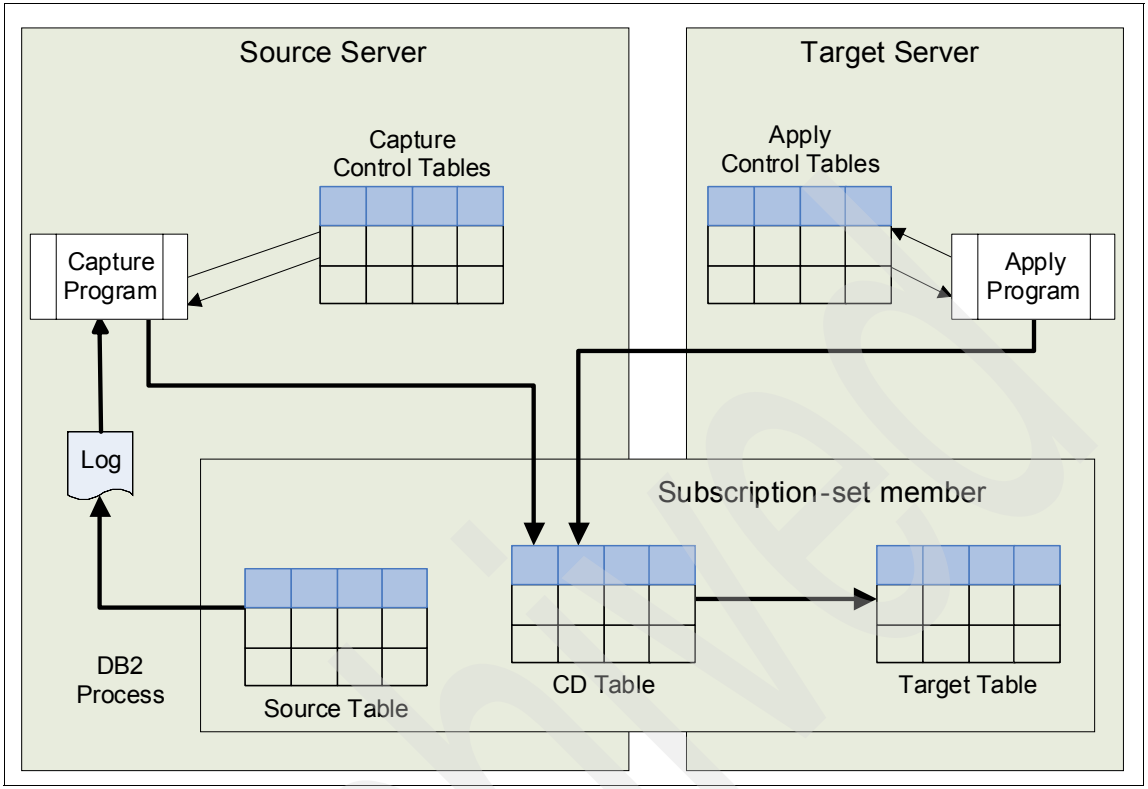


Figure 5-3 Infrastructure for a simple configuration of DB2 SQL replication

The infrastructure is:

- The Capture program uses a set of DB2 tables called *capture control tables*. These tables contain information about replication sources and the current position of the Capture program in the DB2 recovery log. In most cases, the control tables for a Capture program need to be on the same DB2 server as the sources associated with the program.
- The Apply program uses a set of DB2 tables called *apply control tables*. These tables contain information about your targets and where their corresponding sources are located. The control tables for the Apply program usually reside on the system where the Apply program runs. Unlike with the Capture program, you can create multiple Apply programs that use the same set of control tables. Each Apply program is identified in these control tables by a name called an *apply qualifier*.

5.3.2 How SQL replication works

The general approach of DB2 SQL replicati.on is:

1. Register the source in SQL replication.
2. Subscribe sets in SQL replication.
3. Capture data from DB2 sources.
4. Apply data in DB2 targets.

Register source in SQL replication

First of all, you should know what you want to replicate to the target from the source database. So you should tell SQL replication about them by registering them. You can register sources that are DB2 tables and views, or tables on non-DB2 relational databases.

Note: When you register source tables on non-DB2 relational databases, you use SQL replication together with DB2 Relational Connect. You map your source database to a federated database and you create nicknames for each source table.

The replication control tables are playing as the core of replication, where the components of replication communicate by creating, reading, and updating the control tables' data. The control tables are viewable and manually able to be updated. The design divides the control tables into three sets based on the functionality they are related to: capture, apply, and monitoring. Each set of control tables is independent. The sets can be stored in separate databases, in separate instances, and on separate servers.

Replication sources are selected from the objects of the capture control server and registered to the capture control server, which implies that your capture control server must be the one where your source data for replication resides. Remote journaled tables on DB2 UDB for iSeries is the only exception to this rule. The capture control tables should already be created for that server, and the capture control server should already be added to the replication center.

Subscribe sets in SQL replication

After you register your sources, you create subscription sets, in which you pair your sources with targets. Each source-target pair is referred to as a member of the subscription set in which it is created. You can use subscription sets to schedule the replication of data in one or more source-target pairs from one source server to one target server.

The subscribe sets define the rules about how to keep the consistency between source and target objects, so that the SQL replication engine can follow these

rules to do SQL replication. Generally, we need to give SQL replication the following details:

- Which Apply program to sue for processing the subscription set
- Where the source tables or views are stored
- Where is the location of target tables
- The frequency to replicate data from sources to targets
  - Interval timing
  - Continuous replication
  - Event timing
- Whether to use data blocking
- Whether to issue one COMMIT for all applied data or to issue interim commits
- Whether to transform data in the subscription set with SQL scripts or stored procedures

So, when you create a subscription set, the following are some of the major attributes defined:

- A name for the subscription set
- The source and target server name
- The apply qualifier
- When to start replication, how often to replicate, and whether to use interval timing, event timing, or both
- Data blocking, if you expect large volumes of changes

Note: If the subscription set is for replication from or to a non-DB2 server, such as Informix®, the source or target server name will be the name of the DB2 ESE or DB2 Connect™ EE database containing the server definition for the non-DB2 source/target server.

Archived

When you create a subscription set, you map sources to targets as part of creating subscription sets. A subscription set groups together one or more source-target pairs, also called *subscription-set members*. Figure 5-4 depicts a simple subscription-set member, where a source table named *catalog* is mapped to a target table named catalog. The data replicated to the target is first staged in the CD table for the source.

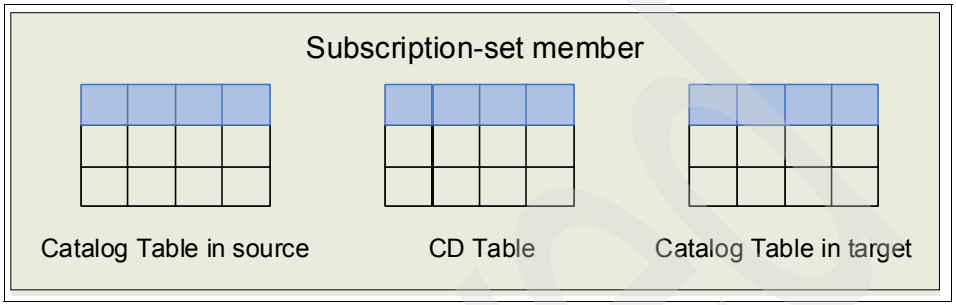


Figure 5-4 Subscription-set sample in SQL replication

Capture data from DB2 sources

After you register your DB2 replication sources and create subscription sets for the source and target, you can start capturing the changes made to your sources. Here you can use a program called the Capture program to do this. The basic operations on Capture are starting, stopping, and querying the Capture programs.

After you start the Capture program and that program receives signals from the Apply program to indicate that the sources and targets are synchronized, the Capture program reads the DB2 log sequentially for changes to the source tables in which you are interested. If it reads a change to one of your source tables, the Capture program adds the change to the corresponding database transaction that it is retaining in memory. Transactions in memory are potentially subsets of the corresponding transactions in the log. They contain only changes to your source tables. The Capture program collects changes in memory until it reads either a ROLLBACK or a COMMIT statement for the transaction in which those changes are made, and it only stores the changes associated with the committed transaction.

For example, if you registered the table catalog as replication sources, SQL replication created a CD table as part of the registration process. After you start the Capture program, you can go through the steps (Figure 5-5) to complete capture progress:

1. Capture program receives signals from the Apply program to indicate that the target tables are synchronized with the source tables.
2. Application A makes a series of changes to the table catalog.
3. The Capture program reads the DB2 logs for changes to those source tables.
4. The Capture program collects these changes in memory.
5. Application A issues a COMMIT statement.
6. When the Capture program reads this statement, it appends a copy of each change to the CD table for the table catalog.

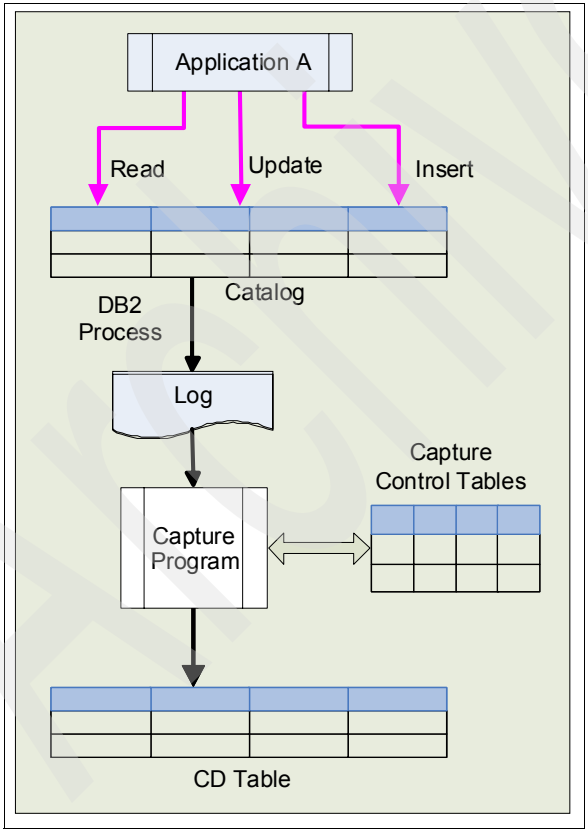


Figure 5-5 Capture data from source table

Apply data in DB2 targets

Apply program in SQL replication can help you to start replicating data from sources to targets. To start replicating data from sources to targets, start the Apply program. After you do so, the Apply program begins processing the subscription sets that you assigned to it. The Apply program processes any active subscription sets one at a time, according to the scheduling or event criteria that you specified when you created the subscription sets.

There are two approaches by which the Apply program can process each member of a subscription set:

- For sources that you registered for change-capture replication
  - The first time that the Apply program processes the corresponding subscription set, the Apply program can populate the targets with the content of the sources. You can tell the Apply program to call one or more utilities, such as the EXPORT and LOAD utilities, to populate the targets. The utilities that the Apply program chooses depends on the platform on which the Apply program is running.
  - Then, at the time intervals that you specified when you created your subscription sets or whenever an event occurs, the Apply program reads the CD tables of your sources for rows that were inserted into the CD table since the Apply program last looked. The rows in the CD tables indicate whether they are records of deletes, updates, or inserts to the corresponding sources.
  - The Apply program uses the data from the CD table to insert, update, and delete rows in the target. Predicates are used to identify the rows to be updated or deleted.
- For sources that you registered for full-refresh replication

  At intervals that you specify, the Apply program populates the targets with the content of the sources. You can tell the Apply program to call one or more utilities, such as the EXPORT and LOAD utilities, to refresh your target tables.

Archive

Figure 5-6 shows how the Apply program applies staging data to the target objects.

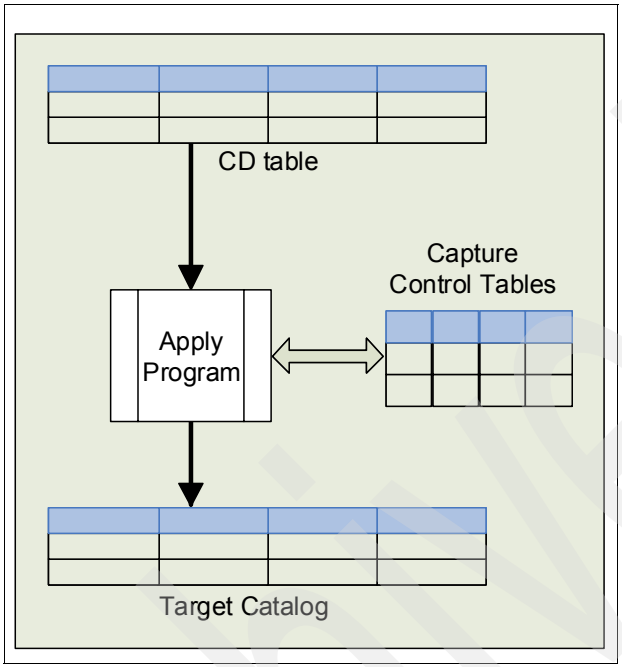


Figure 5-6 Apply data to target table

For more details about SQL replication, refer to the DB2 Information Center at:

http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod_overview/iiyrcintrsbdd.html

Archived

Archived

6

WebSphere Application Server High Availability

WebSphere Commerce utilizes the underlying WebSphere Application Server failover and recovery features to achieve High Availability at the application server level.

This is done through the workload management (WLM) mechanism that is provided by IBM WebSphere Application Server Network Deployment. If you do not use WLM with your application servers (and you do not use any other clustering software), your system cannot provide failover support. In this case, your Web or Java clients will fail if your application server fails.

This chapter gives an introduction to the WLM and failover capabilities of IBM WebSphere Application Server Network Deployment. For more details on this topic, refer to *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392.

Topics discussed in this chapter include:

- Introduction to availability
- Session management using WebSphere Commerce and WebSphere Application Server
- WebSphere workload management defined

© Copyright IBM Corp. 2008. All rights reserved.

- WebSphere Application Server clustering
- WebSphere Commerce clustering

Archived

6.1 Introduction to availability

Also known as *resiliency*, availability is the description of the system's ability to respond to requests no matter the circumstances. Availability requires that the topology provide some degree of process redundancy in order to eliminate single points of failure. Whereas vertical scalability (multiple application servers on one system) can provide this by creating multiple processes, the physical machine then becomes a single point of failure. For this reason, a High Availability topology typically involves horizontal scaling across multiple machines or LPARs.

For more information see IBM Redbook, *IBM WebSphere Application Server Network DeploymentWebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688, for running DMgr and node agents as OS services (Windows®, UNIX).

6.1.1 Hardware-based High Availability

Using multiple machines for WebSphere Application Server eliminates a given application server process as a single point of failure.

Typically, the only single point of failure in a WebSphere cell is the Deployment Manager, where all central administration is performed. However, a failure of the Deployment Manager only impacts the ability to change the cell configuration to run the Tivoli Performance Viewer (which is included in the administrative console in WebSphere V6).

A number of alternatives exist to provide High Availability for the Deployment Manager, including the possible use of an external High Availability solution. However, the minimal impact of a Deployment Manager outage typically does not require the use of such a solution. Some customers even choose to run their production environment without an active Deployment Manager. For this reason, this book does not discuss any detailed setup to configure Network Deployment High Availability using external clustering software. If you are interested in learning how this can be done, refer to IBM Redbook, *WebSphere Application ServerWebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

6.1.2 Workload management

IBM WebSphere Application Server Network Deployment workload management optimizes the distribution of incoming requests between application servers that are able to handle a client request. WebSphere workload management is based on application server clusters containing multiple application servers, so-called

Archived

cluster members. An application deployed to a cluster runs on all cluster members concurrently. The workload is distributed based on weights that are assigned to each cluster member. Thus, more powerful machines can be configured to receive more requests than smaller systems.

Should an application server fail in the cluster, workload management (WLM) can also manage to failover existing client requests to another available cluster member.

In addition, WLM enables servers to be transparently maintained and upgraded while applications remain available for users. You can add additional cluster members to a cluster at any point, providing scalability and performance if an existing environment is not able to handle the workload any more. For more details, see 6.2, “WebSphere workload management defined” on page 64.

6.1.3 Failover

The proposition to have multiple servers (potentially on multiple independent machines) naturally leads to the potential for the system to provide failover. That is, if any one machine or server in the system were to fail for any reason, the system should continue to operate with the remaining servers. The load balancing property should ensure that the client load gets redistributed to the remaining servers, each of which will take on a proportionally higher percentage of the total load. Of course, such an arrangement assumes that the system is designed with some degree of overcapacity, so that the remaining servers are indeed sufficient to process the total expected client load.

Ideally, the failover aspect should be totally transparent to clients of the system. When a server fails, any client that is currently interacting with that server should be automatically redirected to one of the remaining servers, without any interruption of service and without requiring any special action on the part of that client. In practice, however, most failover solutions might not be completely transparent. For example, a client that is currently in the middle of an operation when a server fails might receive an error from that operation, and might be required a retry (at which point the client would be connected to another, still available server). Or the client might observe a pause or delay in processing, before the processing of its requests resumes automatically with a different server. The important point in failover is that each client, and the set of clients as a whole, is able to eventually continue to take advantage of the system and receive service, even if some of the servers fail and become unavailable. Conversely, when a previously failed server becomes available again, the system might transparently start using that server again to process a portion of the total client load.

The failover aspect is also sometimes called fault tolerance, in that it allows the system to survive a variety of failures or faults. It should be noted, however, that failover is only one technique in the much broader field of fault tolerance, and that no such technique can make a system 100% safe against every possible failure. The goal is to greatly minimize the *probability* of system failure, but keep in mind that the *possibility* of system failure cannot be completely eliminated.

Note that in the context of discussions on failover, the term *server* often refers to a physical machine. However, WebSphere vertical scaling also allows for one server process on a given machine to fail independently, while other processes on that same machine continue to operate normally.

6.1.4 HAManager

WebSphere Application Server V6 introduces a new concept for advanced failover and thus higher availability, called the High Availability Manager (HAManager). The HAManager enhances the availability of WebSphere singleton services such as transaction services or message services. It runs as a service within each application server process that monitors the health of WebSphere clusters. In the event of a server failure, the HAManager will failover the singleton service and recover any in-flight transactions. See *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392.

6.1.5 Session management

Web browsers and e-commerce sites use HTTP to communicate. In the case of an HTTP client interacting with a servlet, the state information associated with a series of client requests is represented as an HTTP session, and identified by a session ID. The Servlet 2.3 specification defines that, after a session has been created, all following requests need to go to the same application server that created the session.

However, in a clustered environment, there is more than one application server that can serve the client request. Therefore, the Web server plug-in needs to read a request and be able to identify which cluster member should handle it. *Session identifiers* are used to do this. They allow the plug-in to pick the correct cluster member and Web container to retrieve the current session object.

The session manager module that is part of each Web container is responsible for managing HTTP sessions, providing storage for session data, allocating session IDs, and tracking the session ID associated with each client request.

WebSphere Commerce supports two types of session management: cookie-based and URL rewriting.

Cookie-based session management

When cookie-based session management is used, a message (cookie) containing a user's information is sent to the browser by the Web server. This cookie is sent back to the server when the user tries to access certain pages. By sending back the cookie, the server is able to identify the user and retrieves the user's session from the session database, thus maintaining the user's session. A cookie-based session ends when the user logs off or closes the browser. Cookie-based session management is secure and has performance benefits. Cookie-based session management is secure because it uses an identification tag that only flows over SSL. Cookie-based session management offers significant performance benefits because the WebSphere Commerce caching mechanism only supports cookie-based sessions, and not URL rewriting. We recommend cookie-based session management for customer sessions.

If you are not using URL rewriting and you want to ensure that users have cookies enabled on their browsers, check **Cookie acceptance test** on the Session Management page of Configuration Manager. This informs the customer that if their browser does not support cookies, or if they have turned off cookies, they need a browser that supports cookies to browse the WebSphere Commerce site.

For security reasons, cookie-based session management uses two types of cookies, as discussed in the following sections.

Non-secure session cookie

This is used to manage session data. This contains an activity identifier that points to attributes such as the negotiated language, current store, and customer's preferred currency at the time that the cookie is constructed. This cookie can flow between the browser and server under either a SSL or a non-SSL connection. There are two types of non-secure session cookies:

- A WebSphere Application Server session cookie is based on the servlet HTTP session standard. WebSphere Application Server cookies persist to memory or to the database in a multinode deployment.
- A WebSphere Commerce session cookie is internal to WebSphere Commerce and does not persist to the database.

To select which type of cookie to use, select **WebSphere Commerce** or **WebSphere Application Server** for the Cookie session manager parameter on the Session Management page of Configuration Manager.

By default, a WebSphere Commerce instance is configured to use WebSphere Commerce session management. However, some custom applications may need to store more session data than the Internet Explorer®'s browser limit. In

this case, you may wish to switch to WebSphere Application Server's cookie management mechanism.

WebSphere Application Server session manager provides for the storage of session-related information either in-memory within the application server, in which case it cannot be shared with other application servers; in a back-end database, shared by all application servers; or by using memory-to-memory replication.

Database persistence using WebSphere Application Server session management

Storing session information in a database, sometimes referred to as *persistent sessions* or *session clustering*, is one method to share distributed session information among cluster members. With this option, whenever an application server receives a request associated with a session ID, which is not in memory, it can obtain it by accessing the back-end database, and can then serve the request. When this option is not enabled, and another clustering mechanism is not used, if any load distribution mechanism happens to route an HTTP request to an application server other than the one where the session was originally created, that server would be unable to access the session, and would thus not produce correct results in response to that request. One drawback to the database solution, just as with application data, is that it provides a single point of failure, so it should be implemented in conjunction with hardware clustering products such as IBM HACMP, TSA, or solutions such as database replication. Another drawback is the performance hit, caused by database disk I/O operations and network communications.

Memory-to-memory session replication using WebSphere Application Server session management

Memory-to-memory replication enables the sharing of sessions between application servers without using a database. It uses the built-in Data Replication Service (DRS) of WebSphere to replicate session information stored in memory to other members of the cluster.

Using this functionality removes the single point of failure that is present in persistent sessions through a database solution that has not been made highly available using clustering software. The sharing of session state is handled by creating a replication domain and then configuring the Web container to use that replication domain to replicate session state information to the specified number of application servers. The administrator can define how many replicas should exist in the domain (either a single replica, a defined number, or the entire domain).

Memory-to-memory replication also incurs a performance hit, primarily because of the overhead of network communications. Additionally, because copies of the session object reside in application server memory, this reduces the available heap for application requests and usually results in more frequent garbage collection cycles by the application server JVM.

Note: Depending on your application (for example, the session size) and on your hardware resources, memory-to-memory replication or database persistence might be the better solution for your environment. Refer to *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, for additional information about resource requirements and performance for either option.

Conclusion

Storing session state in a persistent database or using memory-to-memory replication provides a degree of fault tolerance to the system. If an application server crashes or stops, any session state that it might have been working on would normally still be available either in the back-end database or in another still-running application server's memory, so that other application servers can take over and continue processing subsequent client requests associated with that session.

You can find more information about this topic, including traces and logs, in 6.3, "Web container clustering and failover (Web server plugin)" on page 66, and Chapter 6 of *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392

Secure authentication cookie

This is used to manage authentication data. An authentication cookie flows over SSL and is timestamped for maximum security. This is the cookie used to authenticate the user whenever a sensitive command is executed, for example, the DoPaymentCmd, which asks for a user's credit card number. There is minimal risk that this cookie could be stolen and used by an unauthorized user. Authentication code cookies are always generated by WebSphere Commerce whenever cookie-based session management is in use.

Both the session and authentication code cookies are required to view secure pages.

For cookie errors, the CookieErrorView is called under the following circumstances:

- The user has logged in from another location with the same logon ID.
- The cookie became corrupted or was tampered with, or both.
- If cookie acceptance is set to true and the user's browser does not support cookies.

URL rewriting

With URL rewriting, all links that are returned to the browser or that get redirected have the session ID appended to them. When the user clicks these links, the rewritten form of the URL is sent to the server as part of the client's request. The servlet engine recognizes the session ID in the URL and saves it for obtaining the proper object for this user. To use URL rewriting, HTML files (files with .html or .htm extensions) cannot be used for links. To use URL rewriting, JSP pages must be used for display purposes. A session with URL rewriting expires when the customer logs off.

Because URLs returned to the browser contain session IDs, another user with access to the browser history (for example, on a shared computer) could potentially gain access to sensitive information exchanged during a session—if the session has been left active. To prevent such unauthorized access, site developers should consider adding a notice to their site urging customers to always log off at the end of their visit, thereby ending (expiring) their session, particularly on a shared computer.

WebSphere Commerce dynamic caching and URL rewriting cannot interoperate. With URL rewriting turned on, you need to disable WebSphere Commerce dynamic caching.

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

For more information about WebSphere Commerce session management see:

http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/concepts/csesmsession_mgmt.htm

6.2 WebSphere workload management defined

Workload management is implemented in IBM WebSphere Application Server Network Deployment V6 by using application server clusters and cluster members. These cluster members can all reside on a single node (system) or can be distributed across multiple nodes (or LPARs).

You might have Web clients or thick Java/C++ clients. When using clustered WebSphere Application Servers, your clients can be redirected either automatically or manually (depending on the nature of the failure) to another healthy server in the case of a failure of a clustered application server.

Workload management (WLM) is the WebSphere facility to provide load balancing and affinity between application servers in a WebSphere clustered environment. It optimizes the distribution of processing tasks in the WebSphere Application Server environment. Incoming work requests are distributed to the application servers that can most effectively process the requests.

Workload management is also a procedure for improving performance, scalability, and reliability of an application. It provides failover when servers are not available. WebSphere uses workload management to send requests to alternate members of the cluster. WebSphere also routes concurrent requests from a user to the application server that serviced the first request, as EJB™ calls, and session state will be in memory of this application server.

WLM is most effective when the deployment topology comprises application servers on multiple machines, because such a topology provides both failover and improved scalability. It can also be used to improve scalability in topologies where a system comprises multiple servers on a single, high-capacity machine. In either case, it enables the system to make the most effective use of the available computing resources.

Two types of requests can be workload managed in IBM WebSphere Application Server Network Deployment V6:

- *HTTP requests* can be distributed across multiple Web containers. When an HTTP request reaches the HTTP server, a decision must be made. Some requests for static content might be handled by the HTTP server. Requests for dynamic content or some static content will be passed to a Web container running in an application server. Whether the request should be handled or passed to WebSphere is decided by the IBM HTTP Server Plug-in, which runs in-process with the HTTP server. We refer to this as *Plug-in WLM*. For these WebSphere requests, High Availability for the Web container becomes an important piece of the failover solution. See 6.3, “Web container clustering and failover (Web server plugin)” on page 66, for more information.

- *EJB requests* can be distributed across multiple EJB containers. When an EJB client makes calls from the Web container or client container or from outside, the request is handled by the EJB container in one of the clustered application servers. If that server fails, the client request is redirected to another available server. We refer to this as *EJS WLM*.

6.2.1 Distributing workloads

The ability to route a request to any server in a group of clustered application servers allows the servers to share work and improve throughput of client requests. Requests can be evenly distributed to servers to prevent workload imbalances in which one or more servers has idle or low activity while others are overburdened. This load balancing activity is a benefit of workload management.

Thus, the proposed configuration should ensure that each machine or server in the configuration processes a fair share of the overall client load that is being processed by the system as a whole. In other words, it is not efficient to have one machine overloaded while another machine is mostly idle. If all machines have roughly the same capacity (for example, CPU power), each should process a roughly equal share of the load. Otherwise, there likely needs to be a provision for workload to be distributed in proportion to the processing power available on each machine.

Using weighted definitions of cluster members allows nodes to have different hardware resources and still participate in a cluster. The weight specifies that the application server with a higher weight will be more likely to serve the request faster, and workload management will consequently send more requests to that node.

With several cluster members available to handle requests, it is more likely that failures will not negatively affect throughput and reliability. With cluster members distributed to various nodes, an entire machine can fail without any application downtime. Requests can be routed to other nodes if one node fails. Clustering also allows for maintenance of nodes without stopping application functionality.

This section only gives you an introduction into WebSphere WLM. The available WLM policies and how requests are distributed among available servers are described in great detail in Chapter 6 and Chapter 7 of *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392.

Archive

6.2.2 Benefits

Workload management provides the following benefits to WebSphere applications:

- It balances client processing requests, allowing incoming work requests to be distributed according to a configured WLM selection policy.
- It provides failover capability by redirecting client requests to a running server when one or more servers are unavailable. This improves the availability of applications and administrative services.
- It enables systems to be scaled up to serve a higher client load than provided by the basic configuration. With clusters and cluster members, additional instances of servers can easily be added to the configuration. See 6.4, “WebSphere Application Server clustering” on page 72, for details.
- It enables servers to be transparently maintained and upgraded while applications remain available for users.
- It centralizes administration of application servers and other objects.

6.3 Web container clustering and failover (Web server plugin)

Each HTTP server is configured to run the IBM HTTP Server Plug-in. The cluster members can all reside on a single node or can be distributed across multiple nodes in the WebSphere cell (vertical or horizontal scaling).

Archived

Each request coming into the Web server is passed through the plug-in, which uses its configuration information to determine whether the request should be routed to WebSphere, and if so, to which application server (that is, to which Web container) the request should be routed to (Figure 6-1). The communication between the plug-in and the application servers can be either HTTP or HTTPS. The Web server plug-in distributes requests around cluster members that are not available.

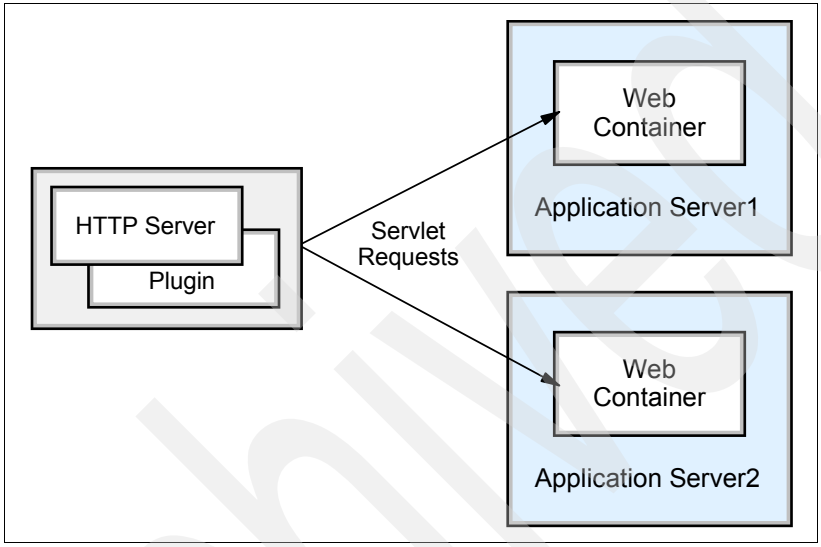


Figure 6-1 Plug-in (Web container) workload management

The plug-in is a module, which is loaded by the Web server when it initializes, and that it bypasses Web containers that are not available. It uses the following mechanisms for WLM and failover:

- Application server clustering, which creates server process redundancy for failover support. All application servers in a cluster host the same applications.
- The workload management routing technique built into the IBM HTTP Server Plug-in. It controls the routing of client requests among redundant server processes. This routing is based purely on the weights associated with the cluster members. If all cluster members have identical weights, the plug-in sends an equal number of requests to all members of the cluster, when assuming no session affinity. If the weights are different, the plug-in routes requests to those cluster members with the higher weight value more often.
- Session management and failover mechanism, which provides HTTP session data for redundant server processes.

Archived

Thus, satisfactory failover support for Web clients can only be achieved by the use of all three mechanisms.

6.3.1 Session management and failover inside the plug-in

As you know, the plug-in always attempts to route a request that contains session information to the application server that processed the previous requests. However, if the server that contains the session is not available to the plug-in when it forwards the request, then the plug-in can route the request to an alternate server. The alternate server can then retrieve the distributed session information according to the chosen distribution method (database or memory-to-memory replication).

There are three methods of identifying a user's session to the application server: Cookies, URL rewriting, and SSL ID. Example 6-1 shows a JSESSIONID cookie, which consists of four parts:

- Cache ID (0000)
- Session ID (A2MB4lJozU_VM8lffsMNfdR)
- Separator (:)
- Clone ID (v544d0o0 = application server ID)

Example 6-1 Example of a session identifier - JSESSIONID cookie

```
JSESSIONID=0000A2MB4IJozU_VM8IffsMNfdR:v544d0o0
```

In case of a failover, the clone ID of the failover server is appended at the end, also separated by a colon. When the original server becomes available again, the request falls back and is handled by the original server.

Archived

Figure 6-2 and the subsequent step-by-step explanation explain how the plug-in performs the failover.

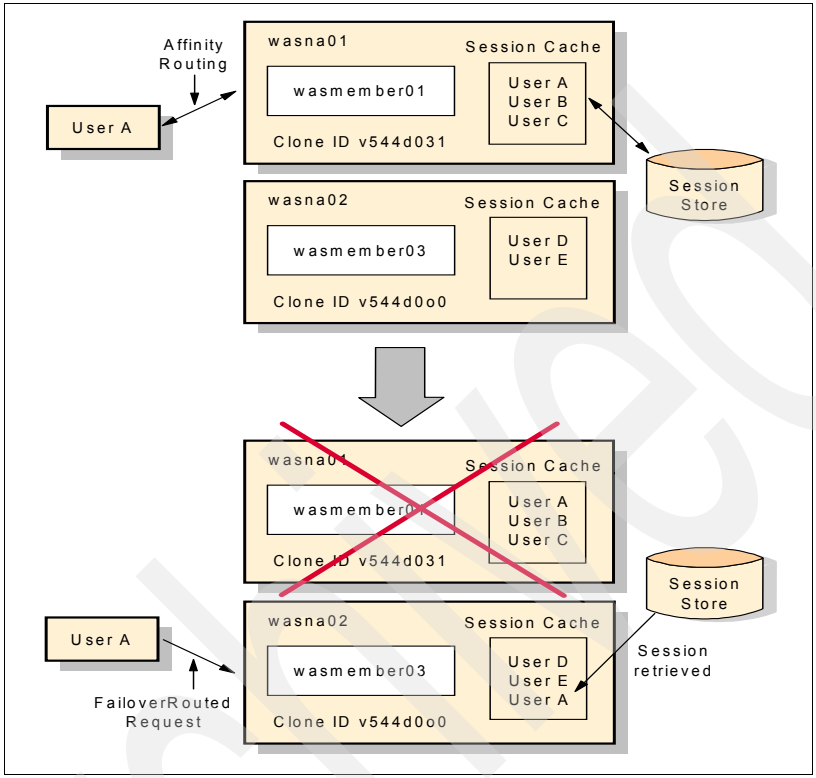


Figure 6-2 Session management example

Using Figure 6-2, the steps involved to find a failover application server are:

1. The plug-in processes a request from user A to http://http1/snoop. The request also contains a JSESSION cookie with a session ID and clone ID of v544d031.
2. The plug-in matches the virtual host and URI to the cluster wascluster01 (composed by servers wasmember01 and wasmember03, each one located in a different machine).
3. The plug-in checks for session affinity and finds the clone ID of v544d031 in the request's JSESSIONID cookie.
4. The plug-in searches for the clone ID of v544d031 in the plug-cfg.xml's list of primary servers and matches the clone ID to the wasmember01 application server.

5. The plug-in checks to see whether wasmember01 has been marked down. In our case, it has not been marked down yet.
6. The plug-in attempts to get a stream to wasmember01. Finding the server is not responding, Web1 is marked as down and the retry timer is started.
7. The plug-in checks the session identifier again.
8. The plug-in checks the servers. When it reaches wasmember01, it finds that it is marked down and the retry timer is not 0, so it skips wasmember01 and checks the next cluster member in the primary server list.
9. The plug-in selects wasmember03 (clone ID v544d0o0) and attempts to get a stream to it. The plug-in either opens a stream or gets an existing one from the queue.
10. The request is sent and received successfully to wasmember03 (which retrieves the session information from the persistent session database or has it in-memory because of a previous replication) and sent back to user A.

For additional information about plug-in failover behavior, read WebSphere plug-in behavior in Chapter 6 of *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392. This section discusses many failure situations in detail and includes information about logs and traces.

6.3.2 Web container failures

In a clustered environment with several cluster members, an unavailable application server does not mean an interruption of the service. When the plug-in has selected a cluster member to handle a request it will attempt to communicate with the cluster member. There are, however, a number of situations in which the plug-in might not be able to complete a request to a specific application server. If this communication is unsuccessful or breaks, then the plug-in marks the cluster member as down and attempts to find another cluster member to handle the request. Web container failures are detected based on TCP response values or lack of response to a plug-in request.

The marking of the cluster member as down means that, should that cluster member be chosen as part of a workload management policy or in session affinity, the plug-in will not try to connect to it. The plug-in knows that it is marked as down and ignores it.

Some example scenarios when the plug-in cannot connect to a cluster member are:

- Expected application server failures (The cluster member has been brought down intentionally for maintenance, for example.)
- Unexpected server process failures (The application server JVM has crashed, for example.)
- Server network problems between the plug-in and the cluster member (A router is broken, for example.)
- System problems (whether expected), such as system shutdown or power failures
- The cluster member is overloaded and cannot process the request (For example, because the system is too small to handle a large number of clients, or because the server weight is inappropriate.)

In the first two failure cases described, the physical machine where the Web container is supposed to be running is still available, although the WebContainer Inbound Chain is not available. When the plug-in attempts to connect to the WebContainer Inbound Chain to process a request for a Web resource, the machine will refuse the connection, causing the plug-in to mark the application server as down.

In the third and fourth events, however, the physical machine is no longer available to provide any kind of response. In these events, if non-blocking connection is not enabled, the plug-in waits for the local operating system to time out the request before marking the application server unavailable. While the plug-in is waiting for this connection to time out, requests routed to the failed application server appear to hang. The default value for the TCP timeout varies based on the operating system. While these values can be modified at the operating system level, adjustments should be made with great care. Modifications might result in unintended consequences in both WebSphere and other network dependent applications running on the machine. This problem can be eliminated by enabling non-blocking connection. Refer to “Connection timeout” on page 408 for more information.

In the fifth case, overloading can make a healthy server unavailable. To avoid overloading of servers, you can define the maximum number of connections that are allowed from HTTP servers to the application server. This is explained in “Maximum number of connections” on page 412.

6.3.3 Web server plug-in failover tuning

IBM HTTP Server Plug-in uses an XML configuration file called plugin-cfg.xml to determine information about the WebSphere cell it is serving. There are some

settings in the plug-in file that directly affect how the plug-in works in a workload management environment. In WebSphere V6, all of these settings can be modified using the administrative console. Plug-in file tags related to workload management and failover are:

- You can change the workload distribution policy in the configuration file.
- You can change the retry interval for connecting to a cluster member marked as down.
- You can change connection timeout settings to bypass the operating system timeout.
- You can divide the servers into a primary server list and a backup server list. This is a feature available since WebSphere Application Server V5, also called two-level failover support.
- You can change the maximum number of connections that will be allowed to a server from a given plug-in. If this attribute is set to either zero or -1, there is no limit to the number of pending connections to the application servers. The default value is -1.
- You can change the refresh interval for the reloading of the plug-in configuration file.

For details on these performance tuning options, see 18.1.4, “IBM HTTP Server Plug-in” on page 395.

6.4 WebSphere Application Server clustering

A *cluster* is a set of application servers that are managed together and participate in workload management. Application servers participating in a cluster can be on the same node or on different nodes. A network deployment cell can contain no clusters, or have many clusters depending on the need of the administration of the cell. The cluster is a logical representation of the application servers. It is not necessarily associated with any node, and does not correspond to any real server process running on any node. A cluster contains only application servers, and the weighted workload capacity associated with those servers.

The cluster member template is created when the first member is added to the cluster. Subsequent configuration changes to the first member will not affect the template, which will be used when creating more members. When creating a cluster, it is possible to select an existing application server as the template for the cluster without adding that application server into the new cluster (the chosen application server is used only as a template, and is not affected in any way by the cluster creation).

Cluster members can be added to a cluster in various ways, during cluster creation and afterwards. During cluster creation, one existing application server can be added to the cluster or one or more new application servers can be created and added to the cluster. There is also the possibility of adding additional members to an existing cluster later on. Depending on the capacity of your systems, you can define different weights for the various cluster members.

Cluster members are required to have identical application components, but they can be sized differently in terms of weight, heap size, and other environmental factors. You must be careful though not to change anything that might result in different application behavior on each cluster member. This concept allows large enterprise machines to belong to a cluster that also contains smaller machines, albeit of different weights.

Starting or stopping the cluster starts or stops all cluster members automatically and changes to the application are propagated to all application servers in the cluster.

Figure 6-3 shows an example of a possible configuration that includes server clusters. Server cluster 1 has two cluster members on node B only. Server cluster 2, which is completely independent of server cluster 1, has two cluster members on node A and three cluster members on node B. Finally, node A also contains a free-standing application server that is not a member of any cluster.

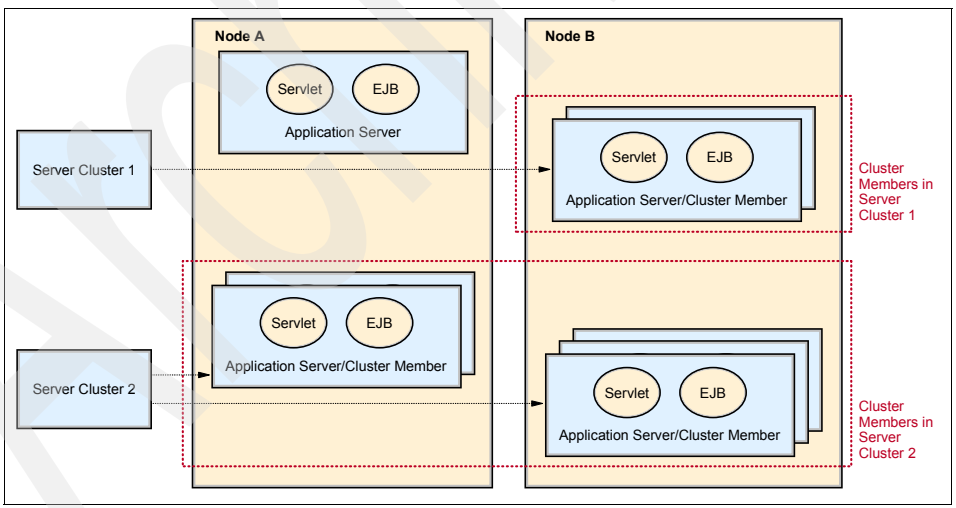


Figure 6-3 Server clusters and cluster members

Clustering for scalability and failover

Clustering is an effective way to perform vertical and horizontal scaling of application servers.

Vertical scaling

In *vertical scaling*, shown in Figure 6-4, multiple cluster members for an application server are defined on the same physical machine, or node, which might allow the machine's processing power to be more efficiently allocated.

Even if a single JVM can fully utilize the processing power of the machine, you might still want to have more than one cluster member on the machine for other reasons, such as using vertical clustering for process availability. If a JVM reaches a table/memory limit (or if there is some similar problem), then the presence of another process provides for failover.

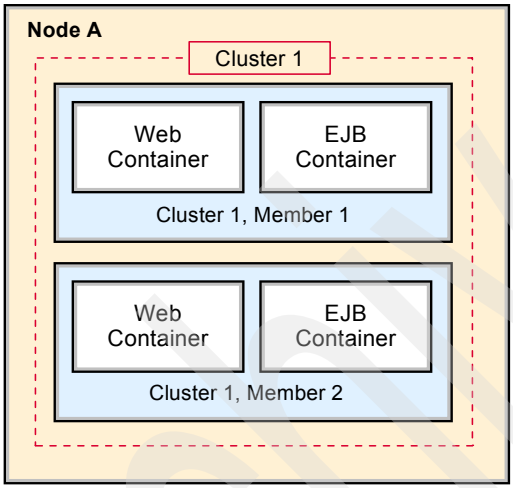


Figure 6-4 Vertical scaling

We recommend that you avoid using *rules of thumb* when determining the number of cluster members for a given machine. The only way to determine what is correct for your environment and application is to tune a single instance of an application server for throughput and performance, then add it to a cluster, and incrementally add additional cluster members. Test performance and throughput as each member is added to the cluster. Always monitor memory usage when you are configuring a vertical scaling topology and do not exceed the available physical memory on a machine.

In general, 85% (or more) utilization of the CPU on a large server shows that there is little, if any, performance benefit to be realized from adding additional cluster members. In conjunction with 6.2.1, "Distributing workloads" on page 65, you should ensure that the required business capacity is maintained even in failure scenarios. For example, in the basic three node cluster (the so-called *rule of three*, which maintains system redundancy even while a single node is offline)

this would mean that no node exceeds 66% utilization at peak load, so that with the loss of one node, the surviving two nodes will not be overwhelmed.

Horizontal scaling

In *horizontal scaling*, shown in Figure 6-5, cluster members are created on multiple physical machines (or LPARs). This allows a single WebSphere application to run on several machines while still presenting a single system image, making the most effective use of the resources of a distributed computing environment. Horizontal scaling is especially effective in environments that contain many smaller, less powerful machines. Client requests that overwhelm a single machine can be distributed over several machines in the system.

Failover is another important benefit of horizontal scaling. If a machine becomes unavailable, its workload can be routed to other machines containing cluster members.

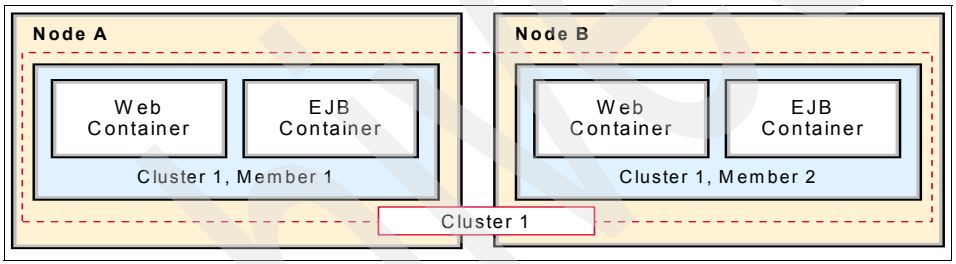


Figure 6-5 Horizontal scaling

Horizontal scaling can handle application server process failures and hardware failures (or maintenance) without significant interruption to client service.

Archived

Combining vertical and horizontal scaling

WebSphere applications can combine horizontal and vertical scaling to reap the benefits of both scaling techniques, as shown in Figure 6-6.

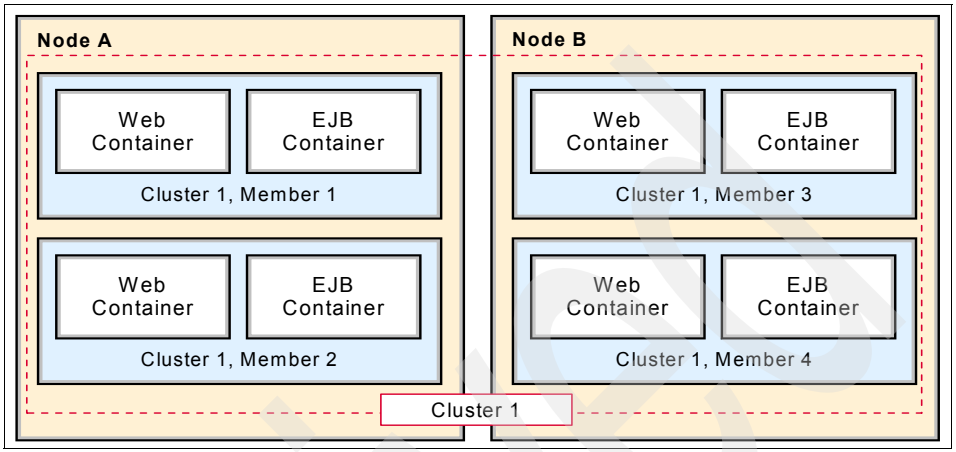


Figure 6-6 Vertical and horizontal scaling

Secure application cluster members

The workload management service has its own built-in security, which works with the WebSphere Application Server security service to protect cluster member resources. If security is needed for your production environment, enable security before you create a cluster for the application server. This enables security for all of the members in that cluster.

The EJB method permissions, Web resource security constraints, and security roles defined in an enterprise application are used to protect EJBs and servlets in the application server cluster. Refer to *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316, for more information.

6.5 WebSphere Commerce cell and cluster setup

In this book, we show you how to configure a complete WebSphere Commerce cell and cluster.

At the high level, the steps to set up a WebSphere Commerce cluster at the WebSphere Application Server tier are:

1. Configure a IBM WebSphere Application Server Network Deployment deployment manager.
2. Federate the first WebSphere Application Server node.

3. Federate an additional WebSphere Application Server node.
4. Create a WebSphere Application Server cluster.

For detailed implementation steps refer to:

https://www.ibm.com/developerworks/websphere/library/tutorials/0804_clustering1/0804_clustering1.html

Archived

Archived

7

Web tier High Availability

In this chapter, we describe how High Availability can be achieved for the Web server tier in WebSphere Commerce architectures. As described in Chapter 3, “Scenario for this book” on page 27, the Web tier typically consists of multiple Web servers and a *Load Balancer*, or *IP sprayer*, which distributes requests to the Web servers. As we will see, a Load Balancer will increase both availability and performance of a site if correctly set up.

To avoid having just another single point of failure, the Load Balancer itself needs to be made highly available, too.

© Copyright IBM Corp. 2008. All rights reserved.

Figure 7-1 shows the Web server tier as the highlighted portion of our sample topology (see Chapter 3, “Scenario for this book” on page 27).

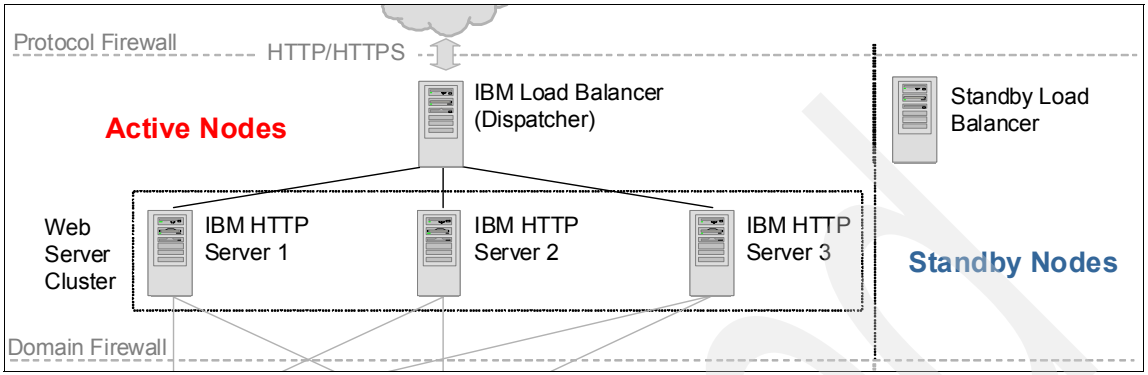


Figure 7-1 Web server tier

Archived

7.1 Introduction to Web server High Availability

A mechanism of providing request distribution and failover for incoming HTTP requests is required. Without this mechanism, the Web server becomes a single point of failure. Also, scalability is limited to the size of the hardware used to host the Web server.

7.1.1 Available solutions

A likely solution is to employ an IP sprayer to distribute HTTP requests to any number of active Web servers.

We prefer such a solution over active/passive clustering solutions where each Web server would have a passive standby node, monitoring the active one and taking over from it in case it experiences an outage. The reason for this is that an IP sprayer is very likely to be utilized for highly frequented sites, providing both High Availability and better performance, and saving hardware costs for passive standby nodes.

IBM WebSphere Edge Components provide a software-based Load Balancer as part of IBM WebSphere Application Server Network Deployment V6. There are a number of software or hardware-based solutions available from other vendors, too, for example:

- Astaro
- Barracuda Networks
- CAI Networks
- Cisco
- Citrix
- Coyote Point Systems
- Crescendo Networks
- DBAM Systems
- Elfiq Networks
- FatPipe Networks
- F5 Networks
- Foundry Networks
- jetNEXUS
- Juniper® Networks
- KEMP Technologies
- Nortel
- Radware
- Zeus Technology
- Sentral Systems Ltd

Archived

In this book, we describe how to use IBM WebSphere Edge Components Load Balancer for WebSphere Commerce. As IBM WebSphere Edge Components are bundled with IBM WebSphere Application Server Network Deployment, many of our customers use them rather than buying an extra third-party solution.

7.1.2 IBM WebSphere Edge Components Load Balancer

IBM WebSphere Edge Components Load Balancer consists of multiple components that can be used separately or together.

For IP spraying, we only need to use the *Dispatcher* component. See Chapter 4, "Introduction to IBM WebSphere Edge Components," in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, for a detailed introduction to Load Balancer as part of IBM WebSphere Edge Components.

Note: We only describe the features that are essentially necessary for our scenarios. We do not cover optional features that are deprecated when using IBM WebSphere Application Server Network Deployment V6.1. Most of the features of IBM WebSphere Edge Components V6.0 are deprecated in V6.1, where they are provided by the IBM WebSphere Application Server Network Deployment proxy server and the IBM HTTP Server Plug-in.

Deprecated features include all components except Dispatcher, and all Dispatcher forwarding methods except MAC. As at the time of writing this book, WebSphere Commerce does not support IBM WebSphere Application Server Network Deployment V6.1, though we still describe some features that are deprecated with V6.1.

For a complete list of deprecated features in IBM WebSphere Application Server Network Deployment V6.1, visit the following URL of the V6.1 Info Center:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rmig_deprecationlist.html`

Dispatcher has internal components that are responsible for various load balancing tasks. The components and tasks are:

- *Executor* is the core component of Dispatcher, and is responsible for the load distribution.
- *Manager* is the component responsible for providing weight values of each balanced server to Executor, so it can make its load balancing decision.
- *Advisors* are lightweight clients that run on the Dispatcher server, and they are aware of the protocol used by the back-end servers. Load Balancer

provides advisors for HTTP, HTTPS, FTP, and LDAP, among others. Advisors periodically measure the response time for each server and provide the results to the manager as input for calculating server weights.

- *Metric Server* is a component that is installed and runs in each back-end server (for example, Web server) and that provides values for the server where it is running (for example, memory and CPU usage), which are then also sent to the manager.
- *Dispatcher* can be administered through a command-line interface (**dscontrol**) and through a GUI (**lbadmin**).

Figure 7-2 shows the Dispatcher components and their interactions.

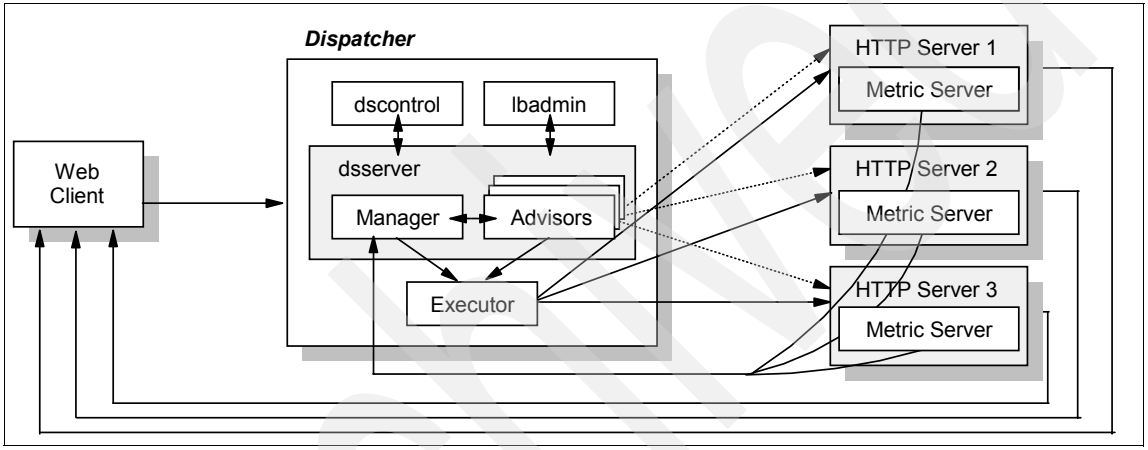


Figure 7-2 Dispatcher components interaction (from WebSphere Application Server V6 Scalability and Performance Handbook, SG24-6392)

All requests are routed to Dispatcher, which in turn sprays them among the members of the Web server cluster (IP spraying). The Web server cluster consists of identical Web servers running on different physical machines (or in different LPARs). All Web servers need to be set up in the same way to serve static content and route requests to the application server tier for WebSphere Commerce. See 11.1, “Add additional Web servers” on page 186. In the event of a failure of one of the Web servers, Dispatcher discontinues directing work to the failed server. Dispatcher provides two IP spraying methods:

1. MAC forwarding is the fastest forwarding method because Dispatcher receives only the incoming traffic. All outbound traffic is sent directly from the balanced server to the client. Each Web server in the topology is configured with at least one physical IP address and a loopback alias configured with a shared virtual IP address, also called the *cluster address*. The cluster address is configured as an alias of the Load Balancer’s network interface. HTTP

clients make HTTP requests to this virtual IP address, and Dispatcher forwards the requests to the Web servers by changing the source and destination MAC address of the packet. The IP addresses remain the same. Because the IP header is not changed, the Web servers send their responses directly to the client, as shown in Figure 7-3.

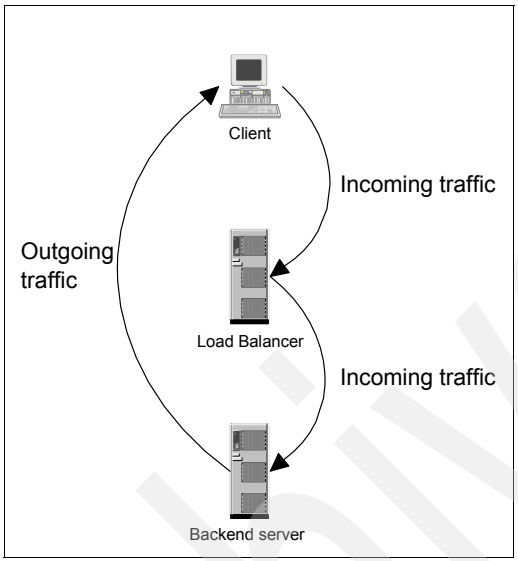


Figure 7-3 MAC forwarding - network flow (from WebSphere Application Server V6 Scalability and Performance Handbook, SG24-6392)

However, the Load Balancer and the Web servers must all be on the same IP subnet. Also, for server affinity, which is a useful feature in performance tuning (see Chapter 19, “Monitor and tune Load Balancer” on page 417), source IP affinity (across multiple ports) is the only available option.

2. Network Address Translation (NAT) and Content Based Routing (CBR) allow the Load Balancer and the Web servers to be on different IP subnets. Cookie and SSL session ID based server affinity are also possible.

Archived

This method does not use loopback aliases. Instead, a *return address* is configured as an additional alias of the Load Balancer's network interface. The requests made to the cluster address are changed at the IP layer, such that the Web server's IP address is put into the request as the destination and the return address as the source for each packet. The packets can then be routed to the Web server, and the Web server will send the response back to the Load Balancer, which needs to change the IP headers of the response packets so that they can be routed back to the client, as shown in Figure 7-4.

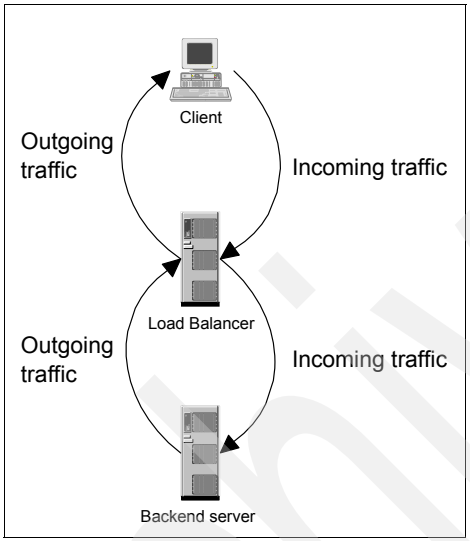


Figure 7-4 NAT forwarding - network flow (from WebSphere Application Server V6 Scalability and Performance Handbook, SG24-6392)

We describe how to set up Load Balancer for our scenario in 8.6, "Install Load Balancer" on page 140. For detailed instructions on how to set up the two different forwarding methods, refer to 11.2.1, "MAC forwarding" on page 193, and 11.2.2, "NAT forwarding" on page 210.

7.2 Introduction to Load Balancer High Availability

Being the entry point into your WebSphere system, it is extremely important that your Load Balancer is highly available. Otherwise, it would be a single point of failure.

While it is possible to use external clustering software (see Chapter 4, "External clustering software" on page 33) to monitor the Load Balancer node and provide failover to a backup node, IBM WebSphere Edge Components Load Balancer

Archived

provides a built-in High Availability solution, which allows you to configure a backup Load Balancer server with exactly the same configuration. If the primary Load Balancer server fails, the backup server will take over the load balancing for all clusters.

The two Load Balancer servers need connectivity to the same clients and to the same cluster of servers, as well as connectivity between themselves. Both Load Balancer servers must be running the same operating systems, and they must be connected to the same network.

The two Load Balancer servers are referred to as the *primary* server and the *backup* (or standby) server. These are the *roles* that are associated with each server during the configuration.

Load Balancer servers run in a specific state: one server is *active* and the other server is in *standby* state. This means that the Load Balancer server that is in active state is the one that is distributing the workload.

The Load Balancer server that is in the standby state monitors the active one. If the active server fails, the standby server performs a failover. It switches to the active state, takes over the cluster IP alias, and starts load balancing the cluster. So the state of a server changes when a failure occurs, but the roles do not change during a failure.

A Load Balancer server can be the primary server for one Web server cluster while acting as standby for another cluster. If the primary Load Balancer of that other cluster is also the standby for the first cluster, the resulting configuration is called *mutual High Availability*.

However, it is not possible to have two primary servers for one cluster, both doing active load balancing at the same time (for example, an active/active configuration), as the virtual IP alias for the cluster can only be configured on one server in the network.

We describe how to set up High Availability for IBM WebSphere Edge Components Load Balancer in 11.3, “Configure Load Balancer High Availability” on page 226.

Archived

Part 3

Install and configure a High Availability WebSphere Commerce system

In our example of setting up a highly available and multi-tier WebSphere Commerce environment, we take the following steps:

1. Configuration of a Multi-tier WebSphere Commerce environment
   a. Configuring the database server
   b. Configuring the Web server
   c. Configuring the application server
   d. Creating a WebSphere Commerce Instance
   e. Publishing a consumer direct sample store

© Copyright IBM Corp. 2008. All rights reserved.

2. Configuration of High Available Database (HADB) servers
    a. Configuring the HADR on a primary/standby database
    b. Enabling client reroute in a HADR environment
    c. Installing Tivoli System Automation
    d. Defining and administering a TSA cluster
    e. Enabling instance and HADR with TSA
3. Configuration of a WebSphere Application Server cluster
    a. Configuring an IBM WebSphere Application Server Network Deployment Manager
    b. Federating the first WebSphere Application Server Node
    c. Federating an additional WebSphere Application Server Node
    d. Creating a WebSphere Application Server Cluster
4. Configuration of an IBM HTTP Server cluster
    a. Installing an additional IBM HTTP Server
    b. Configuring an additional IBM HTTP Server
    c. Introduction to the WebSphere Load Balancer
    d. Installing the WebSphere Load Balancer Dispatcher Component
    e. Configuring the WebSphere Dispatcher Component
    f. Reconfiguring the IBM HTTP Servers and the Plug-in
    g. Verifying the IBM HTTP Server Cluster

The detailed implementation steps of configurations of HADB are found in 9.1, “HADR” on page 146.

The detailed implementation steps of configurations of IBM HTTP Server Cluster are found in 8.5, “Install IBM HTTP Server” on page 127.

Implementation details of all other sections are found in the whitepaper *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0* at:

http://www.ibm.com/developerworks/websphere/library/tutorials/0804_clustering1/0804_clustering1.html

Archived

8

Base product and fix pack installations for all tiers

In this chapter, we describe how to install the products needed for a highly available and highly performing WebSphere Commerce environment. We:

- Explain how to install IBM DB2 Universal Database on the database nodes.
- Explain how to install the primary WebSphere Commerce node and additional WebSphere Commerce nodes. While the base installation on the other tiers is identical for all nodes in the tier, the installation of the additional WebSphere Commerce nodes differs from the primary node.
- Show you how to install and configure an IBM WebSphere Application Server Network Deployment Deployment Manager node.
- Describe the installation of the Web server nodes, using IBM HTTP Server.
- Describe the installation of the primary Load Balancer and Standby Load Balancer node, using IBM WebSphere Edge Components Load Balancer.

© Copyright IBM Corp. 2008. All rights reserved.

Figure 8-1 highlights the primary nodes. The primary database, Commerce, and Web server nodes are necessary to run a basic WebSphere Commerce environment. The primary Network Deployment Manager and Load Balancer nodes will then be needed for basic High Availability of Web and application servers, while the standby nodes will later enable High Availability of the Network Deployment Manager, Load Balancer, and database nodes.

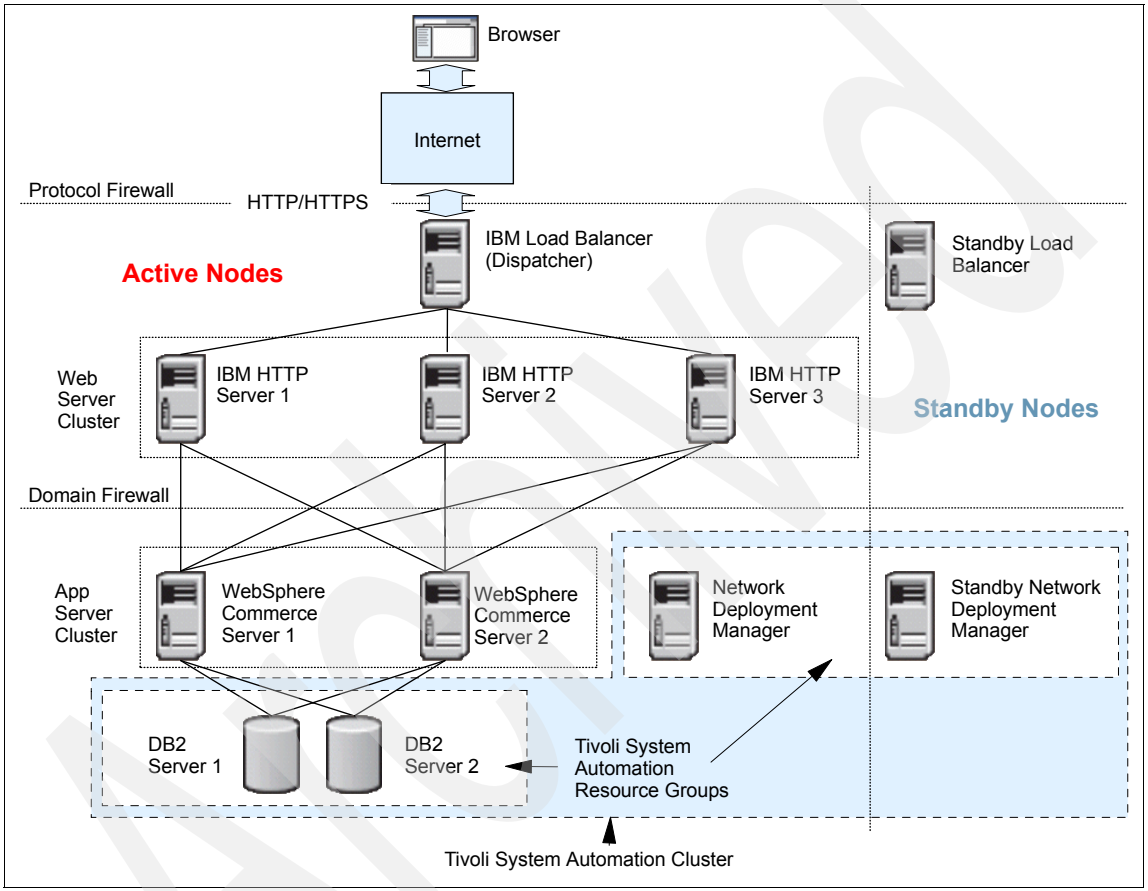


Figure 8-1 Primary nodes

Archived

8.1 Database nodes

DB2 products are available for the following platforms: AIX, HP-UX Version 11i v2 (B.11.23) for Itanium-based™ systems, Linux (Intel®), Linux (iSeries and pSeries), Linux (S/390®, zSeries), Linux (AMD™), and the Solaris™ Operating Environment. In this chapter, we only discuss the installation and configuration steps for DB2 in AIX operating system.

8.1.1 DB2 installation prerequisites

Generally, before we start to install DB2 UDB in AIX operating system, there are some pre-install requirements, some of which are WebSphere Commerce specific and the others that are DB2 required.

WebSphere Commerce prerequisites checking

First of all, we have to make sure that the operating system meets the following prerequisites:

- For AIX 5.3, check your operating system level by issuing the following command:

  ```
  oslevel -r
  ```

  This command might return the following value:

  ```
  5300-04
  ```

  If the output from the command does not end in -01 or higher, it means you are not at the correct AIX maintenance level for WebSphere Commerce. Obtain the correct maintenance level from IBM@ eServer™ pSeries@ Support Web site:

  http://www-912.ibm.com/eserver/support/fixes/fixcentral/main/pseries/aix

- The following AIX APARs are installed:
  - IY67744 (DB2 users only)
  - IY68989 (Oracle users only)
  - IY58143
- You can query your system to see if a particular APAR is installed with the following command:

  ```
  instfix -v -i -k APAR_number
  ```

  For example:

  ```
  instfix -v -i -k IY67744
  ```

Archived

You will be able to get a output as in Figure 8-2.

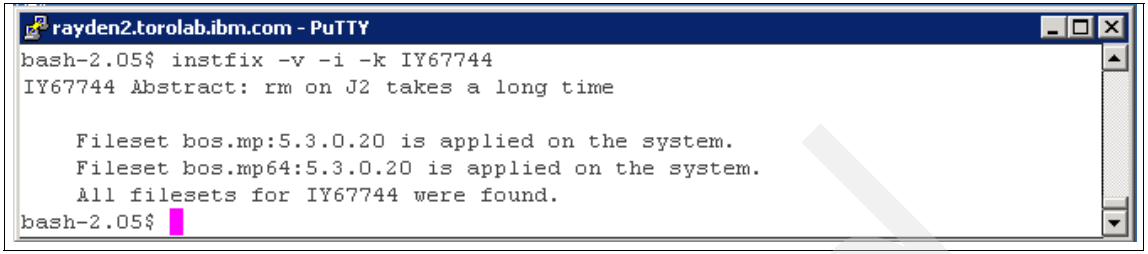


Figure 8-2 instfix result

- Ensure that the stack quota limit is at least 32768. To check the current limit, type the following command in a command window:

  ```
  ulimit -a
  ```

  If the value returned for the stack is less than 32768, increase it to this level by running the following command:

  ```
  ulimit -s 32768
  ```

- Ensure that asynchronous I/O is enabled. This is especially important for logically partitioned (LPAR) systems where asynchronous I/O is enabled on the first LPAR but not on subsequent LPAR.
- If the database server and WebSphere Commerce server are on different machines, you have to take care of the time stamp. The time stamp for both database server and WebSphere Commerce should be same. The server times that are synchronized with each other will avoid the available order items being treated as future orders. On the other hand, in a High Availability environment, it is essential to keep the synchronization between the primary and standby servers, so it is necessary to keep the time stamp synchronized.

  If the time stamps on your servers are different, run the following commands:

  a. Define a local standard time server:

     ```
     startsrc -s xntpd
     ```

  b. Synchronize the client server with the time server:

     ```
     setclock TimeServerName
     ```

  c. Stop the daemon in time server:

     ```
     stopsrc -s xntpd
     ```

Archived

DB2 prerequisites checking

The recommended approach to do DB2 installation prerequisites checking includes two steps:

1. Verify that your environment meets the disk, memory, and installation requirements for the operating system and DB2 product that you are planning to install.
2. Review the DB2 Release Notes for information about the functionalities of this release and how to perform the installation in a different platform.

Generally, we have to consider the following items before installing DB2:

- Disk requirements

  The disk space required for your product depends on the type of installation that you choose and the type of file system that you have. Remember to include disk space for required software, communication products, and documentation.

- Memory requirements

  The basic memory requirement for a system running just DB2 UDB and the DB2 GUI tools is 512 MB, but we recommend 1 GB of RAM memory for improved performance. These requirements do not include any additional memory requirements for other software that is running on your system.

- The appropriate IBM Software Development Kit for Java level for DB2 UDB

  The SDK is installed whenever a component that requires Java is being installed. However, if the installer detects that SDK is already installed, it will not install it again. The SDK is installed in its own directory and does not overwrite any previous levels of the SDK. In cases where 64-bit Java is required, a message appears telling you that Java 64-bit is required.

  You can issue the version of JDK™ by issuing the following command, to make sure that the JDK level is aligned with the requirement of a specific DB2 UDB version:

  ```
  java -version
  ```

  The sample output is displayed as shown in Figure 8-3.

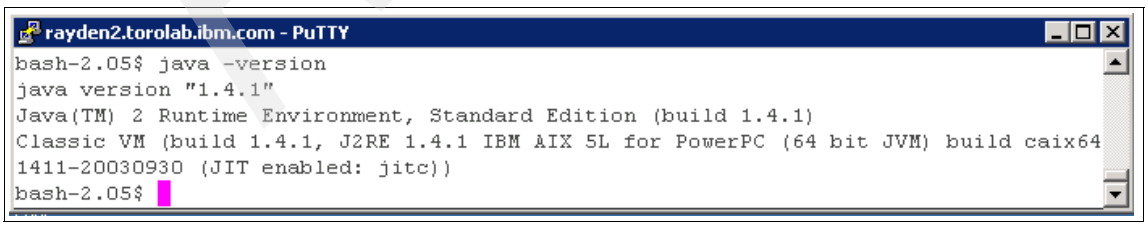


Figure 8-3 Java version display

For the most up-to-date SDK for Java information, see the Java support for DB2 UDB Web page at:

http://www-1.ibm.com/support/docview.wss?rs=71&uid=swg21251460

8.1.2 Base product installation

After downloading the DB2 Enterprise Server Edition v8.2 with the latest fix pack installation package or using the installation CD, follow the steps listed below to install the base product installation:

1. Log into AIX as root. From the command line, run the db2 install:

   ```
   root> ./db2setup
   ```

2. From the Setup Launchpad, choose **Installation Prerequisites** from the panel on the left, read through the installation prerequisites and confirm that they are all satisfied. If so, close Installation Prerequisites.
3. From the Setup Launchpad, choose **Install Products** from the left panel, select the **DB2 UDB Enterprise Server Edition**, and click **Next**.
4. From the DB2 Setup Wizard, choose **Accept** from the software license agreement, and click **Next**.

Archived

5. As in Figure 8-4, choose **Typical** from the Select the installation type panel, and choose the desired capabilities from the Additional Functions section. Click **Next**.

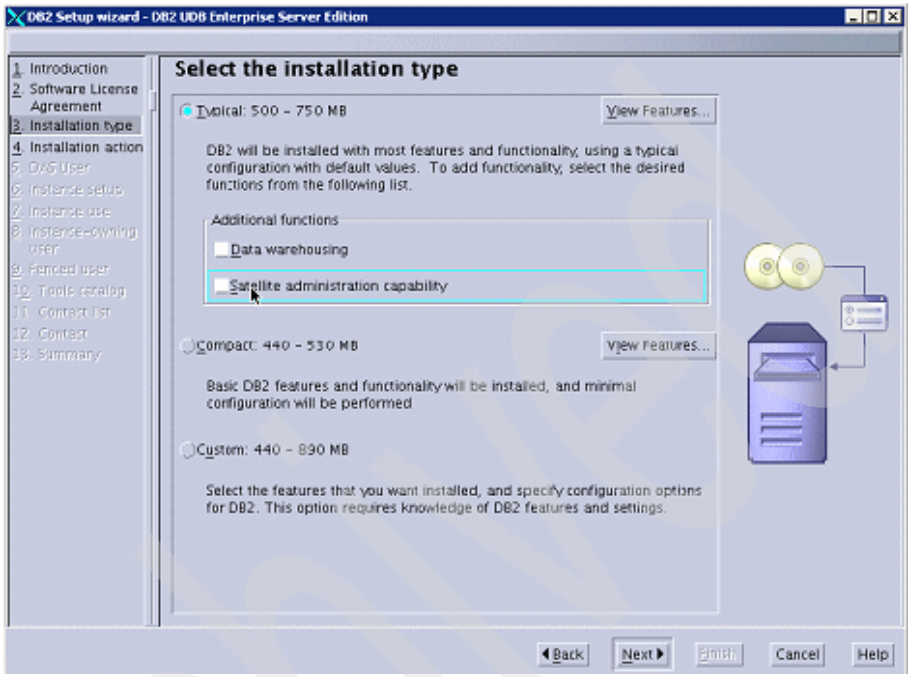


Figure 8-4 Select the installation type

6. Choose **Install DB2 UDB Enterprise Server Edition** on this computer from the Select the installation action panel, and click **Next**.

Archive

7. In Figure 8-5, select **New User** from the Set user information for the DB2 Administration Server panel. Leave the defaults and enter your password. Click **Next**.

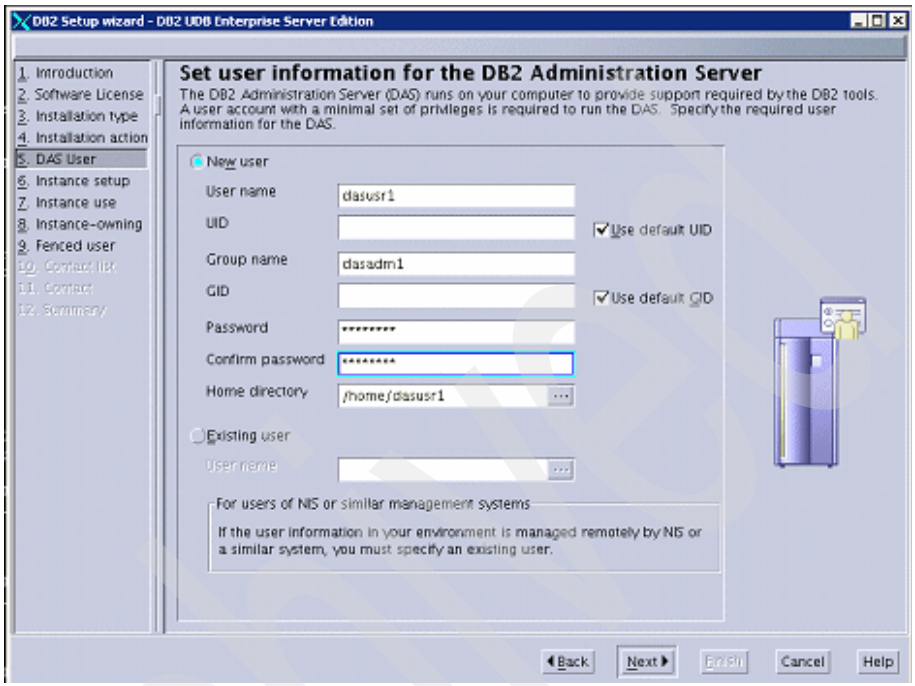


Figure 8-5 Add user information for DB2 installation

8. Select **Create a DB2 instance - 64 bit** from the Set up a DB2 instance panel.
9. Select **Single-partition instance** from the Select how the instance will be used panel. Click **Next**.

10. In Figure 8-6, select **New User** from the Set user information for the DB2 instance owner panel. Leave the defaults and enter your password. Click **Next**.

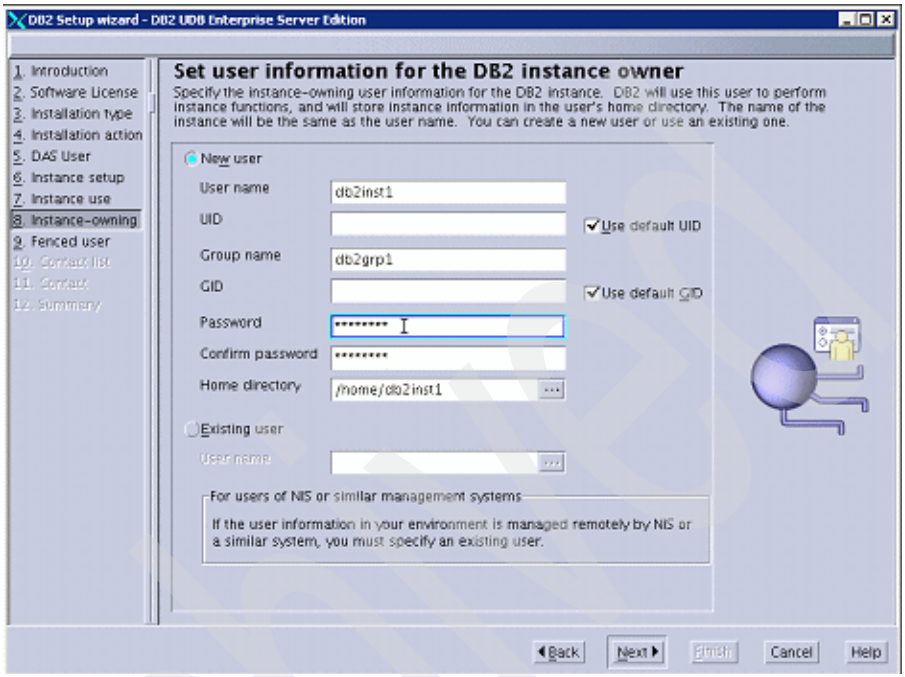


Figure 8-6 Set information for DB2 instance user

Archive

11. As in Figure 8-7, select **New User** from the Set user information for the fenced user panel. Leave the defaults and enter your password. Click **Next**.

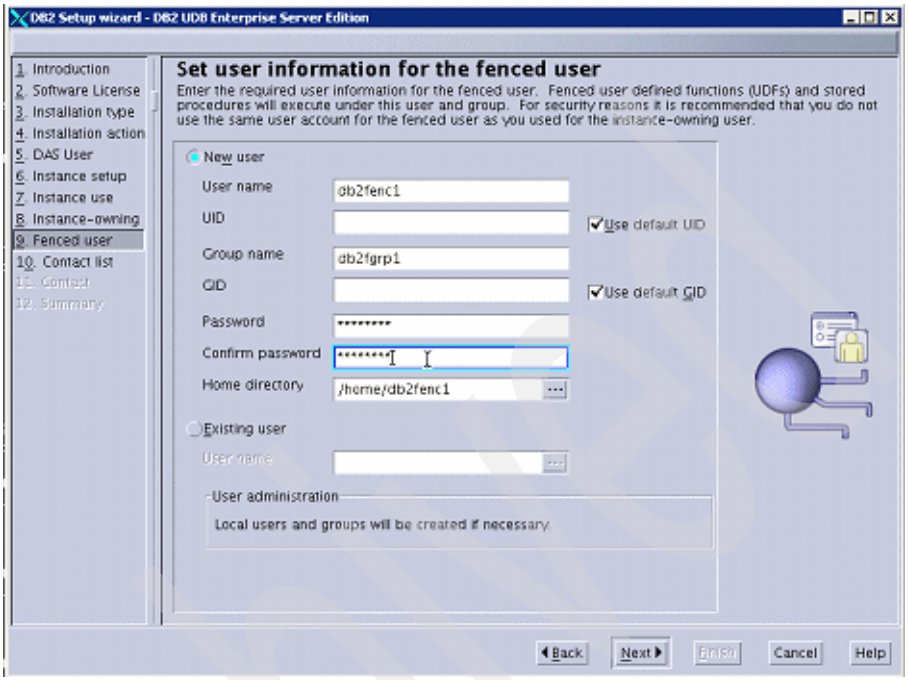


Figure 8-7 Set information for DB2 fenced user

12. Select **Use a local database** from the Prepare the DB2 tools catalog panel.
13. Keep the defaults on the Specify a local database to store the DB2 tools catalog panel and click **Next**.
14. Select **Local - Create a contract list on this system** from the Set up the Administration contact list panel. Click **Next**.

Archive

15. As in Figure 8-8, review your information for corrections on the Start copying files panel. Click **Finish**.

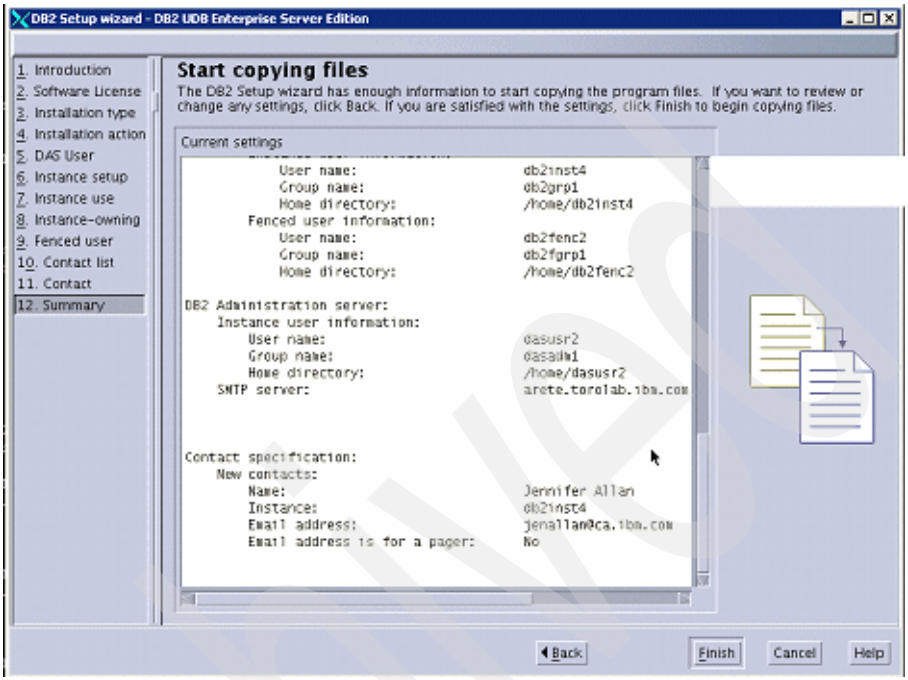


Figure 8-8 Review information before copying files

Archive

16. Wait for the installation and instance creation to complete (Figure 8-9).

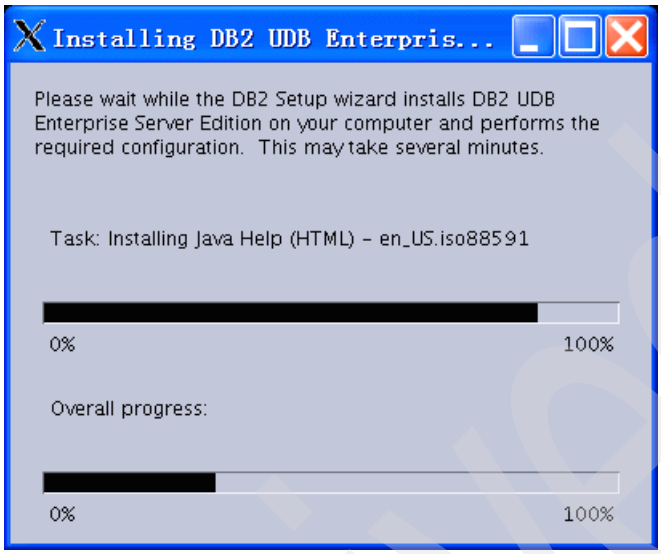


Figure 8-9 DB2 installation status

Archived

8.1.3 Manually create DB2 64-bit instance

After installation, a Setup Complete panel will be displayed, as in Figure 8-10.

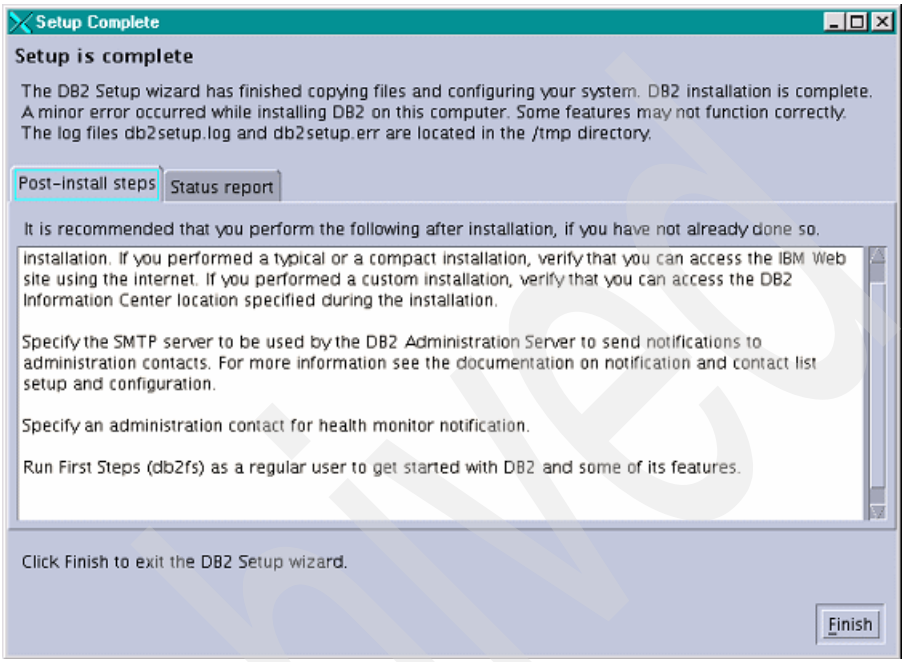


Figure 8-10 Setup Complete

Also, you might get some indication from the Status report tab, as in Example 8-1.

Example 8-1 Status report for DB2 installation

```
Enabling Asynchronous I/O:.......Success
Checking license agreement acceptance:.......Success
Installing DB2 file sets:.......Success
Registering DB2 licenses:.......Success
Setting default global profile registry variables:.......Success
Creating the DB2 Administration Server:.......Success
Initializing instance list:.......Success
Creating DB2 instances:.......Failure
Building list of databases to create:.......Success
Creating DB2 databases:.......Failure
Configuring the DB2 Administration Server:.......Success
Updating global profile registry:.......Success
Creating DB2 tools catalog:.......Failure
```

Archive

Troubleshooting

In some cases, the DB2 installer may fail to create the 64-bit instance during the installation process. The workaround is to apply the latest DB2 fix pack, then manually create the 64-bit instance again. However, since the installer created the db2 instance, fence user, and group, if you would like to use the same names, you must manually remove them first before creating the instance again.

In our test, the installer failed to create a DB2 64-bit instance during the DB2 installation process, so we took the manual steps to create the instance using the instance creation wizard. Follow steps below:

1. Initiate the DB2 Instance Setup wizard. During the Instance-Owning step, select **Existing user db2inst1** and click **Next**.
2. For the Fenced user step, select **Existing user dbfenc1** and click **Next**.
3. For the instance TCP/IP step, as shown in Figure 8-11, enter the Service name as `db2c_db2inst1`, while the port number is 50000. Click **Next**.

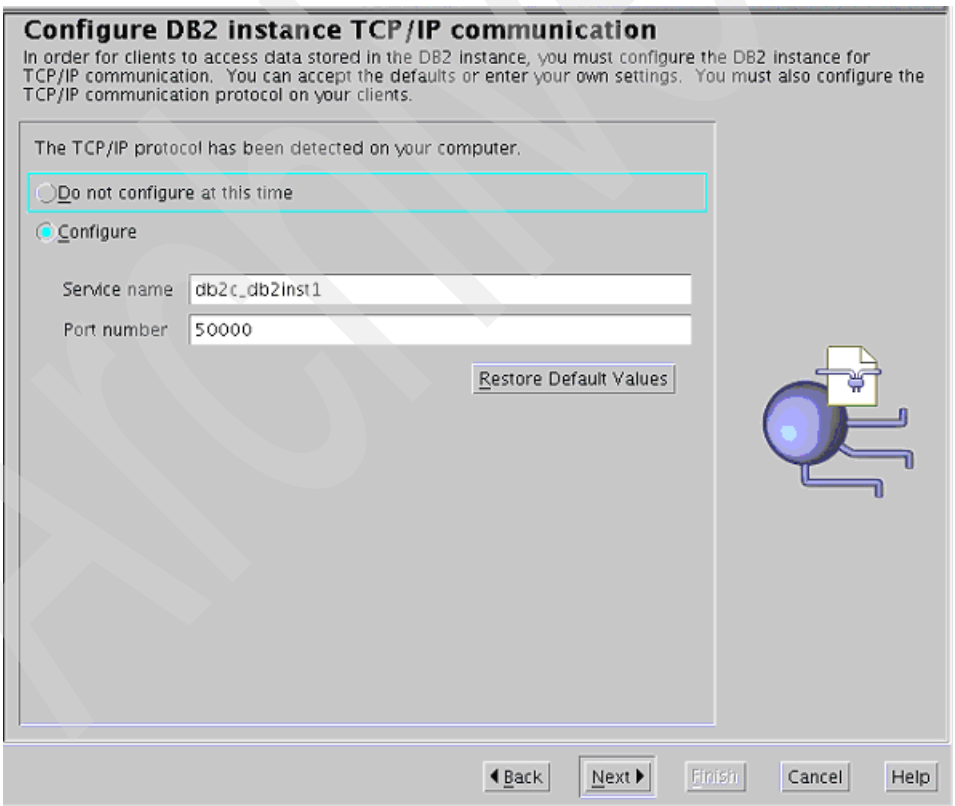


Figure 8-11 DB2 instance TCP/IP communication configuration

4. After the instance creation process finished, check the Status report tab to make sure that instance is successfully created.
5. If you get a status report like Figure 8-12, you have to analyze the root cause of the failure.

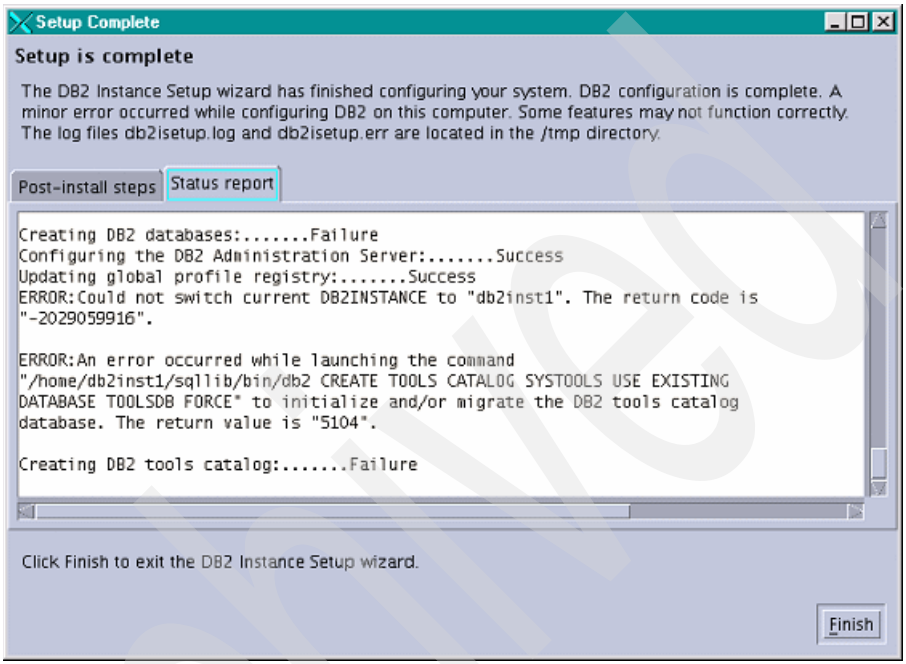


Figure 8-12 Failed status report

6. Check the db2setup.log and db2icrt.log. You could get some indication of the problem, as in Example 8-2 and Example 8-3.

Example 8-2 db2setup.log

```
Command to be run: "/usr/opt/db2_08_01/instance/db2icrt -a SERVER -s ese -u
db2fenc1 -w 64 -p db2c_db2inst1 db2inst1".
ERROR:An error occurred while creating the instance "db2inst1". The return
codeis "1". Create the instance manually using the command "db2icrt"
```

Example 8-3 db2icrt.log

```
Could not load program db2:
Could not load module /usr/opt/db2_08_01/lib/libdb2.a(shr_64.o).
        Dependent module /usr/opt/db2_08_01/lib/libdb2trcapi.a(shr_64.o) could
not be loaded.
        Member shr_64.o is not found in archive
Could not load module db2.
```

```
       Dependent module /usr/opt/db2_08_01/lib/libdb2.a(shr_64.o) could not be
loaded.
Could not load module .
Update DBM cfg SYSADM_GROUP errcode = 255
DBI1281E The database manager configuration file could not be initialized
```

7. In that case, an explanation according to the symptom is that an error occurred when attempting to initialize the database manager configuration file. So that a DB2 instance cannot be created or migrated. We can compare the problematic module on a working DB2 machine (A) with the broken DB2 machine (B). We find that the size and time stamp are different. Example 8-4 and Example 8-5 are the sample comparison results for machine A and B.

Example 8-4 Comparison for libdb2trcapi.a module

```
Database node 1:/usr/opt/db2_08_01/lib>ls -atl |grep libdb2trcapi.a
-r--r--r--   1 bin      bin           56372 Nov 13 21:56 libdb2trcapi.a

Database node 2:/usr/opt/db2_08_01/lib>ls -alt |grep libdb2trcapi.a
-r--r--r--   1 bin      bin           54678 Aug 15 2004  libdb2trcapi.a
```

Example 8-5 Comparison for libdb2.a module

```
Database node 1:/usr/opt/db2_08_01/lib>ls -atl |grep libdb2.a
lrwxrwxrwx   1 root     system           27 Feb 19 20:33 libdb2dasflist.a ->
../                                                     das/lib/libdb2dasflist.a
lrwxrwxrwx   1 root     system           25 Feb 19 20:33 libdb2dasgcf.a ->
../da                                                      s/lib/libdb2dasgcf.a
lrwxrwxrwx   1 root     system           17 Feb 16 20:30 libdb2.a ->
../lib64/li                                                      bdb2.a
-r--r--r--   1 bin      bin            2028 Nov 13 21:56 libdb2lai.a
-r--r--r--   1 bin      bin          674799 Nov 13 21:56 libdb2dasapi.a
-r--r--r--   1 bin      bin          177530 Nov 13 21:56 libdb2dascmn.a
-r--r--r--   1 bin      bin            3352 Nov 13 21:56 libdb2daswrap.a
-r--r--r--   1 bin      bin           94306 Nov 13 21:55 libdb2dasjutil.a

Database node 2:/usr/opt/db2_08_01/lib>ls -atl |grep libdb2.a
lrwxrwxrwx   1 root     system           27 Mar 14 14:03 libdb2dasflist.a ->
../das/lib/libdb2dasflist.a
lrwxrwxrwx   1 root     system           25 Mar 14 14:03 libdb2dasgcf.a ->
../das/lib/libdb2dasgcf.a
lrwxrwxrwx   1 root     system           17 Mar 14 13:59 libdb2.a ->
../lib64/libdb2.a
-r--r--r--   1 bin      bin            2013 Aug 15 2004  libdb2lai.a
-r--r--r--   1 bin      bin          650631 Aug 15 2004  libdb2dasapi.a
-r--r--r--   1 bin      bin          160125 Aug 15 2004  libdb2dascmn.a
-r--r--r--   1 bin      bin            3336 Aug 15 2004  libdb2daswrap.a
-r--r--r--   1 bin      bin           93525 Aug 15 2004  libdb2dasjutil.a
```

Archive

A potential solution for this symptom is to install the latest DB2 fix pack.

8.1.4 Installation of DB2 fix pack

After you download the latest DB2 fix pack (such as fix pack 14), as root user, we can issue the following command to start the fix pack installation:

```
rayden2:/usr/XiaoQing/DB2_Fixpack10/fixpak.s050811 # ./installFixPak
```

After fix pack installation, we can get the installation summary, as in Example 8-6.

Example 8-6 Fix pack installation summary

```
+-----------------------------------------------------------------------------+
                            Summaries:
+-----------------------------------------------------------------------------+

Installation Summary
--------------------
Name                           Level          Part       Event      Result
-------------------------------------------------------------------------------
db2_08_01.ch.en_US.iso88591 8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.cc                8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.sqlproc           8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.repl              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.pext              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.msg.en_US.iso8859 8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.ldap              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.jhlp.en_US.iso885 8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.jdbc              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.inst              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.icut              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.icuc              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.fs                8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.essg              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.dj                8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.db2.samples       8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.db2.rte           8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.das               8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.cs.rte            8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.conv              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.conn              8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.cnvucs            8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.client            8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.cj                8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.ca                8.1.1.128      USR        APPLY      SUCCESS
db2_08_01.db2.engn          8.1.1.128      USR        APPLY      SUCCESS
```

Archived

```
db2_08_01.ca                 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.cc                 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.ch.en_US.iso88591 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.db2.samples        8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.fs                 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.icuc               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.icut               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.jhlp.en_US.iso885 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.msg.en_US.iso8859 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.pext               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.client             8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.cj                 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.cnvucs             8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.conv               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.db2.rte            8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.jdbc               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.ldap               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.repl               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.sqlproc            8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.conn               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.cs.rte             8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.das                8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.db2.engn           8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.dj                 8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.essg               8.1.1.128       USR         COMMIT      SUCCESS
db2_08_01.inst               8.1.1.128       USR         COMMIT      SUCCESS

Log saved in /tmp/installFixPak.log.8.1.1.128
```

Make sure that DB2 fix pack 14 is installed successfully, then apply the same approach to create the DB2 instance manually as described in 8.1.3, “Manually create DB2 64-bit instance” on page 101.

8.2 WebSphere Commerce node 1

We use AIX as the operating system for the Web servers. All instructions, paths, and commands therefore apply to AIX.

Installation prerequisites

Follow the prerequisites section of the *WebSphere Commerce Installation Guide*.

You must first create your WebSphere Application Server/WebSphere Commerce *non-root_user* before starting the installation:

1. Create the *non-root_user*, for example, “wasuser”.

2. Create the *non-root_group*, for example, “wasgroup”, and assign the *non-root_user* to *non-root_group*.

The installation must be done as root.

Install WebSphere Commerce using the installation wizard

Perform the following instructions to install Commerce using the install wizard:

1. Run the installer from your Installation_Source_Directory using the setup command:

   `Hostname:/<Installation_Source_Directory>./setup`

   The relevant images of the installation follow.

Archived

You are introduced to the WCS software stack (Figure 8-13).

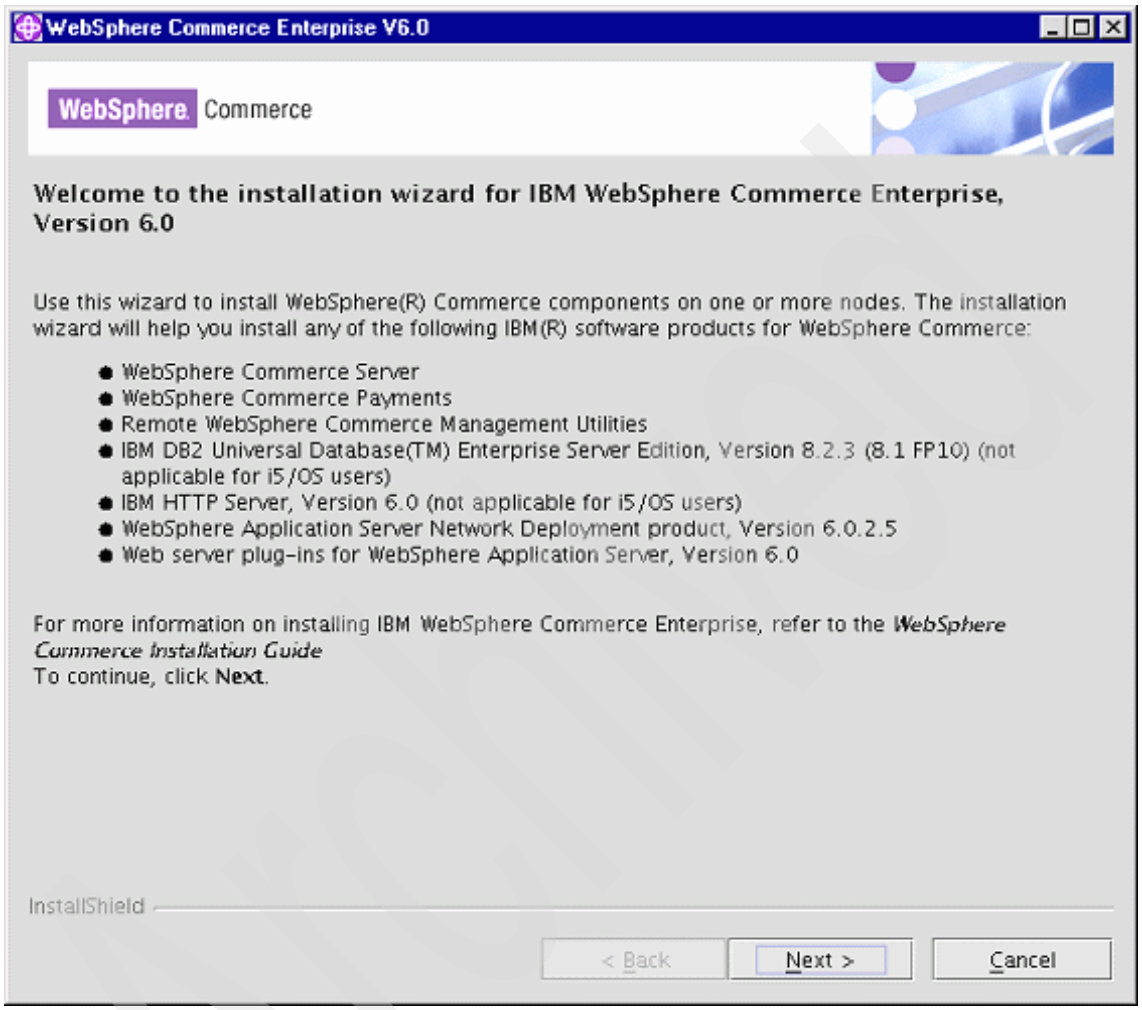


Figure 8-13 WebSphere Commerce Software Stack

2. Select **Custom install** (Figure 8-14).

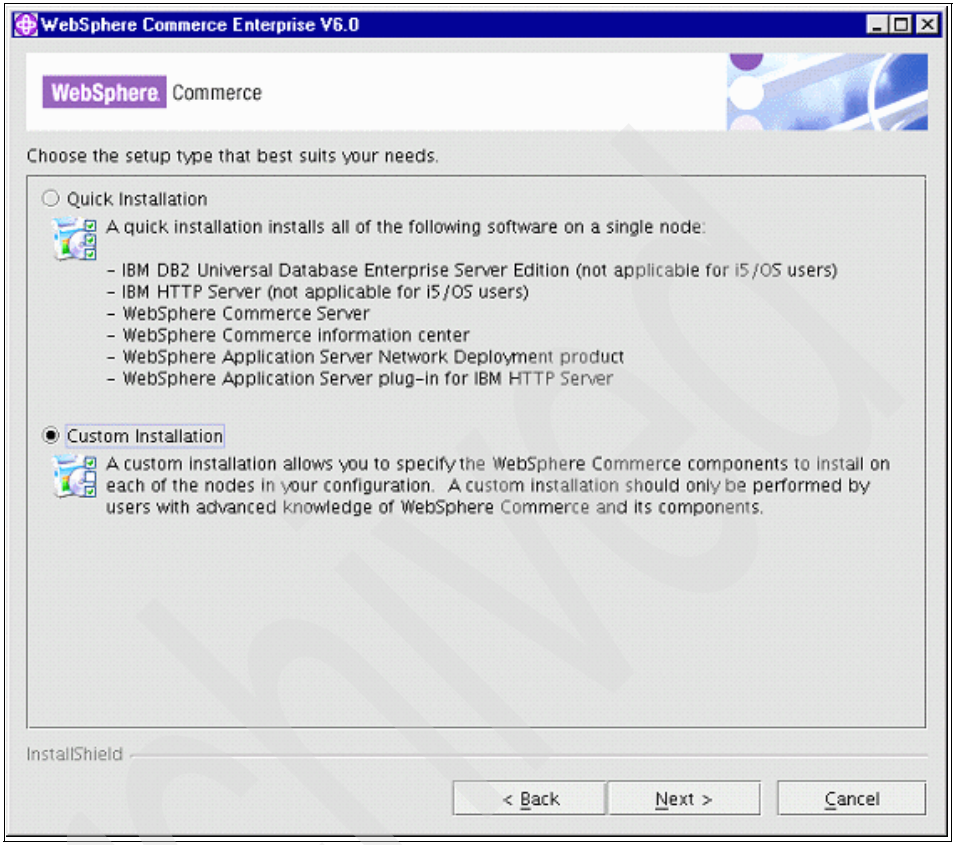


Figure 8-14 WebSphere Commerce installation type selection

Archive

3. Select the components to install (Figure 8-15).

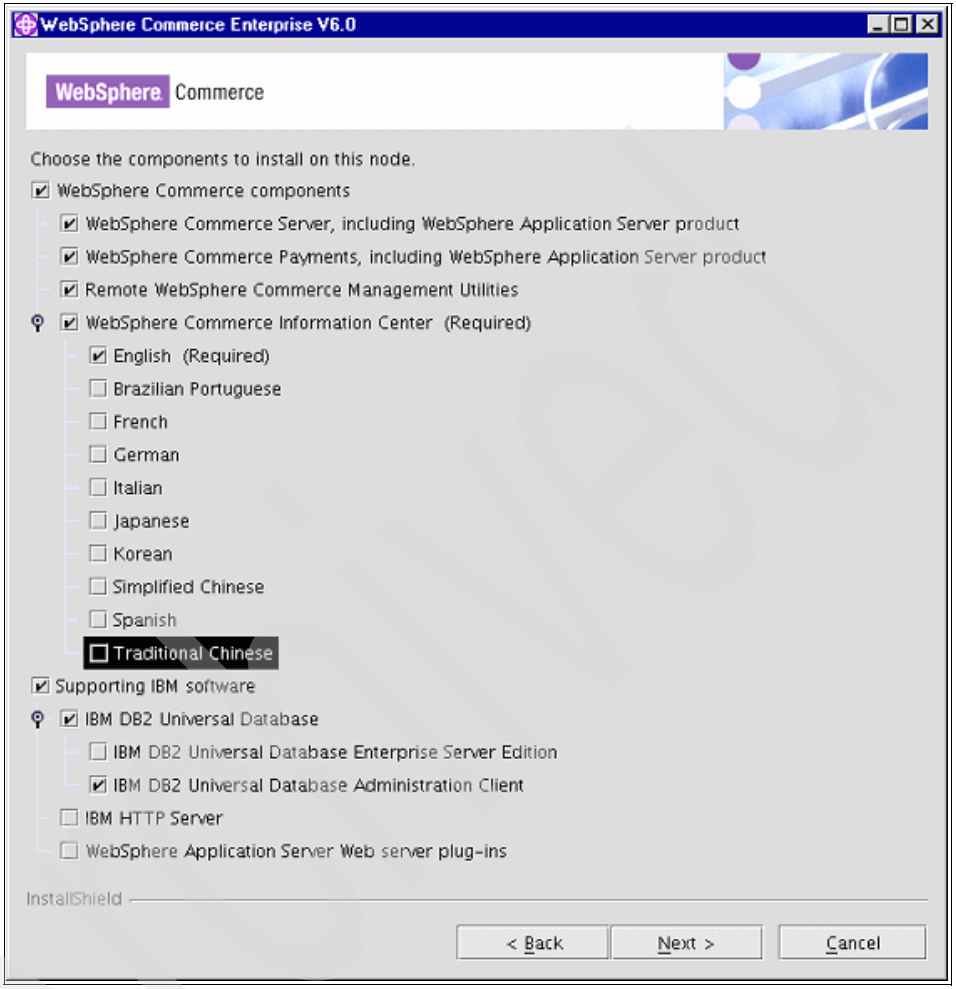


Figure 8-15 WebSphere Commerce component selection

For a WebSphere Commerce node, select **WebSphere Application Server** (WCS), the DB2 Administration Client (to connect to a remote database).

Since the database and the WebServer are remote, do not select:

- The IBM DB2 Universal Database
- The IBM HTTP Server
- The Web server plug-ins

4. Specify the WebSphere Application Server and WCS installation directories *WAS_Install_Dir* and *WC_Install_Dir*. We use the default directories (Figure 8-16).

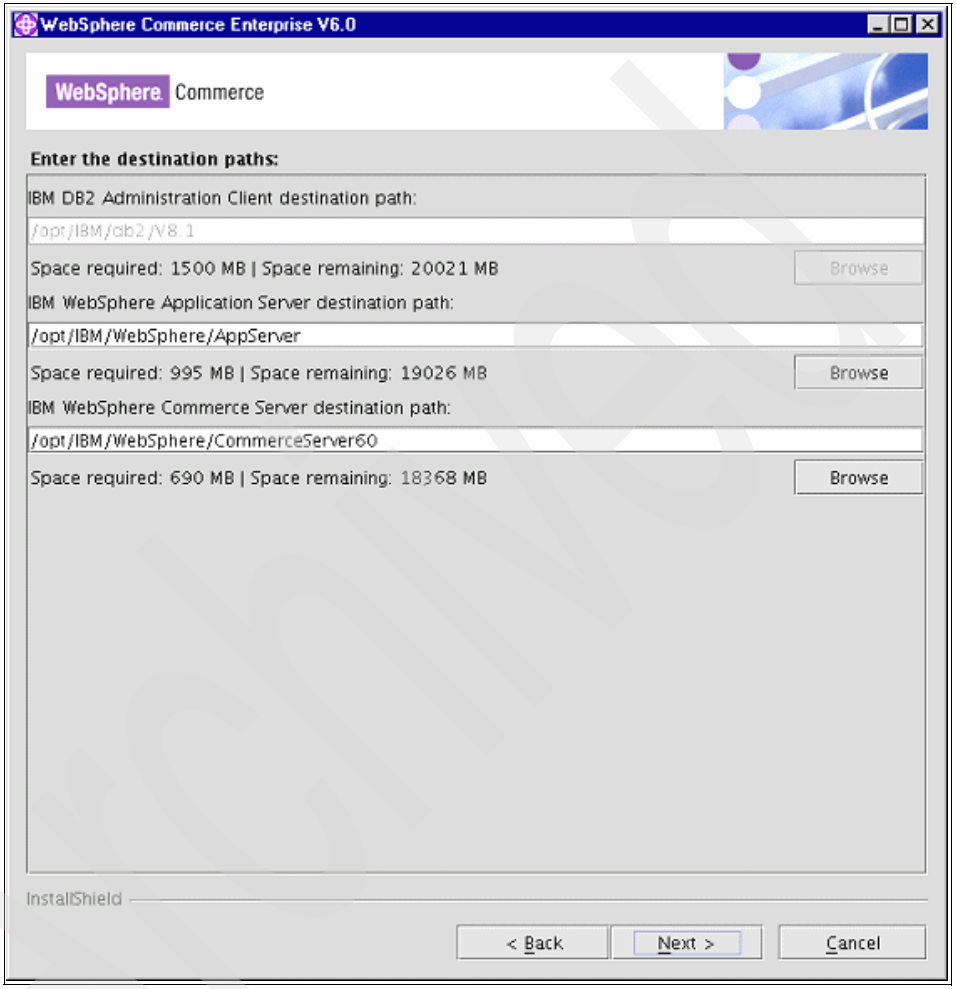


Figure 8-16 WebSphere Commerce component installation directory

5. Specify the database user information (Figure 8-17).

   This step creates a DB2 user on the WebSphere Commerce node. You must ensure that this user, group, and directory do not pre-exist on the WebSphere Commerce node. This user is used to connect to the remote database. This user should either be the DB2 instance or schema owner, as defined on the remote database server.

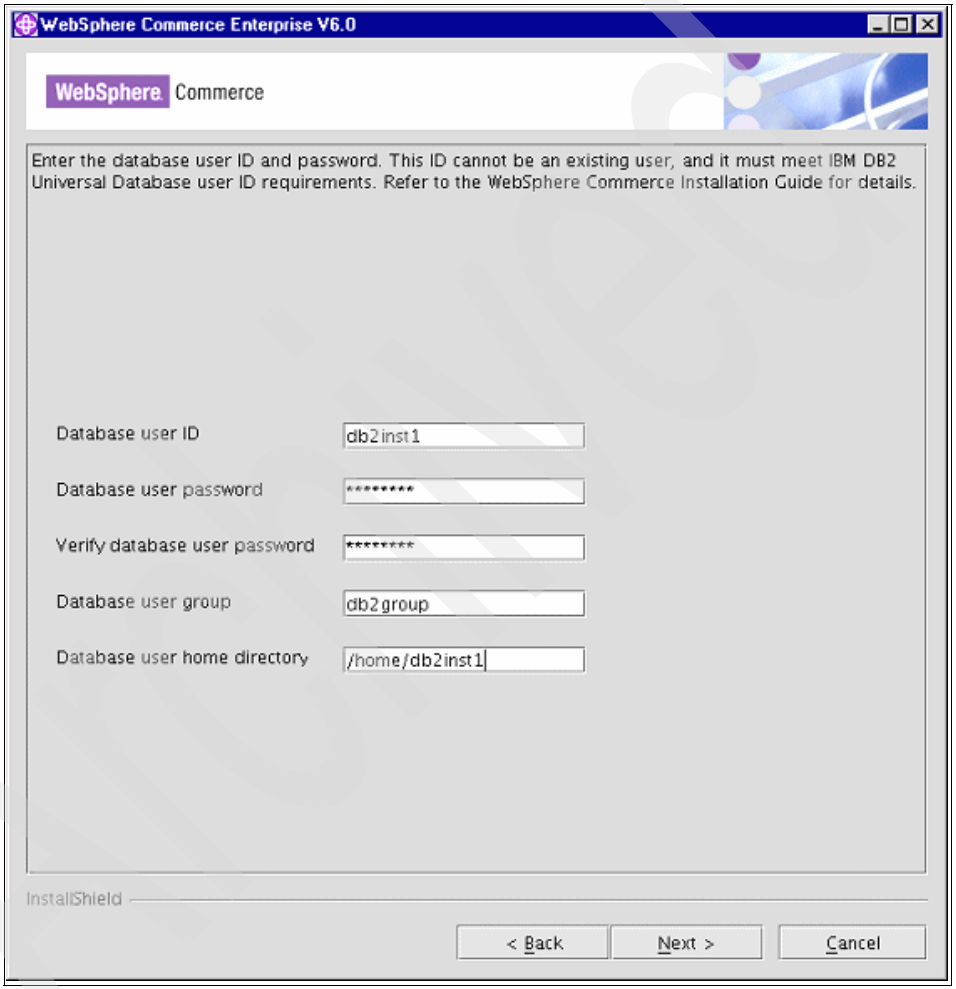


Figure 8-17 Input db2 user information

6. Specify the ConfigManager password (Figure 8-18).

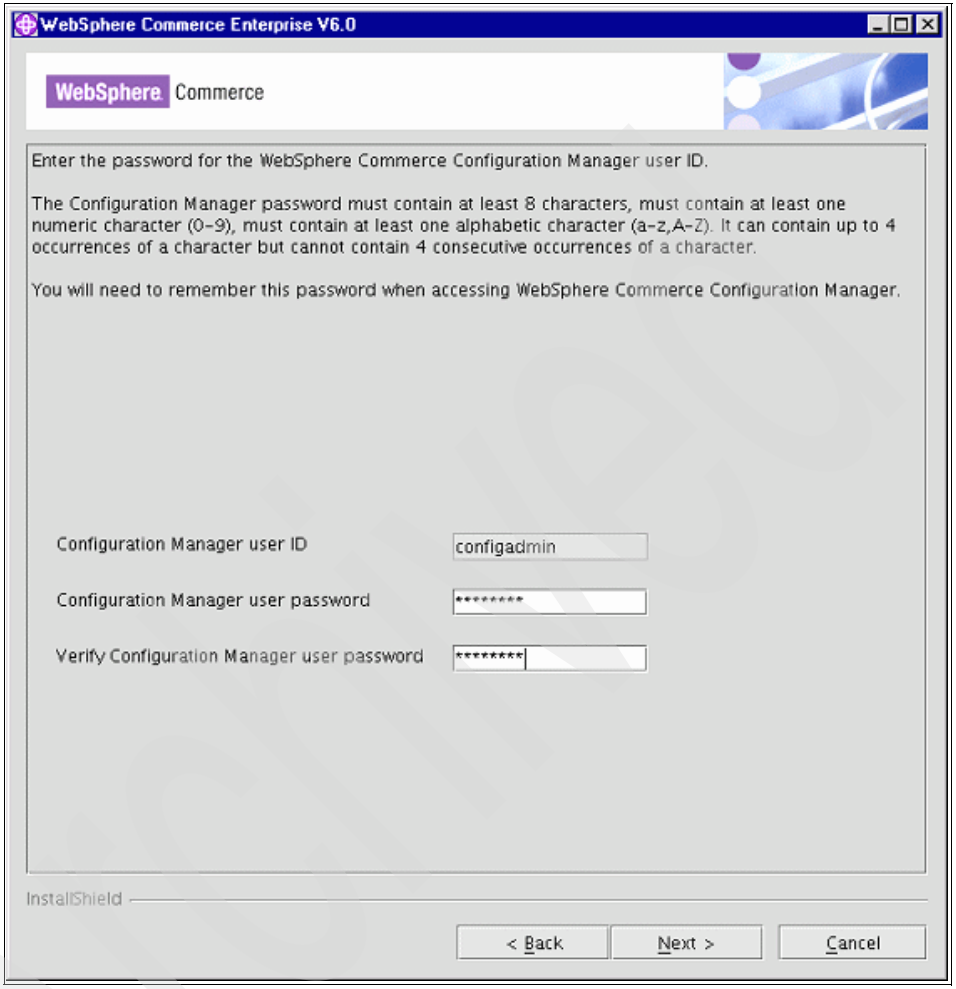


Figure 8-18 Input WebSphere Commerce configuration manager information

7. Specify *non-root_user* information (Figure 8-19). This must be an existing user/group/directory on the WebSphere Commerce node.

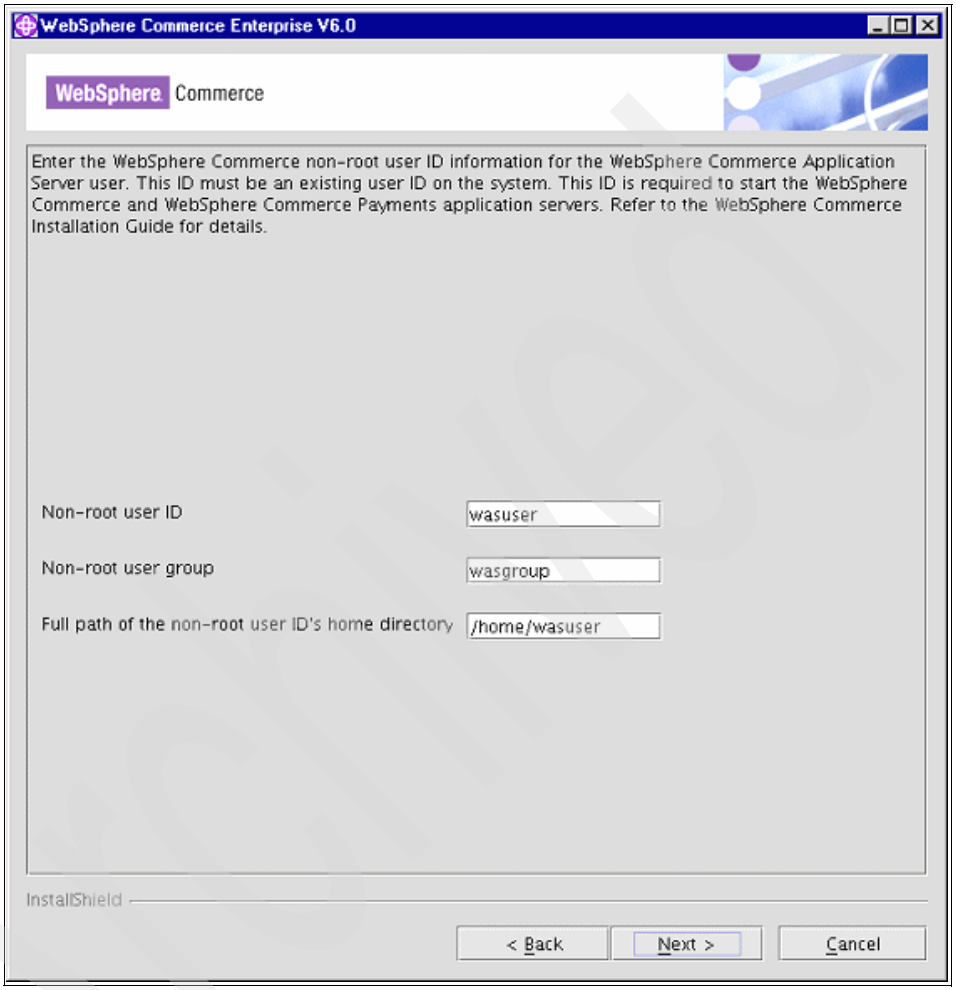


Figure 8-19 Input non-root_user information

8. You may wish to save the configurations in a response file (Figure 8-20) so you can use it to install WebSphere Commerce with the exact same settings on another machine. Click **Next**.

   This may come in handy also if you need to re-install WebSphere Commerce on the same machine.

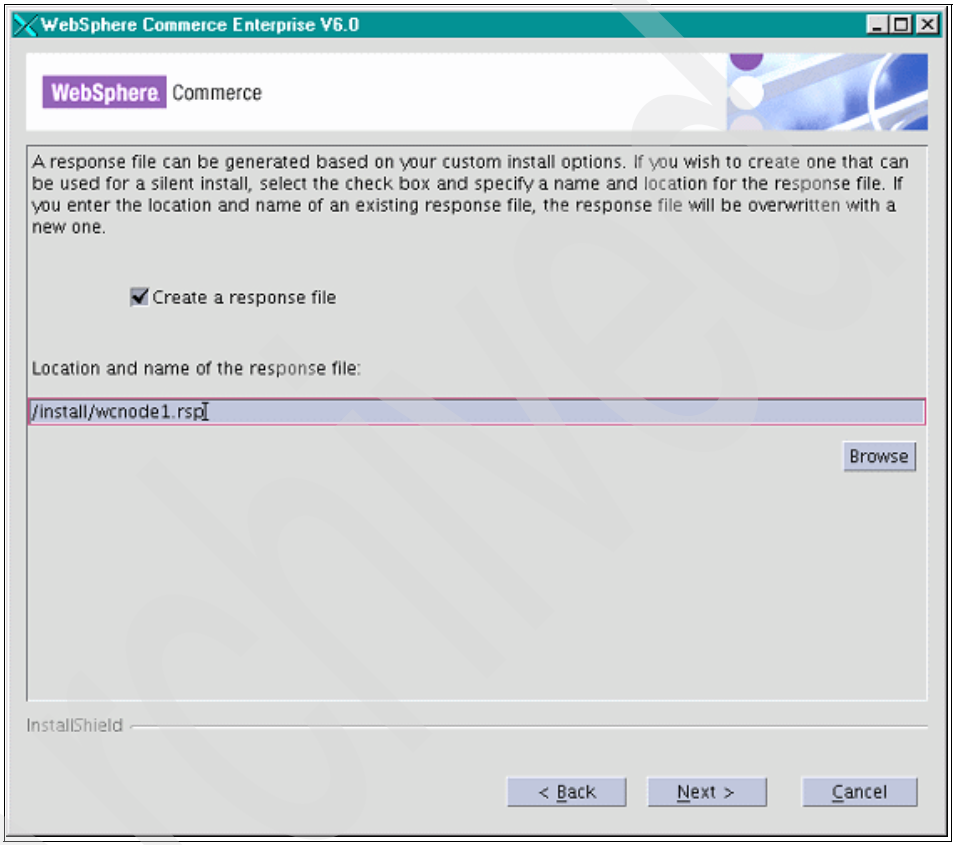


Figure 8-20 Option to create a response file for the WC installation

9. Check your settings with the installation summary. See Figure 8-21.

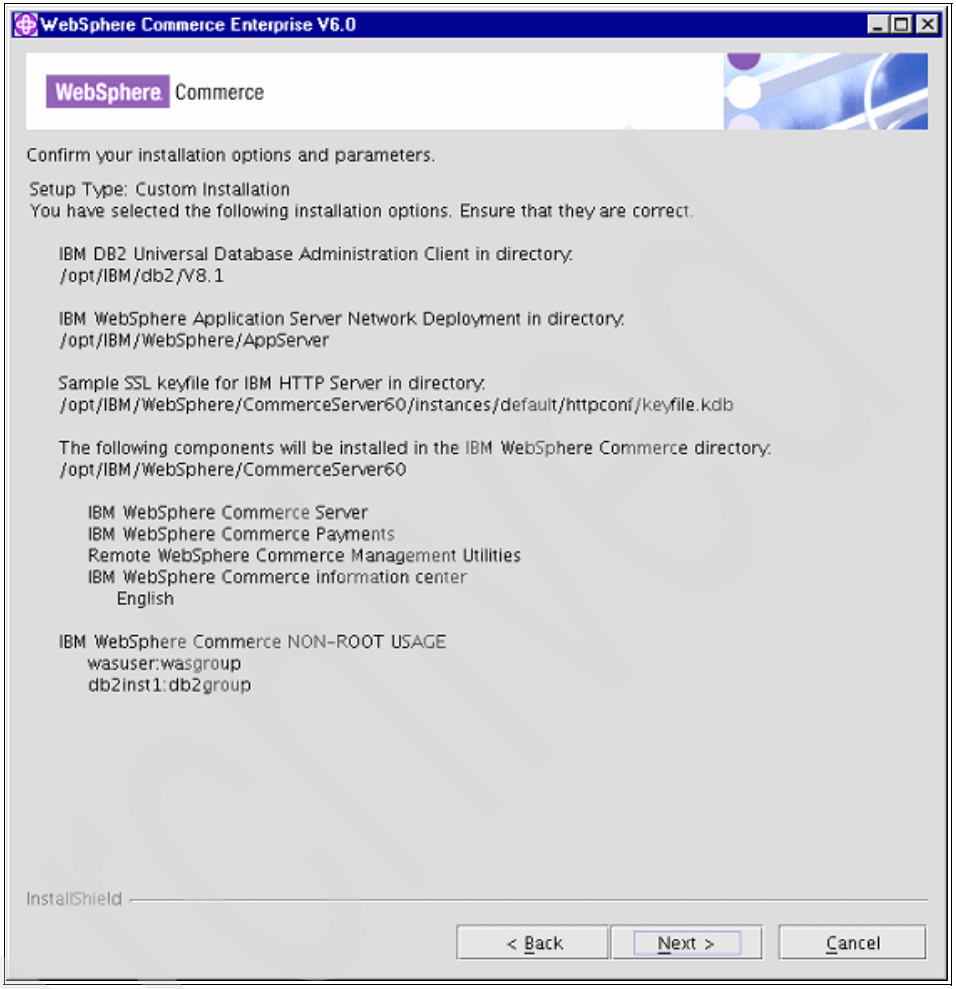


Figure 8-21 WebSphere Commerce installation summary

10. Upon successful installation you should get to the window shown in Figure 8-22.

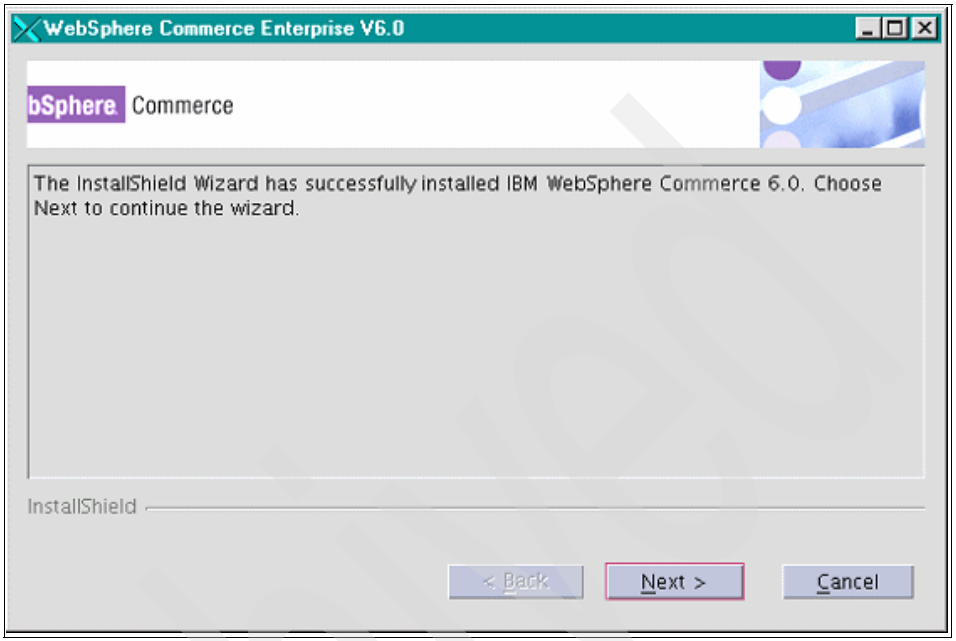


Figure 8-22 WC installation confirmation

11. Once the installation is complete, update the WebSphere Application Server to the latest fix pack.

8.3 Additional WebSphere Commerce nodes

On any additional WebSphere Commerce node, the following must be done before you can federate the additional node and create a new cluster member on it:

- Install the basic WebSphere Commerce product, WebSphere Commerce fix pack, WebSphere Application Server fix pack, and additional APARs. Follow the same steps in “WebSphere Commerce node 1” on page 106.
- You do not need to create a WebSphere Commerce instance on the additional nodes using the WebSphere Commerce configuration manager, but you will need to create a WebSphere Application Server profile.

8.4 Configure a WebSphere Network Deployment Manager

To set up a WebSphere Network Deployment Manager:

1. Install IBM WebSphere Application Server Network Deployment.
2. Create a new Network Deployment Manager profile.

8.4.1 Install IBM WebSphere Application Server Network Deployment

IBM WebSphere Application Server Network Deployment must be installed using the WebSphere Commerce installer. Note that you can also install the WebSphere Commerce product at the same time if the Deployment Manager shares the same physical machine with a WebSphere Commerce node.

Follow the same steps as in 8.2, “WebSphere Commerce node 1” on page 106.

8.4.2 Create the WebSphere Network Deployment Manager Profile

Perform the following steps to create the Deployment Manager profile:

1. Launch the profile creation wizard.

   There are several ways to launch the wizard. From First steps, select **Profile creation wizards** or launch the profile creation wizard directly.

   Detailed instructions are documented at the WebSphere Application Server information center:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tpro_instances.html

   In our example, we launched the profile creation wizard directly. As root, from *WAS_Install_Dir*/bin/ProfileCreator, issue the following command:

   ```
   ./pctAIX.bin
   ```

Archived

The Profile creation wizard launches (Figure 8-23). Click **Next**.

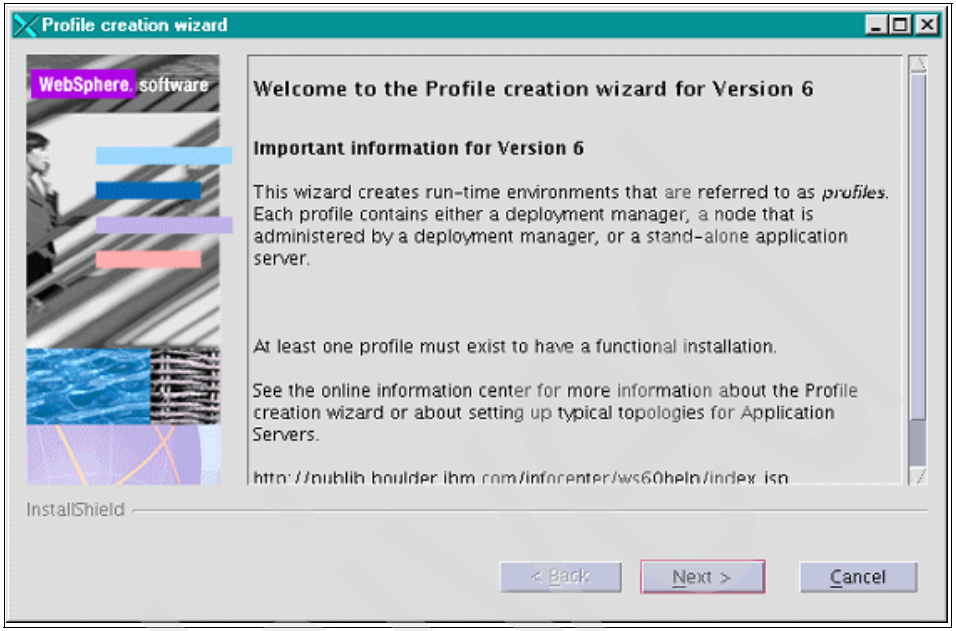


Figure 8-23 WebSphere Application Server profile creation wizard

Archive

2. Select **Create a deployment manager profile**. See Figure 8-24.

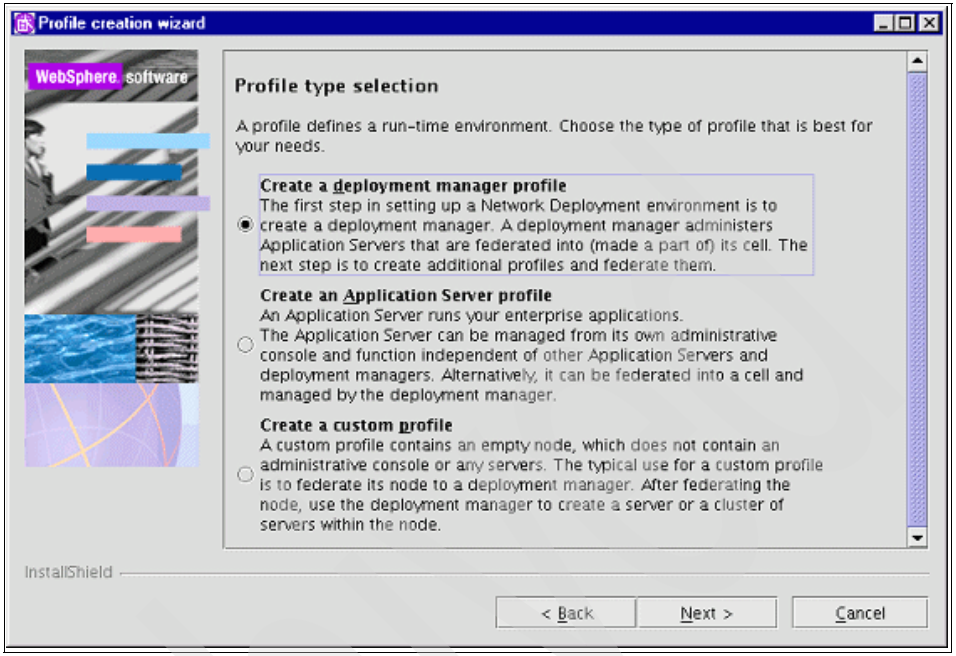


Figure 8-24 Select Create a deployment manager profile

Archive

3. Provide the profile name. See Figure 8-25.

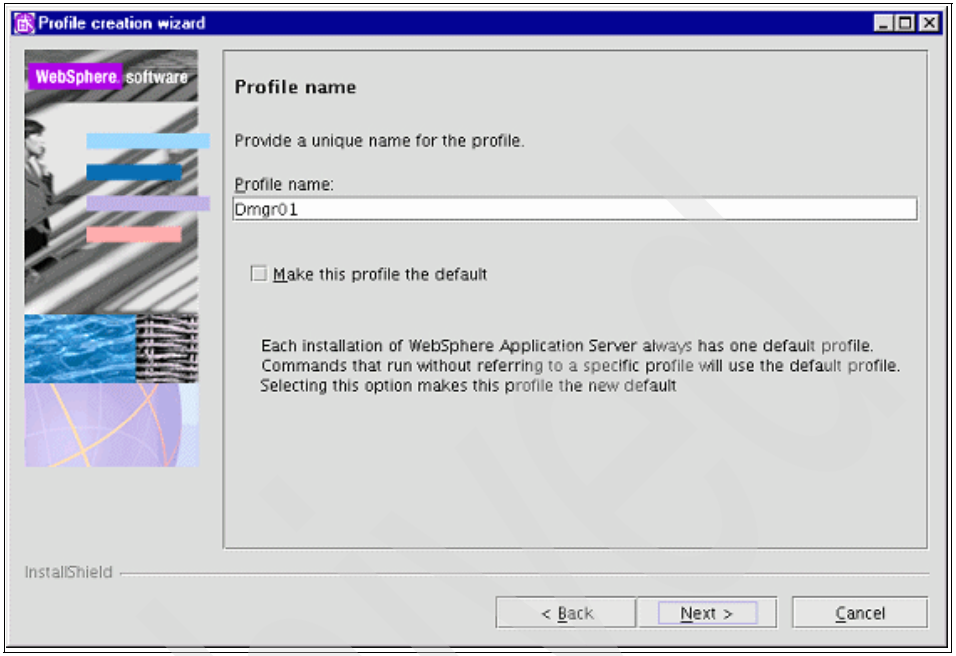


Figure 8-25 Profile creation wizard - enter profile name for deployment manager profile

Archive

4. Provide the profile directory *ND_Profile_Dir*. You may leave the default value, as shown in Figure 8-26.

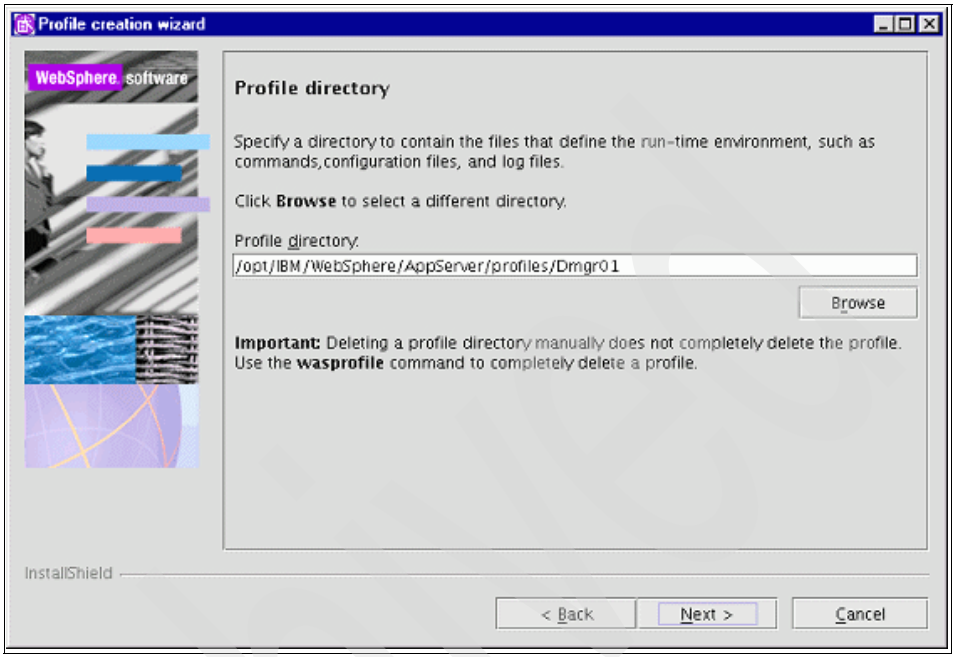


Figure 8-26 Input the directory for the new profile

Archive

5. Provide the node name, the host name, and the cell name. See Figure 8-27.

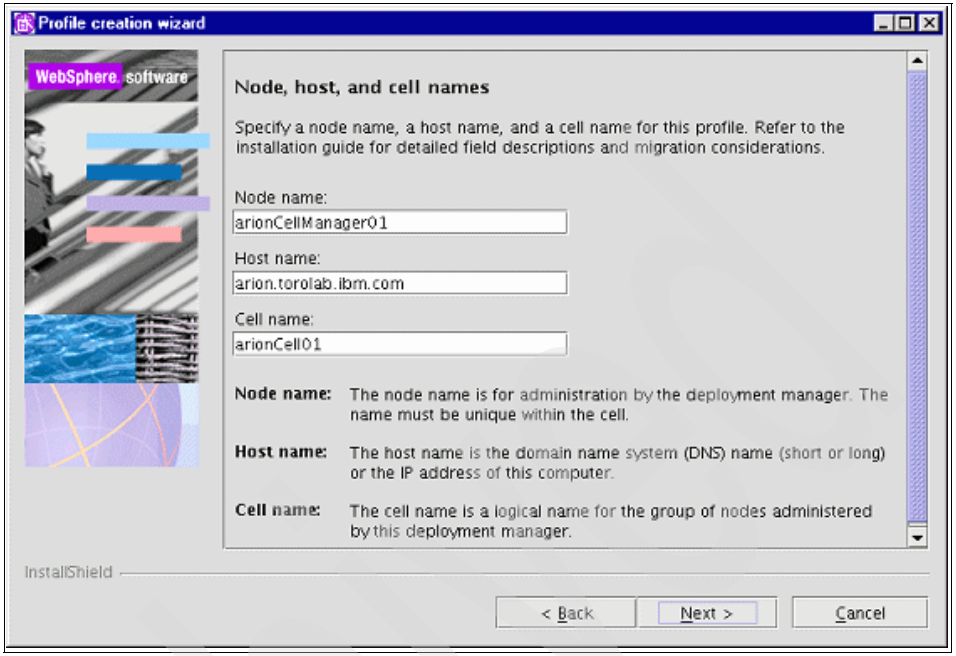


Figure 8-27 Profile creation wizard - node, host, and cell names

Archive

6. Specify the ports to use. Use default ports unless you have a business reason for not doing so. See Figure 8-28.

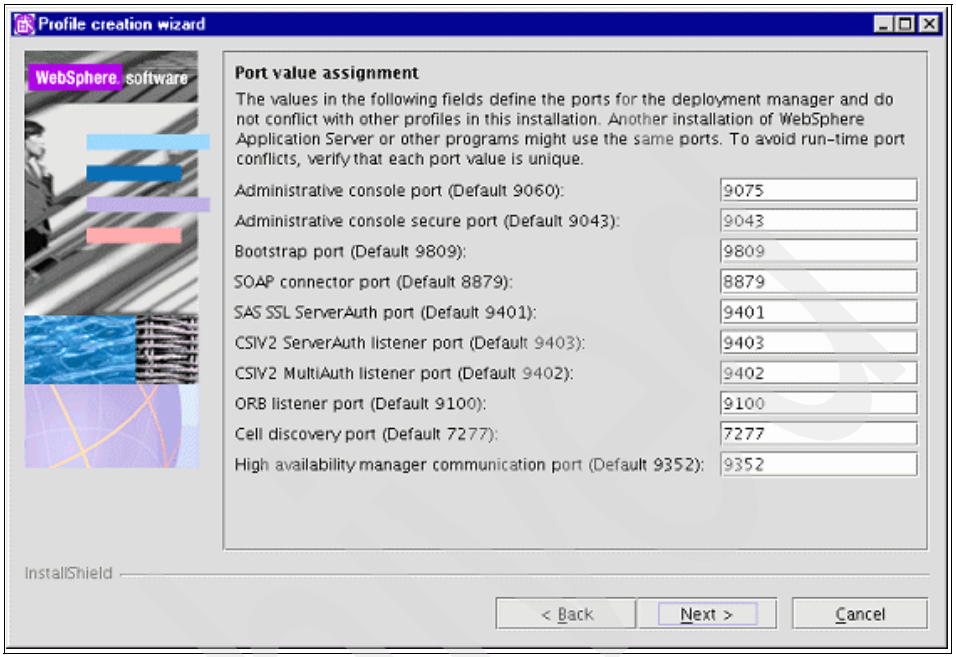


Figure 8-28 Port assignment

Note that the administrative console port here is 9075. By default it is 9060.

To access the administrative console, use:

`http://`*`Deployment_Manager_Hostname`*`:9075/ibm/console`

Archive

7. Verify your profile information on the summary window (Figure 8-29).

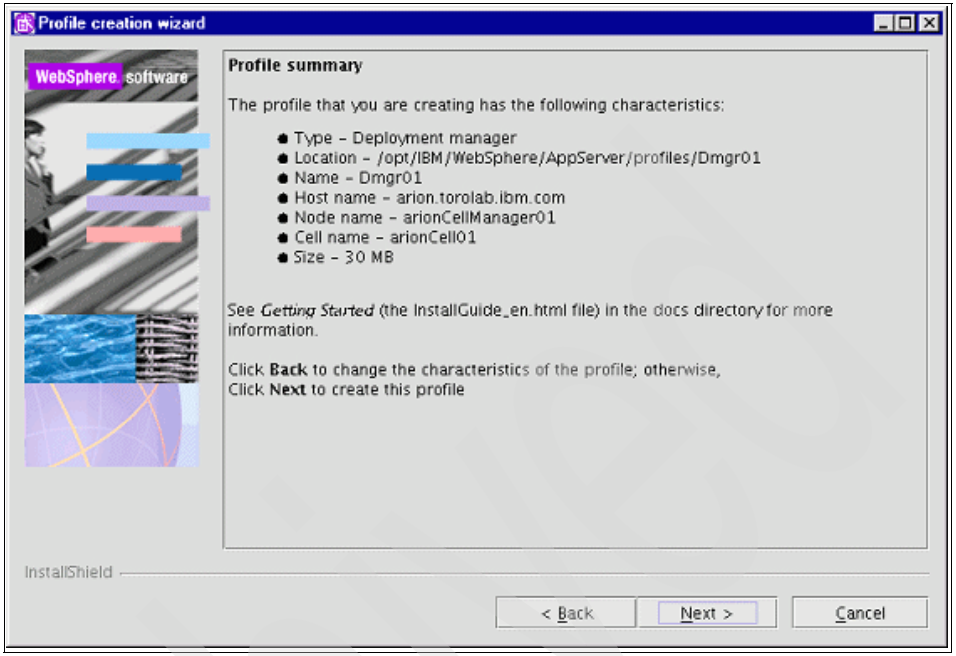


Figure 8-29 Profile creation summary

Archive

8. Create the profile. You should see the following window (Figure 8-30) once the profile creation process completes.

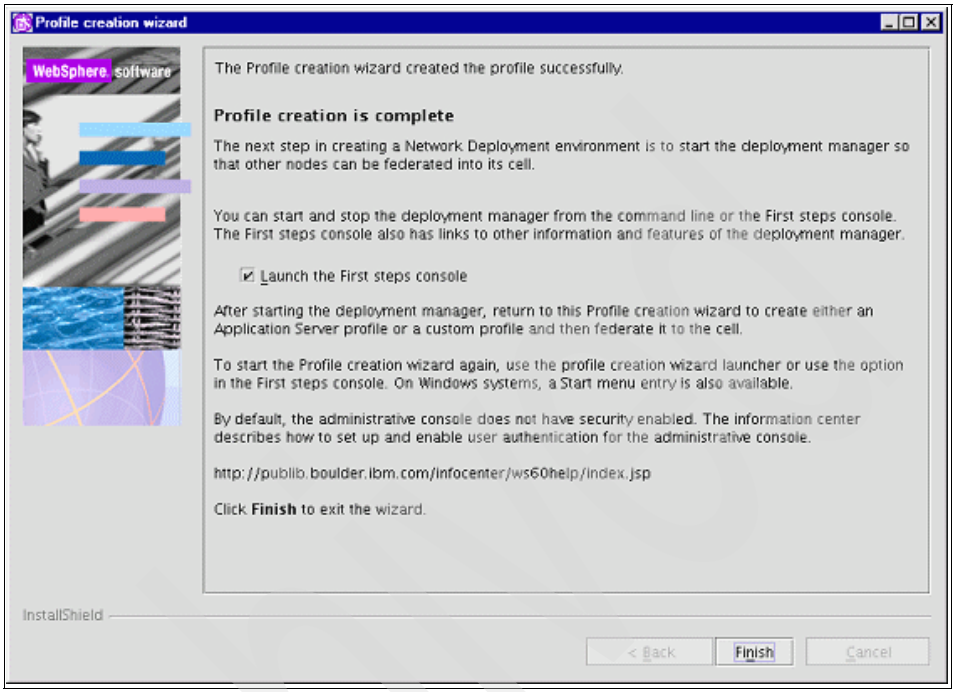


Figure 8-30 Profile creation completes

10. Verify the profile.

Back to the first steps launch pad, you may select to verify. See Figure 8-31.

Figure 8-31 Verify the new profile

11. The URL to access the profile should be:

```
http://<deployment_manager_hostname>:9075/ibm/console
```

8.5 Install IBM HTTP Server

Although other Web server software like IBM HTTP Server 2.0.47.1, Microsoft® Internet Information Services 5.0/6.0 (for Windows), or Sun™ Java™ System Web server 6.1.1 (for AIX, Solaris, Windows) is supported by WebSphere Commerce V6, we recommend using IBM HTTP Server V6.0.x (matching the Version of IBM WebSphere Application Server Network Deployment being used.

Archived

In our scenario the version number is 6.0.2.19.). On iSeries, you need IBM HTTP Server for iSeries.

As we recommend separating the Web server and Application Server tiers, we describe in this chapter how to install only the IBM HTTP Server V6 and the IBM HTTP Server Plug-in for WebSphere Application Server V6.

To ensure High Availability, you should set up multiple Web server nodes as described in 7.1, “Introduction to Web server High Availability” on page 81. The installation procedure is the same for all Web server nodes, as we are using only active Web server nodes, rather than active/passive pairs.

We use AIX as the operating system for the Web servers. All instructions, paths, and commands therefore apply to AIX.

8.5.1 Base installation

For the base installation you need the WebSphere Commerce V6 installation package (which consists of five CDs: WebSphere Commerce disk 1 and 2, IBM WebSphere Application Server Network Deployment disk 1 and 2, and IBM DB2 Database and Administration Client).

Perform the following steps to install the 6.0.0.0 versions of IBM HTTP Server and IBM HTTP Server Plug-in for WebSphere Application Server, using the graphical installer that is included in the WebSphere Commerce V6 package:

1. Make sure that the root user on your Web server nodes has its umask set to 022. Log on as root, and at the prompt, type `umask` to check the umask. If it the result is not 022, change root’s default umask accordingly, for example, by using **`smitty`**.

Archived

2. As root, run the WebSphere Commerce installation wizard on WebSphere Commerce disk 1 from a directory different from any of the CDs or CD images:

   ```
   /tmp/WC60/disk1/setup.sh
   ```

   A language dialog is displayed, as shown in Figure 8-32.

Figure 8-32 Language selection

We choose **English** and click **OK**. The WebSphere Commerce Launchpad is displayed, as shown in Figure 8-33.

Figure 8-33 Launchpad

Archived

3. Click **Install Product**. Another language dialog is shown for the installation, where we select **English** and click **OK**. This brings up the WebSphere Commerce installation wizard.
4. On the welcome panel, click **Next**.
5. In the license agreement window, choose **I accept both the IBM and non-IBM terms** and click **Next**. The wizard checks the system prerequisites and asks for the setup type.
6. In the setup type window, choose **Custom** and click **Next**.

Archived

7. In the component selection window, select **IBM HTTP Server** under **Supporting IBM software**. This automatically also selects **WebSphere Application Server Web server plug-ins**. See Figure 8-34.

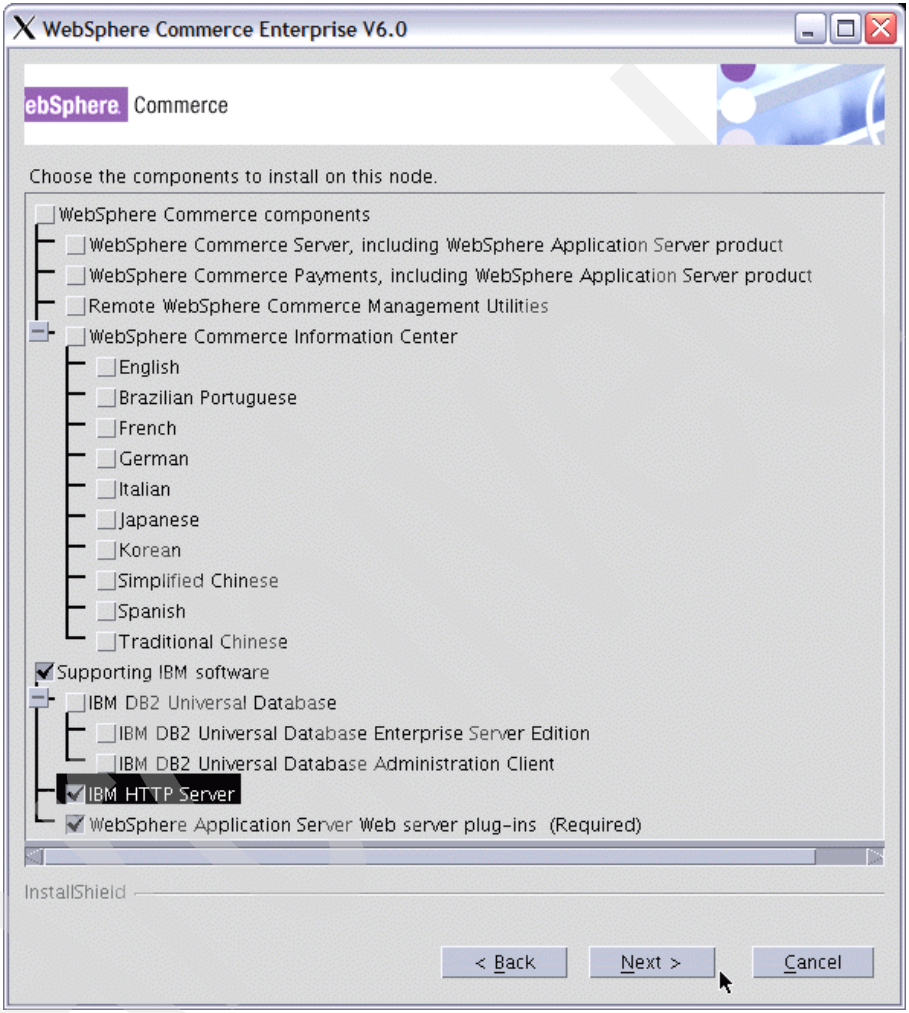


Figure 8-34 Component selection window

8. Click **Next**. The destination paths window is displayed.

9. You may choose to accept the default installation paths referred to as *IHS_Install_Dir* and *Plugin_Install_Dir* in the /usr/IBMIHS for IBM HTTP Server or /usr/IBM/WebSphere/Plugins for IBM HTTP Server Plug-in, or choose your own paths. We kept the defaults, as shown in Figure 8-35.

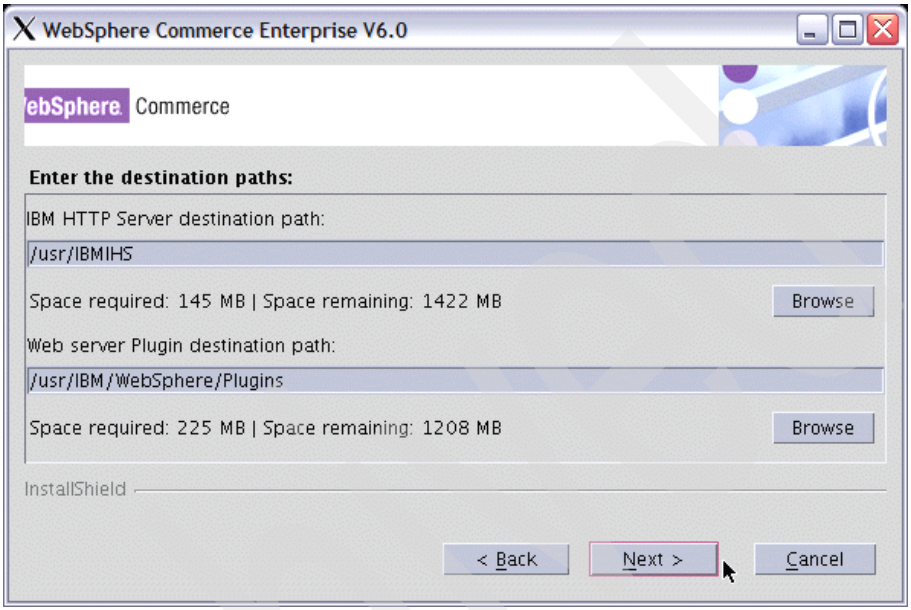


Figure 8-35 Destination paths window

Archive

10. Click **Next**. The installation is performed. It should end with a success message, as shown in Figure 8-36.

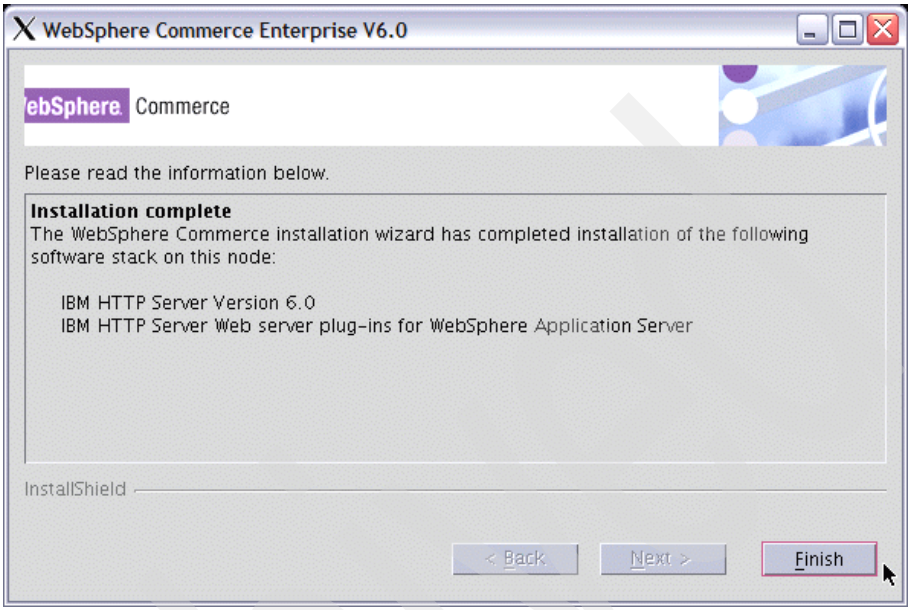


Figure 8-36 Installation complete message

11. Click **Finish**.

8.5.2 Install fixes

You should upgrade all products to the latest fix levels. We describe how to upgrade both IBM HTTP Server and IBM HTTP Server Plug-in to version 6.0.2.19. In order to do this, you need the following packages:

- Refresh pack 6.0.2 for IBM HTTP Server.
- Refresh pack 6.0.2 for IBM HTTP Server Plug-in.
- Fix pack 6.0.2.19 for IBM HTTP Server.
- Fix pack 6.0.2.19 for IBM HTTP Server Plug-in.
- Fix pack 6.0.2.19 for Java SDK (to be applied to both IBM HTTP Server and IBM HTTP Server Plug-in).

Attention: The requirement to use a single Update Installer for V6.0.2 release and V6.1 releases was introduced with Fix Pack 21 (6.0.2.21). Starting with Fix Pack 6.0.2.21, the Update Installer is no longer packaged with the Fix Pack itself. You must use Updated Installer V6.1.0.9 (or later) to install Fix Pack 21(6.0.2.21) and later releases. The Update Installer can be downloaded from:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg24012718

The latest refresh pack and fix pack are available from:

http://www-1.ibm.com/support/docview.wss?uid=swg27004980#ver60

We need to apply the refresh pack first and then the fix pack to all applicable components. While we show how to use the graphical update installer, you may also use the console or silent versions according to the installation instructions included in each package, for example, in case you do not have an X-Windows environment.

Again, the installation is the same on all Web server nodes:

1. Log on as root.
2. Make sure that IBM HTTP Server is not running.

   The following steps need to be performed for all the packages listed above, in the same order as above. As the Java SDK fix needs to be applied to both IBM HTTP Server and IBM HTTP Server Plug-in, the sequence of steps needs to be gone through six times altogether. The examples and graphics are for refresh pack 6.0.2 for IBM HTTP Server.
3. Change to the product installation directory (AIX: *IHS_Install_Dir* for IBM HTTP Server, *Plugin_Install_Dir* for IBM HTTP Server Plug-in) and unpack the package into that directory. For example, for IBM HTTP Server:

   ```
   cd IHS_Install_Dir
   tar -xf /tmp/6.0-WS-WASIHS-AixPPC32-RP0000002.tar
   ```
4. Start the update installer GUI from the updateinstaller directory under the product installation directory:

   ```
   # cd updateinstaller
   # ./update
   ```

   Note: Run **`update -console`** to run the update installer in console mode. You will see the same screens and be given the same options as in the GUI.

Note: For a silent install, perform the following steps:

1. Edit the response file (updateinstaller/responsefiles/install.txt). Follow the instructions in the file. You can find the package to be installed in the updateinstaller/maintenance directory. In the response file, this package needs to be specified using -W maintenance.package="..." using an absolute path and file name in double quotation marks, for example:

   ```
   -W
   maintenance.package="IHS_Install_Dir/updateinstaller/maintenanc
   e /6.0-WS-WASIHS-AixPPC32-RP0000002.pak"
   ```

2. Run:

   ```
   ./update -options responsefiles/install.txt -silent
   ```

3. When using the GUI installer, the installer window is displayed, as shown in Figure 8-37.

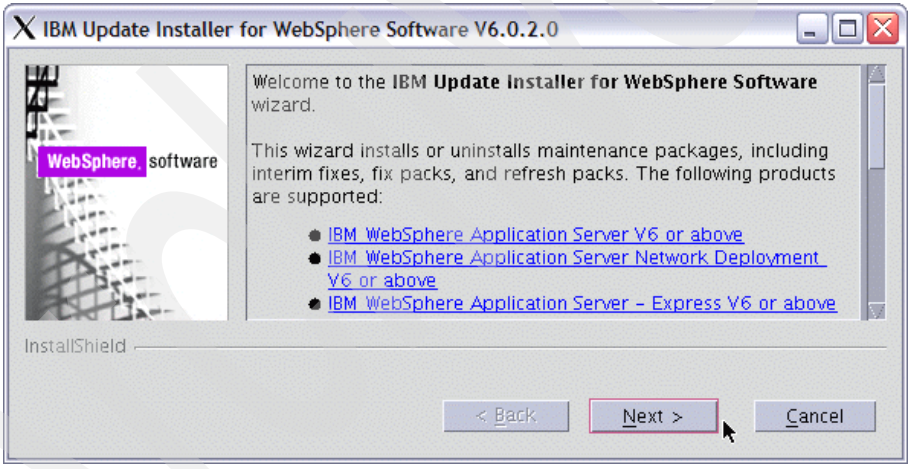


Figure 8-37 Update installer window

4. Click **Next**. On the next panel, the installation location of the product to be fixed can be entered. When started from the updateinstaller directory inside the product directory, the correct directory is already displayed as the default value (for example, *IHS_Install_Dir* for IBM HTTP Server and *Plugin_Install_Dir* for IBM HTTP Server Plug-in), as shown in Figure 8-38.

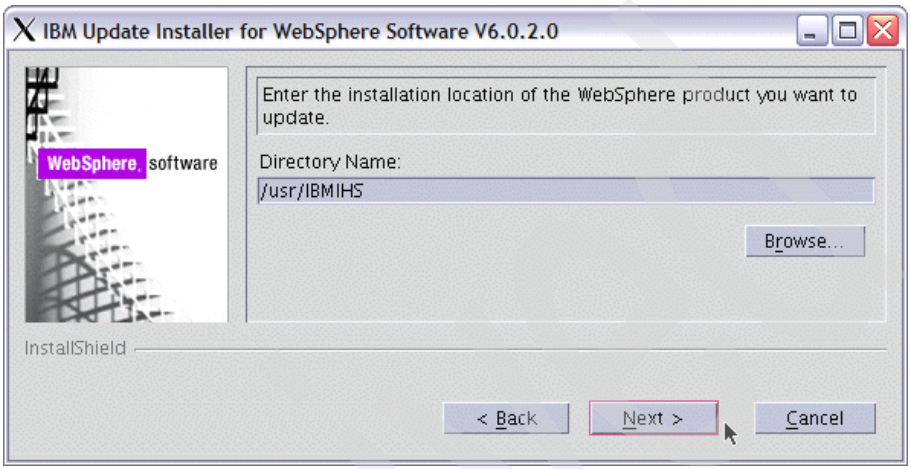


Figure 8-38 Product installation directory

5. Click **Next**. In the next window, you are asked whether you would like to install or uninstall a maintenance package. Choose **Install maintenance package** (this is the default).
6. Click **Next**. In the next window, you need to select the package to install. Although the correct package should be preselected by default, we recommend checking the filename.

Archive

7. Click **Browse** and select the correct package file (ending in .pak) from the maintenance directory, which is opened by default. The package has the same name as the .tar archive. Figure 8-39 shows the file selection window.

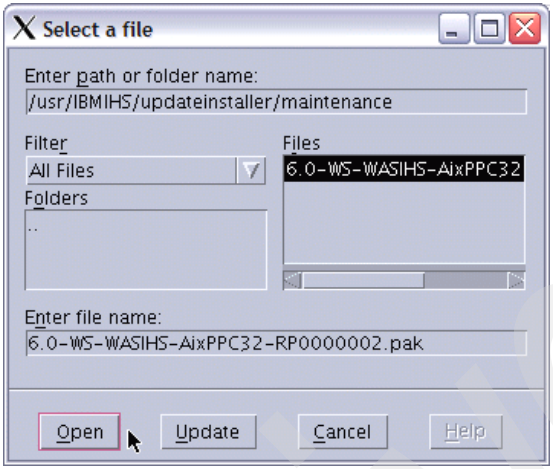


Figure 8-39 Maintenance package file selection window

8. Select the correct package and click **Open**. This will take you back to the installation package window, as shown in Figure 8-40.

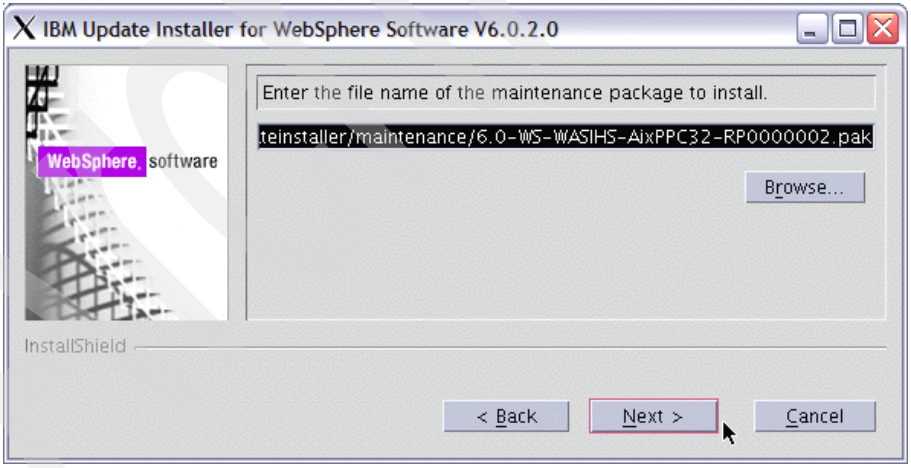


Figure 8-40 Installation package window showing the correct installation package

Archived

9. Click **Next**. The upgrade summary window is displayed. See Figure 8-41.

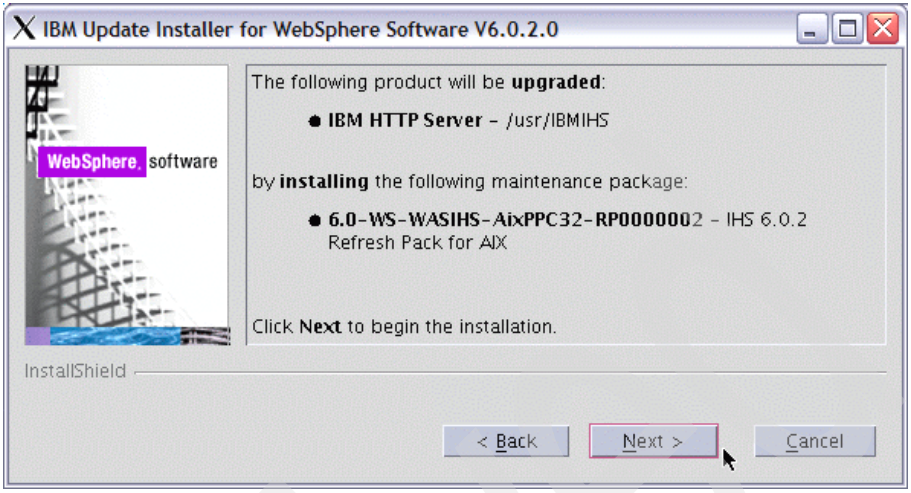


Figure 8-41 Upgrade summary window

10. Click **Next** to begin the installation. Upon successful installation, a success message is displayed, as shown in Figure 8-42.

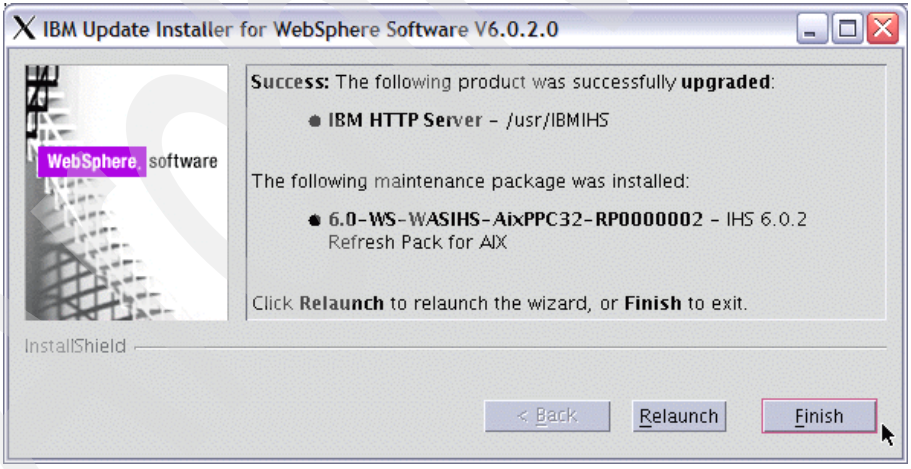


Figure 8-42 Upgrade summary window

11. Repeat steps 3 on page 134 to 10 on page 138 for all packages. Make sure to use the correct installation directories and package names for IBM HTTP Server and IBM HTTP Server Plug-in in step 3 on page 134.

The installation of IBM HTTP Server is now complete. Make sure that you install all your Web server nodes identically.

Note: In some cases, the Java Runtime Environment (JRE™), which is used by update installer itself, needs to be updated as part of the refresh pack or fix pack installation. In this case, update installer displays a message saying that it will copy the JRE and then relaunch itself, as shown in Figure 8-43, to update the IBM HTTP Server Plug-in with refresh pack 6.0.2. After the relaunch you need to select the .pak file again to apply the fix.

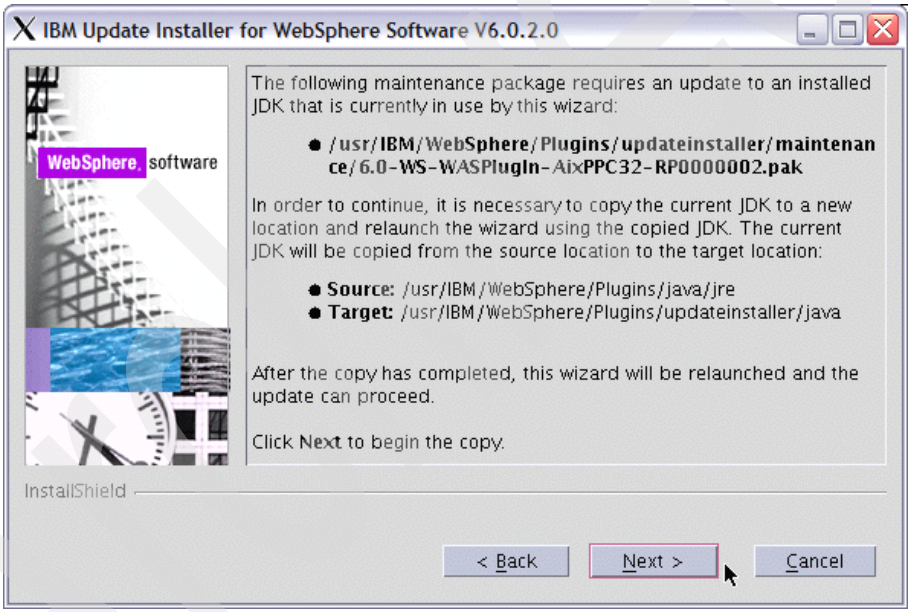


Figure 8-43 Update installer information about updating the currently used JDK

Note: The update installer of the latest fix pack can be used to install interim fixes, too. These come as .pak files, which can be copied into the updateinstaller/maintenance directory in the installation directory of the product to be updated. When running update installer, specify a .pak file, as described in step 7 above, to install an interim fix.

8.6 Install Load Balancer

In this section we describe how to install IBM WebSphere Edge Components Load Balancer V6.0.2 on AIX and Solaris, which are the two operating systems that we used for our tests. For the installation you need both the 6.0 and 6.0.2 versions of Load Balancer.

Load Balancer V6.0 ships as part of the IBM WebSphere Edge Components V6.0. Refer to *Concepts, Planning, and Installation for Edge Components Version 6.0,* GC31-6855, for detailed information about installation, supported platforms, and product requirements.

Load Balancer refresh packs (6.0.x) are available separately from:

http://www-1.ibm.com/support/docview.wss?uid=swg27004980#ver60

From there, follow the links to the current WebSphere Application Server V6.0 refresh pack for your operating system. There you should find a link to *Updates for Edge Components,* GI10-3353, which contains detailed prerequisite information and installation instructions for IBM WebSphere Edge Components updates on all supported operating systems.

The installation procedure is the same for both MAC and NAT forwarding methods (see 7.1, “Introduction to Web server High Availability” on page 81). In both cases, Load Balancer can either be installed on a dedicated machine, or it can be installed *collocated* with a Web server, on a Web server machine. Load Balancer topologies are explained in detail in section 4.5, “Load Balancer topologies,” in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392.

However, we recommend installing Load Balancer on a dedicated machine.

From the IBM WebSphere Edge Components V6.0 package, only the license file needs to be installed. Then Load Balancer 6.0.2 is installed directly.

Important: Before starting the installation, you should have Java Runtime V1.4.2 or later installed on your system.

8.6.1 Install the license

We install the Load Balancer license and other basic files using the graphical installer that is included in the IBM WebSphere Edge Components V6.0.

1. Mount the installation media and start the Edge Components installer by running **`install`**.

   The installer window opens, as shown in Figure 8-44.

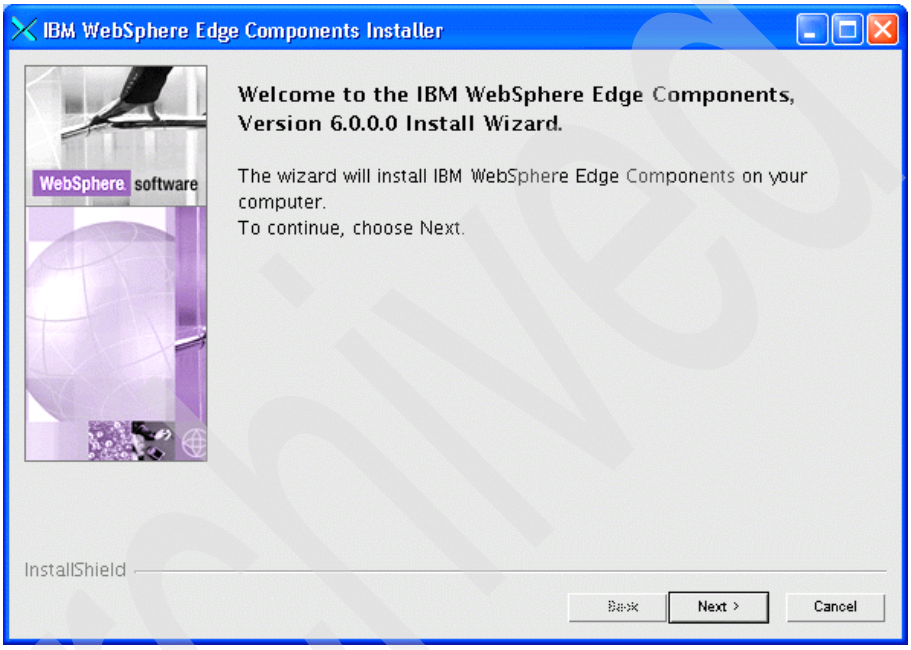


Figure 8-44 Installer window

2. Click **Next** and select **I accept the terms in this license agreement** in the Software License Agreement window.
3. Click **Next**. The installer now checks your system. This takes a few minutes.

Archived

4. Click **Next** when the installer is done with checking. On the language support window, do not select anything except **English**, as shown in Figure 8-45.

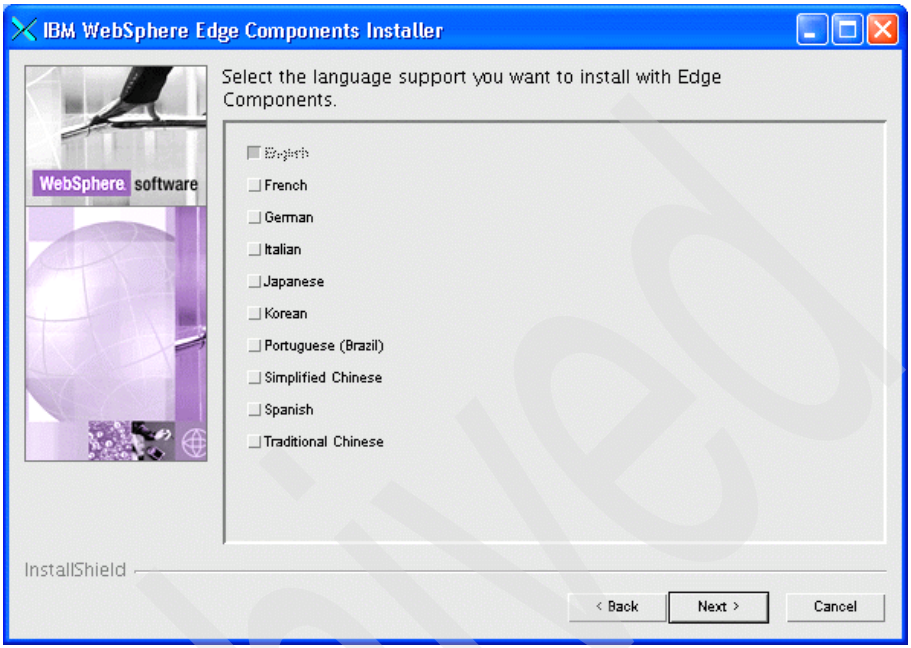


Figure 8-45 Language selection window

5. Click **Next**. In the setup type window, choose **Custom**.

Archived

6. Click **Next**. In the features selection window, only select **License**, as shown in Figure 8-46.

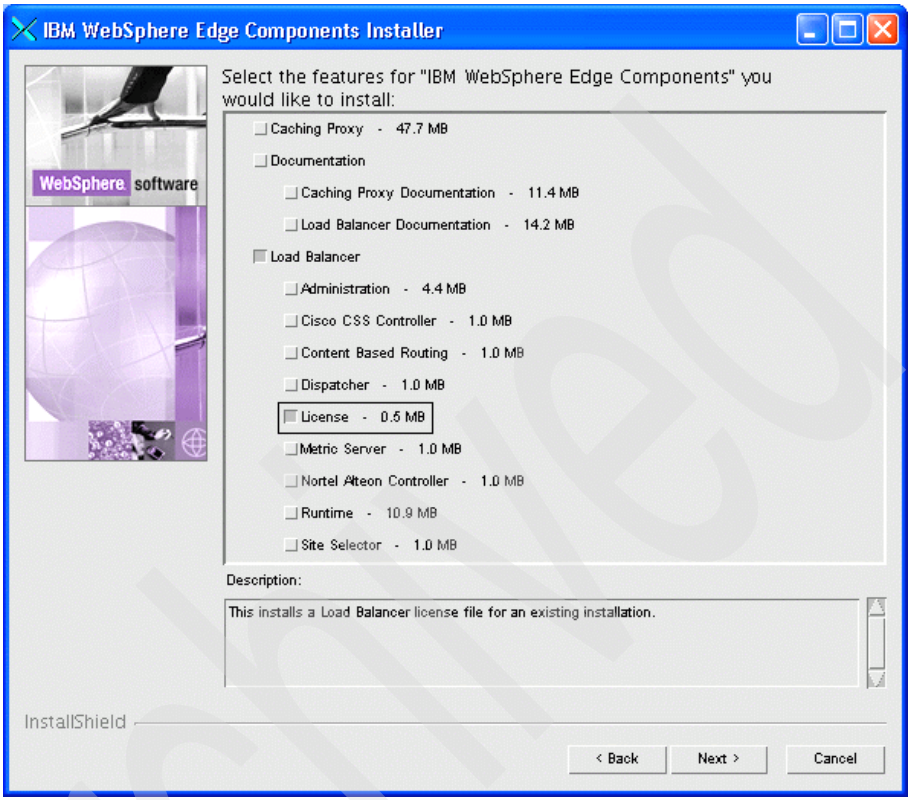


Figure 8-46 Features selection window

7. Click **Next**. Verify the installation options in the summary. Only Load Balancer License should be listed as a feature to install.
8. Click **Next**. The installer installs the Load Balancer license in /opt/ibm/edge/lb on UNIX systems. The installation path cannot be changed.

8.6.2 Install Load Balancer refresh pack

Refresh packs for Load Balancer are downloaded as .tar archives. There is no graphical installer for the refresh packs. Components must be installed using an operating system specific command. The following steps describe the installation for refresh pack 6.0.2 on AIX and Solaris:

1. Switch to root and extract the archive to a temporary directory.

Archived

2. From the temporary directory, issue the following commands in order to install Load Balancer and its Dispatcher component:

   – On AIX

   ```
   inutoc .
   installp -acXd . ibmlb.base.rte
   installp -acXd . ibmlb.admin.rte
   installp -acXd . ibmlb.lb.driver
   installp -acXd . ibmlb.disp.rte
   installp -acXd . ibmlb.doc.rte
   installp -acXd . ibmlb.msg.en_US.*
   ```

   – On Solaris

   ```
   pkgadd -d . ibmlbbase
   pkgadd -d . ibmlbadm
   pkgadd -d . ibmlbdisp
   pkgadd -d . ibmlbdoc
   ```

   Each command should output a success message, as shown in Example 8-7, for the installation of the ibmlb.admin.rte module on AIX.

Example 8-7 Installation success message on AIX

```
+-----------------------------------------------------------------------------+
                            Summaries:
+-----------------------------------------------------------------------------+

Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
ibmlb.admin.rte             6.0.2.0         USR         APPLY       SUCCESS
```

Example 8-8 shows the message on Solaris for the ibmlbbase module.

Example 8-8 Installation success message on Solaris

```
[ verifying class <ibmlbbase> ]

Installation of <ibmlbbase> was successful.
```

The installation of Load Balancer is now complete.

Archived

9

High Availability solution for IBM DB2 Universal Database

In Chapter 5, “Database tier High Availability” on page 39, we introduced the High Availability solution for the DB2 tier, which includes DB2 HADR, SQL Replication, and HACMP. After we understand the theory of these solutions, we give general instructions for setting up your DB2 HA clustering environment by using the DB2 HADR feature.

© Copyright IBM Corp. 2008. All rights reserved.

9.1 HADR

The first solution for database High Availability that we tested is High Availability Disaster Recovery (HADR). Refer to the scenario diagram E:\Stage\7512ch003.fm. This is a DB2-specific solution to achieve the High Availability in the databaset tier in Commerce topology.

9.1.1 Configuring HADR on a primary/standby database

You can configure HADR using the command-line processor (CLP), the Set Up High Availability disaster recovery (HADR) wizard in Control Center, or by the corresponding application programming interfaces (APIs). Use the following procedure to configure primary and standby databases for High Availability disaster recovery using the CLP:

1. Enable log archiving. Configure other DB2 parameters on the primary database:

```
(P) $ db2 update db configuration for rmall using LOGRETAIN RECOVERY
(P) $ db2 update db configuration for rmall using LOGINDEXBUILD ON
(P) $ db2 update db configuration for rmall using INDEXREC RESTART
(P) $ db2 update db configuration for rmall using NEWLOGPATH
"/db2log/"
```

> **Note:** We highly recommended that you use the NEWLOGPATH configuration parameter to put database logs on a separate device from the database once the database is created. This protects your database from media failure where the logs are stored, and improves the overall performance of database system.

2. Perform the command in Example 9-1 to restore the database in standby.

Example 9-1 Restore database in standby

```
(S) $ db2 restore db rmall replace history file
```

3. Configure HADR and client reroute on the primary database. As primary instance owner, set HADR-related parameters for the primary database, as shown in Example 9-2.

Example 9-2 primary database parameter configuration

```
(P) $ db2 update db configuration for rmall using HADR_LOCAL_HOST
rayden2
(P) $ db2 update db configuration for rmall using HADR_LOCAL_SVC 18819
```

```
(P) $ db2 update db configuration for rmall using HADR_REMOTE_HOST
salmon
(P) $ db2 update db configuration for rmall using HADR_REMOTE_SVC 18820
(P) $ db2 update db configuration for rmall using HADR_REMOTE_INST
db2inst1
(P) $ db2 update db configuration for rmall using HADR_SYNCMODE
NEARSYNC
```

Note: The configuration for this article uses the NEARSYNC synchronization mode. Transaction response time is shorter with NEARSYNC than with the other synchronous modes. However, protection against data loss is greater with other synchronization modes.

4. Configure HADR and client reroute on the standby database. As the standby instance owner, set HADR-related parameters for the standby database as shown in Example 9-3.

Example 9-3 standby database parameter configuration

```
(S) $ db2 update db configuration for rmall using HADR_LOCAL_HOST
salmonsalmon
(S) $ db2 update db configuration for rmall using HADR_LOCAL_SVC 18820
(S) $ db2 update db configuration for rmall using HADR_REMOTE_HOST
rayden2
(S) $ db2 update db configuration for rmall using HADR_REMOTE_SVC 18819
(S) $ db2 update db configuration for rmall using HADR_REMOTE_INST
db2inst1
(S) $ db2 update db configuration for rmall using HADR_SYNCMODE
NEARSYNC
```

5. Start HADR. You can use the commands shown in Example 9-4

Example 9-4 Start HADR

```
(S) $ db2 start hadr on db rmall as standby
(P) $ db2 start hadr on db rmall as primary
```

Note: We recommend starting HADR on the standby server before you start HADR on a primary database server. When you start HADR on a primary database server, the database server waits up to HADR_TIMEOUT seconds for a standby database server to connect to it. If there is still no standby database server connected after HADR_TIMEOUT seconds have passed, the HADR start on the primary database server fails.

Archive

6. Verify that HADR has been successfully configured:
   - Review the db2diag.log on both the primary and the standby database server to see whether HADR is configured correctly.
   - Examine the HADR status from a snapshot of both the primary database and the standby database. Figure 9-1 and E:\Stage\7512ch009.fm show the HADR section returned by the GET SNAPSHOT command.

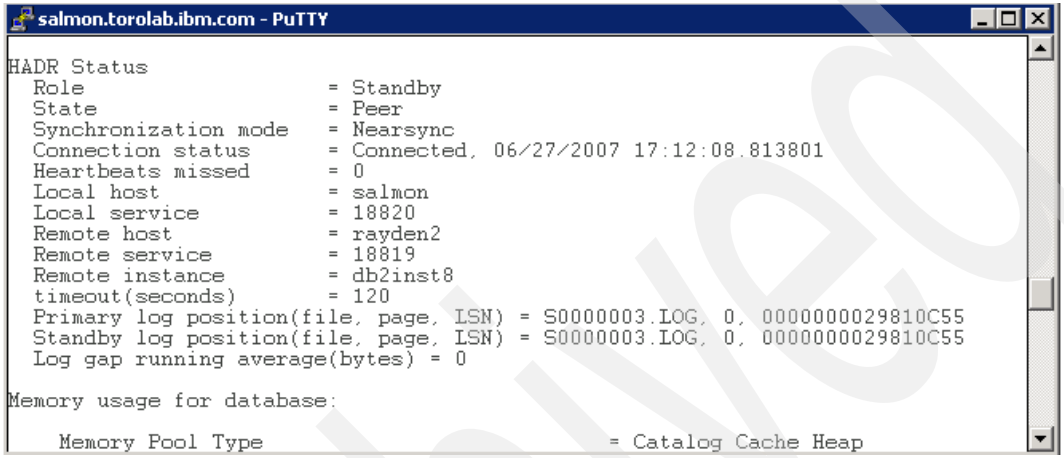


Figure 9-1 HADR standby server status

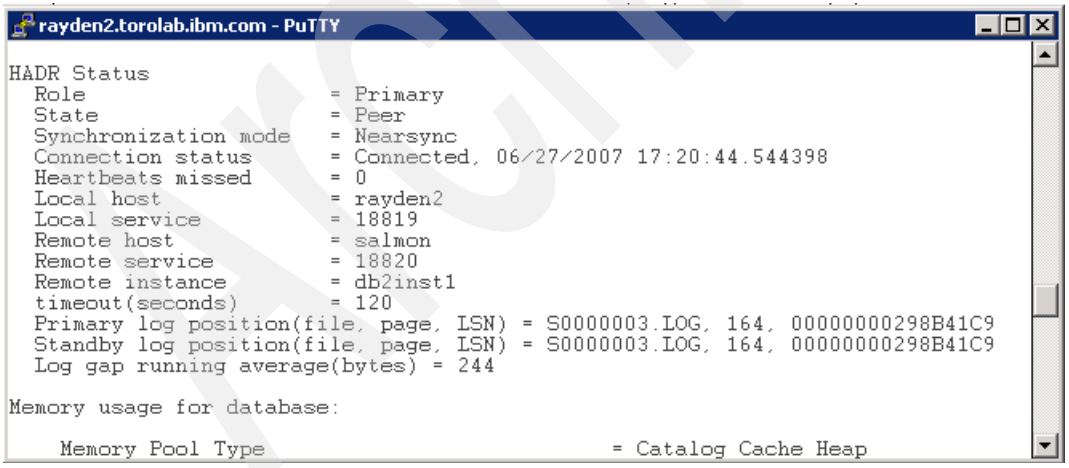


Figure 9-2 HADR primary server status

Archived

– Check the DB2 log to see whether you can see the records shown in Example 9-5 (for the primary database) and Example 9-6 on page 151 (for the standby database).

Example 9-5 DB2 log from the primary database

```
2007-07-03-21.26.46.707671-240 I61036A394        LEVEL: Warning
PID     : 565478               TID  : 1          PROC : db2agent
(RMALL) 0
INSTANCE: db2inst1 NODE : 000         DB   : RMALL
APPHDL  : 0-28                 APPID: *LOCAL.db2inst1.070704012646
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21151
MESSAGE : Info: HADR Startup has begun.

2007-07-03-21.26.46.789144-240 E61431A333        LEVEL: Event
PID     : 618662               TID  : 1          PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000        DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to None (was None)

2007-07-03-21.26.46.825540-240 E61765A335        LEVEL: Event
PID     : 618662               TID  : 1          PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000        DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to P-Boot (was None)

2007-07-03-21.26.46.848887-240 I62101A317        LEVEL: Warning
PID     : 618662               TID  : 1          PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000        DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP,
probe:20301
MESSAGE : Info: Primary Started.

2007-07-03-21.26.47.452229-240 E62419A353        LEVEL: Event
PID     : 618662               TID  : 1          PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000        DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
```

Archived

```
CHANGE  : HADR state set to P-RemoteCatchupPending (was P-Boot)

2007-07-03-21.26.47.462966-240 I62773A398         LEVEL: Warning
PID     : 565478               TID  : 1           PROC : db2agent
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
APPHDL  : 0-28                 APPID: *LOCAL.db2inst1.070704012646
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21152
MESSAGE : Info: HADR Startup has completed.

2007-07-03-21.26.47.664441-240 E63172A362         LEVEL: Event
PID     : 618662               TID  : 1           PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to P-RemoteCatchup (was
P-RemoteCatchupPending)

2007-07-03-21.26.47.671841-240 I63535A336         LEVEL: Warning
PID     : 618662               TID  : 1           PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP,
probe:20445
MESSAGE : remote catchup starts at 000000002A89500C

2007-07-03-21.26.47.769458-240 I63872A353         LEVEL: Warning
PID     : 618662               TID  : 1           PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrTransitionPtoNPeer, probe:10645
MESSAGE : near peer catchup starts at 000000002B751D8B

2007-07-03-21.26.48.379809-240 E64226A352         LEVEL: Event
PID     : 618662               TID  : 1           PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to P-NearlyPeer (was P-RemoteCatchup)

2007-07-03-21.26.48.389134-240 E64579A343         LEVEL: Event
```

Archived

```
PID     : 618662               TID  : 1           PROC : db2hadrp
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to P-Peer (was P-NearlyPeer)
```

Example 9-6 DB2 log from the standby database

```
2007-07-03-21.17.04.600117-240 I568803A394        LEVEL: Warning
PID     : 24210                TID  : 1           PROC : db2agent
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
APPHDL  : 0-120                APPID: *LOCAL.db2inst1.070704011702
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21151
MESSAGE : Info: HADR Startup has begun.

2007-07-03-21.17.04.615717-240 E569198A333        LEVEL: Event
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to None (was None)

2007-07-03-21.17.04.629921-240 E569532A335        LEVEL: Event
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to S-Boot (was None)

2007-07-03-21.17.04.630221-240 I569868A338        LEVEL: Warning
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrStartReplayMaster, probe:21251
MESSAGE : Info: Replaymaster Starting...

2007-07-03-21.17.04.637393-240 I570207A343        LEVEL: Warning
PID     : 30050                TID  : 1           PROC : db2agnti
(RMALL) 0
```

Archived

```
INSTANCE: db2inst1            NODE : 000         DB   : RMALL
APPHDL  : 0-121
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21151
MESSAGE : Info: HADR Startup has begun.

2007-07-03-21.17.04.715572-240 I570551A330       LEVEL: Warning
PID     : 30050               TID  : 1           PROC : db2agnti
(RMALL) 0
INSTANCE: db2inst1            NODE : 000         DB   : RMALL
APPHDL  : 0-121
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:300
MESSAGE : Starting Replay Master on standby.

2007-07-03-21.17.04.715934-240 I570882A340       LEVEL: Warning
PID     : 15422               TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1            NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrStartReplayMaster, probe:21252
MESSAGE : Info: Replaymaster request done.

2007-07-03-21.17.04.716189-240 E571223A345       LEVEL: Event
PID     : 15422               TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1            NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to S-LocalCatchup (was S-Boot)

2007-07-03-21.17.04.716451-240 I571569A317       LEVEL: Warning
PID     : 15422               TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1            NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS,
probe:20341
MESSAGE : Info: Standby Started.

2007-07-03-21.17.04.719812-240 E571887A346       LEVEL: Warning
PID     : 30050               TID  : 1           PROC : db2agnti
(RMALL) 0
INSTANCE: db2inst1            NODE : 000         DB   : RMALL
APPHDL  : 0-121
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:920
MESSAGE : ADM1602W  Rollforward recovery has been initiated.
```

Archived

```
2007-07-03-21.17.04.721055-240 E572234A389        LEVEL: Warning
PID     : 30050                TID  : 1           PROC : db2agnti
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
APPHDL  : 0-121
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:1740
MESSAGE : ADM1603I  DB2 is invoking the forward phase of the database
          rollforward recovery.

2007-07-03-21.17.04.721383-240 I572624A420        LEVEL: Warning
PID     : 30050                TID  : 1           PROC : db2agnti
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
APPHDL  : 0-121
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:720
DATA #1 : String, 103 bytes
Invoking database rollforward forward recovery,
lowtranlsn 000000002A89529F minbufflsn 000000002A0F44C0

2007-07-03-21.17.04.725989-240 I573045A358        LEVEL: Warning
PID     : 30050                TID  : 1           PROC : db2agnti
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
APPHDL  : 0-121
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
MESSAGE : Using parallel recovery with 3 agents 9 QSets 27 queues and
16 chunks

2007-07-03-21.17.04.716208-240 I573404A398        LEVEL: Warning
PID     : 24210                TID  : 1           PROC : db2agent
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
APPHDL  : 0-120                APPID: *LOCAL.db2inst1.070704011702
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup,
probe:21152
MESSAGE : Info: HADR Startup has completed.

2007-07-03-21.17.05.775420-240 I573803A320        LEVEL: Warning
PID     : 41284                TID  : 1           PROC : db2shred
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
APPHDL  : 0-121
FUNCTION: DB2 UDB, recovery manager, sqlpshrEdu, probe:18300
MESSAGE : Maxing hdrLCUEndLsnRequested
```

Archived

```
2007-07-03-21.17.05.819612-240 E574124A361        LEVEL: Event
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to S-RemoteCatchupPending (was S-LocalCatchup)

2007-07-03-21.17.22.845448-240 E574486A369        LEVEL: Event
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to S-RemoteCatchupPending (was
S-RemoteCatchupPending)

2007-07-03-21.17.22.872810-240 E574856A362        LEVEL: Event
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to S-RemoteCatchup (was
S-RemoteCatchupPending)

2007-07-03-21.17.22.873088-240 I575219A335        LEVEL: Warning
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSPrepareLogWrite, probe:10260
MESSAGE : RCUStartLsn 000000002A89539A

2007-07-03-21.17.23.712898-240 E575555A352        LEVEL: Event
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to S-NearlyPeer (was S-RemoteCatchup)

2007-07-03-21.17.23.782792-240 E575908A343        LEVEL: Event
```

Archived

```
PID     : 15422                TID  : 1           PROC : db2hadrs
(RMALL) 0
INSTANCE: db2inst1             NODE : 000         DB   : RMALL
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE  : HADR state set to S-Peer (was S-NearlyPeer)
```

9.1.2 Enabling client reroute in a HADR environment

For a client application to be transparently redirected to an alternate standby database server when there is a loss of communication with the primary database server, specify that alternate server's location on the primary database server. To do this, use the UPDATE ALTERNATE SERVER FOR DATABASE command in the primary database server. Example 9-7 is an example of the steps to specify an alternate database server.

Example 9-7 Enable ACR in DB2 HADR environment

```
(P) $ db2 update alternate server for database rmall using hostname
salmon port 50000
(S) $ db2 update alternate server for database rmall using hostname
rayden2 port 50000
```

The alternate database server information is stored on the primary database server, and loaded into the client's cache upon a successful connection to the primary database server. This means that for a client application to know the standby server, it must first successfully connect to the primary server.

9.1.3 Installing Tivoli System Automation

The following steps give you a brief introduction to implementing the Tivoli System Automation for Multiplatforms policy-based, self-healing capability running on AIX:

1. After you have purchased Tivoli System Automation v2.1, you can download a tar file for the AIX operating system. The name of the archive for AIX platforms is C85W5ML.tar.
2. Download the installation archive into your local directory, and use the tar xvf command to extract the archive. When you have extracted the files, you find the installation wizard in the SAM2100Base/installSAM directory.
3. TSA is contained in several packages that must be installed on every node in the cluster to be automated. TSA for Multiplatforms requires a certain RSCT level to be installed on that system prior to the installation. RSCT is part of

AIX, although not all of the RSCT-related filesets are installed by default within the operating system. For TSA pre-installation checking, a more recent level of RSCT may be required. In this case, for TSA v2.1 installation on AIX v5.2, the required RSCT version is 2.3.7.1. After you install the base filesets, you can download the specific updated filesets from IBM Support Fix Central:

http://www-912.ibm.com/eserver/support/fixes/fixcentral/main/pseries/aix

4. Install the product including the automation adapter with the installSAM script. The installation process is finished automatically.
5. Copy the automatic scripts shipped by DB2 v9 from the DB2 installation package to the local directory in all of the cluster nodes. In this case, they are copied into /software/TSA_Auto_Script.

Note: IBM has created scripts that enable TSA to work seamlessly with the DB2 database. You can download the latest TSA automatic scripts from the DB2 for Linux site.

Archive

6. Change the environment variable. You can use the commands shown in Example 9-8.

Example 9-8 Set environment variables

```
export CT_MANAGEMENT_SCOPE=2
export PATH = $PATH:/usr/sbin/rsct/bin:/usr/opt/IBM/db2_08_01/instance/
:/software/TSA_Auto_Script/
```

7. Confirm that TSA has been installed successfully.

   If TSA has been installed, and if the level of RSCT is correct, then you can start a TSA domain. Use the commands given in Example 9-9 to verify that TSA has been installed successfully and that the level of RSCT is correct. You may get the output shown in Figure 9-3.

Example 9-9 TSA post-installation check

```
lsrpdomain - should show the domain as Online with RSCT level of
2.3.7.1.
lsrpnode - should show all nodes in that domain with RSCT level
2.3.7.1.
lssrc -ls IBM.RecoveryRM - should show an IVN and AVN of 2.1.0.0
```

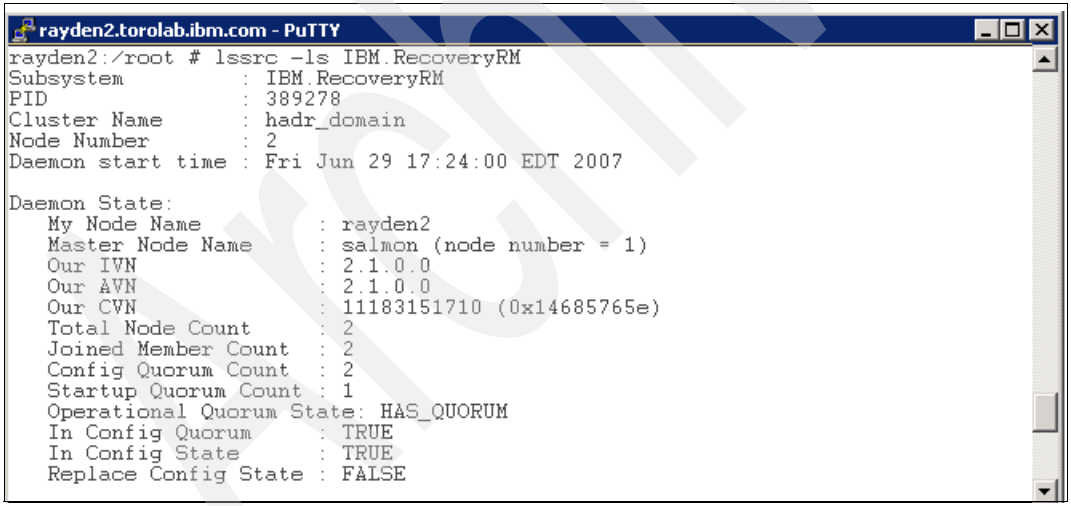


Figure 9-3 TSA post installation check

9.1.4 Defining and administering a TSA cluster

Before configuring your TSA cluster, verify that all of the installations of TSA in your topology know about one another and can communicate with one another in

what is referred to as a TSA cluster domain, then do the following configuration steps:

1. Run the following command as root in each host to prepare the proper security environment between the TSA node so that it is allowed to communicate between the cluster nodes:

   ```
   (P)(S)(H) # preprpnode rayden2 salmon
   ```

2. Issue the following commands to create a RSCT cluster domain:

   ```
   (P) # mkrpdomain HADR_domain rayden2 salmon
   ```

3. Issue the following command to bring the cluster online:

   Note: All future TSA-related commands will run relative to this active domain.

   (P) # startrpdomain HADR_domain

4. In seconds, the cluster starts and you can look up the status of HADR_domain. you can get output, as in Figure 9-4.

   (P) # lsrpdomain

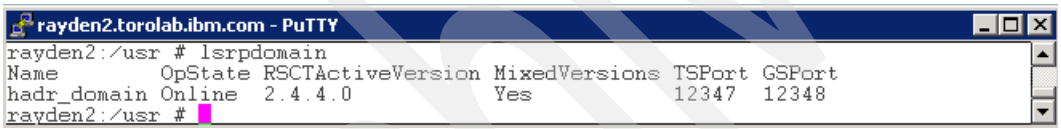


Figure 9-4 TSA cluster domain status

5. Ensure that all nodes are online in the active domain. You can get an output as shown in Figure 9-5.

   ```
   (P) # lsrpnode
   ```

Figure 9-5 TSA cluster nodes status

9.1.5 Enabling instance and HADR with TSA

As the base for automation, the components involved must first be described in a set of RSCT-defined resources. Due to diverse characteristics of resources, there are various RSCT resource classes to accommodate the differences. In a TSA cluster, a resource is any piece of hardware or software that has been

defined to IBM Resource Monitoring and Control (RMC). So in this case, the DB2 database instance and the HADR pair of database are both resources in the cluster, which are configured and registered with TSA for automation management. As explained above, every application needs to be defined as a resource to be managed and automated with TSA. Application resources are usually defined in the generic resource class IBM.Application. In this resource class, there are several attributes that define a resource, but at least three of them are application-specific:

- StartCommand
- StopCommand
- MonitorCommand

These commands may be scripts or binary executables. You must ensure that the scripts are well tested and produce the desired effects within a reasonable period of time. This is necessary because these commands are the only interface between TSA and the application.

The automatic package shipped with DB2 v9 includes several scripts that can control the behavior of the DB2 resources defined in a TSA cluster environment. Here is a description of the scripts.

- For the DB2 database instance:
  - regdb2salin: This script registers the DB2 instance into the TSA cluster environment as a resource.
  - db2_start.ksh, db2_stop.ksh, db2_monitor: These three scripts are registered as part of the TSA resource automation policy. TSA refers to this policy when monitoring DB2 database instances and when responding to predefined events, such as restarting a DB2 database instance when TSA detects that the DB2 database instance has terminated. Example 9-10 is a piece of sample TSA script for the DB2 instance.

Example 9-10 db2_start.ksh

```
......
function activateDatabase
{
   Resource=db2hadr_${db?}-rs
   hn=$(hostname)
   NodeRG1=$(lsrsrc-api -s IBM.Application::'Name="'${Resource?}'"
'::NodeNameList | grep -v $hn | tr "{" " " | tr "}" " " | tr "." " " |
awk '{print $1}' | tail -1)
   if [[ ! -z "$NodeRG1" ]]; then
      # HADR database ...
```

```
     OpState=$(lsrsrc-api -s IBM.Application::'Name="'${Resource?}'"&&
NodeNameList={"'${NodeRG1}'"} '::OpState 2> /dev/null)
      if [[ $OpState == 1 ]]; then
         # HADR is Primary on the other side, start as standby here
         su - ${DB2INSTANCE?} -c "db2 start hadr on db ${db?} as
standby"
      else
         su - ${DB2INSTANCE?} -c "db2 activate database ${db?}"
      fi
   else
      # Not HADR database
      su - ${DB2INSTANCE?} -c "db2 restart database ${db?}" &
      sleep 1
   fi
}
......
```

- For the HADR database pair:
  - reghadrsalin: This script registers the DB2 HADR pair with the TSA environment.
  - hadr_start.ksh, hadr_stop.ksh, hadr_monitor.ksh: These three scripts are registered as part of the TSA automation policy for TSA to monitor and control the behavior of the HADR database pair. Example 9-11 is the script for hadr_start.ksh:

Example 9-11 hadr_start.ksh

```
......
#############################################################
# starthadr()
#############################################################
starthadr()
{
   set_candidate_P_instance
   instance_to_start=${candidate_P_instance}
   HADR_partner_node_state

   $SVC_PROBE ${DB2HADRINSTANCE1?} ${DB2HADRINSTANCE2?}
${DB2HADRDBNAME?} ${VERBOSE?} S
   rc=$?

   if [[ $remote_node_alive == "Online" ]]; then
      # Bring up HADR as Primary on this node
      if [ $rc -eq 1 ]; then
         # already primary
```

```
         rc=0
      elif [ $rc -eq 2 ]; then
         # currently standby,peer
         # takeover (no force)
         logger -i  -p notice -t $0 "su - ${instance_to_start?} -c db2
takeover hadr on db ${DB2HADRDBNAME?}"
         su - ${instance_to_start?} -c "db2 takeover hadr on db
${DB2HADRDBNAME?}"
         $SVC_PROBE ${DB2HADRINSTANCE1?} ${DB2HADRINSTANCE2?}
${DB2HADRDBNAME?} ${VERBOSE?}
         rc1=$?
         if [ $rc1 -ne 1 ]; then
            :
            logger -i  -p err -t $0 "*** Database ${DB2HADRDBNAME} is
in Peer State, TAKEOVER FAILED"
            # Old primary node is still online, offline instance to
prevent split-brain
            # Uncomment following 3 lines to allow takeover by force
            #chrg -o Offline -s "Name =
'${forceRGOfflineInCaseOfByForce?}'"
            #su - ${instance_to_start?} -c "db2 takeover hadr on db
${DB2HADRDBNAME?} by force"
            #logger -i  -p notice -t $0 "NOTICE: Takeover by force
issued, old primary instance offlined to prevent split brain"
         fi

      elif [ $rc -eq 40 ]; then
         :
         logger -i  -p err -t $0 "*** Database ${DB2HADRDBNAME} is not
in Peer State, old Primary machine still Online"
         //–·?ƒ°£°£°£°£°£
         # Uncomment following 3 lines to allow takeover even in case
of non Peer Standby w/ old Primary machine Online
         #chrg -o Offline -s "Name =
'${forceRGOfflineInCaseOfByForce?}'"
         #su - ${instance_to_start?} -c "db2 takeover hadr on db
${DB2HADRDBNAME?} by force "
        #logger -i  -p notice -t $0 "NOTICE: Takeover by force issued,
old primary instance offlined to prevent split brain"
      else
         # current state of HADR is unknown
         # eg. If instance has just gone down, wait until it's
2*monitor period
         # so that instance can be restarted and db ACTIVATEd
         sleep 20
```

```
      fi # Bring up HADR as Primary on this machine

   else
      # Old primary machine is offline
      if [ $rc -eq 2 ]; then
         # Standby is currently in Peer State
         #
         # To bring up standby, will now do a TAKEOVER BY FORCE
         # No need to block until resource group is offline, we have
verified
         # that the node is down already
         # To bring up standby, will now do a TAKEOVER BY FORCE
         :
         logger -i  -p notice -t $0 "su - ${instance_to_start?} -c db2
takeover hadr on db ${DB2HADRDBNAME?} by force"
         su - ${instance_to_start?} -c "db2 takeover hadr on db
${DB2HADRDBNAME?} by force "
         logger -i  -p notice -t $0 "NOTICE: Takeover by force issued"
      elif [ $rc -eq 40 ]; then
         # Standby is currently not in Peer State
         :
         logger -i  -p err -t $0 "*** Database ${DB2HADRDBNAME} is not
in Peer State, old Primary machine Offline"

         # Uncomment following 3 lines to allow takeover even in case
of non Peer Standby w/ old Primary machine Online
         //–·?ƒ°£°£°£°£°£
         #logger -i  -p notice -t $0 "su - ${instance_to_start?} -c db2
takeover hadr on db ${DB2HADRDBNAME?} by force"
         #su - ${instance_to_start?} -c "db2 takeover hadr on db
${DB2HADRDBNAME?} by force "
         #logger -i  -p notice -t $0 "NOTICE: Takeover by force issued"
      fi
   fi # Bring up HADR on this machine

   # Return state
   $SVC_PROBE ${DB2HADRINSTANCE1?} ${DB2HADRINSTANCE2?}
${DB2HADRDBNAME?} ${VERBOSE?} S
   rcs=$?

   # Online succesful must return 0 whilst monitor returns 1
   # for Primary in Peer State and 3 for Primary not Peer
   if [ $rcs -eq 1 ]; then
      rc=0
   elif [ $rcs -eq 3 ]; then
```

```
        rc=0
        # Anything else, map directly from monitor
    else
        rc=$rcs
    fi

    return $rc
}
......
```

Take the following steps:

1. Register the DB2 instance as a resource that can be managed by the TSA cluster:

   ```
   (P)# regdb2salin -a db2inst1 -r -l rayden2
   (S)# regdb2salin -a db2inst1 -r -l salmon
   ```

2. Check the status of the resource group for the db2 instance. You can get an output like Figure 9-6.

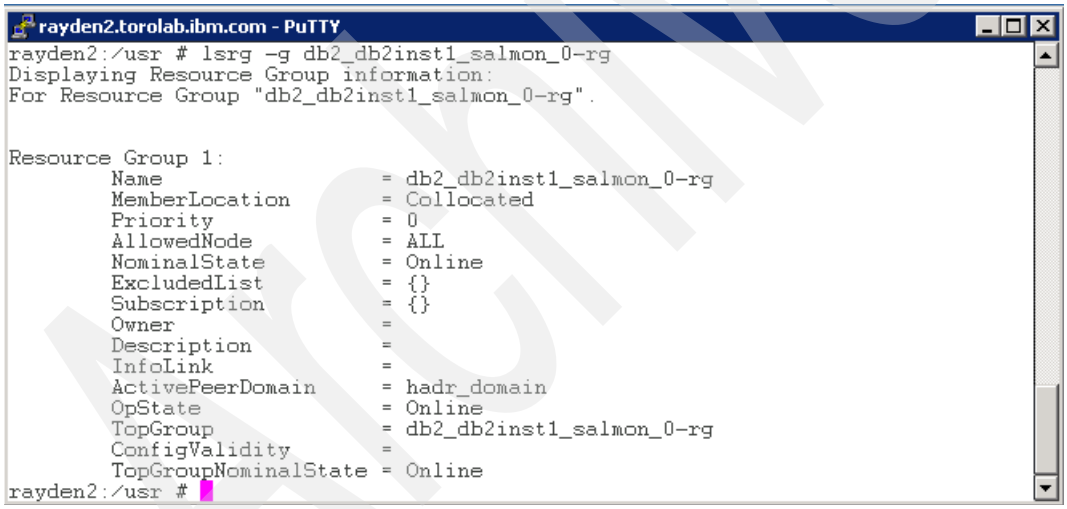


Figure 9-6 Resource group for DB2 instance in TSA cluster domain

Archived

3. Register the DB2 HADR pair into the TSA cluster as a specific resource:

   ```
   (P) reghadrsalin -a db2inst1 -b db2inst1 -d rmall
   ```

4. Check the status of the resource group of HADR. You can get an output like Figure 9-7.

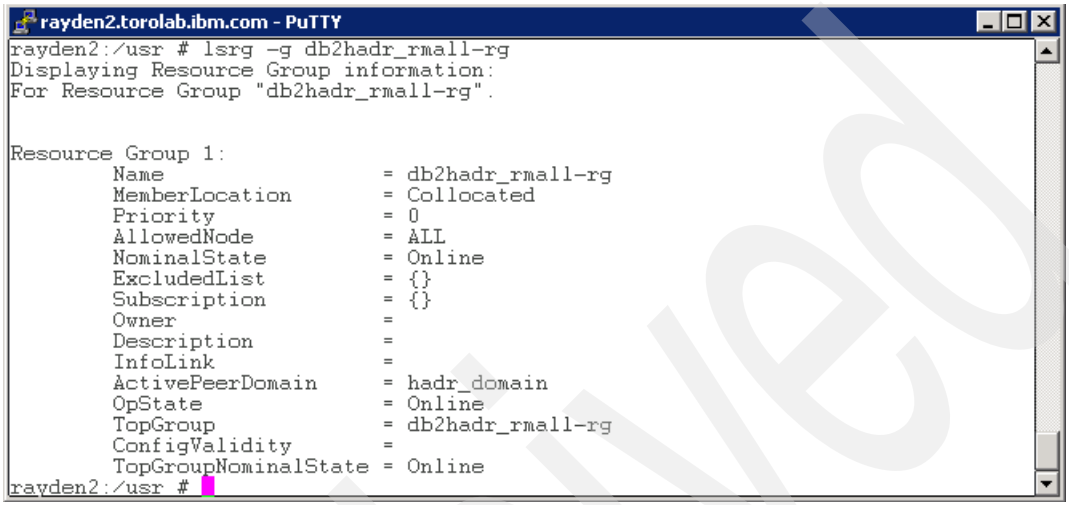


Figure 9-7 Resource group for HADR in TSA cluster domain

Archived

5. Check the status of the entire HADR cluster. Figure 9-8 shows the status of the cluster.

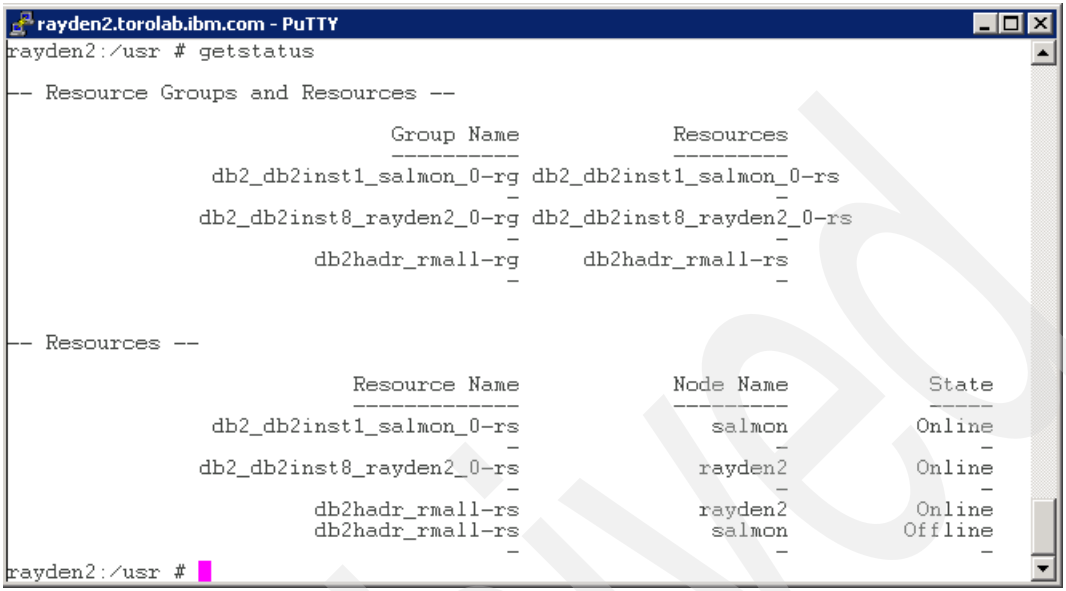


Figure 9-8 Status of the entire cluster

Archived

Archived

10

WebSphere Application Server and WebSphere Commerce federation and clustering

In this chapter, we describe how a WebSphere Commerce instance is created. We also show how to federate the instance to an IBM WebSphere Application Server Network Deployment cell and how to set up the instance to be distributed across multiple nodes of a WebSphere Application Server cluster.

Most of the information needed to perform these actions can be found in the whitepaper *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0*:

http://www.ibm.com/developerworks/websphere/library/tutorials/0804_clustering1/0804_clustering1.html

© Copyright IBM Corp. 2008. All rights reserved.

10.1 Scenario setup as described in the clustering whitepaper

We used the whitepaper *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0* to set up a scenario with one Web server serving an application server cluster with two cluster members, as shown in Figure 10-1.

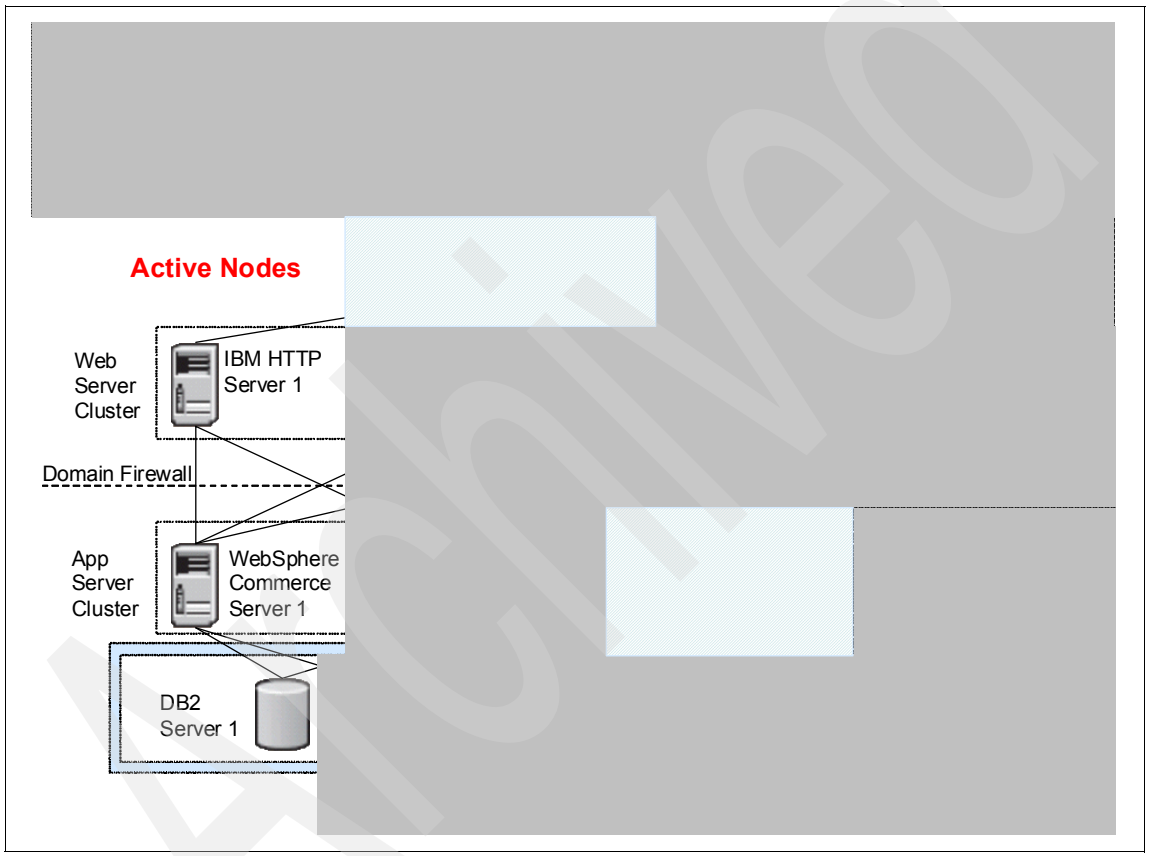


Figure 10-1 Scenario set up by following the Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0 whitepaper

Archived

For the nodes that are greyed out in Figure 10-1 on page 168, refer to the following chapters:

- Chapter 9, “High Availability solution for IBM DB2 Universal Database” on page 145, for the database tier
- Chapter 11, “Web server clustering” on page 185, for the Web tier
- 6.1.1, “Hardware-based High Availability” on page 57, for basic information about High Availability for the Network Deployment Manager

Table 10-1 lists the host names and IP addresses for the nodes used in our scenario.

Table 10-1 Host names and IP addresses for instance clustering

Node	Host name	IP address
Web server node 1	srvb501.torolab.ibm.com	9.26.126.120
WebSphere Commerce Server 1	goro.torolab.ibm.com	9.26.126.90
WebSphere Commerce Server 2	goro2.torolab.ibm.com	9.26.127.133
Network Deployment Manager	evergreen.torolab.ibm.com	9.26.30.222

After the BASE product and fix packs are installed on all the servers, the following tasks need to be performed to create a WebSphere Commerce cluster:

- Create a WebSphere Commerce instance.
- Configure a IBM WebSphere Application Server Network Deployment Manager.
- Federate the First WebSphere Application Server Node.
- Federate an Additional WebSphere Application Server Node.
- Create a WebSphere Application Server Cluster.

Detailed instructions for these tasks are found in sections 2.4, 2.5, 3.1, 3.2, 3.3, and 3.4, respectively, in the whitepaper *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0*.

Although the whitepaper describes the main steps for setting up Web server High Availability using IBM WebSphere Edge Components Load Balancer and additional Web servers, we experimented with additional scenarios (for example, NAT forwarding, High Availability for Load Balancer itself) and decided to give

Archived

the Web tier a special chapter on its own that contains more details than the paper (see Chapter 11, “Web server clustering” on page 185).

For this reason we also provide relevant information about the initial Web server configuration, as taking place during instance creation, in the following section.

Important: Before starting to set up your environment using the *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0* whitepaper, browse through the following section 10.2, “Details on configuring Web server node 1” on page 170, as it contains important information about the Web server setup. Read:

- 10.2.1, “Pre-instance creation tasks” on page 171, before creating the WebSphere Commerce instance
- 10.2.2, “Post instance creation tasks” on page 173, after creating the instance, but before federating it to the Deployment Manager cell
- 10.2.3, “Post federation tasks” on page 176, after federating the instance

10.2 Details on configuring Web server node 1

After installing a Web server node, it needs to be configured for routing requests to the WebSphere Commerce application and for serving WebSphere Commerce specific static content.

Assuming that IBM HTTP Server has been installed on Web server node 1 as described in 8.5, “Install IBM HTTP Server” on page 127, the following sections point out some important configuration details with regard to serving a clustered WebSphere Commerce instance.

When a WebSphere Commerce instance is created (see the whitepaper *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0*), only one remote Web server can be automatically configured. We refer to it as Web server node 1, avoiding the term *primary*, as after the instance configuration is complete, it will not be different from any further Web servers serving the instance. For further Web server nodes, Web server definitions have to be added to the WebSphere Application Server configuration, and configuration files and static content have to be copied from Web server node 1 manually, as described in Chapter 11, “Web server clustering” on page 185. Web server node 1 serves as a model or template for setting up additional Web servers.

10.2.1 Pre-instance creation tasks

This section describes configuration steps to perform *before* creating the WebSphere Commerce instance.

Configure directories, non-root_user, and permissions

As part of WebSphere Commerce instance creation, the instance creation scripts running on the application server attempt to upload Web server configuration files to Web server node 1. This directory can be freely chosen at instance creation time. We choose to store the Web server configuration in the same directory as on WebSphere Commerce node 1. The directory is:

WC_Install_Dir/instances/*Instance_Name*

The instance creation scripts also create the Web server configuration such that the static content for the WebSphere Commerce instance is expected to be in the same directory on the Web server as on the application servers, which is:

WAS_Install_Dir/profiles/*Profile_Name*/installedApps/*Cell_Name*/*Application_Name*

The instance that we use for our scenarios is called *demo*, so the variables in the directory names would have the values listed in Table 10-2.

Table 10-2 Directory variables for the Web servers

Variable	Value
Instance_Name	demo
Profile_Name	demo
Cell_Name	WC_demo_cell
Application_Name	WC_demo.ear

To configure the directories and permissions:

1. We create the two directories mentioned above, as follows (as root):

   ```
   mkdir -p WC_Install_Dir/instances/Profile_Name
   mkdir -p WAS_Install_Dir/profiles/Profile_Name/installedApps
     /Cell_Name/Application_Name
   ```

   WebSphere Commerce attempts to write configuration files to the Web server node 1 as *non-root_user* when creating instances (either using FTP or a mounted directory). For this reason, as well as for security reasons, we recommend that the IBM HTTP Server and IBM HTTP Server Plug-in

Archive

installation directories, as well as any WebSphere Commerce specific content directories on the Web server, are owned by a *non-root_user*.

2. As root, create a non-root group (*non-root_group*) and a *non-root_user* (*non-root_user*) with home directory /home/*non-root_user*, for which the non-root group is the default group. We recommend assigning the same group ID and user ID as for the *non-root_user* on WebSphere Commerce node 1 (see “Installation prerequisites” on page 106). You may use any tool available to you to create the user and the group. In AIX, this task can be performed by using the **smitty** tool. After creating the group and the user, you might still need to create the user's home directory /home/*non-root_user*, if this has not been done by the tools.
3. The user's umask should be 0022. Log in as the new user and at the prompt, type `umask` to check the umask. If it the result is not 022, change the user's default umask accordingly, for example, by using **smitty** or adding the command **umask 0022** to the user's profile ~*non-root_user*/.profile.
4. As root, modify ownership and permissions for the IBM HTTP Server and IBM HTTP Server Plug-in installation directories, as well as for the directories created in step 1 on page 171 above:

```
chown -R non-root_user:non-root_group IHS_Install_Dir
chown -R non-root_user:non-root_group WAS_Install_Dir
chown -R non-root_user:non-root_group WC_Install_Dir
chown -R non-root_user:non-root_group Plugin_Install_Dir
```

Enable remote configuration

The application server node that is used to create WebSphere Commerce instances—WebSphere Commerce node 1—needs remote access to Web server node 1 for configuration.

Remote configuration of Web server node 1 can be done using FTP or a mounted directory as part of WebSphere Commerce instance creation. We recommend FTP, as this method is more flexible, for example, if WebSphere Commerce node 1 and Web server nodes use incompatible file systems.

You need to enable an FTP server on Web server node 1. Ensure that the user *non-root_user* can log in from WebSphere Commerce node 1 (which is used to create the instance). The user *non-root_user* also needs the permissions to create directories as well as files in the WebSphere Commerce instance configuration directory (*WC_Install_Dir*/instances/*Instance_Name*) and the IBM HTTP Server Plug-in installation directory (*Plugin_Install_Dir*).

To test FTP access, log in to your WebSphere Commerce node 1, create a temporary file, and try to upload it, as shown in Example 10-1, for our Web server node 1, srvb501.torolab.ibm.com.

Example 10-1 Testing the FTP connection and permissions for the non-root_user (wasuser in our example)

```
[root:/tmp/ftptest] > touch test.file
[root:/tmp/ftptest] > ftp srvb501.torolab.ibm.com
Connected to srvb501.torolab.ibm.com.
220 srvb501 FTP server (Version 4.2 Fri Oct 7 19:22:01 CDT 2005) ready.
Name (srvb501.torolab.ibm.com:root): wasuser
331 Password required for wasuser.
Password:
230-Last login: Wed Jul  4 19:47:13 EDT 2007 on /dev/pts/2 from
dev01931.de.ibm.com
230 User wasuser logged in.
ftp> cd /usr/IBM/WebSphere/CommerceServer60/instances/demo
250 CWD command successful.
ftp> put test.file
200 PORT command successful.
150 Opening data connection for test.file.
226 Transfer complete.
1 bytes sent in 0.001459 seconds (0.669 Kbytes/s)
local: test.file remote: test.file
ftp> cd /opt/IBM/WebSphere/Plugins
250 CWD command successful.
ftp> put test.file
200 PORT command successful.
150 Opening data connection for test.vi.
226 Transfer complete.
1 bytes sent in 0.001459 seconds (0.669 Kbytes/s)
local: test.file remote: test.file
ftp> quit
221 Goodbye
```

10.2.2 Post instance creation tasks

You need to perform several tasks on Web server node 1 *after* the WebSphere Commerce has been created.

Archived

Check the configuration directory

Check the configuration directory on Web server node 1. Three subdirectories should have been created, as shown in Example 10-2.

Example 10-2 Instance configuration directory (with non-root_user name wasuser)

```
# cd WC_Install_Dir/instances/Profile_Name
# ls -la
total 40
drwxr-xr-x   5 wasuser wasgroup        512 Jul 07 15:15 .
drwxr-xr-x   3 wasuser wasgroup        512 Jul 07 15:04 ..
drwxr-xr-x   2 wasuser wasgroup        512 Jul 07 15:15 httpconf
drwxr-xr-x   2 wasuser wasgroup        512 Jul 07 15:15 httplogs
drwxr-xr-x   2 wasuser wasgroup        512 Jul 07 15:15 web
```

If these subdirectories do not exist, the FTP configuration failed when the instance was created. In that case, you need to copy the three subdirectories from *WC_Install_Dir*/instances/*Instance_Name* on WebSphere Commerce node 1.

Verify the Web server configuration file

During instance creation, the Web server configuration file, httpd.conf, is placed in *WC_Install_Dir*/instances/*Instance_Name*/httpconf. To verify it, open the file using a text editor (for example, vi).

Then verify the locations of the document root directory, the process ID and log files, the SSL key files, and the IBM HTTP Server Plug-in files. The relevant lines are shown in Example 10-3 (indentation indicates that the line should be on one line with the previous line).

Example 10-3 Relevant directives in httpd.conf (not necessarily appearing in this order inside the file)

```
DocumentRoot "WC_Install_Dir/instances/Instance_Name/web"
CustomLog "WC_Install_Dir/instances/Instance_Name/httplogs/access_log"
   common
ErrorLog "WC_Install_Dir/instances/Instance_Name/httplogs/error_log"
PidFile "WC_Install_Dir/instances/Instance_Name/httplogs/httpd.pid"
KeyFile "WC_Install_Dir/instances/Instance_Name/httpconf/keyfile.kdb"
LoadModule was_ap20_module Plugin_Install_Dir/bin/mod_was_ap20_http.so
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
WebSpherePluginConfig "WC_Install_Dir/instances/Instance_Name/httpconf/
   plugin-cfg.xml"
```

Copy static content

The Web server configuration is now almost complete. You only need to copy the static content to be served by the Web server from WebSphere Commerce node 1. Perform the following steps:

1. As the *non-root_user*, on WebSphere Commerce node 1, tar up the WebSphere Commerce instance application with the full path:

   ```
   tar -cvf instance.ear.tar
   WAS_Install_Dir/profiles/Profile_Name/installedApps/Cell_Name/WC_<Instance_Name>.ear
   ```

2. Then FTP (or SCP) this tar file to a temporary directory on Web server node 1, logging in as the <*non-root_user*>.

3. As *non-root_user*, on Web server node 1, untar the tar file:

   ```
   tar -xvf /tmp/instance.ear.tar
   ```

4. On Web server node 1, remove any JSP (*.jsp) and JAR (*.jar) files in *WAS_Install_Dir*/profiles/*Profile_Name*/installedApps/*Cell_Name*/WC_<*Instance_Name*>.ear and all subdirectories.

Activate remote Web server management (optional)

You may configure WebSphere Application Server to manage your Web server using the administrative console. However, we recommend doing so after federating your WebSphere Commerce instance, as the remote management configuration is removed during federation. See “Activate remote Web server management” on page 179 for instructions on how to configure remote Web server management.

Start the Web server

On Web server node 1, as root, start IBM HTTP Server for the new WebSphere Commerce instance, as shown in Example 10-4. The command needs to be on one line.

Example 10-4 Starting the Web server process

```
IHS_Install_Dir/bin/apachectl -k start -f
WC_Install_Dir/instances/Instance_Name/httpconf/httpd.conf
```

Important: IBM HTTP Server must be stopped and restarted by the root user.

10.2.3 Post federation tasks

A new Web server configuration for Web server node 1 is created as part of the reconfiguration of the cell level documents after federating WebSphere Commerce node 1 to the IBM WebSphere Application Server Network Deployment cell used for clustering, as described in the whitepaper *Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0*. We need to verify and update the new Web server definition for using it properly with WebSphere Commerce.

Update the Web server configuration

We use the Network Deployment Manager administration console to verify and to update the Web server configuration.

1. Open the Network Deployment Manager administration console in your browser and navigate to **Servers** → **Web servers** → **webserver1** to display the Web server configuration panel, as shown in Figure 10-2.

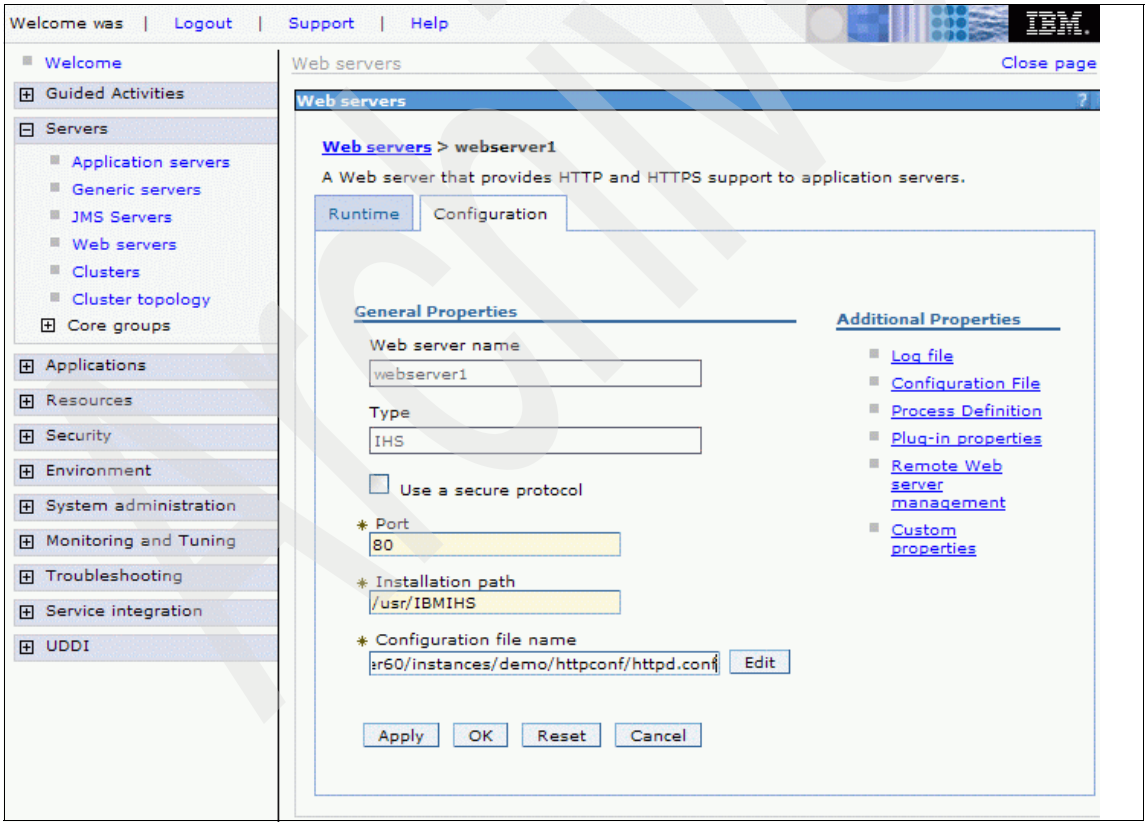


Figure 10-2 Web server configuration panel

2. On the Web server configuration panel for webserver1, make sure that the configuration file name contains the httpd.conf file under a directory named httpconf, which is again under the configuration directory that was specified during instance creation:

   WC_Install_Dir/instances/*Instance_Name*/httpconf/httpd.conf

   If you need to change the path and file name, make sure to click **Apply** after making your changes.

3. Click **Plug-in properties** on the right to show the configuration part for the IBM HTTP Server Plug-in for webserver1, as shown in Figure 10-3.

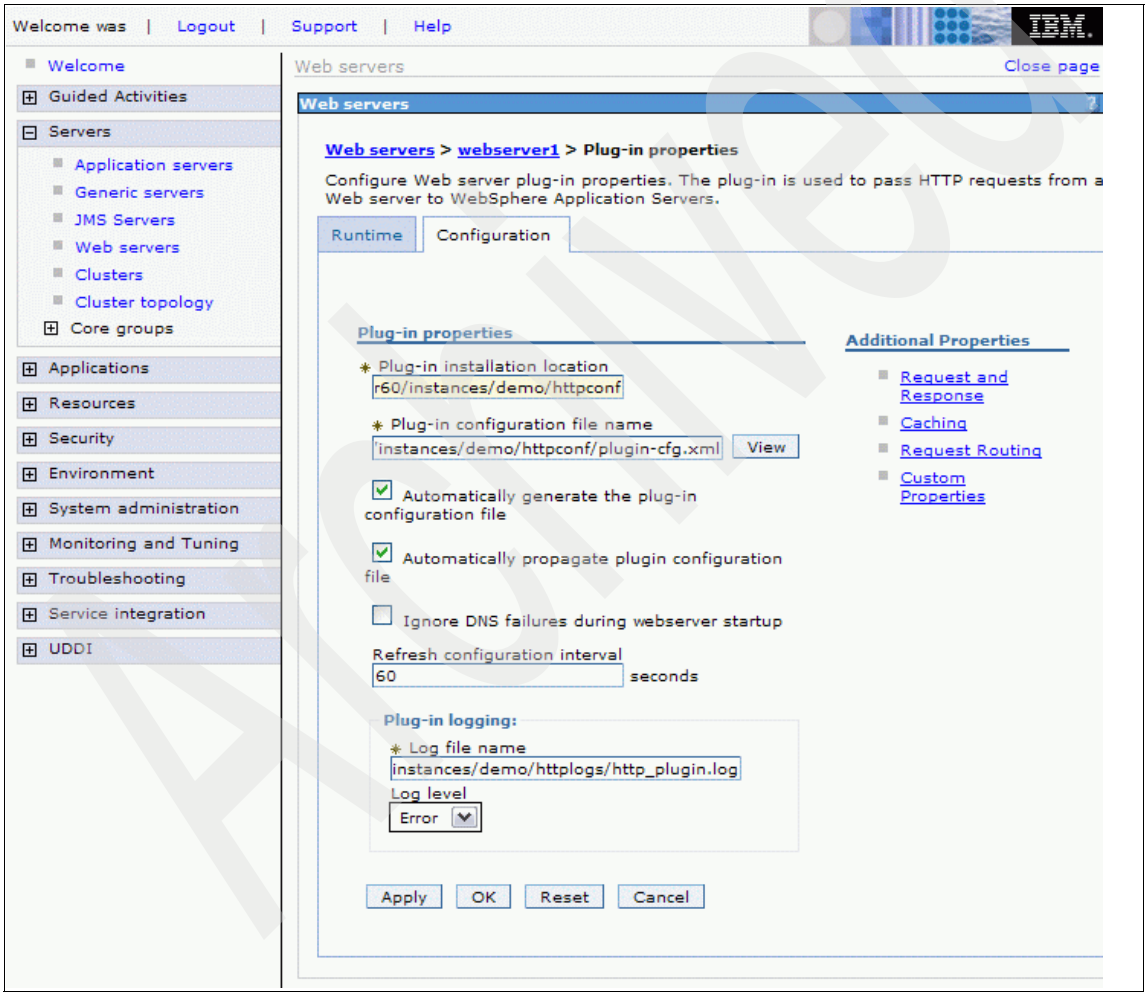


Figure 10-3 IBM HTTP Server Plug-in properties panel

4. The field label *Plug-in installation location* is misleading. The field does not need to point to the actual installation directory for the IBM HTTP Server Plug-in (*Plugin_Install_Dir*), but can point to any existing directory on the Web server. We recommend again using the httpconf directory under the directory specified during instance creation. The directory is used for two purposes:
   - If *Plug-in configuration file name* does *not* contain an absolute path (for example, not starting with /), the actual location for the configuration file, plugin-cfg.xml (which is generated by Network Deployment Manager) is assembled as follows:

     `<Plug-in install location>/config/`<Web server `name>/<Plug-in configuration file name>`

     For example, in our case, if we used the plug-in installation location as shown in Figure 10-3 on page 177, and put just plugin-cfg.xml as plug-in configuration file name, the resulting path would be:

     `WC_Install_Dir``/instances/demo/httpconf/config/webserver1/plugin-cfg.xml`

     Note: If you use an absolute path, clicking the **View** button (on the right side of the Plug-in configuration file name field) displays an empty configuration file. If you want to use this feature, you need to specify a relative name and create the corresponding directory structure on your Web server. In the latter case, we recommend using just plugin-cfg.xml as the file name and creating the config/webserver1 directory under the directory specified in the Plug-in installation location field.

   - In the IBM HTTP Server Plug-in configuration file, two files used for SSL connections, plugin-key.kdb and plugin-key.sth, are referenced using the following paths:

     `<Plug-in install location>``/etc/plugin-key.kdb`
     `<Plug-in install location>``/etc/plugin-key.sth`

   WebSphere Commerce puts the file plugin-cfg.xml into the httpconf subdirectory of the configuration directory specified on the Web server panel of the instance creation wizard.

   As we recommend using *WC_Install_Dir*/instances/*Instance_Name* as the configuration directory, we also recommend using the following field values on the Plug-in properties panel:

   - Plug-in installation location

     `WC_Install_Dir``/instances/`*`Instance_Name`*`/httpconf`

   - Plug-in configuration file name

     `WC_Install_Dir``/instances/`*`Instance_Name`*`/httpconf/plugin-cfg.xml`

WebSphere Commerce also creates the httplogs directory under the configuration directory on the Web server for IBM HTTP Server and IBM HTTP Server Plug-in logs, so we recommend as field value for the Plug-in log file:

```
Plug-in logging > Log file name:
WC_Install_Dir/instances/Instance_Name/httplogs/http_plugin.log
```

Again, if you needed to make any changes on the Plug-in properties panel, click **Apply** after making your changes.

5. You can save your changes now, or continue following the instructions. In the next section, "Activate remote Web server management" on page 179, further changes are made using the Network Deployment Manager administration console before configuration is saved. If you decide to save now, follow steps 4 on page 182 and 5 on page 182 to save the configuration.
6. Log on to Web server node 1 as wasuser.
7. On the Web server, go to the directory that you specified as the plug-in installation location in step 4 on page 178 above and make sure that there is an etc subdirectory, for example:

   ```
   WC_Install_Dir/instances/Instance_Name/httpconf/etc
   ```

   Create the directory if necessary.
8. Copy all plugin-key.* files from their original location, which is the etc directory under the actual IBM HTTP Server Plug-in installation directory (*Plugin_Install_Dir*/etc), to the new etc directory just created.

Note: If you choose to use the actual IBM HTTP Server Plug-in installation directory as the plug-in installation location in step 4 above, you do not need to copy the files. However, we recommend using a dedicated configuration directory for the WebSphere Commerce instance. In a production environment you should then replace the default self-signed SSL certificates by production certificates. This applies both to the Web server certificates (keyfile.*) and to the plug-in certificates (plugin-key.*)

Activate remote Web server management

Once the WebSphere Commerce instance is federated to a Network Deployment cell, the Web server is configured as an unmanaged Web server node. As we use IBM HTTP Server, IBM WebSphere Application Server Network Deployment can still manage the Web server, for example, starting and stopping it and propagating the IBM HTTP Server Plug-in configuration file to the Web server after generating that file. (See 3.4, "Web server topology in a Network Deployment cell," in the *WebSphere Application Server V6 Scalability and*

Performance Handbook, SG24-6392, for more information about managing Web servers.)

Network Deployment Manager needs remote access to the Web server node for administrating it. For remote administration to work, you need to create an administrative user for IBM HTTP Server and start the administrative server as follows:

1. As wasuser, on Web server node 1, create the administrative user ID:

   ```
   # IHS_Install_Dir/bin/htpasswd -cb IHS_Install_Dir/conf/admin.passwd
   ihsadmin ihsadmin_passwd
   ```

2. Modify *IHS_Install_Dir*/conf/admin.conf, and change the run user and group settings from nobody to wasuser, as shown in Example 10-5.

Example 10-5 Setting the run user and group for IBM HTTP Server administrative server

```
# Default user and group settings for the server
User wasuser
Group wasgroup
```

3. As root, start the administrative server:

   ```
   # IHS_Install_Dir/bin/adminctl start
   ```

   Note: The administrative server is not needed at runtime, so we recommend *not* adding this command to any system startup scripts. If you still want to run the administrative server automatically, you may do so, but you need to be aware of the security implications of running the administrative server and configure your firewalls to restrict access to the administrative server.

To be able to use Network Deployment Manager to start and stop your Web server and to update the IBM HTTP Server Plug-in configuration, you also need to configure remote Web server management in the Network Deployment Manager administrative console as follows:

1. Navigate to **Servers** → **Web servers** → **webserver1** → **Remote Web server management**.

Archived

2. Enter the user name and password for the IBM HTTP Server administrative server, which we configured and started in “Activate remote Web server management (optional)” on page 175, as shown in Figure 10-4. The default port is 8008. This can be changed in *IHS_Install_Dir*/conf/admin.conf on Web server node 1, where the port is specified as the parameter of the listen directive.

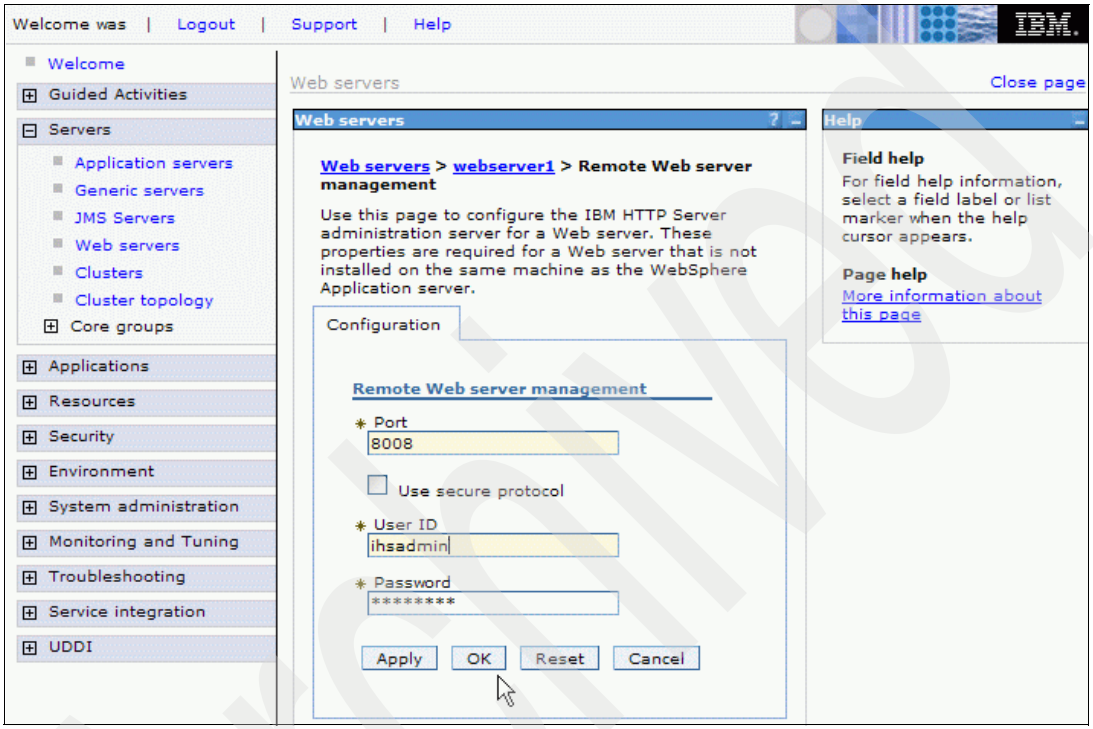


Figure 10-4 Configuring remote Web server management

Archive

3. Click **OK**. A message will be displayed about changes having been made to the configuration, as shown in Figure 10-5.

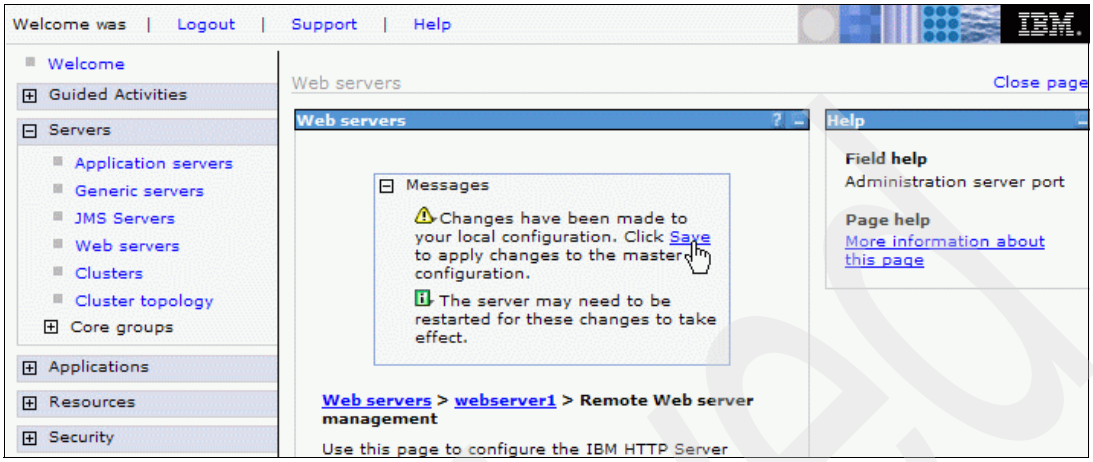


Figure 10-5 Going to the Save workspace changes dialog

4. Click the hyperlinked word **Save**. The configuration saving dialog is displayed (Figure 10-6).

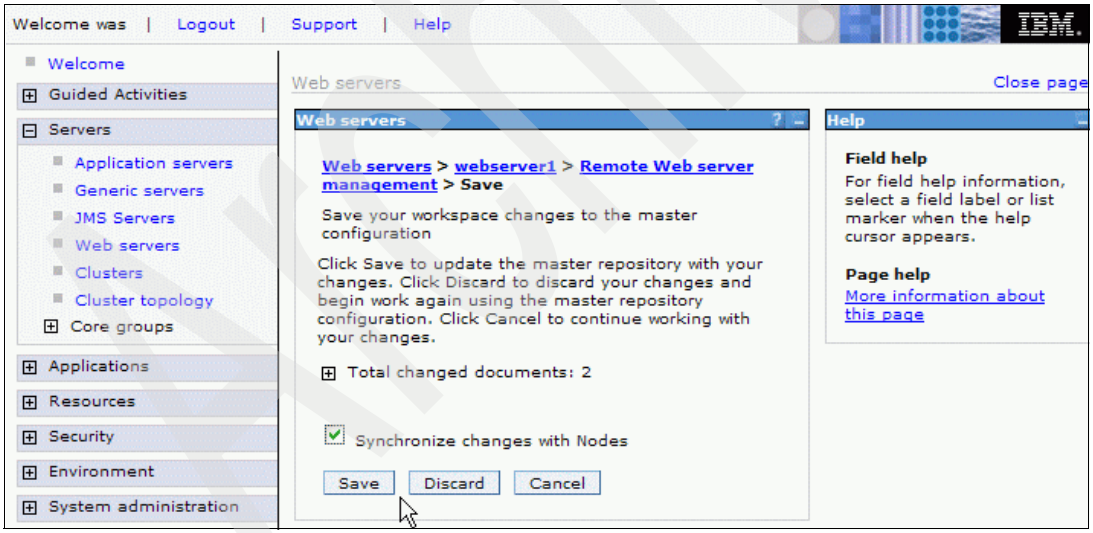


Figure 10-6 Save workspace changes dialog

5. Check **Synchronize changes with Nodes**, then click **Save**.

The Web server can now be started and stopped, and the IBM HTTP Server Plug-in configuration can be propagated to it through the **Server** → **Web servers** panel in the administrative console.

Propagate IBM HTTP Server Plug-in configuration

If automatic generation and propagation of IBM HTTP Server Plug-in configuration and remote Web server management are activated (see above), the plug-in configuration is automatically updated on all Web servers. If automatic propagation is not activated for the Web server, the plug-in has to be manually propagated to the Web server by performing the following steps:

1. Navigate to **Servers** → **Web servers** (Figure 10-7).

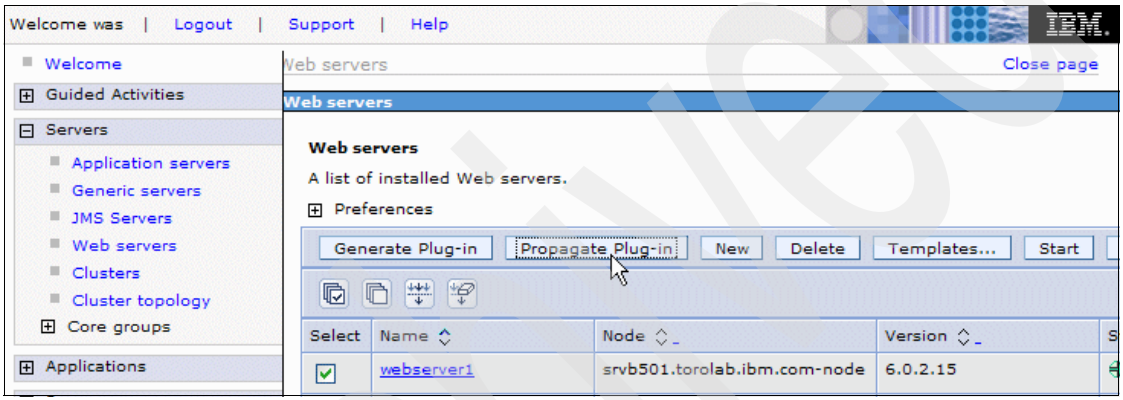


Figure 10-7 Web servers page

2. Under **Select**, check all webserver1 for which automatic propagation is not activated.
3. Click **Generate Plug-in**, then **Propagate Plug-in**.

   This still requires that remote management is set up for the Web server and that the IBM HTTP Server administrative server is running on the Web server (see Figure 10-7).

Archived

If remote management is disabled, the plug-in configuration file can still be automatically generated, but it needs then to be copied manually to the Web server. The path to the generated configuration file on the Network Deployment Manager is displayed after clicking **Generate Plug-in**, as shown in Figure 10-8.

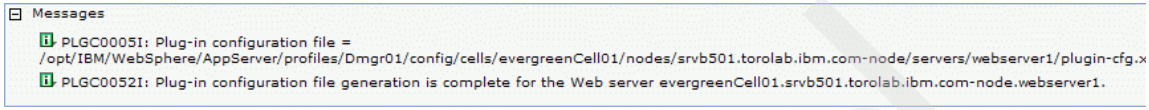


Figure 10-8 Plug-in generation message

From there, you need to copy the plugin-cfg.xml file to your Web server. On the Web server, update the WebSpherePluginConfig directive in the httpd.conf file to point to the copied file.

The Web server configuration is now complete.

Restart the Web server

To apply the configuration changes, restart the Web server. If remote management is enabled, you may do so by selecting **webserver1** on the Web servers page (**Servers** → **Web servers**, as shown in Figure 10-7 on page 183), then clicking **Stop** and then **Start**.

If you do not use remote management, log on to Web server node 1as root and issue the two commands given in Example 10-6, where each command needs to be on one line.

Example 10-6 Restarting IBM HTTP Server manually

```
IHS_Install_Dir/bin/apachectl -k stop -f
WC_Install_Dir/instances/demo/httpconf/httpd.conf
IHS_Install_Dir/bin/apachectl -k start -f
WC_Install_Dir/instances/demo/httpconf/httpd.conf
```

Archived

11

Web server clustering

In this chapter, we describe how to add additional Web servers for our WebSphere Commerce instance. We also show how to configure IBM WebSphere Edge Components Load Balancer to distribute HTTP requests between the Web servers and monitor the Web servers to be able to detect outages and automatically stop routing traffic to failed Web servers.

We also show how Load Balancer can be made highly available using a primary and a backup server that are able to monitor each other by exchanging heartbeat messages.

Figure 11-1 highlights the nodes that we configure in this chapter.

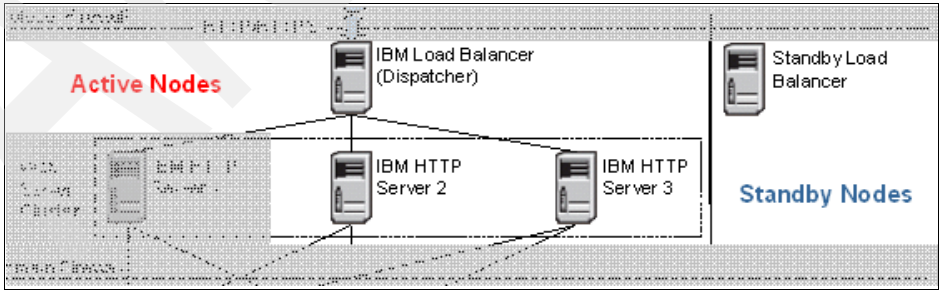


Figure 11-1 Web tier additional configuration in this chapter

© Copyright IBM Corp. 2008. All rights reserved.

11.1 Add additional Web servers

As we only use active Web server nodes managed by Load Balancer, rather than having active/standby pairs of machines (see 7.1, “Introduction to Web server High Availability” on page 81), the configuration is identical on all Web server machines. All necessary files can be copied from Web server node 1, which was configured as part of WebSphere Commerce instance creation.

We recommend adding additional Web servers only after federation (see Clustering WebSphere Commerce V6.0 with WebSphere Application Server V6.0), as the Web server configurations are removed from the WebSphere Application Server cell during federation.

We describe how to add one Web server node to the cell configuration of a federated WebSphere Commerce instance. We refer to it as Web server node 2. In our scenario, its host name is srvb504.torolab.ibm.com and the IP address is 9.26.127.157.

11.1.1 Preparation

Before we configure Web server node 2 to serve content and route requests for our WebSphere Commerce instance, it needs to prepared as follows:

1. Create directories and a *non-root_user*, and set permissions for your new Web server exactly as described for Web server node 1 in “Configure directories, non-root_user, and permissions” on page 171.
2. Make sure that the Web server’s host name is resolvable to its IP address on your Network Deployment Manager node and all application server nodes, either by using DNS or by adding a line to your hosts file (/etc/hosts on AIX, Linux, and other UNIX-like operating systems).
3. If you have an inbound firewall, configure it so that Web server node 2, like Web server node 1, is able to initiate TCP connections to the application server nodes and to accept TCP connections from Network Deployment Manager.

11.1.2 Copy files from Web server node 1

We need to copy the WebSphere Commerce configuration and static content to Web server node 2:

1. On Web server node 1, as wasuser, create tar files for the configuration and the static content:

   ```
   tar -cvf instanceconfig.tar WC_Install_Dir/instances/Instance_Name
   ```

```
tar -cvf instance.ear.tar WAS_Install_Dir/profiles/Profile_Name/
   installedApps/Cell_Name/WC_<Instance_Name>.ear
```

2. Copy the files to /tmp on Web server node 2, for example, using FTP or SCP, and make sure that wasuser can read the copies.
3. As wasuser, untar the files to the same directories:

```
tar -xvf /tmp/instanceconfig.tar
tar -xvf /tmp/instance.ear.tar
```

4. Verify that all copied directories and files are owned by wasuser.

11.1.3 Modify the Web server configuration

The Web server configuration file, httpd.conf, is copied from Web server node 1 to *WC_Install_Dir*/instances/*Instance_Name*/httpconf/httpd.conf. After copying, it still contains the host name of Web server node 1 in Listen, VirtualHost, and ServerName statements.

Open the *WC_Install_Dir*/instances/*Instance_Name*/httpconf/httpd.conf file in a text editor and replace all occurrences of the host name of Web server node 1 by the host name of Web server node 2.

11.1.4 Add the new Web server to the cell configuration

Perform the following steps to add the new Web server to your Network Deployment Manager cell configuration:

1. Log on to your Network Deployment Manager as *non-root_user*.
2. Change to the WebSphere Application Server installation directory (*ND_Install_Dir*, the default is /usr/IBM/WebSphere/AppServer on AIX, /opt/IBM/WebSphere/AppServer on Linux).
3. Execute the following command on one line:

```
./wsadmin.sh -profileName DmgrProfileName -f
configureWebServerDefinition.jacl WebServerName IHS
'IHS_Install_Dir'
'WC_Install_Dir/instances/Instance_Name/httpconf/httpd.conf' 80
MAP_ALL 'WC_Install_Dir/instances/Instance_Name/httpconf/' unmanaged
WebServerNodeName WebServerHostName WebServerOS
```

Archived

For example, our Web server node 2 (srvb504.torolab.ibm.com, running on AIX) is added as webserver2 by the command shown in Example 11-1.

Example 11-1 Adding a Web server definition to the cell configuration

```
./wsadmin.sh -profileName Dmgr01 -f configureWebserverDefinition.jacl webserver2 IHS
'/usr/IBMIHS'
'/usr/IBM/WebSphere/CommerceServer60/instances/demo/httpconf/httpd.conf' 80 MAP_ALL
'/usr/IBM/WebSphere/CommerceServer60/instances/demo/httpconf/' unmanaged
srvb504.torolab.ibm.com srvb504.torolab.ibm.com aix
```

In the Network Deployment Manager administrative console, you should now see two Web servers when navigating to **Servers** → **Web servers**, as shown in Figure 11-2.

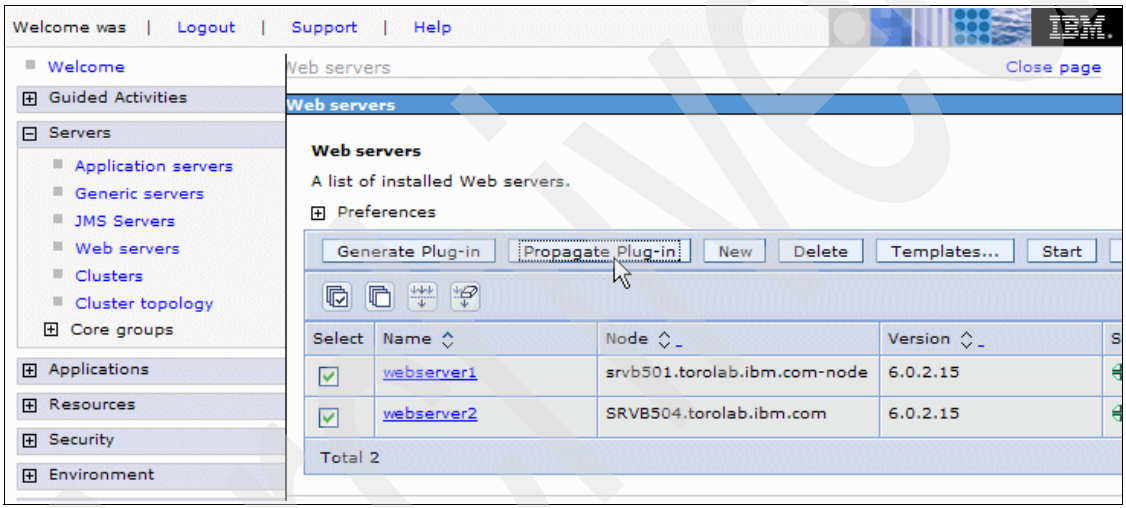


Figure 11-2 Web servers page

4. For the new Web server, verify and change the configuration, and activate remote Web server management as described for the first Web server in “Update the Web server configuration” on page 176 and “Activate remote Web server management (optional)” on page 175. Do not propagate the plug-in configuration at this point.

Archive

5. Verify the application mapping by navigating to **Applications** → **Enterprise Applications** → **WC_<*Instance_Name*> > Map modules to servers**. The modules of the WebSphere Commerce instance application (WC_demo in our case) should be mapped to all Web servers and to the application server cluster. This is shown for each module in the Server column, as shown in Figure 11-3.

Figure 11-3 Map modules to server panel - Server column

If this is not the case, perform the following steps:

a. Select the cluster and all Web servers in the Clusters and Servers combo box, as shown in Figure 11-4.

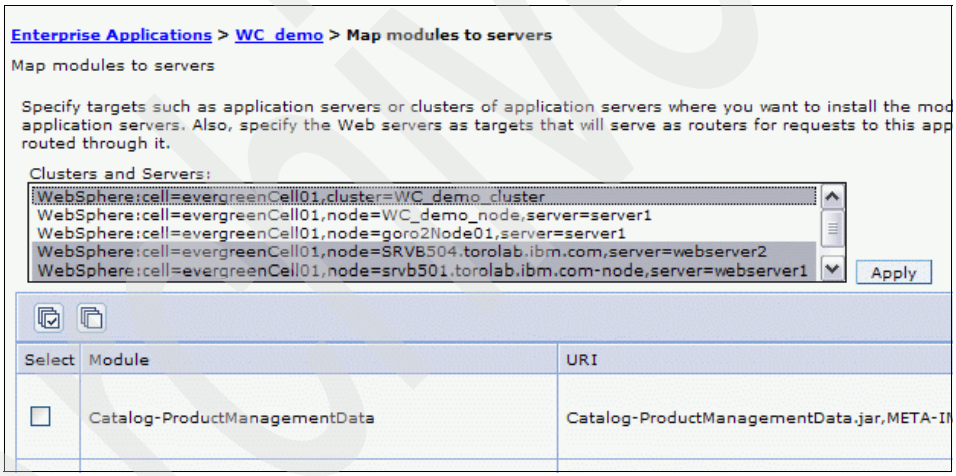


Figure 11-4 Selecting the servers for mapping

b. Then click the Select all icon, as shown in Figure 11-5.

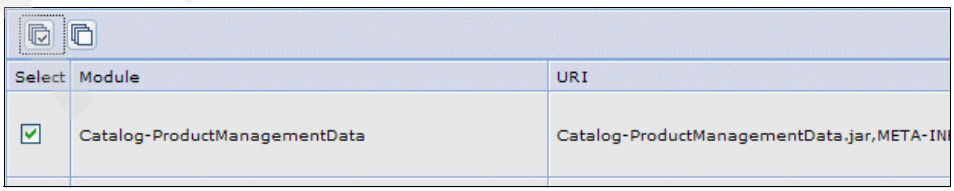


Figure 11-5 Selecting all application modules

Archived

c. Click **Apply** (Figure 11-4 on page 189) and verify the mappings in the Server column.

d. Click **OK** below the list of application modules.

6. To add virtual host aliases for the new Web server, navigate to **Environments** → **Virtual Hosts**.

7. For each WebSphere Commerce virtual host (VH_<*Instance_Name*>...), click the virtual host name, then click **Host Aliases**. Add host aliases for the new Web server for each port of the virtual host by clicking **New** and adding the Web server and port information, as shown in Figure 11-6.

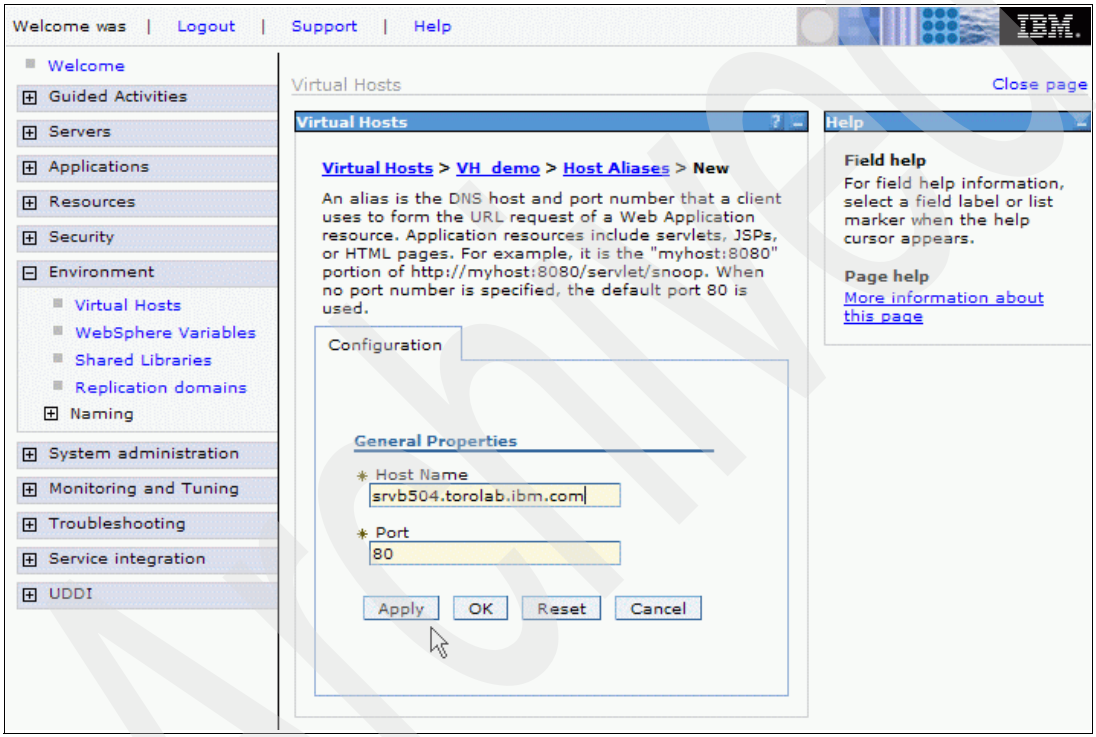


Figure 11-6 Adding a host alias to the VH_<Instance_Name> virtual host

Click **OK** and repeat this for each port of the virtual host. Also, add host aliases for the short name of the Web servers. Figure 11-7 shows the complete host alias list for the VH_<*Instance_Name*> virtual host for our two Web servers (our instance name is *demo*).

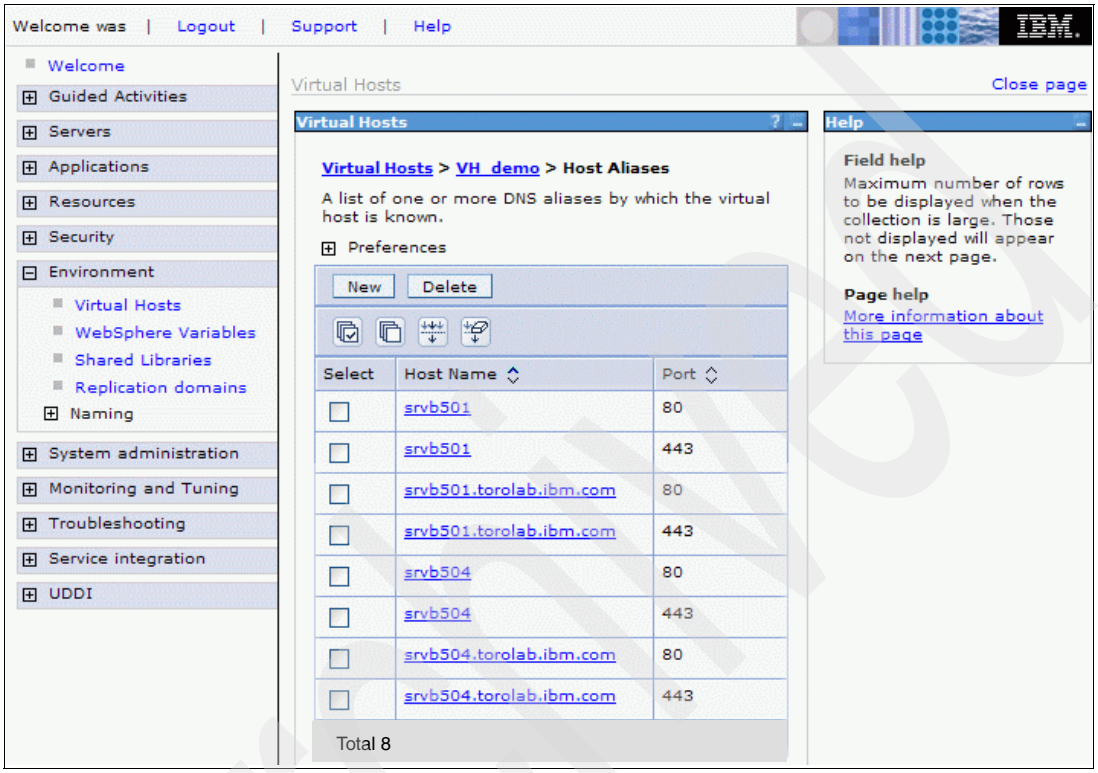


Figure 11-7 Complete host alias list for VH_<Instance_Name>

Add host aliases for the Web server's short and full qualified host names to all VH_<*Instance_Name*>... virtual hosts for the relevant ports. For example, for the VH_<*Instance_Name*>_tools virtual host, add aliases for port 8000. For VH_<*Instance_Name*>_admin, add aliases for port 8002, and so on.

Archived

8. To save the configuration changes, click **Save**, as shown in Figure 11-8.

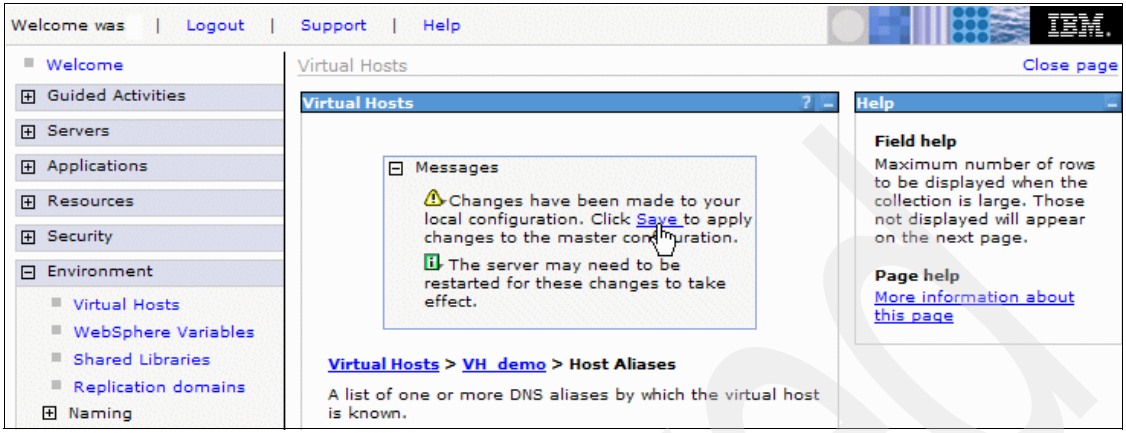


Figure 11-8 Saving the configuration - step 1

9. On the save page, check **Synchronize changes with Nodes** and click the **Save** button, as shown in Figure 11-9.

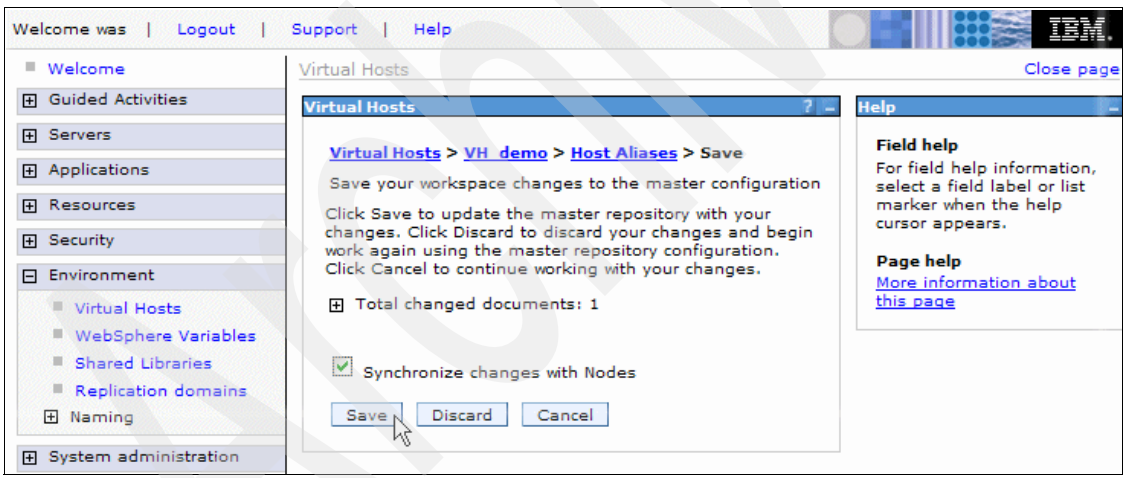


Figure 11-9 Virtual Hosts

The new Web server configuration is now complete and saved.

10. Propagate the IBM HTTP Server Plug-in configuration to the new Web server as explained for Web server node 1 in ""Propagate IBM HTTP Server Plug-in configuration" on page 183.

11. Start (or restart) the new Web server as explained for Web server node 1 in "Restart the Web server" on page 184.

The store and administration pages of your WebSphere Commerce application should now be accessible through the new Web server node. Refer to the following sections for instructions on how to configure load balancing for all your Web servers using IBM WebSphere Edge Components Load Balancer.

11.2 Configure Load Balancer

We describe how you can configure Load Balancer and your Web servers to set up a Web server cluster for Load Balancer to balance and monitor.

11.2.1 MAC forwarding

We set up MAC forwarding as described in 5.2, “Load Balancer configuration: basic scenario,” in the *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, and add configuration steps specific to WebSphere Commerce.

Table 11-1 lists the host names and IP addresses of our nodes.

Table 11-1 Host names and IP addresses for MAC forwarding

Node/cluster	Host name	IP address
Web server node 1	srvb501.torolab.ibm.com	9.26.126.120
Web server node 2	srvb504.torolab.ibm.com	9.26.127.157
Load Balancer cluster	wcha.torolab.ibm.com	9.26.126.103

For the following setup, we assume that Load Balancer is installed on Web server 2 (collocation).

Configure the Load Balancer cluster using the GUI

The configuration can be done using the Load Balancer graphical user interface (**lbadmin**) or using the command-line interface (**dscontrol**). We first explain how to do it using the GUI, and later we show the commands (which give you the same result).

In order to send commands through the GUI or through the command line interface to Load Balancer, you need to start the component element that receives those commands and executes them.

1. Start the Dispatcher server in order to start configuring it. To do so, run the following command:

```
dsserver
```

2. Open the Load Balancer GUI by running the following command:

   ```
   lbadmin
   ```

   The Load Balancer GUI is a Java client that can also be installed on a client machine, so the administrator can work remotely.

3. When the Load Balancer administration tool comes up, right-click **Dispatcher** in the left pane and select **Connect to Host**, as shown in Figure 11-10.

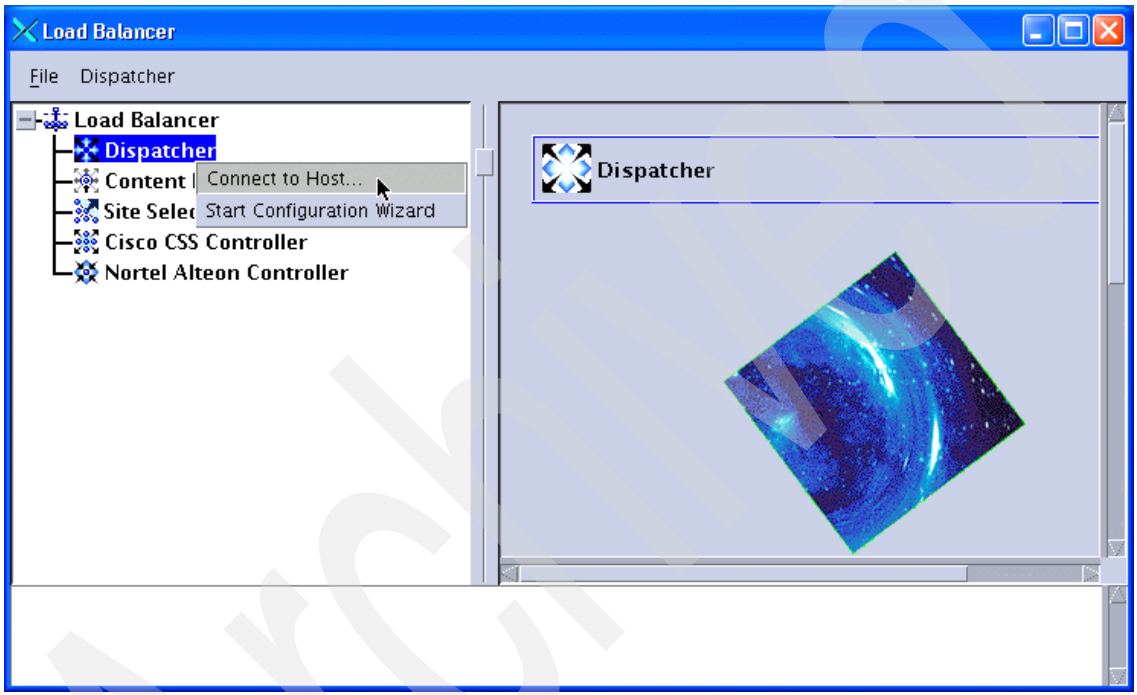


Figure 11-10 Load Balancer administration console

4. A pop-up window is displayed, prompting you for the Load Balancerserver that you want to connect to. Select the host name of the Load Balancer server, as shown in Figure 11-11.

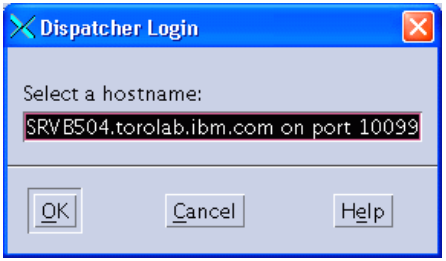


Figure 11-11 Selecting the Load Balancer server

After connecting to the Load Balancer server, a new entry is added to the GUI window in the left pane, containing the host name of the selected server. All the configuration we perform from now on is added to this element in a tree structure.

Archived

5. Now we need to start the Executor component, which is the component that actually distributes the load to the servers. Right-click **Host:srvb504.torolab.ibm.com** and select **Start Executor**, as shown in Figure 11-12.

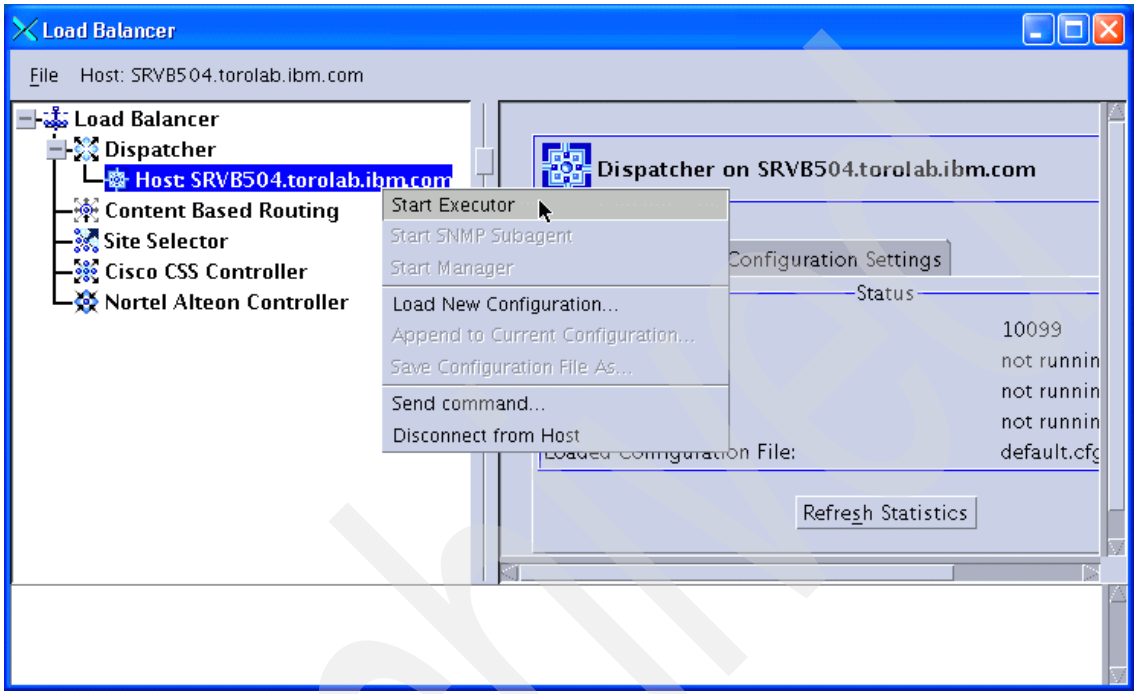


Figure 11-12 Starting the Executor

If Executor is started successfully, a new item named Executor is added to the left pane. In our scenario, the Load Balancer IP address is 9.26.126.103, so this IP address is shown in this new item as well.

> **Tip:** For every action that you perform, you can see a message in the bottom pane of the GUI window that confirms whether the action was performed successfully.

6. The next thing that we need to do is to add our cluster. In our scenario, we have a cluster called wcha.torolab.ibm.com (9.26.126.103),and this cluster contains two Web servers, srvb501.torolab.ibm.com (9.26.126.120) and srvb504.torolab.ibm.com (9.26.127.157).

Right-click **Executor: 9.26.127.157** and select **Add Cluster**, as shown in Figure 11-13.

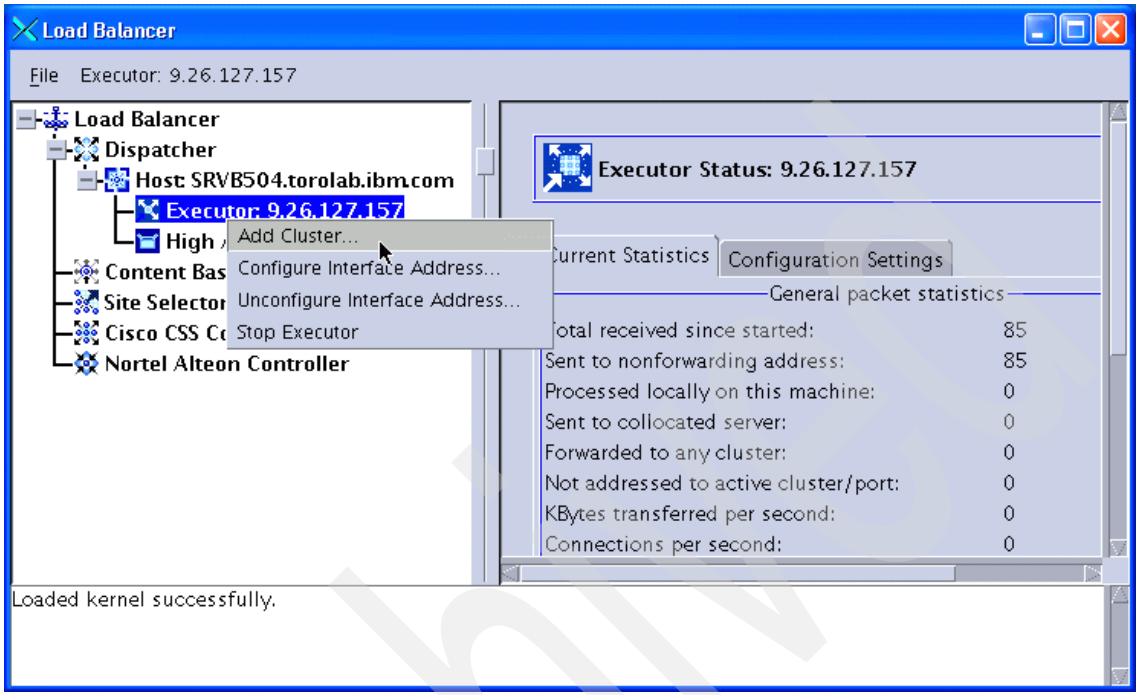


Figure 11-13 Adding a cluster

7. A new window is displayed, prompting for the necessary information to add the new cluster. Type the name of the cluster in the Cluster field (we recommend using the host name). Then type the cluster IP address in the Cluster address field, and make sure that the Load Balancer's IP address (which in our scenario is the same address as that of Web server node 2) is selected in the Primary host for the cluster field.

Check the option **Configure this cluster?**, as shown in Figure 11-14. This option is used to create an IP alias in the operating system for the cluster IP address. You can also uncheck this option and add the IP alias manually using operating system tools or commands.

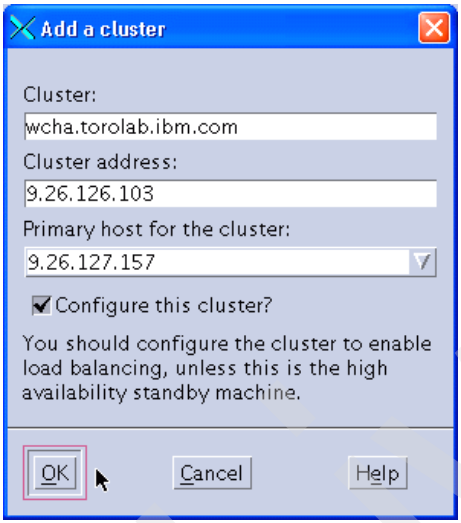


Figure 11-14 Filling in the information for adding a cluster

8. If you checked the Configure this cluster? check box, another window is displayed. Enter the interface identification in the Interface name field (in our server the interface that is associated with the IP address 9.26.126.103 is en0) and the network mask in the Netmask field, as shown in Figure 11-15.

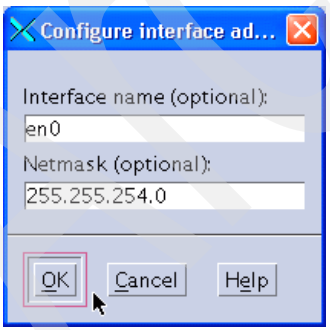


Figure 11-15 Configuring the interface

A new item that identifies your cluster is added to the left pane of the GUI.

Archived

9. Add each port that will be load balanced by Dispatcher. Right-click **Cluster: wcha.torolab.ibm.com** and select **Add Port**, as shown in Figure 11-16 on page 199.

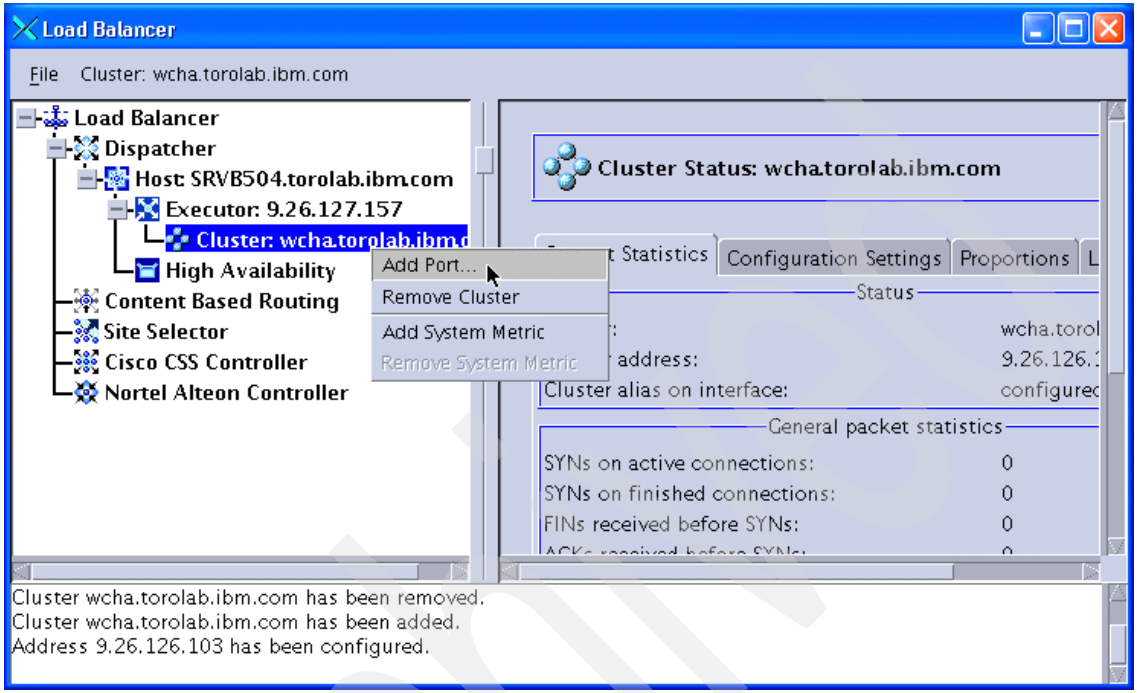


Figure 11-16 Adding a port

The port that we are adding refers to the port that the clients will access. For WebSphere Commerce, clients typically use ports 80 (HTTP) and 443 (HTTPS) to access the stores. You may also want to add the ports for the administrative consoles (8000, 8002, and so on) if you want console access to be load balanced. In our scenario, we use ports 80 and 443.

10. Fill in the number of the port in the Port number field and select **MAC Based Forwarding** in the Forwarding method field, as shown in Figure 11-17.

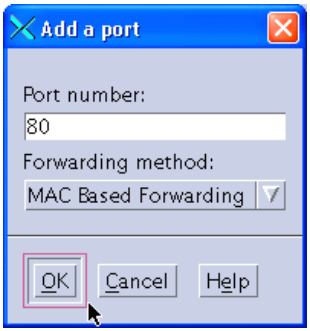


Figure 11-17 Entering port information

11. Add the servers that will receive the load for port 80 of cluster cluster.itso.ibm.com. Right-click **Port:80** and select **Add Server**, as shown in Figure 11-18.

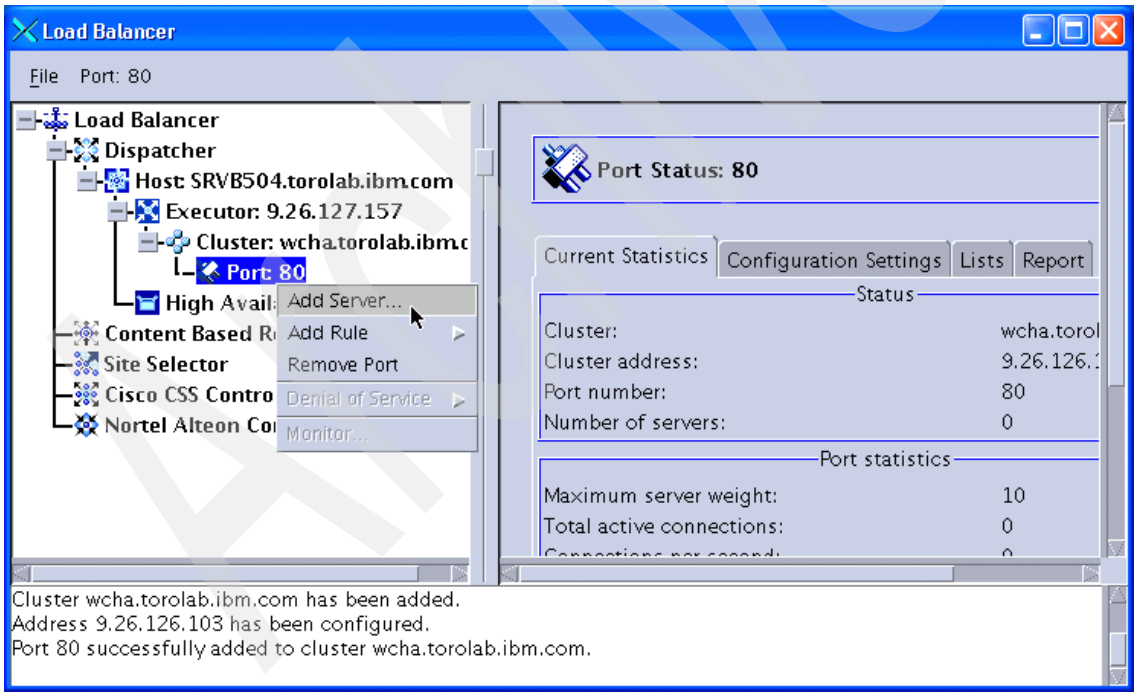


Figure 11-18 Adding a server to a port

The next window prompts you for the information of the first server. Fill in the host name of your Web server in the Server field and enter its IP address in the Server address field, as shown in Figure 11-19 on page 201, for Web server node 1 of our scenario.

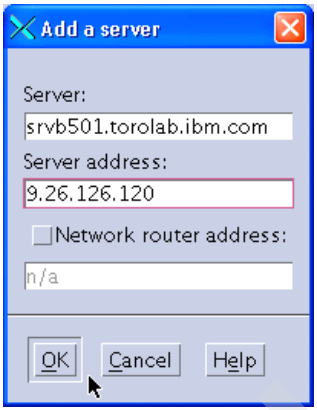


Figure 11-19 Entering balanced server information

Note that the Network router address check box is disabled because we selected MAC forwarding and this forwarding method does not allow load balancing to remote servers.

Repeat step 11 on page 200 for all servers in the cluster.

Repeat step 10 on page 200, including step 11 on page 200, for all load balanced ports.

Archived

After adding all ports and adding all servers to each port, the tree view for our scenario is displayed, as shown in Figure 11-20.

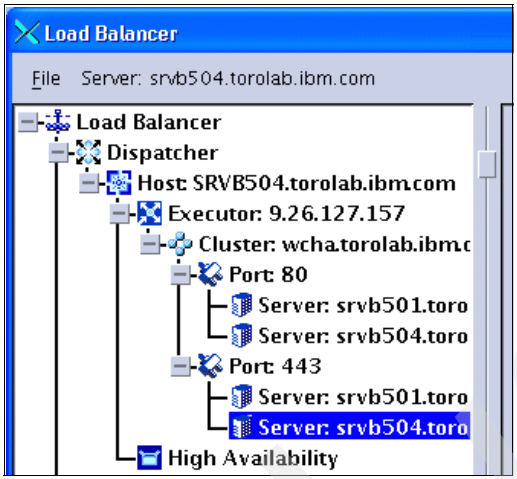


Figure 11-20 Configuration tree view after adding all ports and servers

The load balancing part of the configuration is done. All the information that Dispatcher needs to provide load balancing for our cluster is now configured. But we also need the Manager component because we want to work with dynamic weight values and failure detection.

Archived

12. Therefore, we now need to start the Manager component. Right-click **Host: srvb504.torolab.ibm.com** and select **Start Manager**, as shown in Figure 11-21.

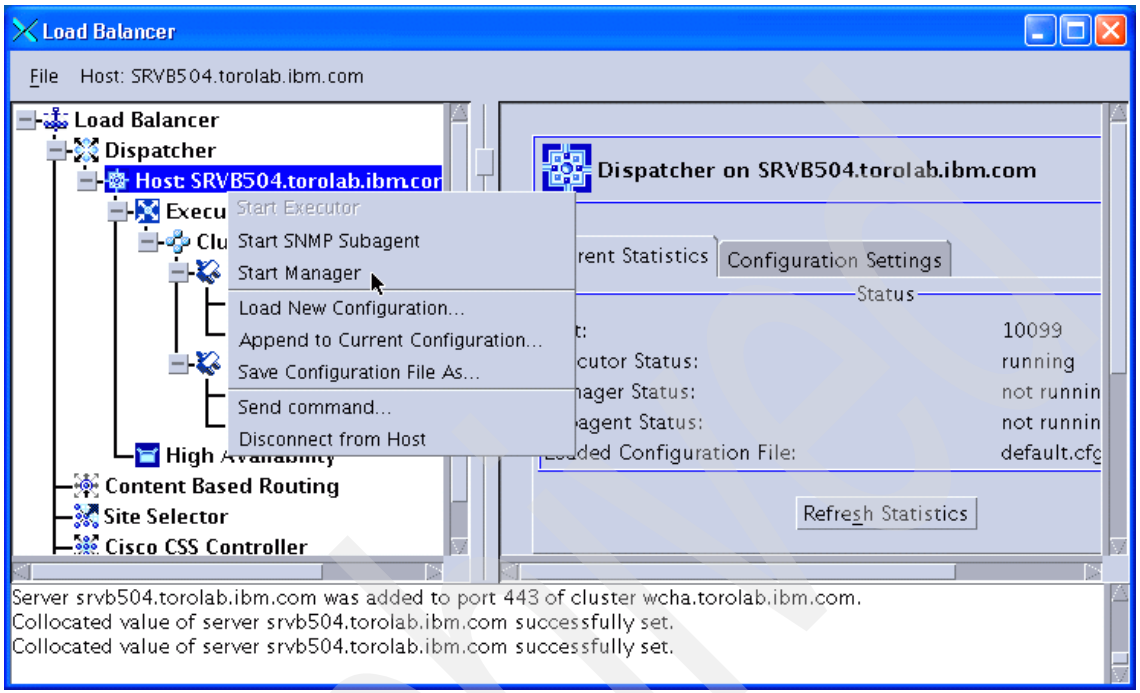


Figure 11-21 Starting the Manager

A window is displayed in which you can select the name of the Manager log file and the metric port, as shown in Figure 11-22 on page 203. We choose the default options.

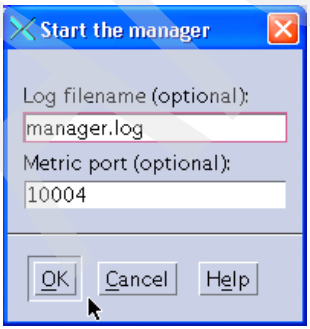


Figure 11-22 Manager options

The Manager needs advisors in order to generate a weight value based on the response time from each server in the cluster. The advisor is also needed in order to detect a failure in the service of any balanced server.

Due to the importance of the advisor, when you start Manager, the Load Balancer GUI automatically displays a pop-up window prompting you to start an advisor.

In our scenario, we are load balancing Web servers using the HTTP and HTTPS protocols. Therefore, we first use the default values, as shown in Figure 11-23, which are HTTP in the Advisor name field and 80 the Port number field.

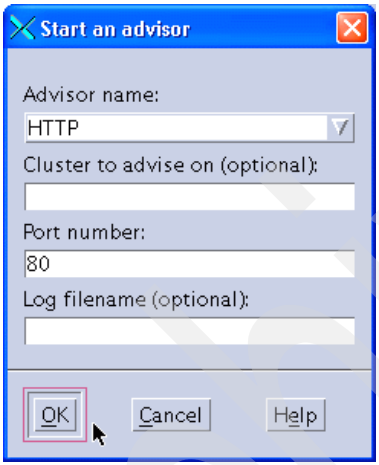


Figure 11-23 Starting the HTTP advisor

You can also choose a specific cluster with which to associate this advisor. By leaving the optional Cluster to advise on field blank, this advisor is automatically associated with all clusters that are load balancing port 80.

If you want to specify a log filename for this advisor, type in the desired name in the Log filename field. The default filename for the HTTP advisor is Http_80.log.

Archived

13. As we are also balancing the HTTPS port 443, we start an advisor for SSL by right-clicking **Manager** and selecting **Start advisor**, as shown in Figure 11-24.

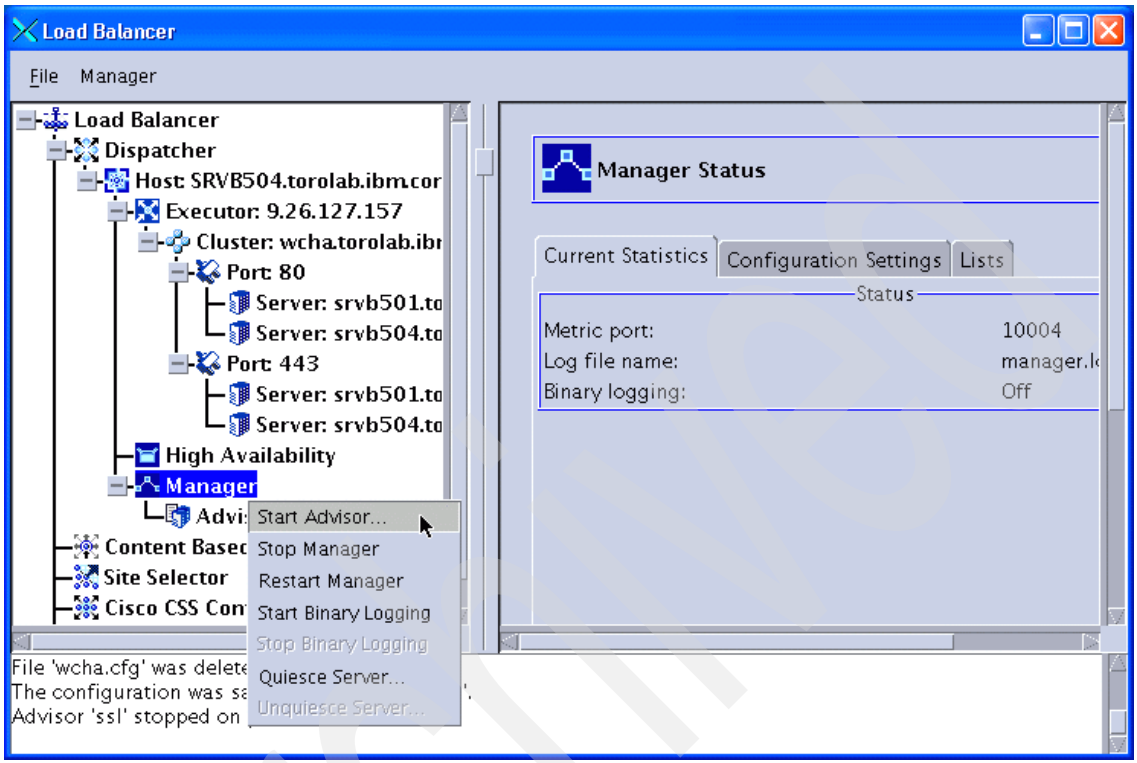


Figure 11-24 Starting an additional advisor

Archive

The pop-up window for specifying the advisor information is displayed again an we choose the SSL advisor, as shown in Figure 11-25.

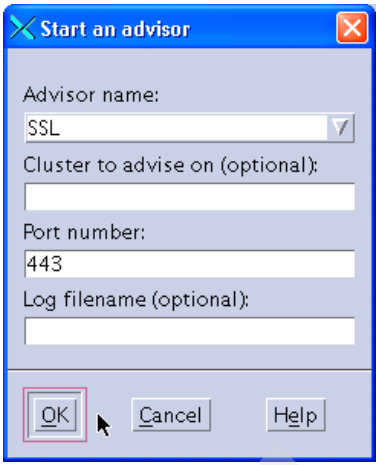


Figure 11-25 Starting the SSL advisor

Again, leaving the Cluster to advise on field blank, the advisor is associated with all clusters load balancing port 443. The default log file name for the SSL advisor is Ssl_443.log.

Note: The SSL advisor is a *lightweight* advisor for SSL connections. It does not establish a full SSL socket connection with the server. The SSL advisor opens a connection, sends an SSL CLIENT_HELLO request, waits for a response, closes the connection, and returns the elapsed time as a load. There is also an HTTPS advisor. It is a *heavyweight* advisor for SSL connections. It performs a full SSL socket connection with the server. The HTTPS advisor opens an SSL connection, sends an HTTPS request, waits for a response, closes the connection, and returns the elapsed time as a load. For more information about advisors, refer to *Load Balancer Administration Guide,* GC31-6858.

14. If Load Balancer is collocated with one of the Web servers, we need to change the server configuration. This is the case in our scenario where Load Balancer is collocated with Web server node 2, srvb504.torolab.ibm.com.

Archived

Select **Server: srvb504.torolab.ibm.com** under Port: 80. On the right pane, select the **Configuration settings** tab and select **yes** in the Collocated drop-down box, as shown in Figure 11-26.

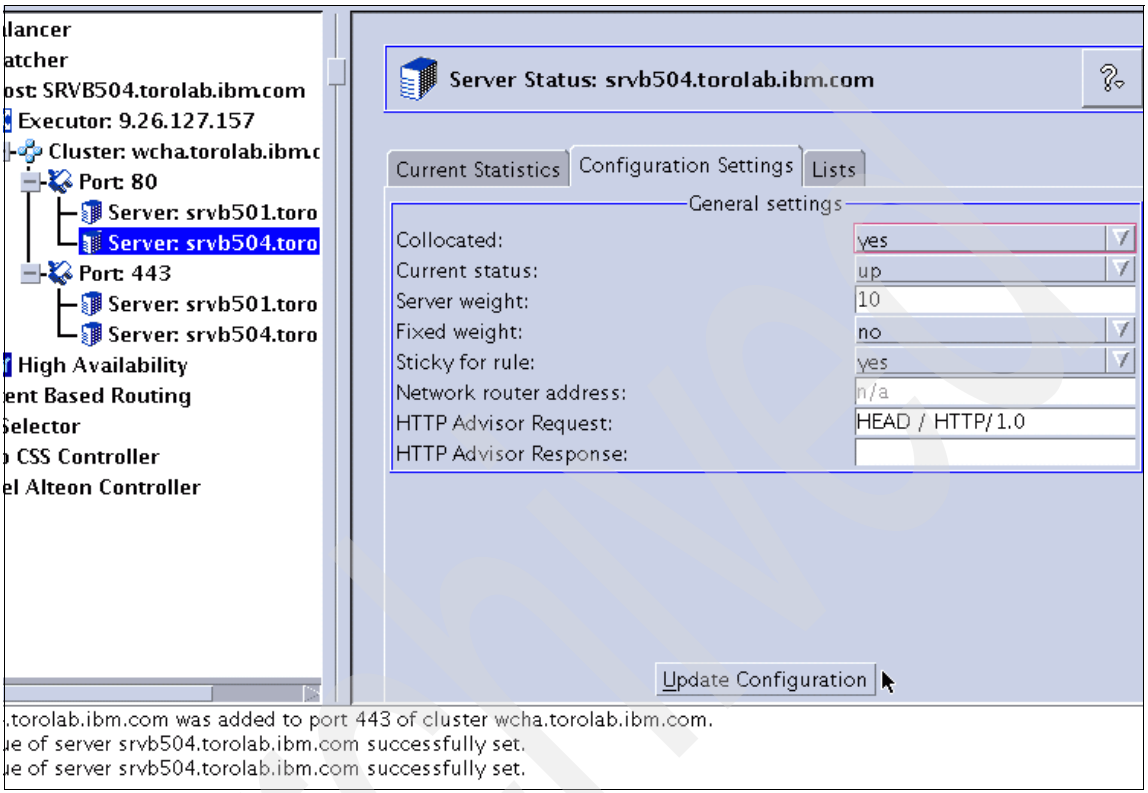


Figure 11-26 Configuring collocation

After making the change, do not forget to click the **Update Configuration** button at the bottom. (If you do not click the button and navigate away from the view, your changes are not saved.)

Repeat step Figure 14 on page 206 for all occurrences of the collocated server under all ports.

We have concluded the basic load balancing configuration, so we should save the configuration performed so far.

15. Right-click **Host: srvb504.torolab.ibm.com** and select **Save Configuration File As**, as shown in Figure 11-27.

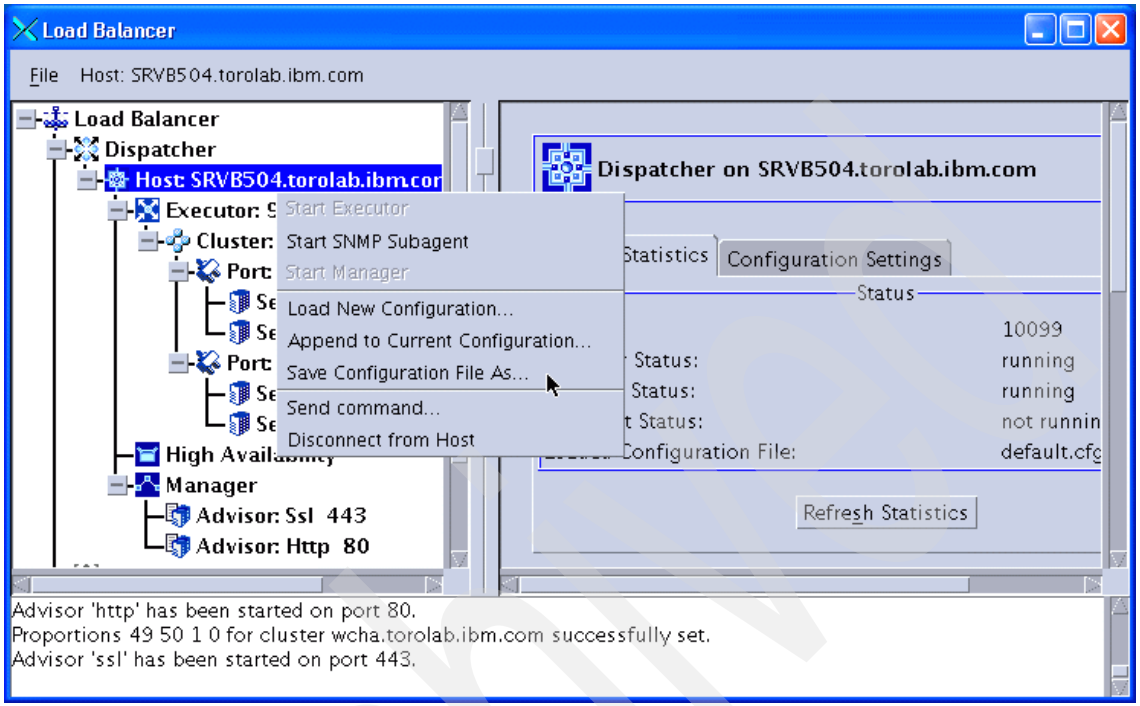


Figure 11-27 Saving the Load Balancer configuration

A pop-up window is displayed (Figure 11-28). In the Filename field, you can either select an existing configuration file (which will be overwritten) or you can enter a new filename.

The filename default.cfg is the default name for Load Balancer. This means that when you start the Dispatcher server (dsserver), it will look for the file default.cfg and, if it exists, it will load it. default.cfg is stored in *<LB_Install_Dir>*/servers/configurations/dispatcher.

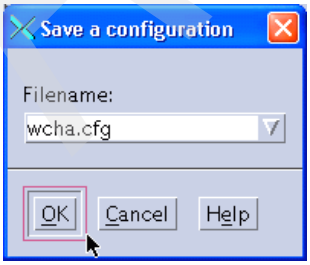


Figure 11-28 Choosing the configuration filename

Command-line configuration

The configuration file, which has been saved after configuring the cluster using the administration GUI, is shown in Example 11-2. It contains the commands for configuring the same cluster as described above. Note that each individual command has to be on one line in the configuration file. However, because of size limitations, some lines might be printed on two lines in our examples.

Example 11-2 Cluster configuration commands

```
dscontrol set loglevel 1
dscontrol executor start

dscontrol cluster add wcha.torolab.ibm.com address 9.26.126.103
primaryhost 9.26.127.157

dscontrol executor configure 9.26.126.103 en0 255.255.254.0

dscontrol port add wcha.torolab.ibm.com:80 reset no

dscontrol server add wcha.torolab.ibm.com:80:srvb504.torolab.ibm.com
address 9.26.127.157
dscontrol server set wcha.torolab.ibm.com:80:srvb504.torolab.ibm.com
collocated y

dscontrol server add wcha.torolab.ibm.com:80:srvb501.torolab.ibm.com
address 9.26.126.120

dscontrol port add wcha.torolab.ibm.com:443 reset no

dscontrol server add wcha.torolab.ibm.com:443:srvb504.torolab.ibm.com
address 9.26.127.157
dscontrol server set wcha.torolab.ibm.com:443:srvb504.torolab.ibm.com
collocated y

dscontrol server add wcha.torolab.ibm.com:443:srvb501.torolab.ibm.com
address 9.26.126.120

dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start Ssl 443 Ssl_443.log
```

Configure Web servers for MAC forwarding

When using MAC forwarding with Load Balancer, traffic for all ports is routed through the primary Load Balancer. Requests will therefore carry the Load

Archived

Balancer cluster IP address as destination. To be able to handle these requests, the Web servers need to be configured as follows.

- Add the Load Balancer cluster IP address as non-advertising alias address to the loopback interface.
- Modify the configuration for WebSphere Commerce instances in httpd.conf.

On AIX, the loopback interface is aliased using the **ifconfig** command. For Web server node 1 in our scenario, the command is issued as follows:

```
# ifconfig lo0 alias 9.26.126.103 netmask 255.255.255.255
```

Important: Make sure you use the netmask option with the netmask 255.255.255.255 when you are adding an IP alias to the loopback device in AIX. If you use an incorrect netmask, or do not use this option at all. A new route will be added to your routing table using the loopback as the gateway, which causes routing problems.

Add this command to the end of the /etc/rc.net file so that it will be run every time the networking configuration is run (for example, after a system reboot).

For examples of how to set up the loopback alias on other operating systems, refer to the *Load Balancer Administration Guide,* GC31-6858. In any case, after adding the alias, check for extra routes that may have been created, and remove them according to the correct procedure for each operating system.

11.2.2 NAT forwarding

In this scenario, we use the same two Web servers that we worked with in 11.2.1, “MAC forwarding” on page 193. However, we use a Load Balancer in a different network, which is installed on a Solaris system. We also add a third Web server, which is in a geographically remote network and in a different domain.

We now use Network Address Translation (NAT) as the forwarding method instead of MAC forwarding. We assume an unconfigured Load Balancer for this scenario. So if you have set up MAC forwarding and plan to use the same machine, then you need to delete the existing configuration. You also need an additional IP address in the same subnet as your cluster address, which is used as the return address.

Archive

Table 11-2 lists the host names and IP addresses of our nodes.

Table 11-2 Host names and IP addresses for NAT forwarding

Node/cluster	Host name	IP address
Web server node 1	srvb501.torolab.ibm.com	9.26.126.120
Web server node 2	srvb504.torolab.ibm.com	9.26.127.157
Web server node 3	m106958f.itso.ral.ibm.com	9.42.171.83
Load Balancer	pistons.torolab.ibm.com	9.26.52.215
Load Balancer cluster	nat1.torolab.ibm.com	9.26.52.154
Load Balancer return addr.	(nat2.torolab.ibm.com)	9.26.52.156

When using Solaris on the Load Balancer machine, install Load Balancer as described in 8.6, “Install Load Balancer” on page 140.

Important: On Solaris 9, we had to install Version 6.0.2.58 of Load Balancer to prevent the machine from crashing when accessing it from a remote browser after completing the configuration. Version 6.0.2.58 can be directly installed on top of the 6.0 license file as described for Version 6.0.2 in 8.6, “Install Load Balancer” on page 140. If you plan to install it on top of an existing installation, you need to remove all packages of the existing installation except the license first.

Configure the Load Balancer cluster using the GUI

To configure NAT forwarding:

1. Start dsserver, connect to the Dispatcher server, and start Executor, as explained in steps 1 on page 193 through 5 on page 196.

2. After starting Executor, click **Executor: 9.26.52.215** in the left pane of the GUI window, and locate the Client gateway address field in the right pane (see Figure 11-29). This field needs to be filled in order to enable NAT/NAPT and CBR forwarding methods. This is the IP address of the gateway router that handles traffic from the cluster back to the client browser.

   In our scenario, we used the default gateway IP address in our configuration, which is 9.26.52.1.

   Important: We are using 9.26.52.1 because we use the same IP address for inbound and outbound traffic. In case you are not using a common IP address, you need to specify the IP address of the router that serves as the first hop on the way from the Load Balancer to the client.

   Enter the correct IP address and click **Update Configuration**.

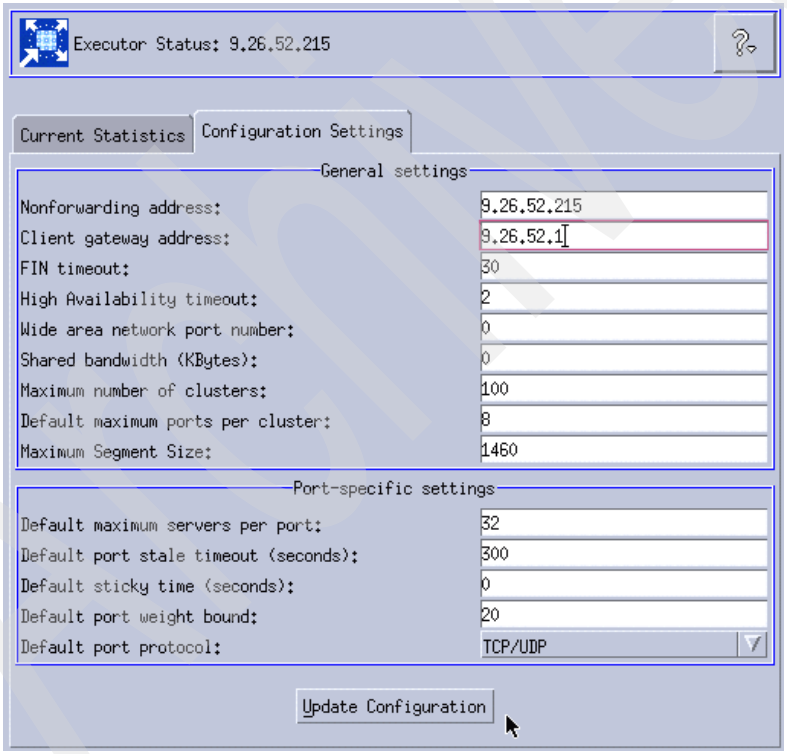


Figure 11-29 Configuring the client gateway address

3. After configuring the client gateway address, we need to configure the cluster. Right-click **Executor: 9.26.52.215** in the left pane of the GUI and select **Add Cluster**, as shown in Figure 11-30 on page 213.

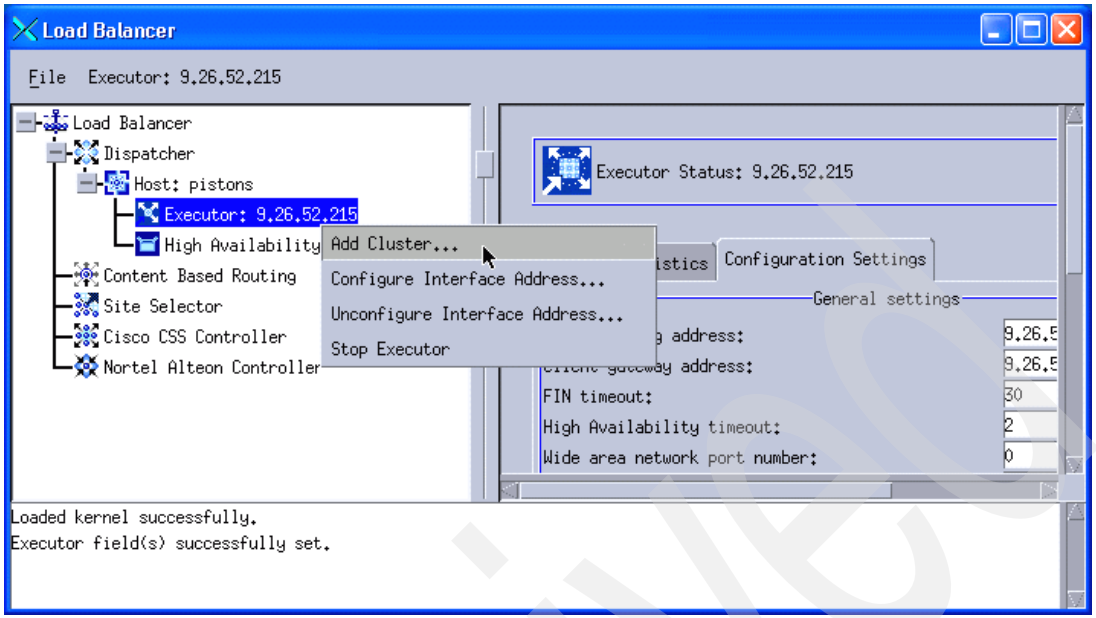


Figure 11-30 Add a cluster - NAT configuration

The information provided for the cluster creation is similar to what we used in the MAC scenario (see steps 6 on page 196 through 8 on page 198). Details are shown in Figure 11-31.

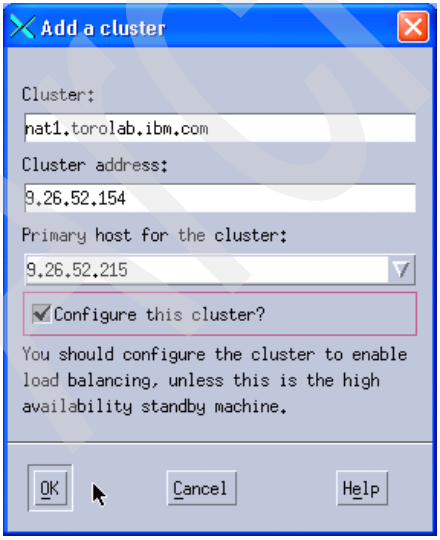


Figure 11-31 Cluster information

If you selected the Configure this cluster? check box, you need to also provide the information to add the IP alias, as shown in Figure 11-32. Note that on Solaris 9, the Ethernet network interface is called hme0.

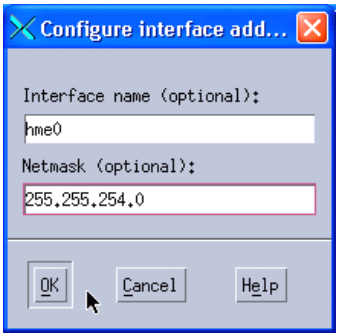


Figure 11-32 Configuring the interface

4. Right-click **Cluster: nat1.torolab.ibm.com** in the left pane and select **Add Port**, as shown in Figure 11-33.

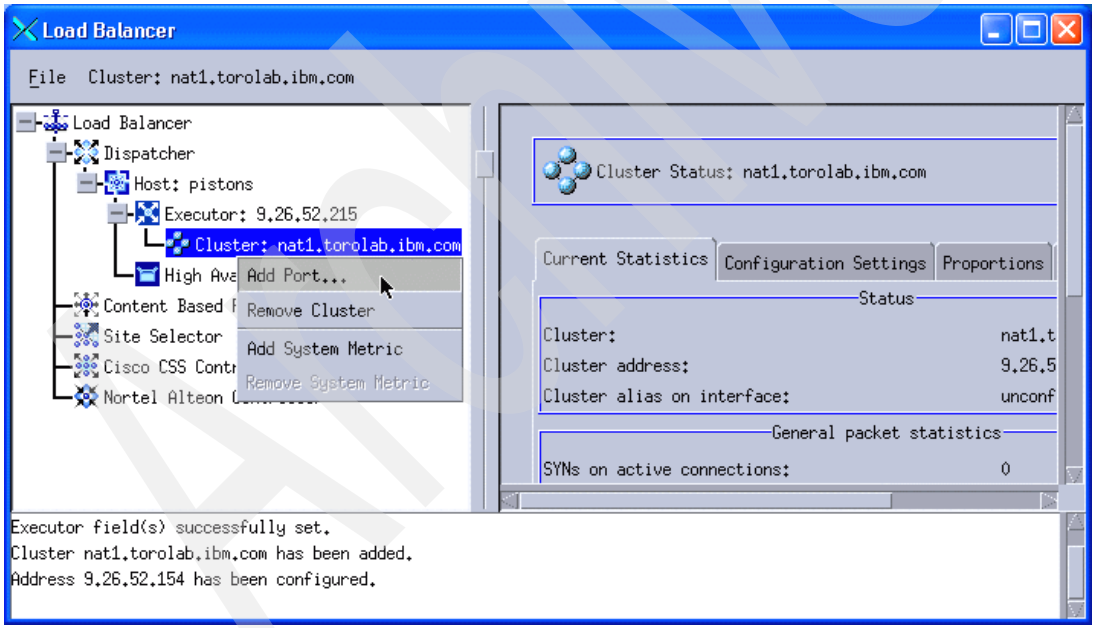


Figure 11-33 Add a port

Arrived

A pop-up window is displayed similar to the one that we had in the NAT scenario. Type 80 into the Port number field, select **NAT / NAPT** as the Forwarding method, and click **OK**. See Figure 11-34.

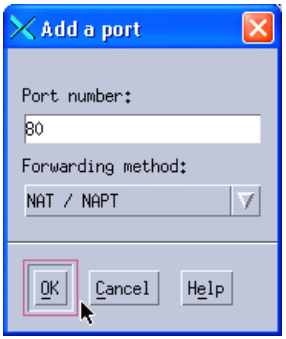


Figure 11-34 Adding a port - NAT forwarding method

5. Right-click **Port: 80** in the left pane and select **Add Server**, as shown in Figure 11-35.

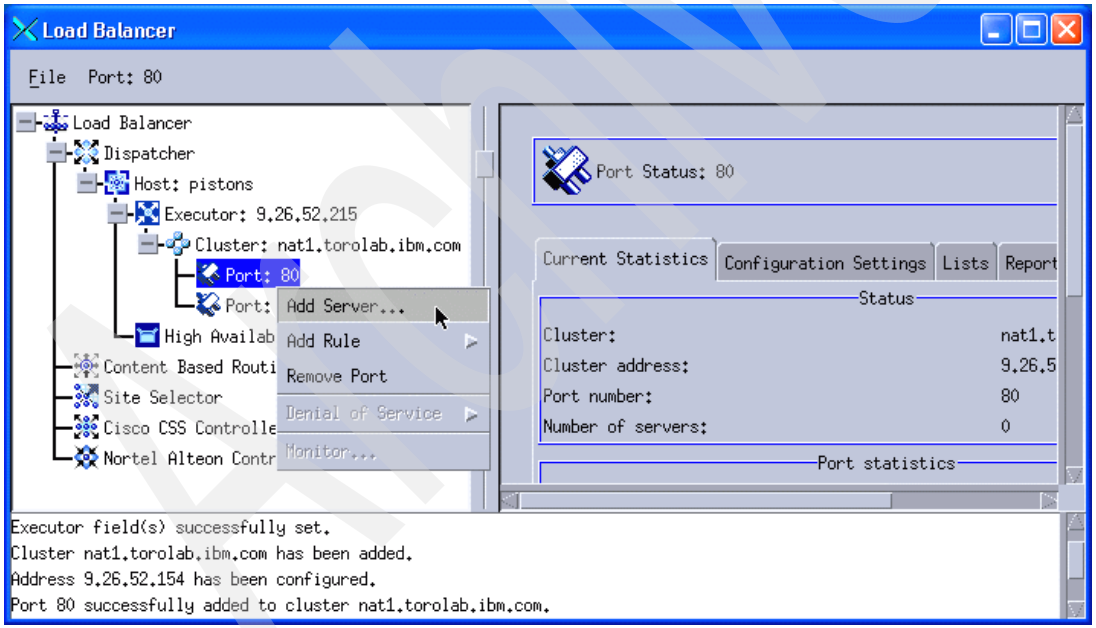


Figure 11-35 Add a server

A pop-up window is displayed, but it contains more fields compared to the one we got when configuring MAC forwarding.

The two extra fields are the return address and the network router address.

The return address is an extra IP address that is used by Dispatcher as the source IP address of the packets that are sent to the Web servers. You cannot use neither the cluster address or the non-forwarding address (NFA) as the return address, so you need one additional IP address for this configuration.

The network router address is the router that serves as the first hop on the way from the Load Balancer to the load balanced server. With our single subnet scenario, it is the same address as the client gateway. This field is provided in the server configuration in case you have several balanced servers spread in different remote networks, and you need a different router IP address to reach each server.

Note: When you configure NAT/NAPT on a collocated server, you need to use the local IP address as the network router address. This tells Dispatcher that the desired server is collocated on the local machine.

We added our first Web server, as shown in Figure 11-36. We used the same values that we used for MAC forwarding (see Figure 11-19 on page 201) and we also provided the IP address we selected as our return address and our default gateway IP address as our network router address.

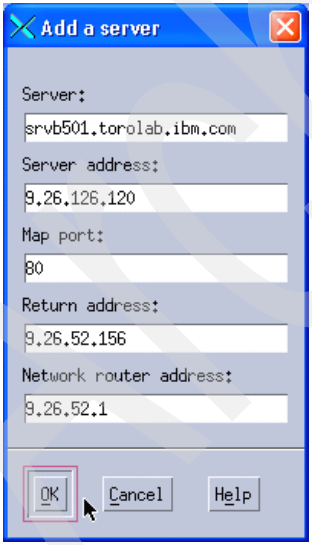


Figure 11-36 Adding the first balanced server

6. Add the second Web server. We used the same values for our second Web server as the ones we used for MAC forwarding, and we also used the same return address and the same network router address, as shown in Figure 11-37.

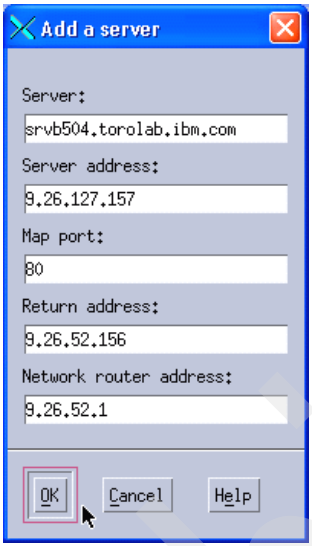


Figure 11-37 Adding the second balanced server

7. We also add a third server in a remote network, as shown in Figure 11-38.

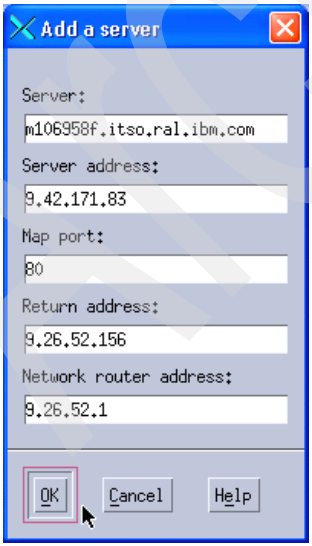


Figure 11-38 Adding the third balanced server

Archived

8. Repeat steps 4 on page 214 through 7 on page 217 for any other port for which you want to provide load balancing. We want to balance our store front (ports 80 and 443), but not the administration consoles (ports 8000, 8002, and so on), so we only add port 443 and add the three Web servers to it. Our final configuration is shown in Figure 11-40 on page 219.
9. Follow steps 12 on page 203 and 13 on page 205 to start the manager and advisors.
10. The last thing that you need to configure is to add the return address to the operating system, so it is able to handle the responses from the balanced servers.

    Important: It is sufficient to configure the return address if you are not in a collocated environment. If, however, your Load Balancer server is collocated, you need to perform additional steps. These differ for each operating system, so refer to Chapter 21, Section "Using collocated servers - Configuring server collocation with Dispatcher's NAT forwarding," in the *Load Balancer Administration Guide,* GC31-6858, for complete instructions for your operating system.

    In the GUI, right-click **Executor** and select **Configure Interface Address**. Enter the return address into the Interface address field (in our example this is 9.26.52.156), type the interface name into the optional Interface name field (in our example this is hme0), and enter the netmask into the Netmask field (in our example this is 255.255.254.0). See Figure 11-39.

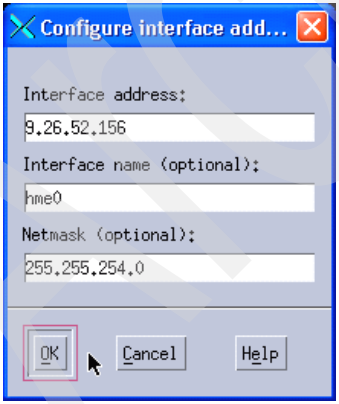


Figure 11-39 Configure the return address

11. Do not forget to save your configuration file as described in step 15 on page 208.

When the configuration is complete, the tree view in the left pane of the GUI should look as shown in Figure 11-40.

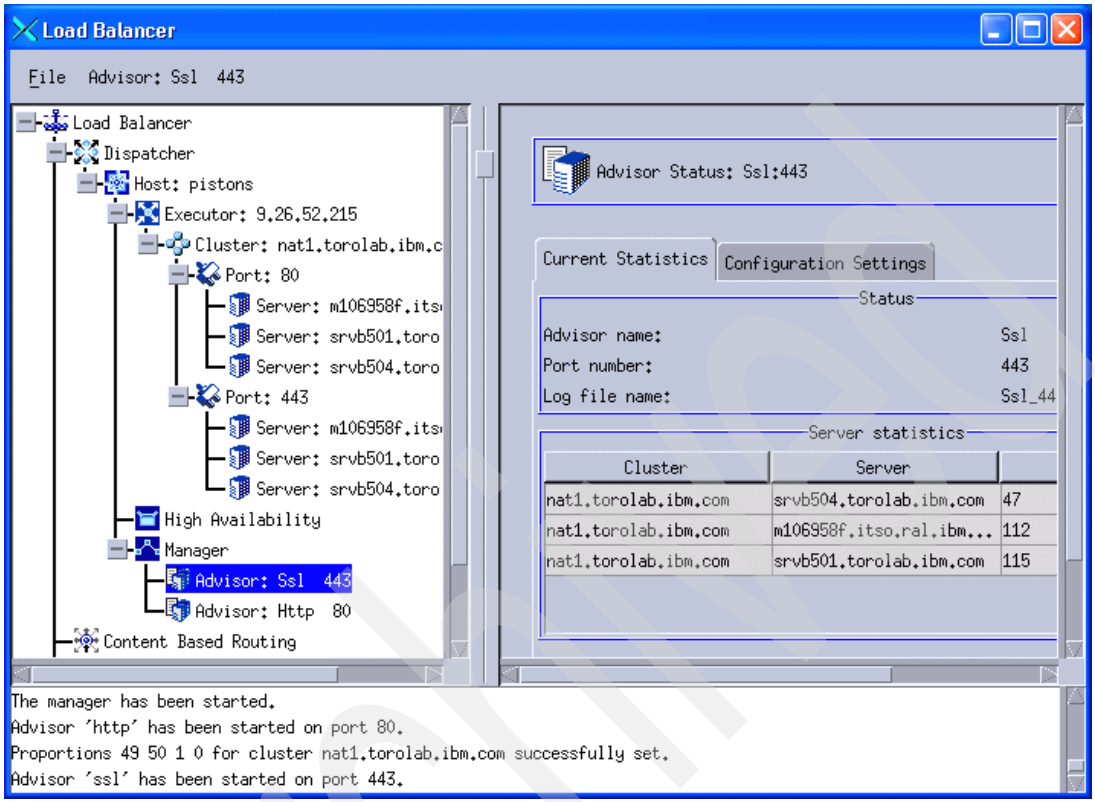


Figure 11-40 Complete NAT configuration

Command-line configuration

Example 11-3 shows the configuration file for this scenario. (Remember that each command must be on one line.)

Example 11-3 Configuration file for NAT forwarding

```
dscontrol set loglevel 1
dscontrol executor start
dscontrol executor set clientgateway 9.26.52.1

dscontrol cluster add nat1.torolab.ibm.com address 9.26.52.154 primaryhost
9.26.52.215
dscontrol executor configure 9.26.52.154 hme0 255.255.254.0

dscontrol port add nat1.torolab.ibm.com:80 method nat reset no
dscontrol port set nat1.torolab.ibm.com:80 porttype tcp
```

```
dscontrol server add nat1.torolab.ibm.com:80:srvb501.torolab.ibm.com address
9.26.126.120 router 9.26.52.1 returnaddress 9.26.52.156
dscontrol executor configure 9.26.52.156 hme0 255.255.254.0

dscontrol server add nat1.torolab.ibm.com:80:srvb504.torolab.ibm.com address
9.26.127.157 router 9.26.52.1 returnaddress 9.26.52.156
dscontrol executor configure 9.26.52.156 hme0 255.255.254.0

dscontrol server add nat1.torolab.ibm.com:80:m106958f.itso.ral.ibm.com address
9.42.171.83 router 9.26.52.1 returnaddress 9.26.52.156
dscontrol executor configure 9.26.52.156 hme0 255.255.254.0

dscontrol port add nat1.torolab.ibm.com:443 method nat reset no
dscontrol port set nat1.torolab.ibm.com:443 porttype tcp

dscontrol server add nat1.torolab.ibm.com:443:srvb501.torolab.ibm.com address
9.26.126.120 router 9.26.52.1 returnaddress 9.26.52.156
dscontrol executor configure 9.26.52.156 hme0 255.255.254.0

dscontrol server add nat1.torolab.ibm.com:443:srvb504.torolab.ibm.com address
9.26.127.157 router 9.26.52.1 returnaddress 9.26.52.156
dscontrol executor configure 9.26.52.156 hme0 255.255.254.0

dscontrol server add nat1.torolab.ibm.com:443:m106958f.itso.ral.ibm.com address
9.42.171.83 router 9.26.52.1 returnaddress 9.26.52.156
dscontrol executor configure 9.26.52.156 hme0 255.255.254.0

dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start Ssl 443 Ssl_443.log
```

Note: You do not need to configure the balanced Web servers as described in “Configure Web servers for MAC forwarding” on page 209 when using the NAT/NAPT forwarding method. This step is only necessary when using the MAC forwarding method.

11.2.3 Configure the Web servers for WebSphere Commerce

Having set up Load Balancer to route requests to the Web servers, and, in case of MAC forwarding, having enabled the Web servers’ network interfaces to accept packets on the Load Balancer cluster IP address (by adding the loopback alias as described in “Configure Web servers for MAC forwarding” on page 209), we now need to configure the Web servers to listen to the cluster IP address.

After creating a WebSphere Commerce instance, the IBM HTTP Server configuration file, httpd.conf, typically contains Listen and VirtualHost statements (as shown in Example 11-4) for Web server node 1 *before* configuring load balancing.

Example 11-4 httpd.conf for Web server node 1 before load balancing

```
...

Listen srvb501.torolab.ibm.com:80
Listen srvb501.torolab.ibm.com:443
...

########## IBM WebSphere Commerce (Do not edit this section) ...
#Instance name : demo
<VirtualHost srvb501.torolab.ibm.com:80>
ServerName srvb501.torolab.ibm.com
Alias   /wcsstore "/usr/.../Stores.war"
Alias   /wcs "/usr/.../CommerceAccelerator.war"
</VirtualHost>
<VirtualHost srvb501.torolab.ibm.com:443>
SSLEnable
SSLClientAuth 0
ServerName srvb501.torolab.ibm.com
Alias   /wcsstore "/usr/.../Stores.war"
Alias   /wcs "/usr/.../CommerceAccelerator.war"
</VirtualHost>
...
```

We need to change the listen statements and the VirtualHost elements. To till be able to access the store directly through the Web servers, we used an asterisk (*) as the host name rather than duplicating the listen statements and VirtualHost elements. Example 11-5 shows the relevant lines of the resulting configuration, which can be used for *all* Web server nodes.

Example 11-5 httpd.conf for Web server node 1 with load balancing

```
...

Listen *:80
Listen *:443
...

########## IBM WebSphere Commerce (Do not edit this section) ...
#Instance name : demo
<VirtualHost *:80>
```

Archived

```
ServerName srvb501.torolab.ibm.com
Alias   /wcsstore "/usr/.../Stores.war"
Alias   /wcs "/usr/.../CommerceAccelerator.war"
</VirtualHost>
<VirtualHost *:443>
SSLEnable
SSLClientAuth 0
ServerName srvb501.torolab.ibm.com
Alias   /wcsstore "/usr/.../Stores.war"
Alias   /wcs "/usr/.../CommerceAccelerator.war"
</VirtualHost>
...
```

After changing httpd.conf on all load-balanced Web servers, the Web servers have to be restarted (see “Restart the Web server” on page 184).

11.2.4 Configure the IBM HTTP Server Plug-in

In order for WebSphere Application Server to handle requests to the Load Balancer cluster, the IBM HTTP Server Plug-in has to be configured on all Web servers such that it accepts the Load Balancer cluster host name on the load-balanced ports.

All we have to do to change the plug-in configuration on all Web servers is to add the virtual host aliases to the relevant virtual hosts. In our scenario, only store access is load balanced, so the only virtual host that needs new aliases for the Load Balancer cluster is VH_<*Instance_Name*>.

We use the Network Deployment Manager administrative console to add the following host aliases to VH_<*Instance_Name*>:

- *:80
- *:443

Again, we use * as a shortcut. When using *, all other host aliases for the actual port may be removed. Of course, you may also add the two new host aliases using the your explicit Load Balancer cluster host name, without removing the existing ones.

To add the host names, use your browser and go to the Network Deployment Manager administrative console to perform the following steps:

1. In the main navigation on the left, select **Environment** → **Virtual Hosts**.
2. Click the **VH_<*Instance_Name*>** virtual host.
3. Click **Host Aliases** to show all aliases for VH_<*Instance_Name*>.

4. Click **New** and enter the new host alias for port 80, using * or the Load Balancer cluster host name, as shown in Figure 11-41, for instance name *demo*.

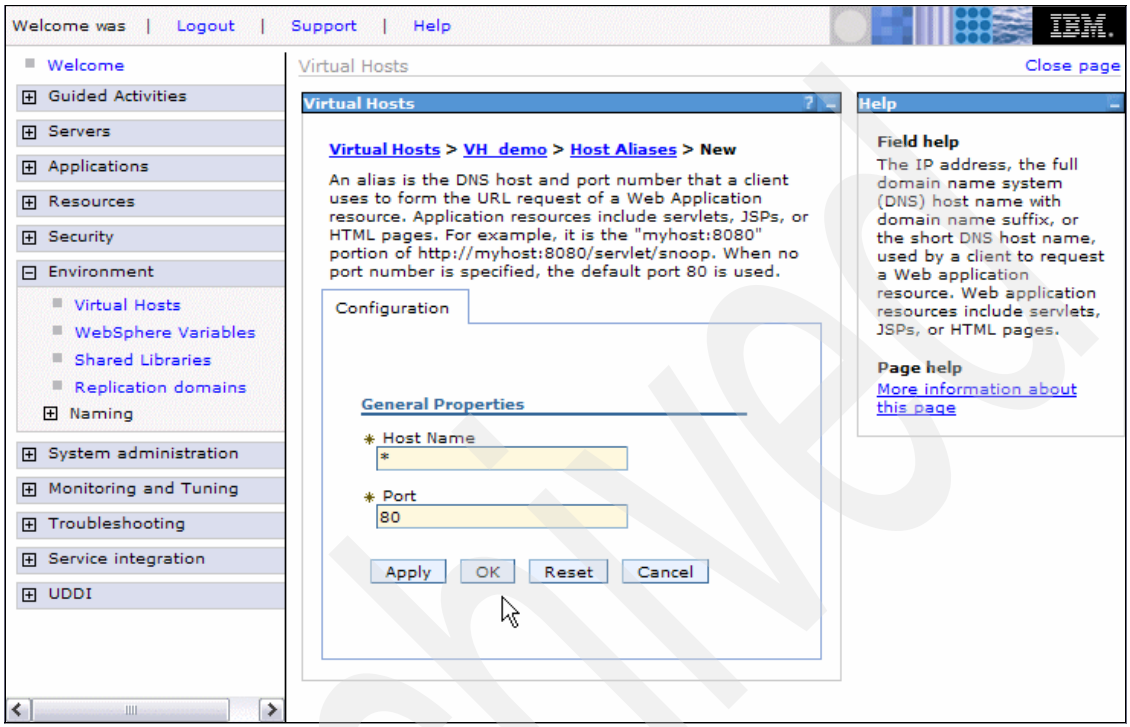


Figure 11-41 Adding a virtual host alias in Network Deployment Manager administrative console

Archive

5. Click **Save** (Figure 11-42).

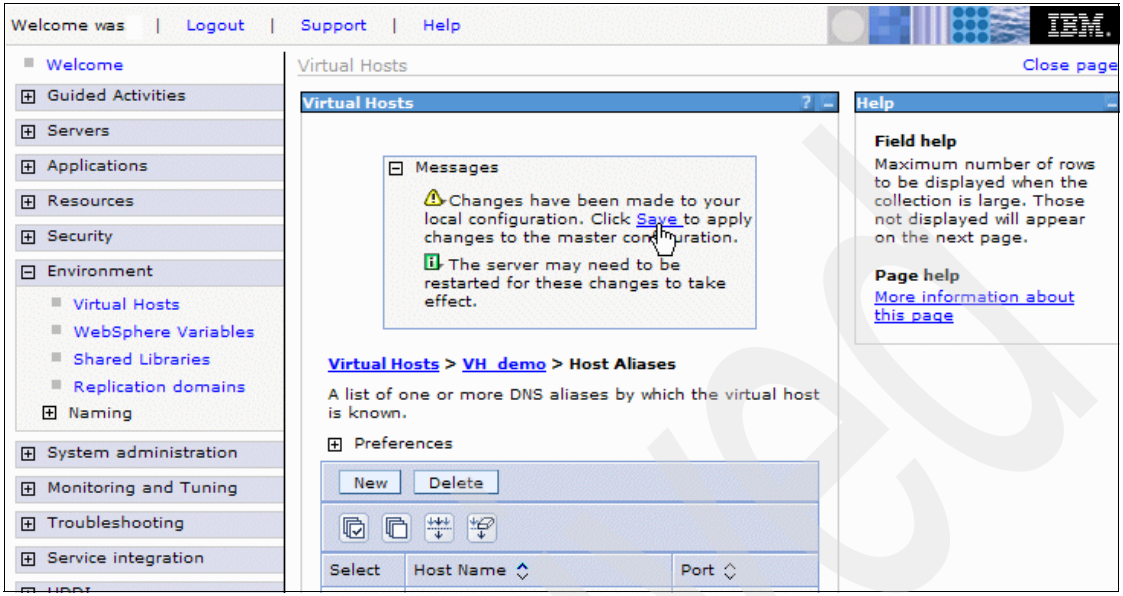


Figure 11-42 Going to the Save workspace changes dialog

6. In the Save workspace changes dialog, check **Synchronize changes with Nodes** and click **Save** (Figure 11-43).

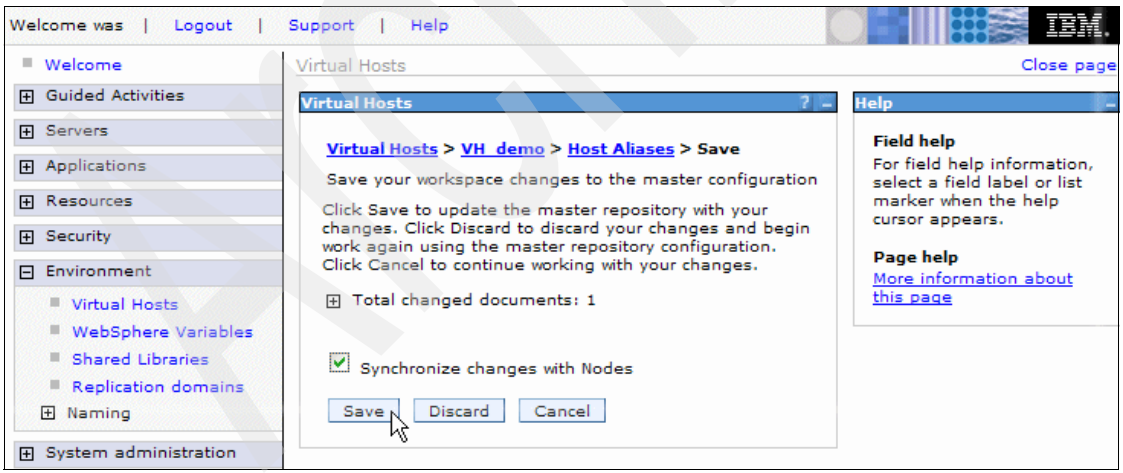


Figure 11-43 Save workspace changes dialog

7. If automatic generation and propagation of the IBM HTTP Server Plug-in configuration, as well as remote Web server management, are enabled (see

Figure 10-3 on page 177 and “Activate remote Web server management (optional)” on page 175), the plug-in configuration is automatically updated on all Web servers. If automatic propagation is not activated for a Web server, the plug-in has to be manually propagated to that Web server by performing the following steps:

a. Navigate to **Servers** → **Web servers** (Figure 11-44).

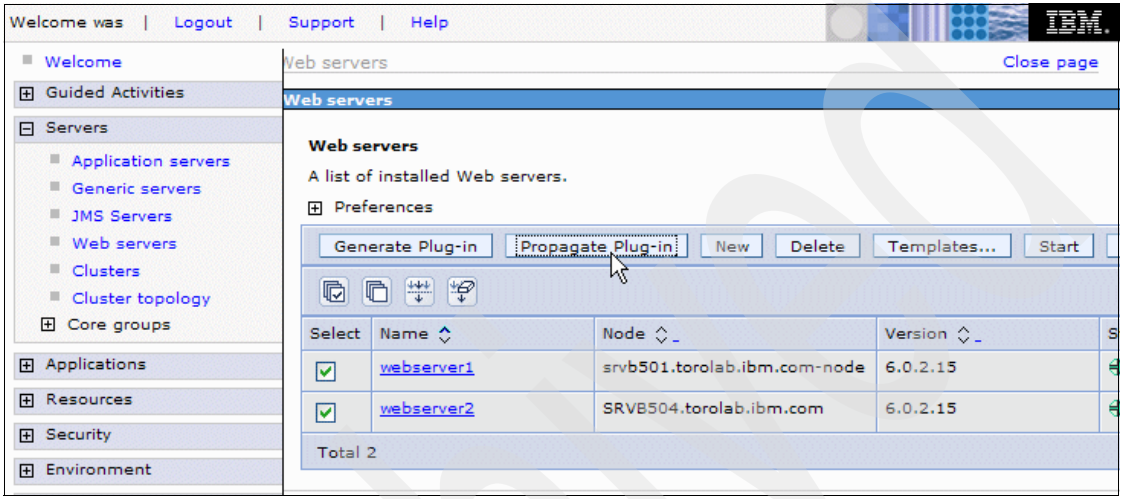


Figure 11-44 Propagating the Plug-in configuration

b. Under Select, check all Web servers for which automatic propagation is not activated.

c. Click **Generate Plug-in**, then **Propagate Plug-in**.

This still requires that remote management is set up for the Web server and that the IBM HTTP Server administrative server is running on the Web server (see above).

If remote management is disabled, the plug-in configuration file can still be automatically generated, but it needs then to be copied manually to the Web server. See “Propagate IBM HTTP Server Plug-in configuration” on page 183 for information about manual propagation.

The Load Balancer configuration is now complete and you should be able to access your store on ports 80 and 443 using the host name that resolves to your cluster IP address.

11.3 Configure Load Balancer High Availability

In this section we explain how to configure High Availability for the MAC forwarding scenario, but we also point out which settings are different for NAT forwarding.

We assume that one of the scenarios is already set up, and that the backup Load Balancer is installed identically to the primary server, as described in 8.6, “Install Load Balancer” on page 140, in the base installation chapter.

The step-by-step configuration instructions are based on the instructions in 5.3, “Load Balancer: High Availability scenario,” in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, which we adapted for load balancing a WebSphere Commerce instance.

11.3.1 Configure basic High Availability

When you are using the High Availability feature of Load Balancer, you do not use the **`dscontrol executor configure`** command in the Dispatcher configuration file. Leaving the command in the configuration would break the High Availability configuration because it adds the IP aliases to the network interface no matter what the state of the Load Balancer server is (active or standby). We need to make sure that the IP aliases are only added to the network interface when a server changes to the active state. For a High Availability configuration, we create scripts that control all IP aliases that need to be added or removed from the network interfaces and the loopback interface, as described in 11.3.4, “Configuring the High Availability scripts” on page 239.

First, we have to remove the cluster IP alias from the existing configuration before proceeding:

1. Open the Load Balancer GUI and connect to the primary server as described in steps 1 on page 193 through 4 on page 195. Make sure that the correct configuration is loaded by checking that the cluster, port, and servers are already configured.

Archived

2. Right-click **Executor** and select **Unconfigure Interface Address**, as shown in Figure 11-45, for the MAC configuration. For NAT, you need to unconfigure both the cluster address and the return address.

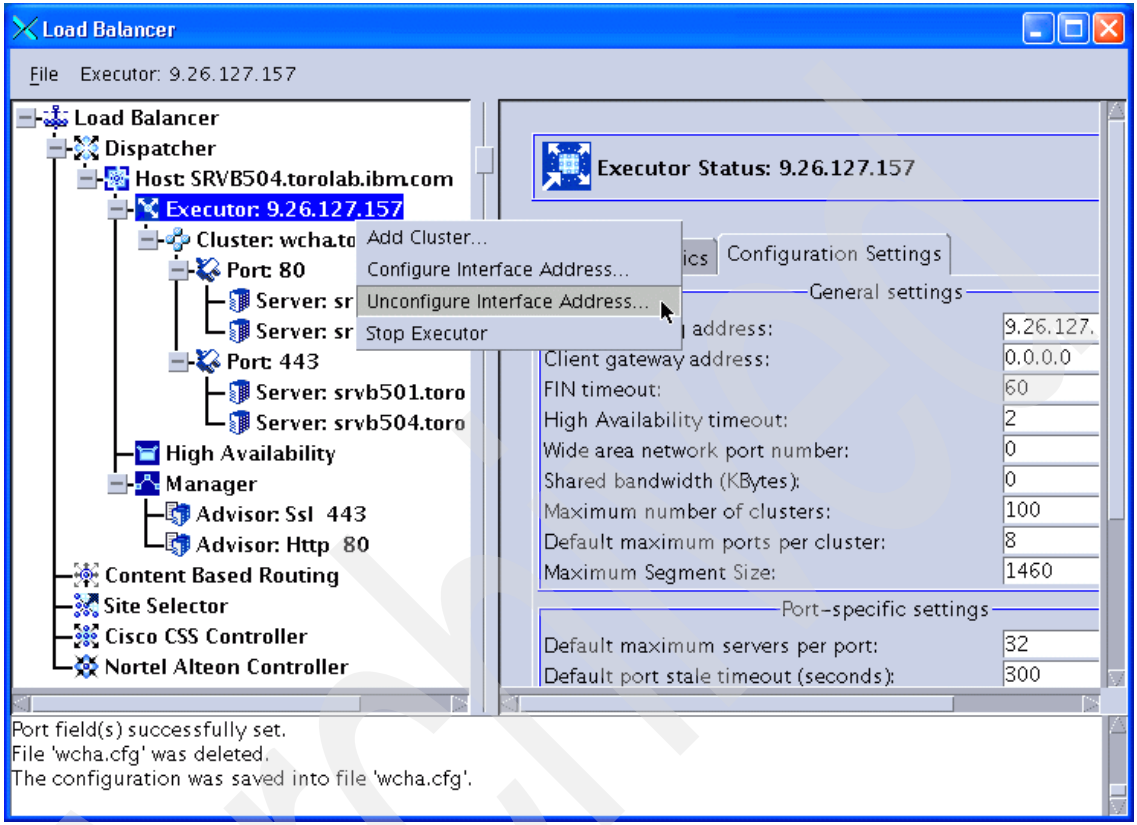


Figure 11-45 Unconfiguring the cluster address

A pop-up window is displayed. Enter the IP address of the cluster in the Interface address field. In our scenario, we want to remove the IP alias for the cluster address 9.26.126.103, as shown in Figure 11-46.

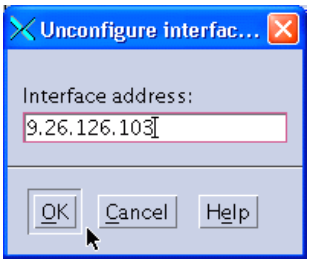


Figure 11-46 Specifying the IP address to unconfigure

3. Save the current configuration, then copy it to the Standby Load Balancer so that you do not need to set up the basic load balancing configuration there again.

Note: The menu choice **Host:... > Save Configuration File As...** prompts you for a filename, but with Load Balancer V6.0.2 on AIX it seems that the currently used configuration file is always overwritten, no matter which filename is specified. Therefore, we recommend making a backup of the file and then having it overwritten with the new configuration. The configuration files are stored in *LoadBalancer_Install_Dir*/servers/configurations/dispatcher on AIX and Solaris.

Archived

4. Now we need to add the High Availability configuration. Right-click **High Availability** in the left pane of the GUI window and select **Add High Availability Backup**, as shown in Figure 11-47.

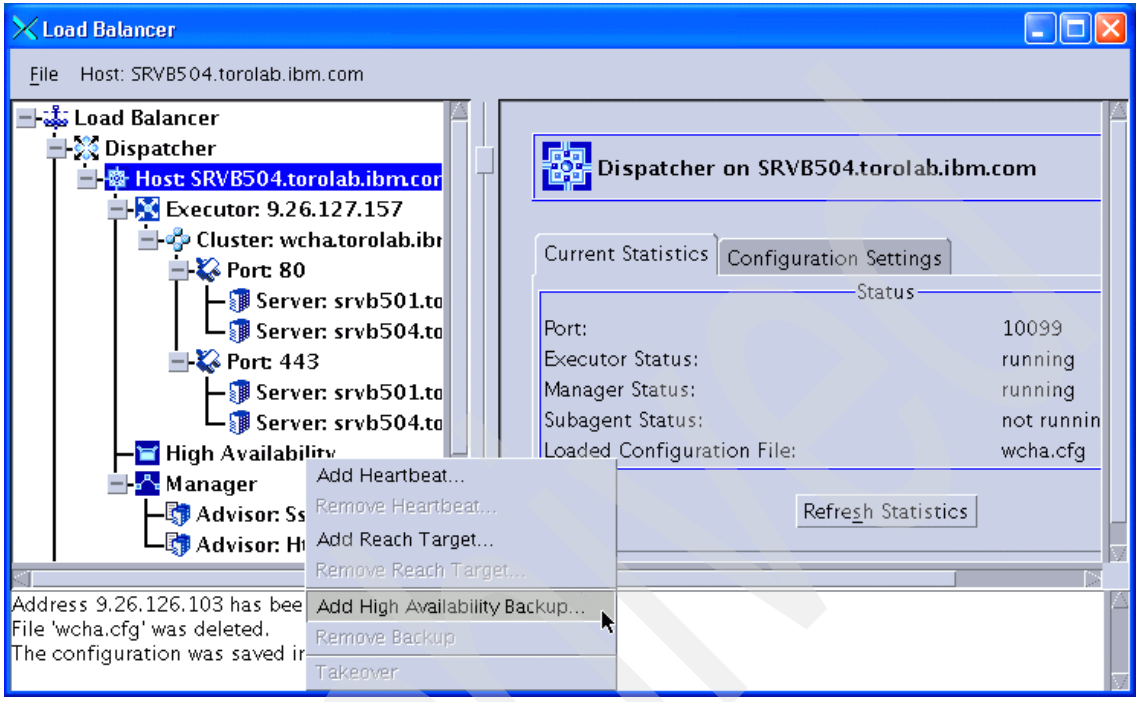


Figure 11-47 Starting the High Availability configuration

Archive

A new pop-up window is displayed, as shown in Figure 11-48.

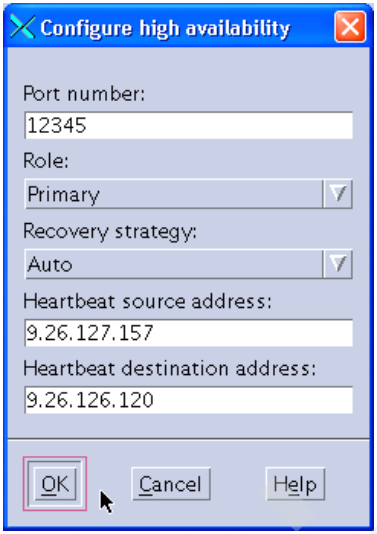


Figure 11-48 Configuring High Availability options

In order to monitor the health of the active server, heartbeats are sent every half second. The failover occurs when the standby server receives no response from the active server within two seconds.

a. Choose a port number that will be used by both Load Balancer servers to exchange the information needed to keep them synchronized and put it in the Port number field. You can choose any port—you just need to make sure that the port number matches on both servers.

b. In the Role field, select the role that this machine will have in the High Availability scenario (Primary, Backup, or Both). In our scenario, this machine is our primary machine, so we select **Primary**.

c. In the Recovery strategy field, choose how your primary machine is going to behave in case the backup machine has taken over. Select **Auto** for automatic takeover as soon as it is up and responding to the network. Select **Manual** for manual takeover (either by selecting **High Availability** → **Takeover** in the tree view of the Load Balancer GUI, or by running the command **`dscontrol high takeover`**).

d. The last thing that you need to add in this window is the heartbeat source address and the heartbeat destination address. The heartbeat is a Generic Route Encapsulation (GRE) packet that is sent from the local machine to the other server in the same High Availability cluster to make sure that it is responding. Enter the IP address of the local machine into

Archived

the Heartbeat source address field, and the IP address of the standby machine into the Heartbeat destination address field.

e. When you are finished, click **OK**.

f. For now, the configuration is done on the primary server. Click **High Availability** in the left pane, and you can see the High Availability status information in the right pane, as shown in Figure 11-49.

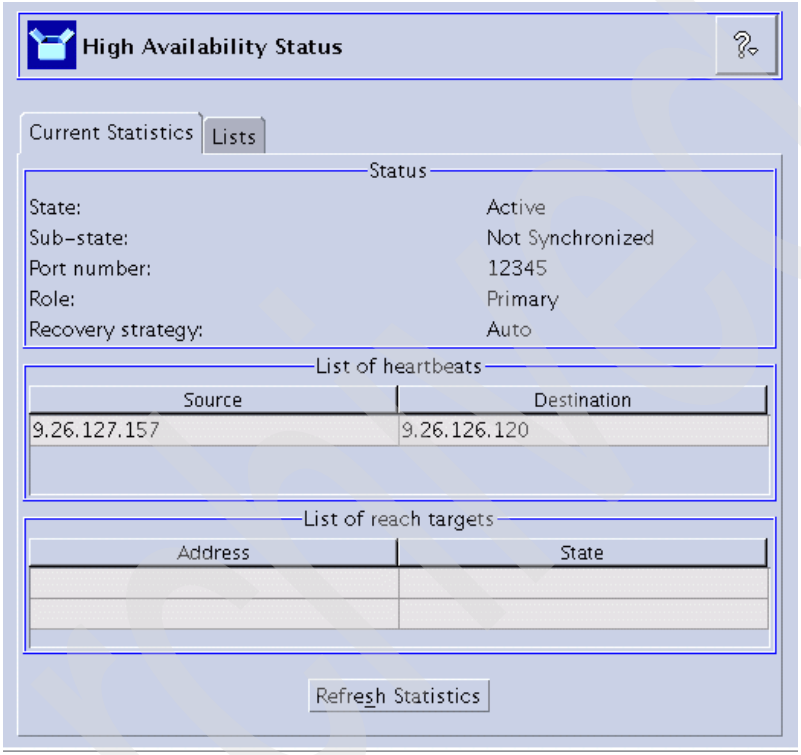


Figure 11-49 High Availability status

The next steps must be performed on the standby Standby Load Balancer server, which in our scenario is the server srvb501.torolab.ibm.com.

5. Open the Load Balancer GUI and connect to the backup server as described in steps 1 on page 193 through 4 on page 195. Select your backup server in the Dispatcher Login pop-up, as shown in Figure 11-50.

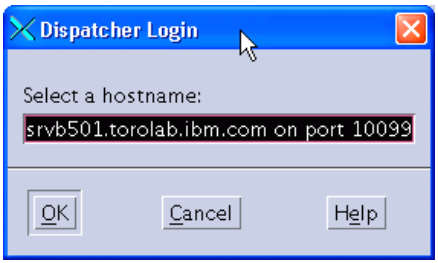


Figure 11-50 Connecting the GUI to a host

6. Load the configuration file that you copied from the primary server in step 3 on page 228. Right-click **Host: srvb501.torolab.ibm.com** and select **Load New Configuration**, as shown in Figure 11-51.

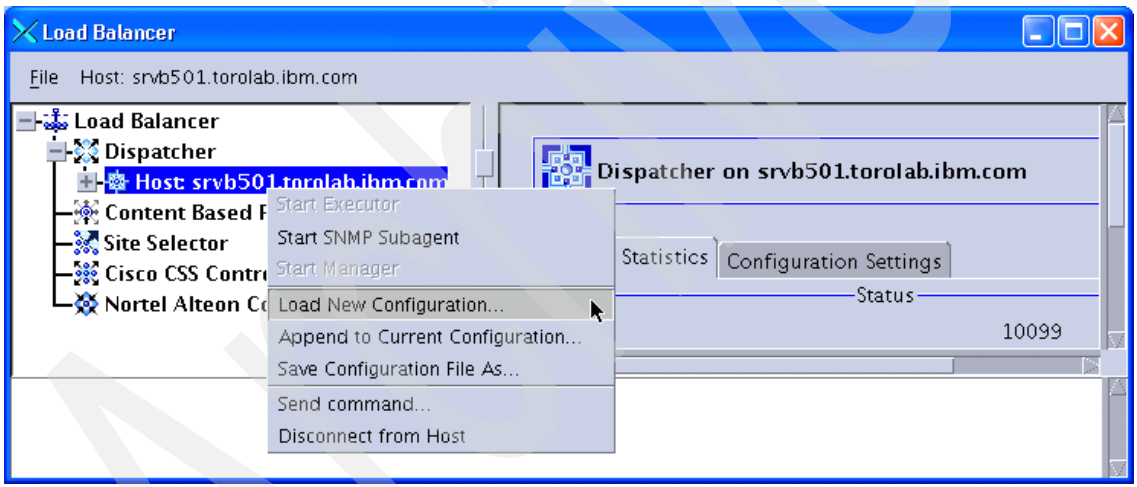


Figure 11-51 Loading a new configuration

Archived

Select the filename in the pop-up window and click **OK**. See Figure 11-52.

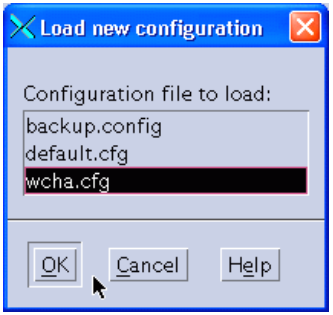


Figure 11-52 Selecting a configuration file

7. We now need to configure the balanced servers' collocation settings. For our scenario, we installed the primary Load Balancer on Web server node 2 and set the collocation flag to yes in step Figure 14 on page 206. On the Standby Load Balancer machine, we need to change this setting back to no.

   Similarly, if the Standby Load Balancer is installed on the same machine as one of the load-balanced Web servers, we need to set that server's collocation flag to yes. This is the case in our scenario where the Standby Load Balancer is installed on Web server node 1.

   The flag has to be changed for all occurrences of the respective balanced server under the balanced ports.

   Again, when changing a server's configuration, do not forget to click the **Update Configuration** button.

8. Add the High Availability information for the backup server. Right-click **High Availability** in the left pane of the GUI window and select **Add High Availability Backup** (this is the same procedure that we performed for the primary server in step 4 on page 229).

   You need to use the same parameters for Port number and Recovery strategy that you used in the configuration of the primary server (see Figure 11-27 on page 208).

   Select **Backup** in the Role field.

Abbrev

For the backup server, the heartbeat source address is the standby server itself, and the heartbeat destination address is the primary server, as shown in Figure 11-53.

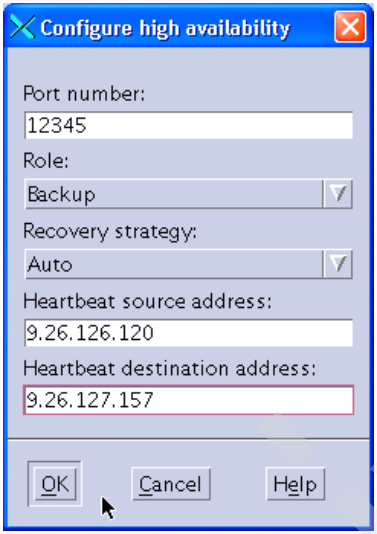


Figure 11-53 Configure High Availability

9. Save the configuration of the primary and backup servers.

Refer to 11.3.3, “Command-line configuration” on page 236, for the complete configuration files of both Load Balancer servers.

For both servers, when selecting High Availability in the tree view on the left side of the GUI, the High Availability status window should now show the correct states (active or standby) and sub-state (synchronized), as shown in Figure 11-54 and Figure 11-55 on page 235.

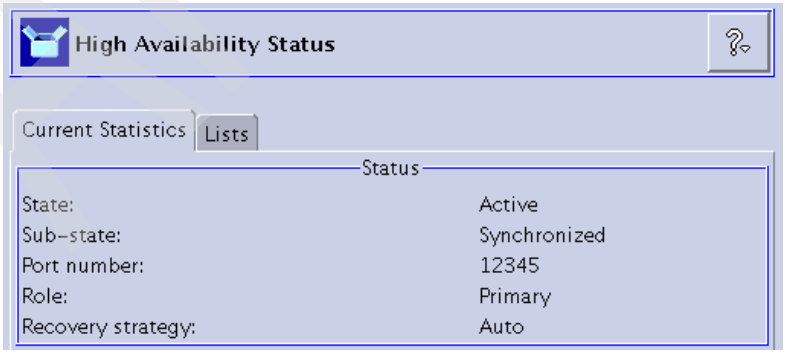


Figure 11-54 Primary Load Balancer status after High Availability configuration

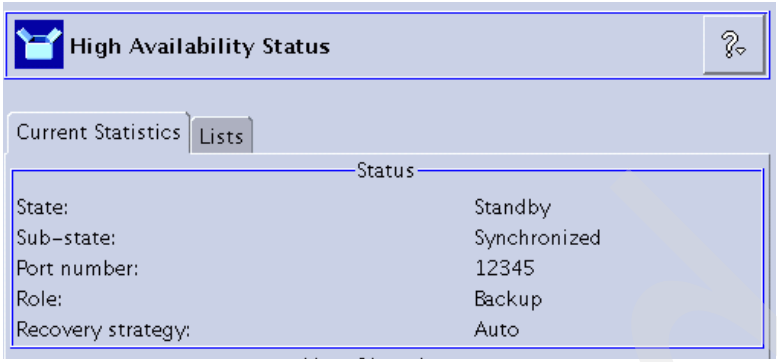


Figure 11-55 Standby Load Balancer status after High Availability configuration

11.3.2 Adding reach targets

A reach target is like a heartbeat, but another machine is used as the destination for the ping packet. The same reach targets are added to the configuration of both primary and backup servers.

If the active Load Balancer cannot reach this target (it does not receive a response from the ping packet), but the standby server still receives responses from it, the standby server will force a failover. Therefore, it is very important that you choose a stable server or network appliance as the reach target. We usually recommend using the default router of the local network (use the IP of the interface that is directly connected to the local network).

If you configure more than one reach target, the Standby Load Balancer will fail over if it receives responses from more reach targets than the active Load Balancer.

In our MAC scenario, we used the IP address of our local router, which is 9.26.126.2. For our NAT scenario we used 9.26.52.1.

Archived

To add this IP address as a reach target IP address, right-click **High Availability** in the left pane of the window and select **Add Reach Target**. A pop-up window is displayed, as shown in Figure 11-56.

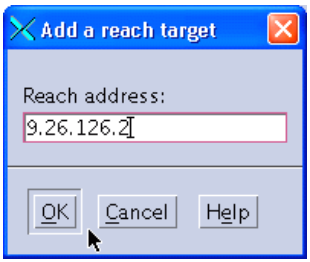


Figure 11-56 Adding a reach target

Note: Make sure to add the reach targets to the active Load Balancer first, and then to the one in standby state. Otherwise, the standby machine could force a failover during the process of setting up the reach targets, because it detects that it can reach more targets than the active machine.

We also recommend that you add the reach target after you have already tested the High Availability configuration (including takeover and failover) and the load balancing. The reason for this is that you could experience unwanted failovers during your initial High Availability tests if the reach target system is unstable.

Save the configurations on both Load Balancer machines after adding the reach targets.

11.3.3 Command-line configuration

The High Availability configuration can be performed from the command line using the statements saved in the configuration file by the GUI.

Archived

Example 11-6 shows the commands for the primary Load Balancer in the MAC forwarding scenario.

Note that each individual command has to be on one line in the configuration file. However, because of size limitations, some lines might be printed on two lines in our examples.

Example 11-6 Primary Load Balancer configuration file for MAC forwarding

```
dscontrol set loglevel 1
dscontrol executor start

dscontrol highavailability heartbeat add 9.26.127.157 9.26.126.120
dscontrol highavailability backup add primary=9.26.127.157 auto 12345
dscontrol highavailability reach add 9.26.126.2

dscontrol cluster add wcha.torolab.ibm.com address 9.26.126.103
primaryhost 9.26.127.157

dscontrol port add wcha.torolab.ibm.com:80 reset no

dscontrol server add wcha.torolab.ibm.com:80:srvb504.torolab.ibm.com
address 9.26.127.157
dscontrol server set wcha.torolab.ibm.com:80:srvb504.torolab.ibm.com
collocated y

dscontrol server add wcha.torolab.ibm.com:80:srvb501.torolab.ibm.com
address 9.26.126.120

dscontrol port add wcha.torolab.ibm.com:443 reset no

dscontrol server add wcha.torolab.ibm.com:443:srvb504.torolab.ibm.com
address 9.26.127.157
dscontrol server set wcha.torolab.ibm.com:443:srvb504.torolab.ibm.com
collocated y

dscontrol server add wcha.torolab.ibm.com:443:srvb501.torolab.ibm.com
address 9.26.126.120

dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start Ssl 443 Ssl_443.log
```

Archived

Example 11-7 shows the commands for the Standby Load Balancer in the MAC forwarding scenario.

Example 11-7 Standby Load Balancer configuration file for MAC forwarding

```
dscontrol set loglevel 1
dscontrol executor start

dscontrol highavailability heartbeat add 9.26.126.120 9.26.127.157
dscontrol highavailability backup add backup auto 12345
dscontrol highavailability reach add 9.26.126.2

dscontrol cluster add wcha.torolab.ibm.com address 9.26.126.103
primaryhost 9.26.127.157

dscontrol port add wcha.torolab.ibm.com:80 reset no

dscontrol server add wcha.torolab.ibm.com:80:srvb504.torolab.ibm.com
address 9.26.127.157

dscontrol server add wcha.torolab.ibm.com:80:srvb501.torolab.ibm.com
address 9.26.126.120
dscontrol server set wcha.torolab.ibm.com:80:srvb501.torolab.ibm.com
collocated y

dscontrol port add wcha.torolab.ibm.com:443 reset no

dscontrol server add wcha.torolab.ibm.com:443:srvb504.torolab.ibm.com
address 9.26.127.157

dscontrol server add wcha.torolab.ibm.com:443:srvb501.torolab.ibm.com
address 9.26.126.120
dscontrol server set wcha.torolab.ibm.com:443:srvb501.torolab.ibm.com
collocated y

dscontrol manager start manager.log 10004
dscontrol advisor start Http 80 Http_80.log
dscontrol advisor start Ssl 443 Ssl_443.log
```

For NAT forwarding without collocation, the configuration files are changed similarly. The three **dscontrol highavailability** lines are added with the respective parameters for primary and backup machine, and all **dscontrol executor configure** lines are removed.

Archived

For NAT forwarding with collocation, refer to the *Load Balancer Administration Guide,* GC31-6858.

11.3.4 Configuring the High Availability scripts

We now need to create the High Availability scripts.

You need to configure at least the goActive, goStandby, and goInOp scripts, but we also show how to use the serverUp, serverDown, and highavailChange scripts.

You can start working with the samples available in the *LoadBalancer_Install_Dir*/servers/samples directory and customize them for your environment, or you can write them from scratch.

All scripts are created in the *LoadBalancer_Install_Dir*/servers/bin directory, and you need to name them exactly as indicated (Load Balancer is case sensitive).

Note: The scripts are identical for the primary and the backup servers, unless there is some particular command that you need to run on each machine. This might be the case, for example, if your backup server collocated while your primary server did not.

We used two AIX servers for the MAC forwarding configuration, so we wrote the scripts using ksh syntax.

For NAT forwarding, we did not configure High Availability completely on our Solaris system, but we point out the main differences here. Again, this applies only if you do not use collocation with NAT forwarding. If you want to use collocated servers with NAT forwarding, refer to the *Load Balancer Administration Guide,* GC31-6858.

Variables

First, we set up a script for initializing some common variables, which is then included by the other scripts. This script is shown in Example 11-8.

Example 11-8 LoadBalancer_Install_Dir/bin/variables script for MAC forwarding

```
#!/bin/ksh
ND_LOGDIR=LoadBalancer_Install_Dir/servers/logs/dispatcher
CLUSTER=9.26.126.103
INTERFACE=en0
NETMASK=255.255.254.0
```

For NAT, you need to provide a variable for the return address, as shown in Example 11-9. We are using bash syntax for Solaris 9.

Example 11-9 LoadBalancer_Install_Dir/bin/variables script for NAT forwarding

```
#!/bin/sh
ND_LOGDIR=LoadBalancer_Install_Dir/servers/logs/dispatcher
CLUSTER=9.26.52.154
INTERFACE=hme0:1
RETURNADDRESS=9.26.52.156
RETURNINTERFACE=hme0:2
NETMASK=255.255.254.0
```

goActive

Executor runs this script when Load Balancer switches to active state. This could be when a failover occurs and the standby server switches to active, or when Load Balancer is started on the primary machine and the recovery strategy is set to automatic.

The script adds the cluster IP alias (9.26.126.103) to the network interface (defined in the INTERFACE variable). As our Load Balancer servers are collated with balanced Web servers, we also need to remove the cluster IP alias from the loopback interface (lo0 on AIX). See Example 11-10.

Example 11-10 goActive script for MAC forwarding (AIX)

```
#!/bin/ksh
. LoadBalancer_Install_Dir/servers/bin/variables
date >> $ND_LOGDIR/ha.log
print "This machine is Active. Aliasing cluster address to NIC. \n" >>
   $ND_LOGDIR/ha.log
ifconfig lo0 delete $CLUSTER
ifconfig $INTERFACE alias $CLUSTER netmask $NETMASK
```

Note: Only delete the loopback alias (“ifconfig lo0 delete $CLUSTER”) if Load Balancer is collocated with a balanced Web server.

For NAT forwarding, you need to add an alias for the return address. See Example 11-11.

Example 11-11 goActive script for NAT forwarding (Solaris 9, no collocation)

```
#!/bin/sh
. LoadBalancer_Install_Dir/servers/bin/variables
date >> $ND_LOGDIR/ha.log
```

```
echo "This machine is Active. Aliasing cluster address to NIC. \n" >>
   $ND_LOGDIR/ha.log
ifconfig $INTERFACE plumb $CLUSTER netmask $NETMASK up
ifconfig $RETURNINTERFACE plumb $RETURNADDRESS netmask $NETMASK up
```

goStandby

Executor runs this script when Load Balancer switches to standby state. This could be when Load Balancer is started on the standby server and the primary server is active.

The script removes the cluster IP alias (9.26.126.103) to the network interface (defined in the INTERFACE variable). Because in our scenario Load Balancer is collocated with balanced Web servers, the cluster IP alias is added to the loopback interface (lo0 on AIX). See Example 11-12.

Example 11-12 goStandby script for MAC forwarding (AIX)

```
#!/bin/ksh
. LoadBalancer_Install_Dir/servers/bin/variables
date >> $ND_LOGDIR/ha.log
print "Going into Standby mode. Deleting the device alias and adding
   the loopback alias.\n" >> $ND_LOGDIR/ha.log
ifconfig $INTERFACE delete $CLUSTER
ifconfig lo0 alias $CLUSTER netmask 255.255.255.255
```

Note: Only add the loopback alias ("ifconfig lo0 alias $CLUSTER netmask 255.255.255.255") if Load Balancer is collocated with a balanced Web server.

For NAT forwarding, you need to remove the alias for both the cluster and the return address, as shown in Example 11-13.

Example 11-13 goStandby script for NAT forwarding (Solaris 9, no collocation)

```
#!/bin/sh
. LoadBalancer_Install_Dir/servers/bin/variables
date >> $ND_LOGDIR/ha.log
echo "Going into Standby mode. Deleting the device alias and adding
   the loopback alias.\n" >> $ND_LOGDIR/ha.log
ifconfig $INTERFACE $CLUSTER netmask $NETMASK down unplumb
ifconfig $RETURNINTERFACE $RETURNADDRESS netmask $NETMASK down unplumb
```

Archived

goInOp

Executor runs this script whenever Executor is stopped or it is first started. We want to set the network interfaces to an initial state here, so the cluster IP alias (9.26.126.103) is removed from the network interface. However, because our Load Balancer servers are collated with balanced Web servers in our MAC forwarding scenario, we need to add the cluster IP alias to the loopback interface (lo0 on AIX) or keep it there. See Example 11-14.

When Executor is started and a configuration is loaded, goActive or goStandby will be called after this script, and the settings for the interfaces are corrected.

Example 11-14 goInOp script for MAC forwarding (AIX)

```
#!/bin/ksh
. LoadBalancer_Install_Dir/servers/bin/variables
date >> $ND_LOGDIR/ha.log
print "Executor has stopped. Removing device alias.\n" >>
   $ND_LOGDIR/ha.log
ifconfig lo0 delete $CLUSTER
ifconfig lo0 alias $CLUSTER netmask 255.255.255.255
ifconfig $INTERFACE delete $CLUSTER
```

Note: Only add the loopback alias ("ifconfig lo0 alias $CLUSTER netmask 255.255.255.255") if Load Balancer is collocated with a balanced Web server.

For NAT forwarding on Solaris (without collocation), the script is shown in Example 11-15.

Example 11-15 goInOp script for NAT forwarding (Solaris 9, no collocation)

```
#!/bin/sh
. LoadBalancer_Install_Dir/servers/bin/variables
date >> $ND_LOGDIR/ha.log
echo "Executor has stopped. Removing device alias.\n" >>
   $ND_LOGDIR/ha.log
ifconfig $INTERFACE $CLUSTER netmask $NETMASK down unplumb
ifconfig $RETURNINTERFACE $RETURNADDRESS netmask $NETMASK down unplumb
```

serverDown

Executor runs this script whenever a balanced server is marked down by Manager. It passes the name of the balanced server as the first parameter to the script. With ksh on AIX, the variable $1 contains the server name in the format <cluster>:<port>:<server>. For example, for Web server node 1 in our scenario:

```
wcha.torolab.ibm.com:80:srvb501.torolab.ibm.com
```

We use this script to record an entry in a log file saying that one of the Web servers was marked down by manager. See Example 11-16.

Example 11-16 serverDown script

```
#!/bin/ksh
DATE=`date`
OUTPUT="$DATE $1 has been marked down."
echo $OUTPUT >> /opt/ibm/edge/lb/servers/logs/lb.log
```

For NAT on Solaris, we use the same script except that we use #!/bin/sh as the execution shell.

serverUp

Executor runs this script whenever a balanced server is marked up by Manager. It passes the name of the balanced server as the first parameter to the script. With ksh on AIX, the variable $1 contains the server name in the format <cluster>:<port>:<server>. For example, for Web server node 1 in our scenario:

```
wcha.torolab.ibm.com:80:srvb501.torolab.ibm.com
```

We use this script to record an entry in a log file saying that one of the Web servers was marked up by Manager. See Example 11-17.

Example 11-17 serverUP script

```
#!/bin/ksh
DATE=`date`
OUTPUT="$DATE $1 has been marked back up."
echo $OUTPUT >> /opt/ibm/edge/lb/servers/logs/lb.log
```

For NAT on Solaris, we use the same script except that we use #!/bin/sh as the execution shell.

highavailChange

Executor runs this script whenever the state of the Load Balancer server changes (from active to standby or from standby to active). It passes the name of the High Availability script that was run as the first parameter to the script. With ksh on AIX, the script name can be used by referencing the variable $1.

Archived

We use this script to record an entry in a log file saying that the state of the local server has changed. See Example 11-18.

Example 11-18 highavailChange script

```
#!/bin/ksh
DATE=`date`
OUTPUT="$DATE LB just ran $1."
echo $OUTPUT >> /opt/ibm/edge/lb/servers/logs/lb.log
```

For NAT on Solaris, we use the same script except that we use #!/bin/sh as the execution shell.

11.3.5 Test Load Balancer High Availability

The High Availability configuration is now complete.

You may test the failover by issuing the command **dscontrol executor stop** on the command line of your primary server.

You should still be able to access your store through the host name that resolves to the cluster IP address. The High Availability status for your standby server, when selecting High Availability in the left pane of the Load Balancer GUI, should now be active while the role is still backup.

Note that **dsserver stop** is not sufficient to stop load balancing. The executor will continue running. To completely stop load balancing, first stop executor, then dsserver.

11.3.6 Starting Dispatcher automatically after a reboot

If you are running Dispatcher on a UNIX system, you must configure the system to run the **dsserver** command after each reboot. Make sure that your configuration filename is default.cfg, because when dsserver is run it automatically loads the default.cfg configuration file.

In our AIX environment, we enabled the automatic startup of dsserver. We added it to the inittab by running the following command:

```
mkitab "ds:2:wait:/usr/bin/dsserver > /dev/console 2>&1"
```

Tip: You can also use other scripts that are run during the server start up process, for example, /etc/rc.tcpip (in AIX systems) or /etc/rc.local (in Linux systems). Make sure that you consult the system administrator to find the most suitable option for your environment.

Archived

Archived

Archived

Part 4

Design with performance in mind

In the previous chapters we talked about High Availability architecture and installation instructions. In this part of the book we talk about developing your custom code with performance in mind. Both these activities can go in parallel, depending on your resource availability.

Performance of an application must be a focus area throughout the project cycle. Traditionally, performance testing has occurred late in the cycle, towards the end of the testing phase, or as part of the deployment process. This does not leave much time to identify and fix any performance issues, particularly if significant architecture changes are required. It is much more difficult and time consuming to improve performance in the later phases of the project cycle than to ensure that the application performs adequately from the beginning.

This section demonstrates the importance of performance testing early in a project, with focus on profiling.

This part of the book include:

- Development performance considerations
- Caching

© Copyright IBM Corp. 2008. All rights reserved.

Archived

12

Development performance considerations

All the performance best practices remain the same as the basic performance practices for developing Java code or SQL queries, and so on. However, here we highlight a few specific performance coding practices that will benefit you during your WebSphere Commerce site development.

Tip: You will find a large number of useful best practices available at:

http://www.ibm.com/developerWorks

© Copyright IBM Corp. 2008. All rights reserved.

12.1 Development best practices for performance

Depending on your custom scenario you may need to focus on improving, for example, catalog browse, search, or shopping cart processing performance. There is a lot of information in the WebSphere Commerce InfoCenter to guide you through those. In this section we discuss some of the more general performance development techniques and strategies.

12.1.1 Access Bean usage

EJB access beans can greatly simplify client access to enterprise beans and alleviate the performance problems associated with remote calls for multiple enterprise bean attributes.

Below we discuss a couple of considerations when working with access beans.

Well formed constructor

Non-default AccessBean constructors are usually mapped to the corresponding EJB create() method, which in turns will trigger database INSERT. For example, XYZAccessBean(int p1, String p2) is mapped to the corresponding create(int p1, String p2) in the EJB Home interface.

Consider the case where a table having three columns is represented by a single EJB entity Bean (XYZBean), where the first two columns are non-nullable. Assume that we have two create() methods:

- create(int p1, string p2) represents the minimum number of non-nullable columns.
- create(int p1, string p2, int p3) represents the full well formed.

  Within a single transaction, the following code snippet will cause 2 SQL calls in the database as marked:

```
XYZBean aBean = new XYZBean(3, “testing”);  // INSERT call to
database

aBean.setP3(4); // UPDATE call to database
```

  While the following will only trigger one SQL call:

```
XYZBean aBean = new XYZBean(3, “testing”, 4);  // INSERT call to
database
```

  Unless it is required by the logic, the second code snippet using the well-formed constructor is preferable over the first one.

Lazy instantiation

All EJB multi-object finders return enum (EJB 1.1) or Collection (EJB 2.0). The actual object is not instantiated until the object is needed. This pattern should be preserved in our EJB and AccessBean when returning relationship objects, for example:

```
AssociatedAccessBean[] getAssociatedBean() should be changed to
Enumeration getAssociatedBean() or Collection getAssociatedBean()
```

A similar pattern should be used in DataBean as well.

Re-use finder results

Finder methods go back to the database to obtain their results. Avoid calling the same finder to find the same results more than once in a transaction by saving the resulting access bean objects and re-using them until the transaction ends.

12.1.2 Java classes and keywords

There are number of performance considerations for writing code in Java. Here we provide a few key ones only.

Tracing

Avoid using System.out.println in the final code. This is a serialized resource that requires file I/O on every call and will impact performance.

When adding WC trace in the code, it should be enclosed in a if-then block if the trace parameters have side effects (for example, creating new objects, expensive operations). For example:

```
if (ECTrace.traceEnabled(ECTrace.Identifiers.COMPONENT_XYZ) {
      .......
}
```

Synchronization

If you do not require synchronized access to resources, avoid using the *synchronized* keyword in method or block. If there is such a need, try to minimize the code block.

Similarly, if you do not require protecting any shared resources, use:

- java.util.ArrayList instead of java.util.Vector
- java.util.HashMap instead of java.util.Hashtable

Archived

Avoid instanceof

Avoid using the instanceof keyword in the code whenever possible, as this is a relatively expensive operation. Try to refactor your code.

Initial capacity and load factor

Specify reasonable initial capacities when creating container objects such as StringBuffers, Lists, Sets, Maps, Vectors, and Hashtables, to avoid unnecessary re-allocation when the initial capacity is exceeded, and to avoid allocating more memory than required.

The capacity for a StringBuffer created using the StringBuffer(String aString) constructor is 16+aString.length(). If you know that the StringBuffer will eventually be longer than that, you will be better off specifying an initial capacity for the StringBuffer and then appending aString.

For HashMaps and Hashtables, take the load factor into account when specifying initial capacities. For example, to allocate a HashMap with sufficient capacity for n entries (using the default load factor of .75), specify an initial capacity of 1+n*100/75.

We do not recommend increasing the default load factor (for example, to 1), since each hash bucket will end up holding more entries, thus slowing down the lookup operation.

Throwing and catching exceptions

Throwing and catching an exception is relatively costly. Use exceptions in exceptional situations, but avoid using them in normal (frequently executed) processing.

Reduce unnecessary memory usage

Avoid allocating memory unnecessarily, to reduce the frequency of garbage collection cycles. This can occur, for example, when A calls B, which calls C, and A has an integer, converts it to a string so it can be passed to B, and B converts it back to an integer so that it can be passed to C. It may be that the string and the second integer object need not have been allocated if the method signature to b accepted an Integer rather than a String.

Use local variables, avoid inappropriate use of non-final getter methods

Accessing member data is slower than accessing a local variable, and calling a non-final or non-private getter method is even slower. There are sometimes good reasons to call non-final or non-private getter methods (it allows subclasses to

override the getter behavior). Be sure that you use them with that in mind. However, calling the same non-final or non-private getter method twice in the same method is slower than necessary (and may have unintentional side effects since you cannot control the behavior of a subclass). Use a local variable to avoid calling the getter more than once per method call. For example, do this:

```
int total = 0;
int amt = getAmount();
if (amt>0) {
total += amt;
}
```

Not this:

```
int total = 0;
if (getAmount()>0) {
total += getAmount();
}
```

12.1.3 JSPs

Here are a couple of considerations when writing JSPs.

Session-aware JSP

By default, all JSPs are session aware, meaning that they can participate in HTTP session-related operations. If your JSP does not need such participation, disable it by using the JSP page directive (for example, <%@ page session="false" >). Otherwise, this will cause the HTTP session to be created if it does not exist.

DataBean activation

When composing a JSP page using multiple JSP fragments, care should be taken if the same databean is used on these fragments to avoid unneccessary activation and instantiation overhead. In these cases, use the the same id attribute value and set the scope attribute value to *request*.

If databeans are not used, do not declare them with the useBean tag, as that incurs some overhead cost.

12.1.4 Registry objects

The Commerce server runtime includes several objects that implement the registry interface, which includes initialize and refresh methods. Some registries (for example, the StoreRegistry) are implemented using the

Archived

AbstractManagedDynamicCacheRegistry abstract class, which provides a LRU lazy cache based on a Hashtable and a LinkedList of most recently used elements. The initial capacity and maximum size of the cache is configurable in the <instance>.xml file via the “initialCapacity” and “regMaxSize” attributes of the “<registry>” element of the “<Registries>” instance property. The default value for “regMaxSize” is 500.

StoreRegistry

The StoreRegistry is a lazy initialization in memory cache of StoreCopy objects. StoreCopy is a class that inherits from StoreAccessBean. Use the StoreRegistry “find” method (for example, StoreRegistry.find(Integer storeId)) to obtain StoreCopy objects when you need *read only* store access beans. Otherwise, If you need to update a Store EJB, then you should instantiate a normal StoreAccessBean (not a StoreCopy) and use the setter methods and the commitCopyHelper method to update the Store object.

The StoreCopy object caches information about a store and its related information:

- Filenames of resource bundles files and the actual resource bundles
- List of supported language IDs
- Member ID of the owning organization of the store
- Access beans for the store default information
- Store description
- Default contract for the store
- Master catalog and other catalogs for the store
- Tax categories for the store
- Store directory tree under the Stores WebApp

If any of the cached information is changed, the StoreRegistry must be updated to invalidate the information cached in the StoreCopy object for that Store. The StoreRegistry is also responsible for invalidating cached store relationship information, cached for each Commerce server in the StoreRelationshipCache object.

RefreshRegistry and UpdateRegistry controller commands

To invalidate information in the StoreRegistry, or in other Registry objects, use either the RefreshRegistry or UpdateRegistry controller commands. These commands schedule a scheduler job for immediate execution, which causes commands to be executed in each Commerce server. The commands end up calling the refresh methods, or the update methods, of the specified registries, in each Commerce server.

When your code calls these controller commands, it should be careful to call the setAccCheck(false) method, to prevent that command from performing access

control checks. That is because, for example, a user who has the authority to add a catalog to a store may not have the authority to execute the UpdateRegistry command directly. However, when a catalog is added to a store, the StoreRegistry must nonetheless be updated for that store.

12.1.5 Database operations

One common coding practice than can lead to deadlocks is to avoid deleting rows and inserting rows within a single transaction. Deleting a row and inserting a row in the same transaction will potentially cause the database to lock the table, reducing concurrency and introducing the potential for more deadlocks.

We recommend restructuring the code to avoid such a situation. One technique that can sometimes be used, depending on the data involved and how it is used by the business logic, is to define a *recycler* class that can delay deletion of operational data until the end of the transaction, in case the object can be re-used simply by changing some of its attribute values, thus avoiding a delete operation followed by an insert operation.

12.1.6 Command execution

Some common coding practices for when working with WebSphere Commerce commands are:

- Batch price retrieval calls.

  Looking up a price via one of the contract price retrieval task commands travels a lengthy code path. If your code has to look up several prices, it is faster to specify all of them in a single call rather than make a separate call for each price retrieval.

- Re-use command instances.

  Task and controller command instances are retrieved from the CommandFactory. They can be executed more than once in the same transaction. Call the reset method to re-initialize a command for re-execution.

12.1.7 Web 2.0 considerations

When you are designing a Web 2.0 store all the Web 1.0 performance considerations apply. In addition, there are a number of additional considerations of which you have to be careful. The store flow and the page design will have a direct impact on both your WebSphere Commerce server capacity utilization and the response time of the page.

Archive

The reason that you may need additional server capacity to support the same site traffic is due to the fact that you may design your page such that the sum total of all the new (asynchronous) calls to WebSphere Commerce may far exceed workload generated by your Web 1.0 site. The key point here is that in this case the additional capacity requirement will purely be client (JSP) driven. The WebSphere Commerce server will still perform as well as it would had the request come from the Web 1.0 client.

The reason response will likely increase is due to JavaScript™ parsing of the Dojo widgets. The more the widgets, the more the parsing. Again, the key point here is that WebSphere will still respond in about the same time. However, most of the additional response time delays will be due to the parsing of Web pages on the client computer running a Web browser. Thus, the more widgets that you have on your page the more response time a client will experience.

Note: For more information about performance consideration when developing Web 2.0 store, refer to:

http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp?topic=/com.ibm.commerce.web20storesolution.refapp.doc/tasks/tsm_web20_extend.html

12.2 Performance best practices for database customizations

Your site design requirement may require you to customize the schema. The base out-of-box WebSphere Commerce schema should only be modified in such a way that your customizations do not impact the functioning and maintenance of WebSphere Commerce. Otherwise, the cost of your customizations could snowball. For example, you might customize your schema in such a way that certain aspects of administration tools may stop functioning properly or, perhaps, fix packs may not be installed successfully, or the WebSphere Commerce migration program may not support your schema customization and thus fail to function.

Generally, you would want to add new indices and tables. In some rare circumstances you may want to add some columns to a table. In such a situation you should first consider using the extra customizable fields available in the table.

In this section we do not go into detail about the pros and cons of database customization, or the detailed methodology. In this section, we discuss the

performance considerations to keep in mind while you design your database customizations.

12.2.1 Table design

When defining a new table, you will need to decide what type of information will be stored in the table, the data types of the attributes, and whether an attribute can have a null value, and determine which attributes require a default value, define the primary key, define additional unique indexes or indexes required for performance reasons, and determine any relationships with other tables.

Design your table with application access in mind (that is, pay attention to the way that the application accesses data). This will help you place the columns in the correct order and decide on what indexes you may need.

- If you need to repeat the same data many times in the table, then consider creating a separate lookup table with the repeated data with a primary key. Then you can reference the new table in your master table (for example, normalize your data).
- Place columns in the order that makes sense upon retrieving. For example, if you always select col1, col2, and col3 in your where clause, then order the columns the same way in the table. This will make it easier when creating indexes on them.
- Order the column with the primary key in mind. The order of the columns in the table must match the order of the columns in the primary key.
- Place variable length columns (Varchar, CLOBS) at end of your tables to avoid fragmentation.

12.2.2 Index design

Having an efficient set of indexes defined for a table will improve performance on queries that access and update information for that table. For practical reasons, you cannot have indexes on every column. This would result in extremely poor performance when rows are inserted or updated in a table. The more indexes you have, the higher the additional overhead to insert, delete, and potentially update. A good start is to define indexes on these columns:

- Columns involved in the WHERE clause of an SQL statement, especially if it is part of a join with another table.
- Columns involved in the ORDER BY or GROUP BY clause. If you have a two-column order by clause, you may want to have a compound index that includes those two columns. The leading index columns would have to exactly match the ordering in the ORDER BY or GROUP BY clause.

- Columns included in the result set. This is an added bonus because it may result in no data pages being accessed to process the query.
- The order of columns in a compound index (an index that involves more than one column) is extremely important. If the leading column of the index is not used in the WHERE clause of an SQL statement, then that index will not be selected by the optimizer to improve performance. For example, suppose the EMPLOYEE table has an index on LASTNAME + FIRSTNAME. The following query will not be able to use that index. Instead, a table scan will be performed:

  ```
  SELECT lastname, firstname, salary FROM employee WHERE FIRSTNAME =
  'John'
  ```

- When creating a compound index, consider placing the column with the most values first and the column with the fewest values last.

Remember that WC queries are dynamic. The access path that you see could easily change, and usually will in a real-world application. Foresight may be more important than DB2 Explain!

12.2.3 Avoiding deadlocks

Good initial design of commands to break down business logic into short transactions, using efficient queries to access information, and defining useful indexes will all help to minimize deadlock.

However, the most important design consideration to avoid deadlock is to perform database access on tables in the same order across all areas of an application. If all processes access table A, B, and C in that order each and every time, it is not possible to create a deadlock condition. Deadlocks occur because another process accesses the tables in the order C, B, A. This requires an understanding of the SQL that is being invoked within your transaction, as well as SQL invoked by other transactions or processes.

The following tips that can help in avoiding deadlocks:

- Avoid using Select for update.
- Make your transactions short and issue the update or delete statements at the end of your EJB transactions.
- Index the columns that appear in the Where clauses in your SQL queries, especially if the tables you are selecting from are large tables by nature.

Archived

12.3 Performance best practices for SQL queries

Optimizing SQL is an extensive, multidimensional topic and requires careful analysis of not only the SQL but also the data and the scenarios using that data.

Here we list some of the key aspects of SQL optimization only that we found beneficial when writing code for WebSphere Commerce.

12.3.1 Reduce the result set as early as possible

Assume the following expressions in a Where clause:

1. Table1.column1 = Table2.column3 (joins all rows of both tables)
2. Table1.Column5 > value1 (returns 60% of table1's rows)
3. Table2.Column3 = value2 (returns 30% of table's rows)
4. Table1.column2 = value3 (returns 2% of table2's rows)

Following the above order will be expensive, especially if table1 and table2 are large tables. A more efficient order is 4, 3, 2, 1.

12.3.2 Avoid using sub-selects and redundant expressions

Let us look at the following real example on subselects:.

```
SELECT distinct(CATENTRY_ID)
from CATENTRY
where MARKFORDELETE=0
AND
1) (CATENTRY_ID
      in (SELECT CATENTRY_ID
           FROM CATGPENREL
          WHERE CATGROUP_ID = 10000000)
          or CATENTRY_ID
2)      in (SELECT CATENTRY_ID_CHILD
             from CATENTREL, CATGPENREL
            where CATENTRY_ID_PARENT=CATENTRY_ID
             and CATGROUP_ID = 10000000))
```

The above SQL has two levels of subselect. This makes the SQL statement very expensive. Optimization is required for such statements. If there are common tables in the From clause between the sub-selects, then it is a good hint that the sub-selects can be converted easily to joins.

Archived

A better and more efficient version of the SQL is:

```
SELECT distinct (c.CATENTRY_ID)
From CATENTRY c , catgpenrel b ,catentrel d
Where
   MARKFORDELETE=0
   AND
   b.CATGROUP_ID = 10000000
   and
   (
1) (c.catentry_id=b.catentry_id)
         or
2) (c.CATENTRY_ID=d.CATENTRY_ID_PARENT
       and b.catentry_id=d.CATENTRY_ID_CHILD))
```

Note that:

- The underlined clause (CATGROUP_ID = 100000000) was repeated and evaluated multiple times in the original statement. In the new tuned SQL, the clause was taken to be outside the scope of the original sub-selects.
- The original subselect #1 was rewritten by moving the table “CATGPENREL” in the “FROM” section to the main SQL. The new form of the subselect is shown as # 1 above. It is a simple join expression.
- For sub-select #2 in the original SQL, the table “CATENTREL” was moved to the main. The other table “CATGPENREL” was moved to the main part of tuning sub-select #1.

12.3.3 IN versus Exists

To quote the Oracle Tuning Reference, “In certain circumstances, it is better to use IN rather than EXISTS. In general, if the selective predicate is in the subquery, then use IN. If the selective predicate is in the parent query, then use EXISTS.”

IN and EXISTS work very differently. In case of IN, the subquery is executed first and then the result is compared with the outer table. The EXISTS clause, however, is evaluated for every value in the outer table. Here is an example for WebSphere Commerce that shows that EXISTS performs much better than IN when the selective predicate is the parent query. The query below was written using IN, and it was running for a long time. According the above role, the query is a good candidate to use EXISTS instead of IN. The selective predicates (T2.REGISTERTYPE IN ('R') AND mr1.ROLE_ID = -29) are in the parent query.

```
SELECT DISTINCT t1.state, t1.member_id, t1.optcounter, t1.type,
t2.field2, t2.registrationupdate, t2.field3, t2.lastorder,
```

```
t2.language_id, t2.prevlastsession, t2.setccurr, t2.dn,
t2.registrationcancel, t2.lastsession, t2.registration, t2.field1,
t2.registertype, t2.profiletype
FROM userreg, mbrrole mr1 , users t2, member t1
WHERE
        t2.registertype IN ('R')
        AND mr1.role_id = -29
        AND (EXISTS
              ( SELECT 1
                FROM mbrrel mr, mbrrole ml
                WHERE (
                          (mr.descendant_id = -1000
                              AND mr.sequence=1
                              AND mr1.orgentity_id = mr.ancestor_id)
                          OR
                          (ml.role_id IN (-1,-20,-27)
                              AND ml.member_id = -1000
                             AND mr1.orgentity_id = ml.orgentity_id)
                 OR
                 (
                              (ml.role_id IN (-1,-20,-27)
                                  AND ml.member_id = -1000
                                  AND mr.ancestor_id =
ml.orgentity_id )
                              AND mr1.orgentity_id =
mr.descendant_id)
                          )
                 )
                 OR mr1.orgentity_id IN
                    (SELECT ancestor_id
                     FROM mbrrel mrl, mbrrole mble
                     WHERE mble.member_id=-1000
                           AND mble.role_id IN  (-1,-20,-27)
                           AND mrl.descendant_id = mble.orgentity_id
                    )
        )
        AND t2.users_id = mr1.member_id
        AND userreg.users_id = t2.users_id
        AND t1.member_id = t2.users_id
        AND t1.type = 'u'
```

Archived

12.3.4 Other important SQL tuning hints

Add the ‘FOR READ ONLY’ clause on selects that are retrieving data that will not be updated. Determine whether additional indexes can help performance.

Retrieve only the information that is needed

Note the following:

- Select only the columns that you need, not all the columns in a table.
- Never use the SELECT * FROM table. If the column order changes or columns are removed or added to a table, then your results will be unpredictable.
- If a large result set is expected, restrict the number of rows returned with the FETCH FIRST n ROWS clause.
- If a small result set is the expectation, use OPTIMIZE FOR n ROWS to help optimization.
- If the FETCH FIRST n ROWS clause is used, consider also using the OPTIMIZE FOR statement.
- In GUIs that allow search criteria to be entered by the user, consider forcing a minimal number of characters to be entered for pattern matching. (This will avoid problems where a user requests product information for all items that contain the letter a in the description.)
- Avoid the use of a leading wildcard in predicates, since this will require a table scan.

Practices to be avoided

Avoid the following practices:

- Avoid using “Distinct” in comparison clauses (that is, where column a in (select distinct b from z)). The “Distinct” is nothing but an overhead.
- Avoid the use of functions or operators on indexed columns in a predicate. For example, the following query will not be able to take advantage of an index on the LASTNAME column:

  ```
  SELECT lastname, firstname FROM employee WHERE UPPER(LASTNAME) =
  ‘BROWN’
  ```

- Avoid using FOR UPDATE, as it can lock the rows for longer times and may cause waits and deadlocks.
- Avoid constructs that require extra table scanning, index scanning, or locking (for example, DISTINCT, ORDER BY, GROUP BY, and so on).
- Avoid scalar functions (for example, Count(*), max(), avg()).

Archived

- Avoid INNER and OUTER joins unless they are warranted. Too many times these are not required and result in the optimizer choosing the wrong path.

Archived

Archived

13

Caching

In general, caching improves response time and reduces system load. Caching techniques have long been used to improve the performance of World Wide Web Internet applications. Most techniques cache static content (content that rarely changes) such as graphic and text files. However, many Web sites serve dynamic content, containing personalized information or data that changes more frequently. Caching dynamic content requires more sophisticated caching techniques, such as those provided by the WebSphere Application Server 6.0 dynamic cache, a built-in service for caching and serving dynamic content.

When designing your application, it is critical to keep caching considerations in mind.

This chapter discusses the benefit of caching on a WebSphere Commerce site.

We discuss:

- Types of caching
  - Dynamic Cache Service (DynaCache)
  - Edge Side Includes (ESI)
- Caching enhancements in WebSphere Commerce 6.0.0.1 and later
- Cache replication strategy for a large scale, highly available WebSphere Commerce system
- Monitoring DynaCache and ESI

© Copyright IBM Corp. 2008. All rights reserved.

13.1 Types of caching

WebSphere Commerce performance can be greatly enhanced by using two types of caching available to WebSphere Application Server and IBM HTTP Server Plug-in:

- Dynamic caching
- Edge Side Includes (ESI) caching

The following sections explain the two mechanisms and direct you to valuable resources for further reading.

13.1.1 Dynamic caching

The dynamic cache service includes:

- Servlet or JSP result cache, to cache entire pages or fragments generated by a Servlet or a JSP page.
- Command cache, to cache command objects.
- Edge Side Includes (ESI) caching, to cache, assemble, and deliver dynamic Web pages at the edge of an enterprise network.
- Invalidation support, to ensure that the content of the cache is correct. Invalidation can be rule based, time based, group based, and programmatic.
- Replication support, to enable cache sharing and replication among multiple servers.
- Disk offload capability, to enable caching large amounts of data, and to preserve cache content while the application server is stopped and restarted. Note that after the application server is restarted and the database restored, we recommend that you clear the disk cache using the cache monitor. This will ensure that information that has become invalid for the new database is removed. Use the cache monitor to clear the cache or, alternatively, use the following URL:

  ```
  http:// host_name/path/DynaCacheInvalidation?clear=true
  ```

The caching behavior of the WebSphere Application Server dynamic cache service is specified by cache policies defined by <cache-entry> elements in cache specification configuration XML (cachespec.xml) files.

As the dynamic cache service places objects in the cache, it labels them with unique identifying strings (cache IDs) constructed according to <cache-id> rules specified in the <cache-entry> elements. Once an object with a particular cache-id is in the cache, a subsequent request for an object with the same cache-id is served from the cache (a cache *hit*). The <cache-id> rules define how

Archived

to construct cache-ids from information associated with an application server request (to execute a Servlet, JSP, or command), including how information may be obtained programmatically from CacheableCommand objects.

Cached objects are removed from the cache according to information provided in their <cache-entry> elements, such as the <timeout>, <priority>, and <invalidation> elements.

The <timeout> and <priority> elements configure expiry and eviction policies. When the available cache memory is full, a least recently used (LRU) caching algorithm removes cached objects with lower priority, or offloads them to disk if the disk offload capability is enabled, before those with higher priority.

The <dependency-id> and <invalidation> elements define rules that generate dependency IDs and invalidation IDs, which together specify that certain objects should be removed from the cache when certain requests (such as those that update cached information) are processed. When an object is cached, its generated dependency IDs are associated with it in the cache. When a request causes invalidation IDs to be generated, all objects associated with those invalidation IDs are removed from the cache.

The <inactivity> element is used to specify a time-to-live (TTL) value for the cache entry based on the last time that the cache entry was accessed. The value is the amount of time, in seconds, to keep the cache entry in the cache after the last cache hit.

The dynamic cache service responds to changes in the cachespec.xml file. When the file is updated, the old policies are replaced. Objects cached through the old policy file are not automatically invalidated from the cache. They are either reused with the new policy or eliminated from the cache through its replacement algorithm.

WebSphere Commerce uses WebSphere command caching internally, such as with MemberGroupsCacheCmdImpl in the preceding cache filter. However, WebSphere Commerce does not support caching of commands that contain non-serializable objects.

WebSphere Application Server dynamic cache

WebSphere Commerce uses the WebSphere Application Server dynamic cache service for caching servlets or JSP files and commands that extend from the WebSphere Application Server CacheableCommand interface. The dynamic cache service, servlet caching, and disk offload are enabled by default during the creation of a WebSphere Commerce instance.

Archive

For more details on dynamic caching for WebSphere Commerce, refer to *Mastering DynaCache in WebSphere Commerce*, SG24-7393-00.

13.1.2 Edge Side Includes (ESI) caching

WebSphere Application Server leverages the Edge Side Includes specification to enable caching and assembly of distributed fragments. Edge Side Includes is a simple mark-up language used to define Web page fragments for dynamic assembly of a Web page at the edge of network.

With the Distributed Fragment Caching and Assembly Support, WebSphere Application Server customers can defer page assembly to any ESI-compliant surrogate server, such as Akamai EdgeSuite service. This may result in a significant performance advantage if fragments can be cached and reused at the surrogate.

You can find detailed information about the ESI specification at the following sources:

- Edge Side Includes W3C Submission

  `http://www.w3.org/Submission/2001/09/`

- EdgeComputing.org

  `http://www.edgecomputing.org/`

- ESI.org

  `http://www.esi.org/`

WebSphere Application Server provides distributed fragment caching and assembly support through the Web server plug-in. WebSphere Application Server uses IBM HTTP Server Plug-in to communicate with the HTTP Server. This plug-in has the ability to cache entire pages or fragments.

With dynamic cache service's external cache control, distributed fragment caching, and assembly support, dynamic content can be exported, cached, and assembled at the most optimal location, closer to the user. More important, WebSphere Application Server can maintain control of the external cache through its invalidation support to ensure the freshness of cached content. As WebSphere Commerce is an application based on WebSphere Application Server, WebSphere Commerce customers can make use of the ESI caching feature to create and serve highly dynamic Web pages without jeopardizing page performance and user experiences.

Archived

ESI Processor

The cache implemented by the ESI processor is an in-memory cache, not a disk cache. Therefore, the cache entries are not saved when the Web server is restarted.

The basic operation of the ESI processor is as follows: When a request is received by the Web server plug-in, it is sent to the ESI processor, unless the ESI processor is disabled. It is enabled by default. If a cache miss occurs, a surrogate-capabilities header is added to the request by the plug-in and the request is forwarded to the WebSphere Application Server. If the dynamic servlet cache is enabled in the application server, and the response is edge cacheable, the application server returns a surrogate-control header in response to the IBM HTTP Server Plug-in.

The value of the surrogate-control response header contains the list of rules that are used by the ESI processor in order to generate the cache ID. The response is then stored in the ESI cache, using the cache ID as the key. For each ESI include tag in the body of the response, a new request is processed such that each nested include results in either a cache hit or another request forwarded to the application server. When all nested includes have been processed, the page is assembled and returned to the client.

Best practices

Review the best practice item listed here:

- Fragment pages when necessary to efficiently edge cache parts of pages that are dependent on different subsets of parameters and attributes (where attributes have to be replaced by cookies).
- Unlike the dynamic cache service that runs within the WebSphere Application Server, the ESI processor does not have access to user HTTP session data to uniquely identify page fragments. The application must be designed such that page fragments can be uniquely identified using *request parameters on the URL, HTTP form data, or HTTP cookies in the request*. In a JSP include, parameters should be included in the URL as query parameters instead of as JSP parameter tags. Since Version 6.0, WebSphere Commerce provides some cookies that can be used to access session data. See “Cookie support” on page 287.
- The parameter needs to be on the parent URL or the child URL. The child URL is created from the parameter passed into the JSP. Using WCParam within the JSP means that we cannot use it as a cache key, thus allowing the ESI processor visibility to these values as query parameters.
- Consideration should be given as to how expensive a given fragment is to compute. Fragments that are expensive to compute provide the best candidates for edge caching and can provide significant performance

benefits. The cache entry sizes for the ESI processor should be large enough to accommodate these expensive fragments. Also, the priority (or the time-to-live value) of these fragments should be raised to ensure that less expensive fragments are removed from the cache first. With ESI configured for the IBM HTTP Server Plug-in, we can use invalidation from Dynacache, so this is less of an issue. This can provide a huge performance boost. But you need to remember that the cache in the plug-in is only in memory. Also, it is important to note that the configuration at the servlet level does not let you define what is edgeable at a low enough level.

- Another important consideration for edge caching is the update rate of a page. Invalidation of cached fragments is a relatively expensive operation. Therefore, very dynamic fragments that are invalidated often may not benefit from caching, and may actually hurt performance.
- Web server tuning considerations when using ESI plug-in cache.
  - Use a thread-based model (see “Threads” on page 397).
  - Use a maximum of 1 GB for cache. (This may cause issues if you are using the static memory cache as well.)

See 13.2, “Set up ESI caching” on page 270, for instruction on how to set up ESI caching for WebSphere Commerce.

13.2 Set up ESI caching

In this section we describe how to set up and configure ESI caching for WebSphere Commerce.

13.2.1 Prerequisites for ESI caching

Your system should be developed and set up according to the best practices above (see “Best practices” on page 269).

Your WebSphere Commerce application must be designed such that page fragments can be uniquely identified using request parameters on the URL, HTTP form data, or HTTP cookies in the request, as ESI processor has no access to HTTP session data.

We explain in “Cookie support” on page 287 how to convert HTTP session information like the attribute DC_userid (the current user) to cookies with WebSphere Commerce.

Archived

13.2.2 Configure ESI caching

In this section we outline the steps necessary to activate and configure ESI caching:

1. Install the DynaCacheEsi application.
2. Start the DynaCacheEsi application.
3. Reconfigure and restart the Web servers.
4. Modify and distribute the DynaCache configuration file.

Install the DynaCacheEsi application

First we need to install the DynaCacheEsi application that is needed for ESI caching (and invalidation) in the IBM HTTP Server Plug-in. Perform the following steps:

1. Open the Network Deployment Manager administrative console and verify whether the DynaCacheEsi application is already installed. To do so, click **Applications** → **Enterprise Applications**. If DynaCacheEsi is not displayed in this list, click **Install**. Otherwise, proceed to “External cache group settings” on page 280.

Archived

2. On the Preparing for the application installation panel, browse to the *WAS_Install_Dir*/installableApps Server path and select the **DynaCacheEsi.ear** file, as shown in Figure 13-1.

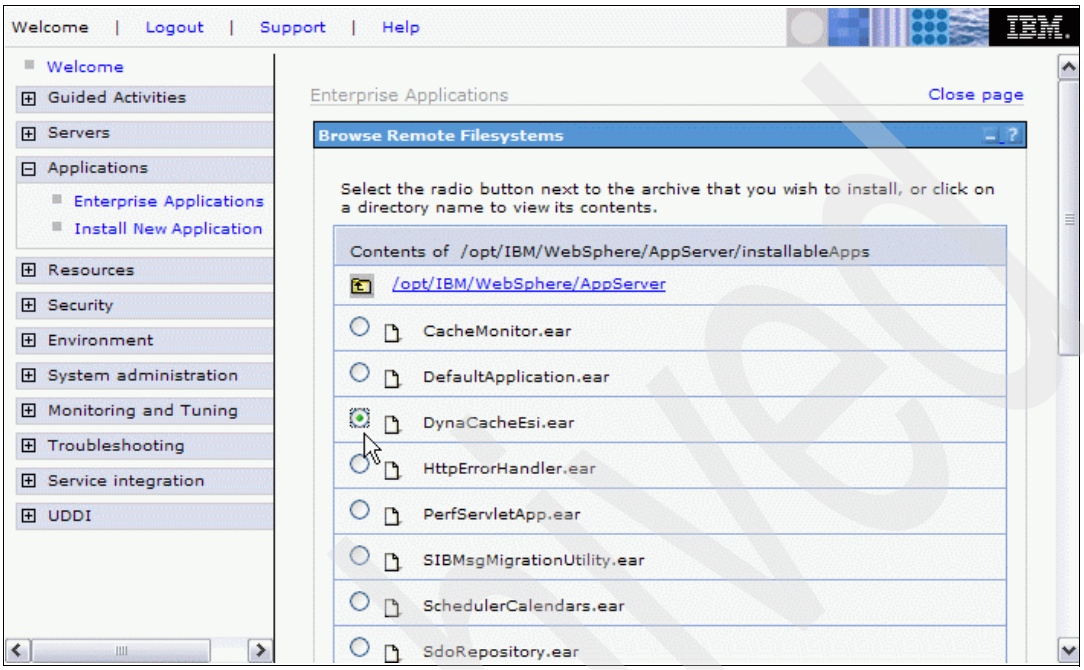


Figure 13-1 Selecting the DynaCacheEsi installation archive

Archived

3. Click **OK**. This takes you back to the Preparing for the application installation panel, as shown in Figure 13-2. Click **Next**.

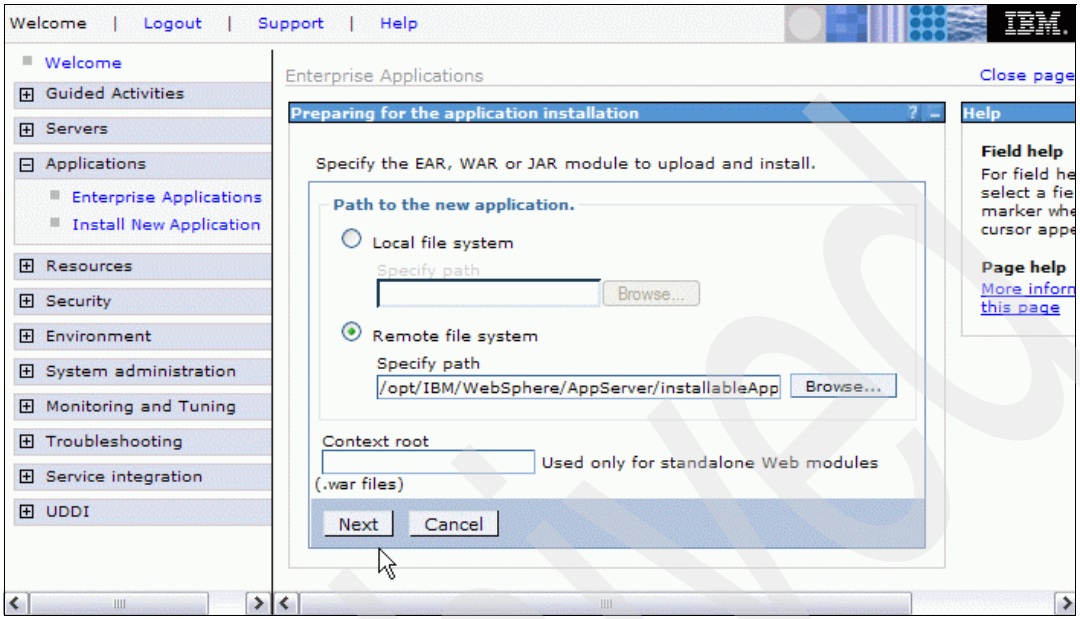


Figure 13-2 Prepare for the application installation panel

Archived

4. On the next panel, enter VH_<*Instance_Name*> for the Virtual Host, as shown in Figure 13-3 for our instance, which is named demo. (DynaCacheEsi needs to be installed using the same virtual host as the cached application.)

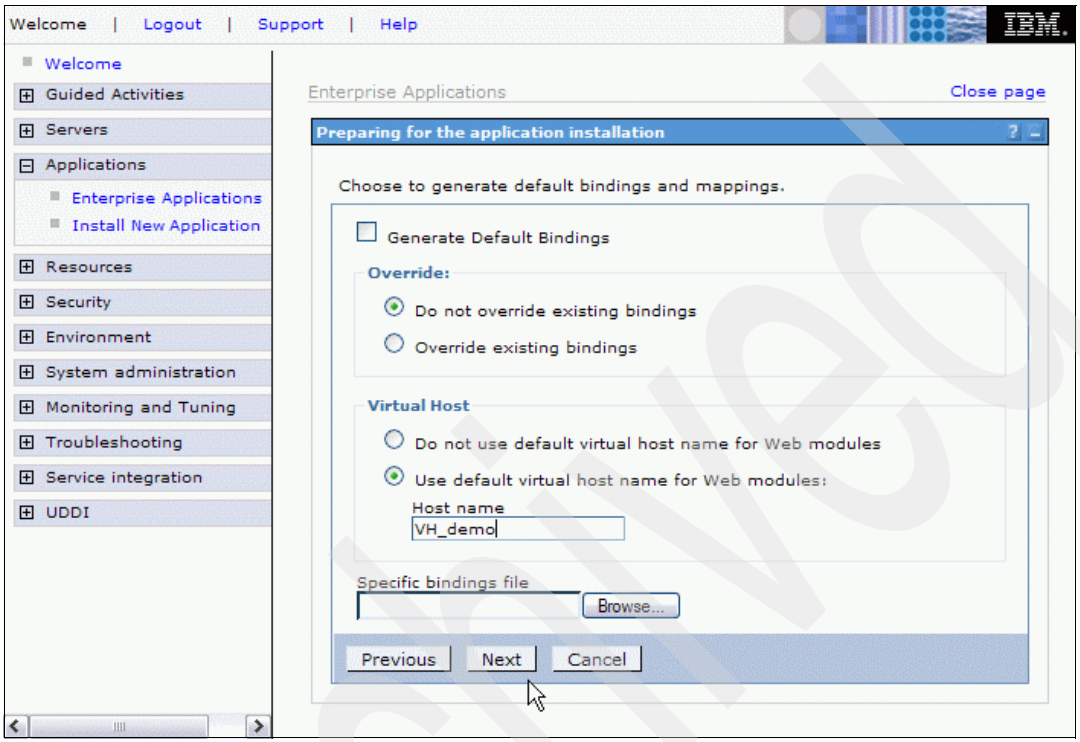


Figure 13-3 Entering the virtual host

Archived

5. Click **Next** → **Continue**. On the Application Security Warnings pane, click **Continue**. Accept the default options on the next panel (step 1) and click **Next** once again. See Figure 13-4.

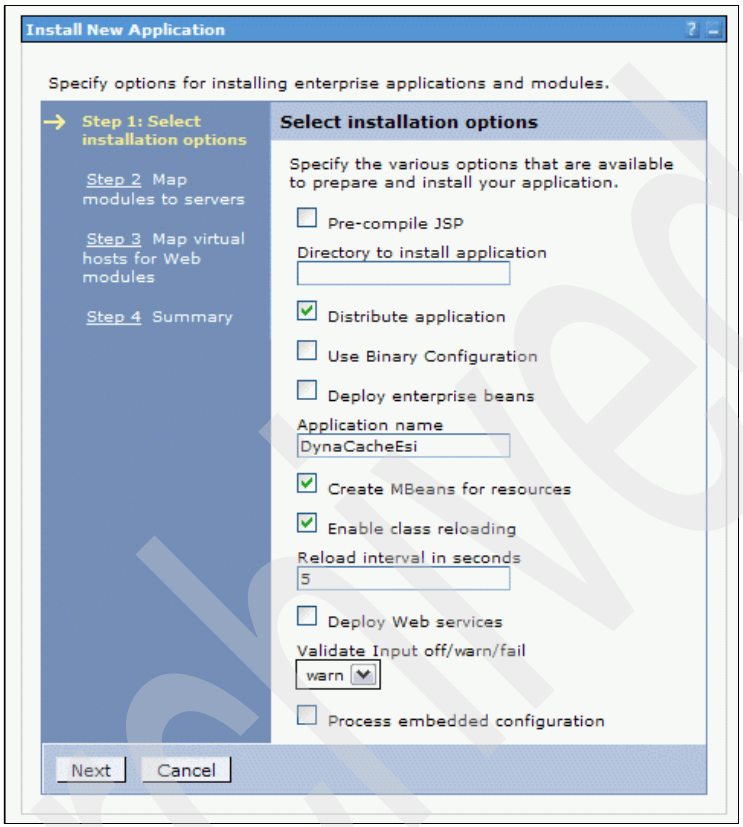


Figure 13-4 DynaCacheEsi installation - installation options

Archived

6. Step 2 - Map modules to application servers.

   On this panel, you need to select your application server cluster and all your Web servers, and click the check box in the Select column for the DynaCacheEsi Module, as shown in Figure 13-5.

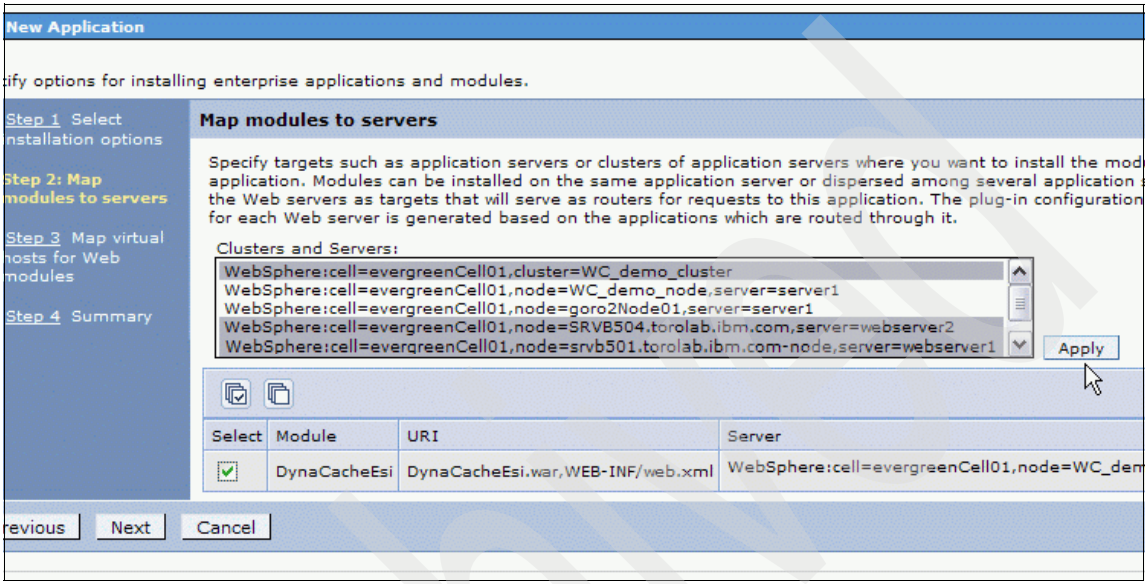


Figure 13-5 DynaCacheEsi installation - Map modules to application servers

7. Click **Apply**. In the server column next to the DynaCacheEsi module, your cluster and Web servers are now listed, as shown in Figure 13-5. Click **Next**.

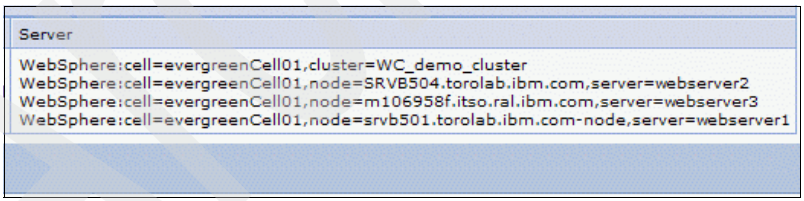


Figure 13-6 Servers selected for mapping

8. Step 3 - Map virtual hosts for Web modules (Figure 13-7).

   Check **DynaCacheEsi** and select the **VH_<*Instance_Name*>** virtual host from the pull-down menu. Then click **Next**.

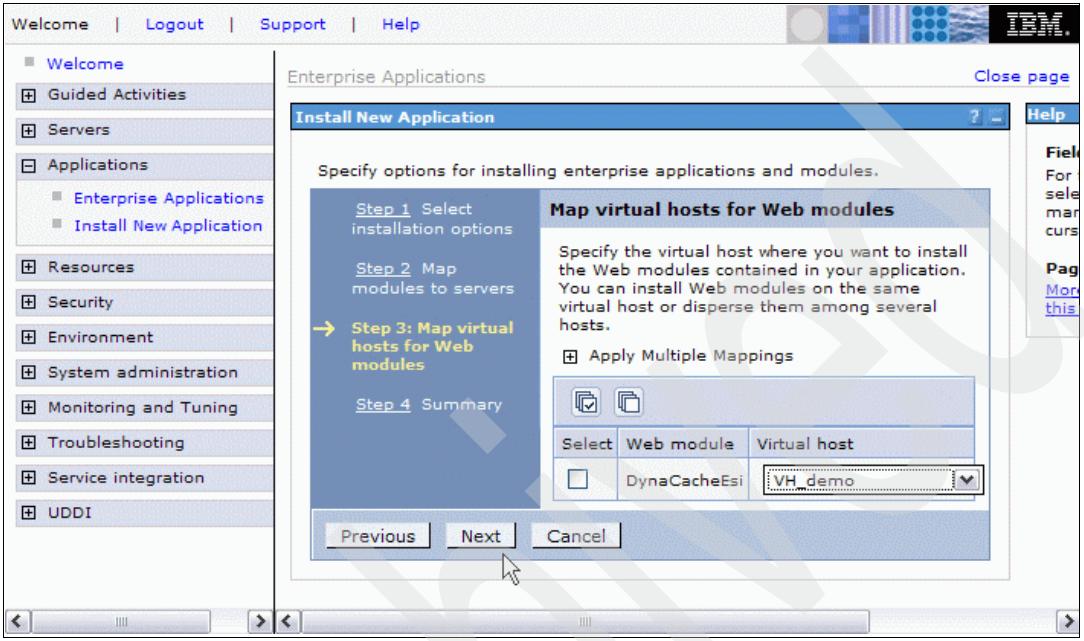


Figure 13-7 DynaCacheEsi installation - Map virtual hosts for Web modules

Archive

9. Confirm the installation on the Summary window, as shown in Figure 13-8, by clicking **Finish**.

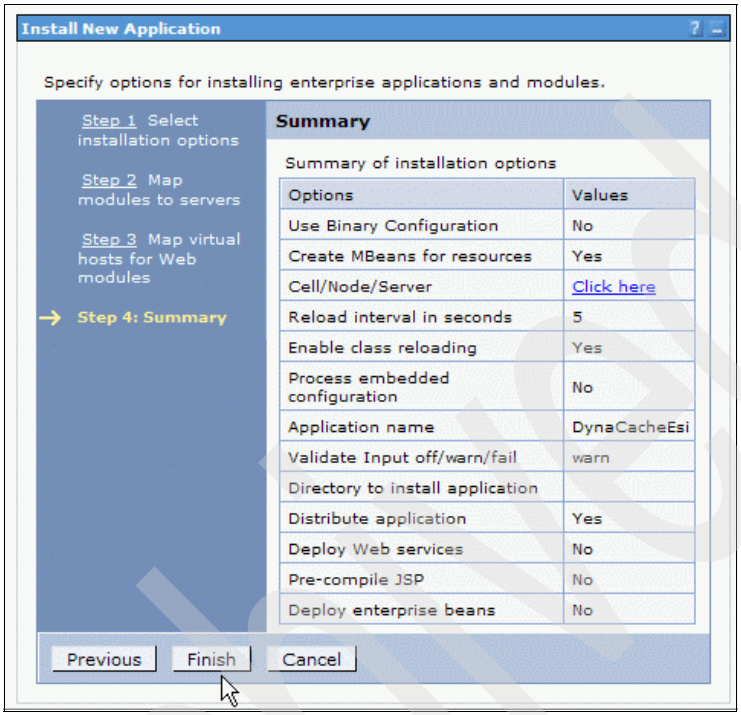


Figure 13-8 DynaCacheEsi installation - Summary

10. Once the installation has completed successfully, click **Save to Master Configuration**, as shown in Figure 13-9.

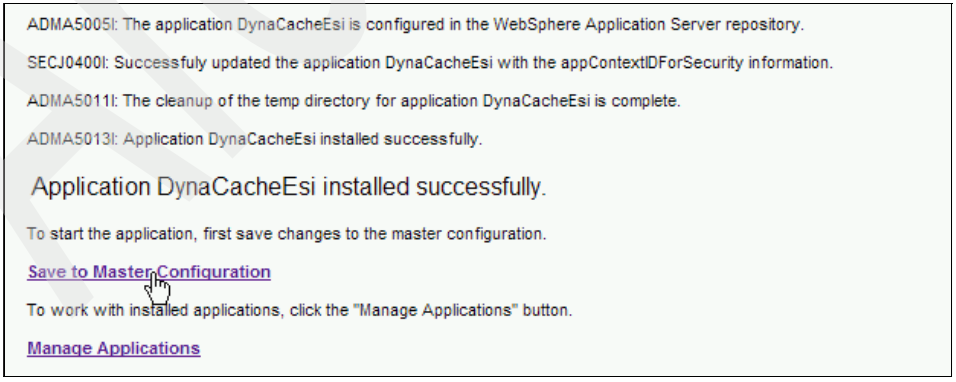


Figure 13-9 Saving your changes after installing DynaCacheEsi

11. On the following page make sure to select **Synchronize changes with nodes**, then click **Save**.

Start the DynaCacheEsi application

To start the DynaCacheEsi application through the Network Deployment Manager administrative console, navigate to **Applications** → **Enterprise Applications**, select **DynaCacheEsi**, and click **Start**, as shown in Figure 13-10.

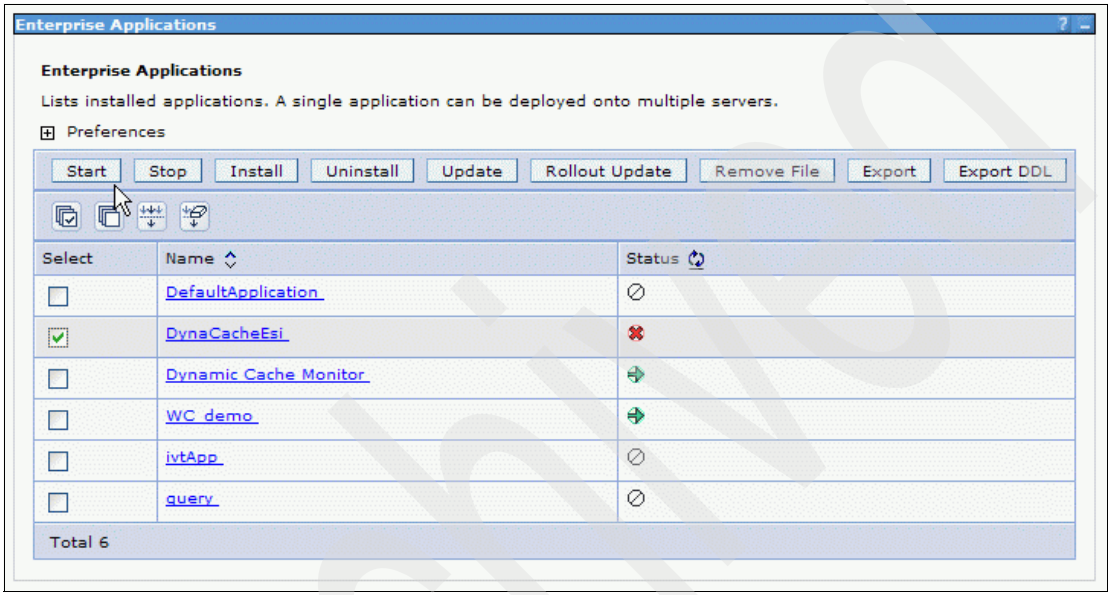


Figure 13-10 Starting the DynaCacheEsi application

When the application has started successfully on all cluster members, a message box is displayed (see Figure 13-11) and the application status is changed to started.

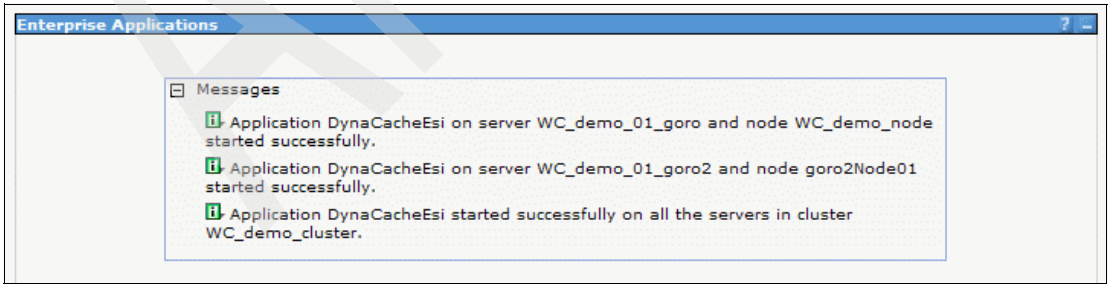


Figure 13-11 DynaCacheEsi application started successfully

External cache group settings

You need to define an external cache group controlled by WebSphere Application Server. To do so:

1. Open the Network Deployment Manager administrative console and click **Servers** → **Application servers**.
2. Select your application server from the list.
3. Select **Container Services** → **Dynamic Cache Service**.
4. Select **External Cache Groups** from the Additional Properties pane.

   This panel (shown in Figure 13-12) allows you to create, delete, or update an existing external cache group.
5. The EsiInvalidator external cache group normally exists by default. If it is not shown, click **New**. This launches the Configuration window where you can specify the name of the external cache group.

> **Tip:** In a production environment, you need to verify, and, if necessary, create the external cache groups on all application servers that are serving the cached application.

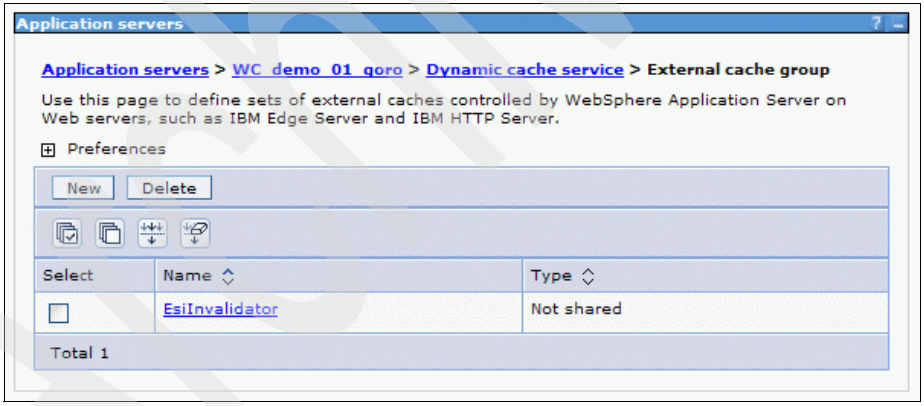


Figure 13-12 External cache groups

6. On the External cache group panel, click **EsiInvalidator**, then select **External cache group members**.

7. For the EsiInvalidator group, there should be one member by default, as shown in Figure 13-13.

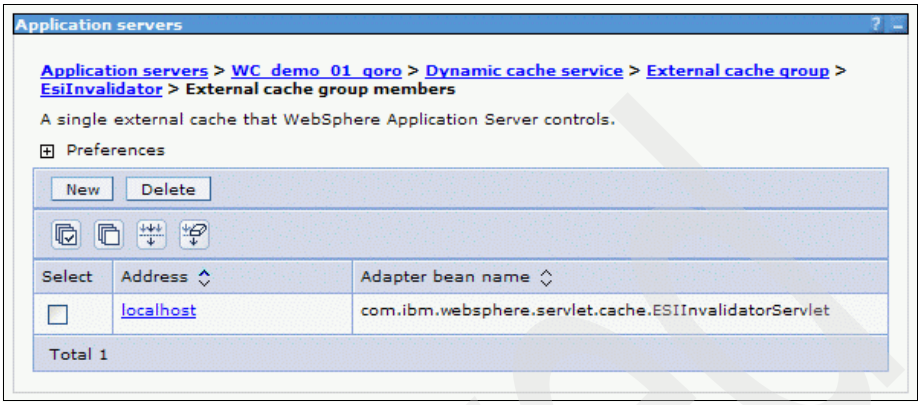


Figure 13-13 External cache group members

8. If you just created the cache group, or if the default external cache group member does not exist, click **New** to create it, and specify the following information:
   - Address:

     `localhost`
   - Adapter Bean Name:

     `com.ibm.websphere.servlet.cache.ESIInvalidatorServlet`

   Then click **OK**.
9. If you have made any changes, make sure to save your changes to the master configuration.

Reconfigure the Web servers

For ESI caching to work properly in the IBM HTTP Server Plug-in, including receiving invalidations from the application servers, you need to change the plug-in configuration file on each Web server. Example 13-1 shows the beginning of the file with ESI caching and invalidations enabled for a maximum cache size of 512 MB.

Example 13-1 The beginning of the file

```
<?xml version-"1.0"?>
<Config ...>
      ...
      <Property Name="esiEnable" Value="true"/>
      <Property Name="esiMaxCacheSize" Value="524288"/>
```

```
      <Property Name="esiInvalidationMonitor" Value="false"/>
      ...
</Config>
```

Note the following:

esiEnable This can be used to disable the ESI processor by setting the value to false. ESI is enabled by default. If ESI is disabled, then the other ESI options are ignored.

esiMaxCacheSize This is the maximum size of the cache in 1 KB units. The default maximum size of the cache is 1 MB (1,024 KB). If the cache is full, the first entry to be evicted from the cache is the entry that is closest to expiration.

esiInvalidationMonitor This specifies whether the ESI processor should receive invalidations from the application server. ESI works well when the Web servers following a threading model are used, and only one process is started. When multiple processes are started, each process caches the responses independently and the cache is not shared. This could lead to a situation where the system's memory is fully used up by the ESI processor. There are three methods by which entries are removed from the ESI cache: First, an entry's expiration timeout could fire. Second, an entry may be purged to make room for newer entries. Or third, the application server could send an explicit invalidation for a group of entries. In order for the third mechanism to be enabled, the esiInvalidationMonitor property must be set to true and the DynaCacheEsi application must be installed on the application server. The DynaCacheEsi application is located in the installableApps directory and is named DynaCacheEsi.ear. If the ESIInvalidationMonitor property is set to true but the DynaCacheEsi application is not installed, then errors will occur in the Web server plug-in and the request will fail.

Follow the steps below for each of your Web servers to configure the plug-in on all Web servers using the Network Deployment Manager administrative console:

1. In the administrative console, navigate to **Servers** → **Web servers** → **<WebServer_Name>** → **Plug-in properties** → **Caching**.

2. Activate both **Enable Edge Side Includes (ESI) processing to cache the responses** and **Enable invalidation monitor to receive navigations** and enter your desired maximum cache size (this is per ESI processor process), as shown in Figure 13-14.

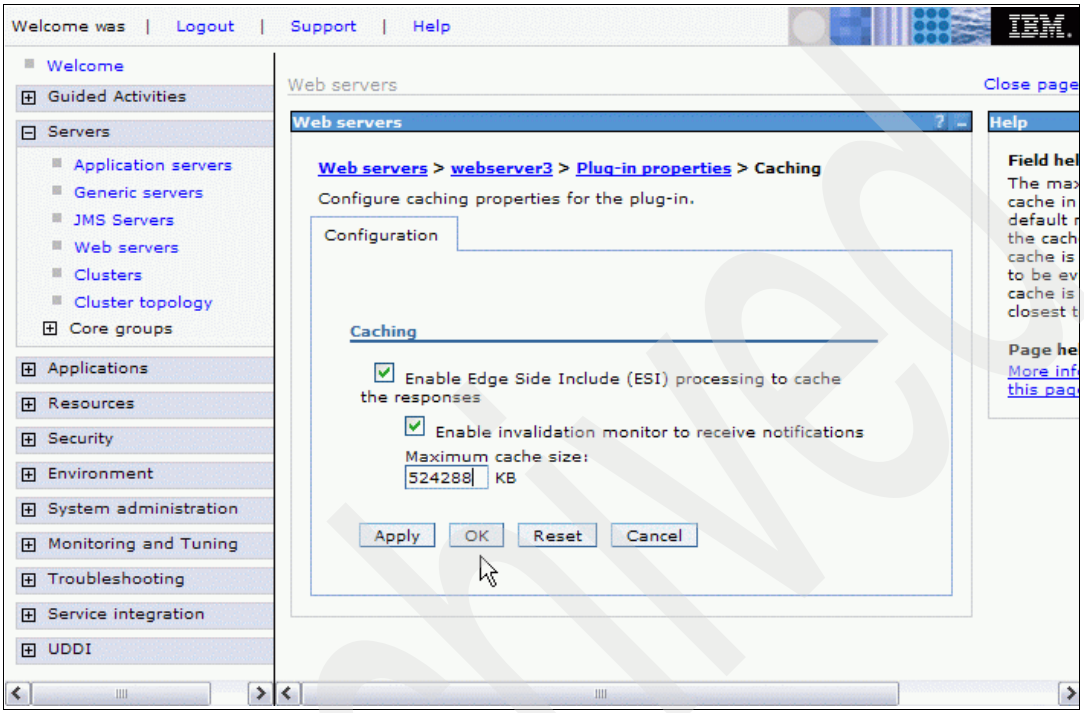


Figure 13-14 Configuring ESI caching

3. Click **OK**.
4. Save your changes to the master configuration.

After performing the steps above for each of your Web servers, you need to propagate the plug-in configuration file, plugin-cfg.xml.

You do not need to explicitly propagate the plug-in configuration file if you make all of the following settings:

- Activate both automatic plug-in configuration generation and propagation (“Update the Web server configuration” on page 176).
- Enable remote Web server management for your Web servers. (Supply an administrative user name and password to Network Deployment Manager, and start IBM HTTP Server administrative server on the Web servers.) See “Activate remote Web server management (optional)” on page 175.

You may still perform the following steps if you have remote Web server management enabled and if the administrative server is running on your Web servers, for example, if you do not want to wait until the plug-in configuration is refreshed automatically, or if automatic generation and propagation are switched off.

1. Navigate to **Servers** → **Web servers**.
2. Select all of your Web servers that have remote Web server management enabled, as shown in Figure 13-15.

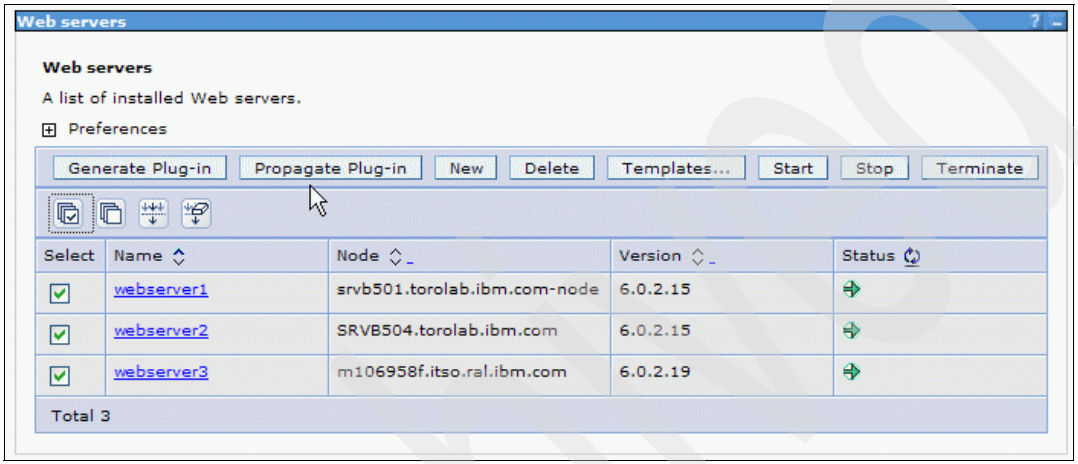


Figure 13-15 Propagating the plug-in configuration for all Web servers

3. Click **Generate Plug-in**, then **Propagate Plug-in**.

If remote Web server management is not enabled, you need to copy the plug-in configuration file manually to your Web servers (see “Propagate IBM HTTP Server Plug-in configuration” on page 183).

After propagating the plug-in configuration, stop and start your Web servers. According to “Testing ESI caching” on page 571, in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, simply reloading the plugin-cfg.xml file is not enough, and a real stop and start of the Web servers is necessary to activate ESI caching and invalidation handling.

Modify the DynaCache configuration file

To cache full pages and fragments on the Web server, you need to update the Dynamic Cache configuration file, cachespec.xml. On WebSphere Commerce node 1, the file can be located as follows:

`WAS_Install_Dir/profiles/Profile_Name/installedApps/Cell_Name/WC_<Instance_Name>.ear/Stores.war/WEB-INF/cachespec.xml`

To change the file, copy the file to your local machine and make the modifications that you need, as described in the following two sections, “Caching full pages” on page 285 and “Caching fragments” on page 286. We do not recommend changing it directly on the server, as updates to the file are effective immediately.

When you are done with editing the file, you can distribute it to all the application servers in your cluster, as described in “Distribute cachespec.xml” on page 289.

Caching full pages

To mark an entry to be cached using ESI, use the property EdgeCacheable. This property also implies the property of consume-subfragments. That is, the page will be cached as a full page, including all its subfragments, unless one of these subfragments is specified to be cacheable separately (refer to “Caching fragments” on page 286 for details). In order to cache pages with ESI, only the parameter and cookie component can be used to define the cache ID.

Example 13-2 shows our cache-entry for the ECActionServlet servlet with a sample cache-id entry for the TopCategoriesDisplay URL, which is used to test full page caching.

Example 13-2 ECActionServlet cache-entry with sample cache-id

```
<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
<cache-entry>
   <class>servlet</class>
   <name>com.ibm.commerce.struts.ECActionServlet.class</name>
   <property name="store-cookies">false</property>
   <property name="save-attributes">false</property>
   <property name="EdgeCacheable">true</property>
...
   <cache-id>
      <component id="" type="pathinfo">
         <required>true</required>
         <value>/TopCategoriesDisplay</value>
      </component>
      <component id="storeId" type="parameter">
         <required>true</required>
      </component>
      <component id="catalogId" type="parameter">
         <required>true</required>
      </component>
      <component id="categoryId" type="parameter">
         <required>false</required>
```

Archived

```
      </component>
   </cache-id>
...
</cache>
```

Caching fragments

Alternate URL is a method for edge caching JavaServer™ Pages (JSP) files and servlet responses that you cannot request externally. Dynamic cache provides support to recognize the presence of an Edge Side Include (ESI) processor and to generate ESI include tags and appropriate cache policies for edge cacheable fragments. However, for a fragment to be edge cacheable, you must be able to externally request it from the application server. In other words, if a user types the URL in her browser with the appropriate parameters and cookies for the fragment, WebSphere Application Server must return the content for that fragment.

The child JSP files are edge cacheable only if you can request these JSP files externally, which is not usually the case. For example, if a child JSP file uses one or more request attributes that are determined and set by the controller servlet, you cannot cache that JSP file on the edge. You can use alternate URL support to overcome this limitation by providing an alternate controller servlet URL used to invoke the JSP file.

The alternate URL for a JSP file or a servlet is set in the cachespec.xml file as a property with the name alternate_url. You can set the alternate URL either on a per cache-entry basis or on a per cache-id basis. It is valid only if the EdgeCacheable property is also set for that entry. If the EdgeCacheable property is not set, the alternate_url property is ignored.

A good example for WebSphere Commerce is a JSP, which displays a personalized mini shopping cart, and that is included by fully cached pages, for example, TopCategoriesDisplay (see above). See Example 13-3 on page 288 for the cachespec.xml entry for the mini shopping cart JSP fragment.

In order to cache the TopCategoriesDisplay page and the mini shopping cart on the edge, we need to construct cache ID rules that only contain URL parameters or cookies. For the store catalog display page, it will not be a problem since all the information needed to cache it is on the URL. However, since the mini current order display is unique per user, we have to use the user's ID as the cache ID.

Since only URL parameters and cookies can be used to define the cache ID rule, and the URL does not contain the user's information, the only other way is to use a cookie that contains the user's ID information as a component of the cache ID.

Cookie support

Since Version 6.0, WebSphere Commerce provides some cookies that can be used as part of the cache-id. The DynaCache Event Listener listens to the session change events triggered by the user ID or store ID change and then performs the following actions:

- Deletes all old session cookies (if they exist).
- Creates new session cookies based on the settings of the com.ibm.commerce.dynacache.DynaCacheCookie object and data obtained from the basic catalogue structure.
- Each cookie is given an expiry period of one day.
- Each cookie is hashed using a one-way hash of the value + merchant key + today's date (yyyymmdd).

To use cookies, perform one of the following steps:

- Single store scenario

  Add the following component entry in the cachespec.xml file inside the <cache-id> tags of <cache-entry> elements that are dependent on session information.

```
<component id="<Component_ID>" type="cookie">
   <required>true</required>
</component>
```

  Where *<Component_ID>* is one of the values listed in Table 13-1.

Table 13-1 Cookie component IDs

Component_ID	**Definition**
WC_LANGID	Language ID
WC_CURRID	Currency ID
WC_PROG	Parent organization
WC_CACHEID1	Contract ID
WC_CACHEID2	Member groups
WC_CACHEID3	Buyer contract ID
WC_CACHEID4	User ID
WC_CACHEID5	User type

Archived

Example 13-3 shows how this would look like for the mini shopping cart display, using WC_CACHEID4 to include the user ID in the cache-id.

Example 13-3 Using cookies in the cache-id (single store)

```
<cache-entry>
   <class>servlet</class>
   <name>/ConsumerDirect/include/MiniShopCart.jsp</name>
   <property name="EdgeCacheable">true</property>
   <property name="alternate_url">/servlet/ConsumerDirect/
include/MiniShopCart.jsp</property>
   <property name="save-attributes">false</property>
   <property name="do-not-consume">false</property>
   <cache-id>
      <component id="WC_CACHEID4" type="cookie">
         <required>true</required>
      </component>
   </cache-id>
</cache-entry>
```

- Multi store scenario

  Insert a <cache-id> entry for each store into the cachespec.xml with the structure and properties shown in Example 13-4 for the mini shopping cart display.

Example 13-4 Using cookies in the cache-id (multi store)

```
<cache-entry>
   <class>servlet</class>
   <name>/ConsumerDirect/include/MiniShopCart.jsp</name>
   <property name="EdgeCacheable">true</property>
   <property name="alternate_url">/servlet/ConsumerDirect/
include/MiniShopCart.jsp</property>
   <property name="save-attributes">false</property>
   <property name="do-not-consume">false</property>
   <cache-id>
      <component id="WC_CACHEID4_10001" type="cookie">
         <required>true</required>
      </component>
   </cache-id>
   <cache-id>
      <component id="WC_CACHEID4_10002" type="cookie">
         <required>true</required>
```

Archived

```
      </component>
    </cache-id>
</cache-entry>
```

At runtime the cookie generator dynamically produces a cookie named based on the storeId. For example, given the storeId 10001 and an English store, the cookie WC_LANGID_10001=-1 will be generated.

Distribute cachespec.xml

Once your cachespec.xml file is complete, you need to distribute it to your application servers. To do this:

1. Open the Network Deployment Manager administrative console and navigate to **Applications** → **Enterprise Applications**.
2. Select your WebSphere Commerce application, WC_<*Instance_Name*>, as shown in Figure 13-16. In our case, the application is WC_demo.

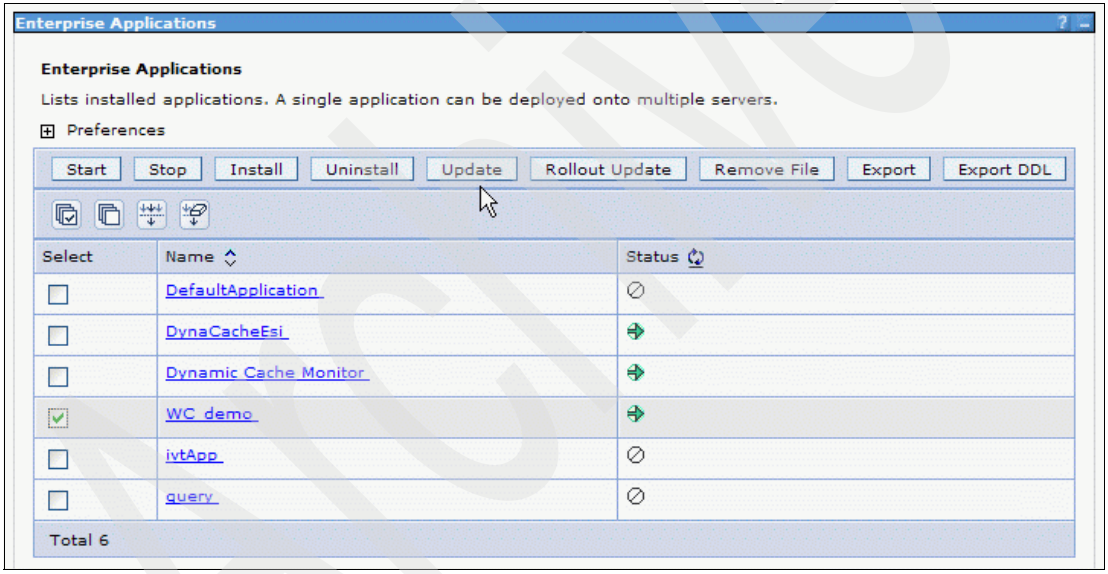


Figure 13-16 Updating an application

3. Click **Update**. On the Preparing for the application installation page, select **Single file** and enter `Stores.war/WEB-INF/cachespec.xml` in the Relative path to file field. The relative path always starts at the root of the Enterprise Application Archive (EAR). In the Upload the new or replacement files box, choose **Local filesystem** and specify your modified cachespec.xml file using the full path and filename.

Figure 13-17 shows the settings for our scenario.

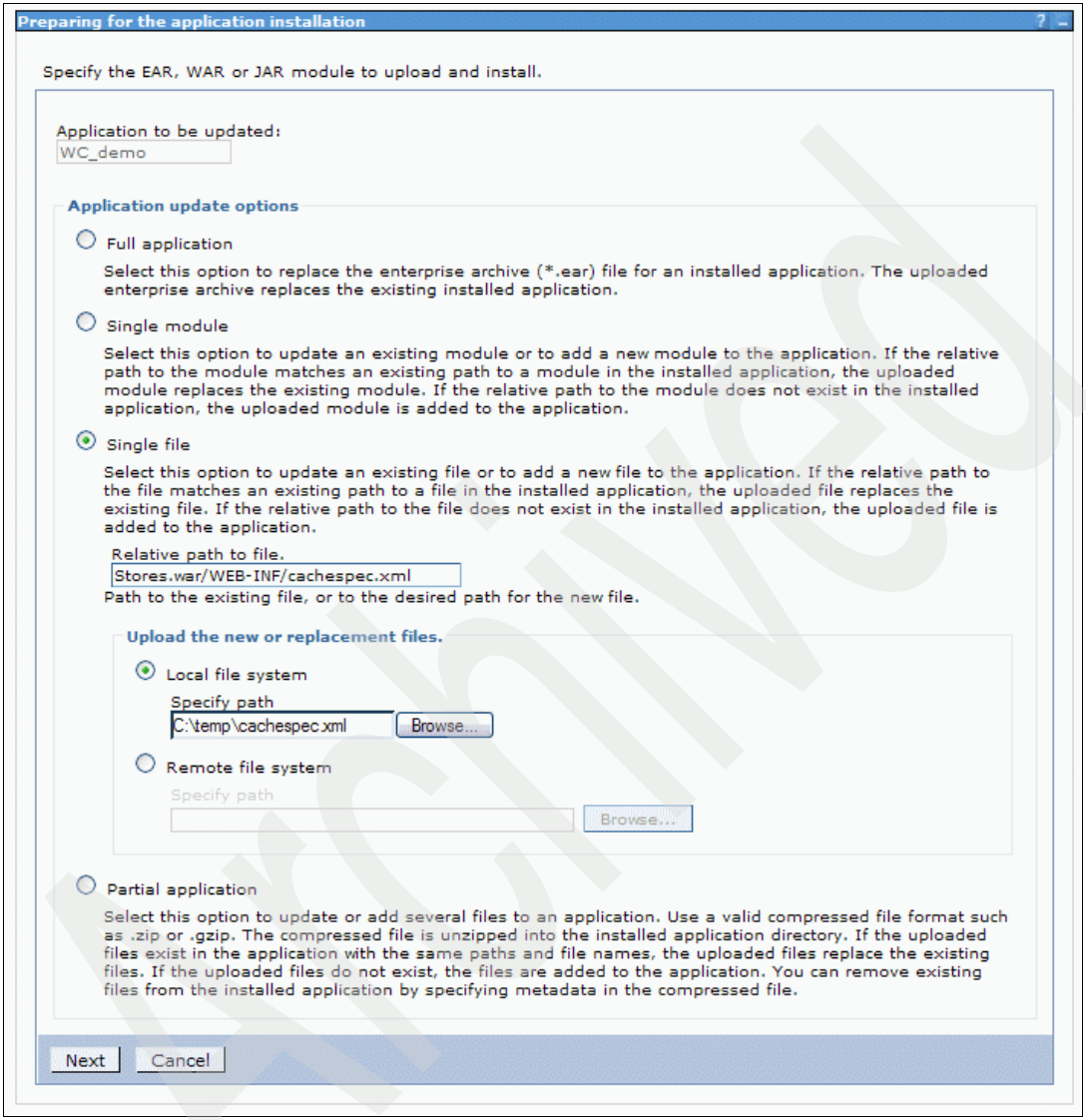


Figure 13-17 Specifying the cachespec.xml update

Archived

4. Click **Next**. A confirmation page is displayed stating the relative path just entered. See Figure 13-18.

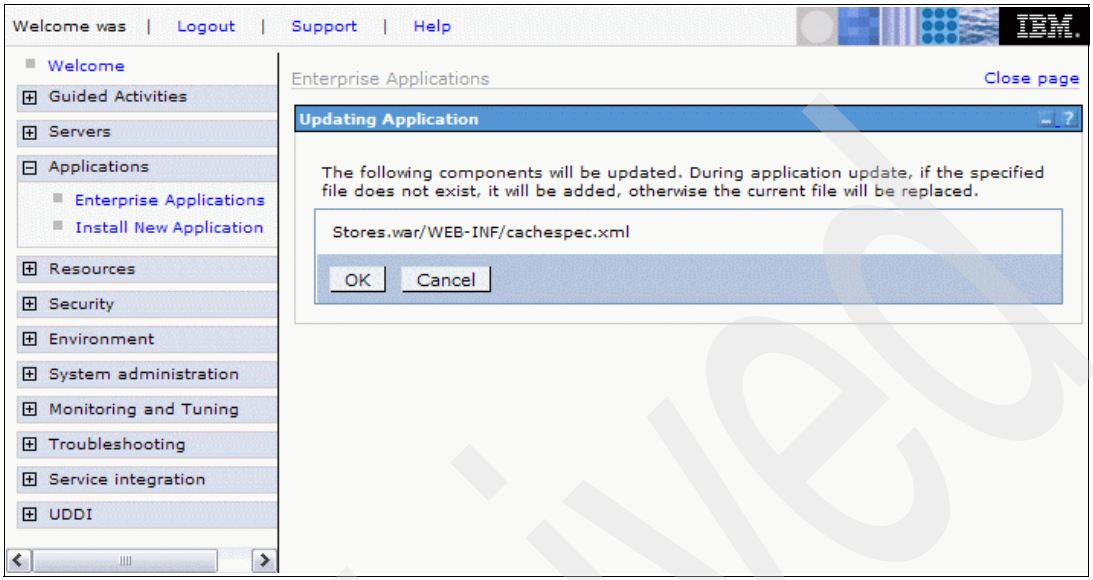


Figure 13-18 Update confirmation

5. Click **OK**. The application is now patched. This step only updates the Network Deployment Manager configuration. The updated file is not yet distributed to the application servers.
6. When the installation has successfully completed, save the changes to the master configuration by clicking **Save to master configuration**.
7. On the Save page, make sure that **Distribute changes to nodes** is activated, then click **Save**. This distributes the updated file to all application servers, where the changes become effective immediately after the file has been copied.

The ESI caching configuration is now complete.

Archived

13.3 Caching enhancements in WebSphere Commerce 6.0.0.1 and later

In WebSphere Commerce 6.0.0.1 and later, you now have the option to enable additional WebSphere Commerce runtime command caching to further enhance performance. For details on how to do this, follow this link:

http://www-1.ibm.com/support/docview.wss?uid=swg21246721

13.4 Cache replication

A large WebSphere Commerce environment utilizes WebSphere Application Server clustering to achieve scalability and high availability. Dynamic cache is implemented to improve application response times and reduce system load. In a clustered environment, each JVM (cluster member) has a cache instance that generates its unique cache content, in memory first with the option to offload excessive cache content to disk.

WebSphere Commerce runtime creates many cached objects in the background. Hence, data replication service is required in a clustered environment to ensure that these runtime cached objects are invalidated across the cluster.

When designing your cache strategy in this highly available environment, a few important questions must be answered:

- What options are available to ensure the consistency of cache content across a cluster?
- What are the pros and cons of each option?
- What does cache replication do?
- How does cache replication work?
- How much performance improvement does cache replication provide?
- What is the performance overhead of cache replication?
- How will you tune cache replication to reap the most benefit out of it and reduce the performance overhead?
- How will you configure cache replication?
- What does WebSphere Commerce recommend with respect to cache replication?

This section answers these questions.

13.4.1 Cache replication

Cache replication is a WebSphere Application Server service that provides the following benefits:

- Data is generated one time and copied or replicated to other servers in the cluster, saving time and resources and aiding in cache consistency.
- Cache entries that are not needed are removed or replaced on all cluster members, again, aiding in cache content consistency across the cluster.

The data replication configuration can exist as part of the Web container dynamic cache configuration accessible through the administrative console, or on a per-cache entry basis through the cachespec.xml file. The sharing policy set on a particular cache-entry overrides the default sharing policy set on the replication domain.

Cache replication can take on various forms:

- PUSH - Cache entries for this object are automatically distributed to the dynamic caches in peer-distributed servers.
- PUSH/PULL - Cache entries for this object are shared between application servers on demand. When an application server generates a cache entry, it broadcasts the cache ID of the created entry to all cooperating application servers. Each server then knows whether an entry exists for any given cache ID. On a given request for that entry, the application server knows whether to generate the entry or pull it from somewhere else. When using PUSH-PULL, if the WebSphere Application Server level is greater than 6.0.2.15, ensure that you have the following APARS: PK32424, PK27694, PK36676.

The dynamic cache service broadcasts cache replication updates asynchronously, based on a configurable batch update interval (on the dynamic cache service administrative console panel) rather than sending them immediately. Invalidations are sent immediately. Distribution of invalidations prevents stale data from residing in a cluster.

13.4.2 In-memory cache

A newly generated cache entry is stored in memory (JVM) until it fills up. The maximum number of cache entries that can be stored in the JVM is controlled by the Cache size parameter, configurable in the administrative console. The default is 2000.

It is extremely important to monitor the heap consumed by the cache using verbose GC tracing or heap dump analysis.

Archived

This helps in determining the size of the in-memory cache. Using the GC logs during performance testing, one should arrive at the appropriate size of the in-memory cache.

The in-memory cache size is limited by the size of the JVM. As a rule of thumb, we recommend leaving 40% of JVM free with caching to avoid excessive fragmentation and out-of-memory problems causing crashes. Gradually increase the cache size and fine-tune the JVM to reduce fragmentation.

13.4.3 Offload to disk

By default, the dynamic cache maintains the number of entries that are configured in memory. If new entries are created while the cache is full, the priorities that are configured for each cache entry, and a least recently used algorithm, are used to remove entries from the cache. In addition to having a cache entry removed from memory when the cache is full, you can enable disk offload to have a cache entry copied to the file system (the location is configurable). Later, if that cache entry is requested, dynamic cache first evicts a cache entry from JVM based on priority and LRU algorithm, and then re-inserts the needed cache entry into memory.

The disk cache size can be controlled in WebSphere Application Server Version 6.0.2.17 or later.

The disk cache size can be controlled in terms of the number of entries or size of disk. You can do so by setting JVM custom properties com.ibm.ws.cache.CacheConfig.diskCacheSizeInGB and com.ibm.ws.cache.CacheConfig.diskCacheSize. The technote found at http://www-1.ibm.com/support/docview.wss?uid=swg27007969 elaborates on all the disk cache enhancements. We recommend that the customer use SAN disks for hosting the disk cache for performance-critical real-time applications.

In theory, disk cache is only limited by file system size. However, in practice you must consider the performance overhead versus performance gain.

Larger disk cache size is dependent on how fast the information can be written to disk and retrieved from it. If you allow disk cache to grow without limit, eventually you will reach a point where it is faster to regenerate a cache entry from the database than to look it up from disk.

The ideal disk cache size must be determined in an endurance test.

A PMI counter, HitsOnDisk, may be used to monitor the number of hits on disk cache.

Archived

13.4.4 FlushToDiskOnStop

Cache entries can also be written to disk when the server is stopped.

There are two ways to achieve this:

- Enable the Flush to disk option in the administrative console.
- Set the com.ibm.ws.cache.flushToDiskOnStop custom JVM property to true.

When this custom property is set to true, even if Flush to disk is not selected in the administrative console, upon a server stop, it will flush in-memory cache to disk and also retain the over flowed offloaded-to-disk cache.

To verify whether FlushToDiskOnStop is enabled or disabled, check the SystemOut.log on server startup for this message:

```
Cache         I   DYNA0061I: Flush to disk on stop is enabled for cache
name "baseCache".
```

or

```
Cache         I   DYNA0061I: Flush to disk on stop is disabled for
cache name "baseCache".
```

If you do not want to retain any cache entries (in-memory or on disk) when the server stops, you must ensure the following:

- The Flush to disk option is not selected in the Dynamic Cache Service Configuration page.
- The custom JVM property com.ibm.ws.cache.flushToDiskOnStop is either not present or is set to false in the JVM Custom Properties Configuration page.

13.4.5 Limitation on invalidation when server is stopped

When a server is stopped, it can no longer receive and process any cache replication requests.

There is a limitation on invalidation when a server is stopped.

Consider this scenario. FlushToDiskOnStop is enabled, and upon a server stop, in-memory and disk cache are preserved on the file system.

One server in the cluster is stopped, but the other ones are running and processing invalidation requests. Often the invalidated cache entry also exists in the disk cache of the server that is stopped. It will not have the ability to remove this cache entry from its disk cache, and therefore when this cluster member starts back up, its disk cache will contain outdated cache entries.

Archived

Because of this limitation, in order to keep the cache content consistent, you may wish to disable FlushToDiskOnStop in a clustered environment. The downside to this is that when a server starts up, its cache content must be rebuilt. As such, the initial response time from this server will be slow.

13.4.6 Performance tuning

We highly recommend that you apply the latest WebSphere Commerce and WebSphere Application Server fix pack to take advantage of recent performance enhancements. In addition to applying the fix packs, you will need to consider the topics discussed in this section when tuning for dynamic cache in a clustered environment.

Control DRS message size

Cache replication distributes cache entries and invalidations in the form of DRS messages.

In the past when we had no ability to control DRS message size, depending on the size of your cache, the DRS messages generated can be very large, often times hundreds of KB or several MB in some extreme cases. A busy WebSphere Commerce JVM is usually fragmented. Finding this much contiguous space in a fragmented JVM is often impossible, and when this happens, an Out Of Memory condition is triggered and causes the JVM to crash.

A solution is now available to address this problem.

With two new APARs or WebSphere Application Server fix pack 6.0.2.19 or later, you can now control the DRS message size and replicate smaller DRS objects and in batches.

The two APARs are:

- PK32201: OutOfMemory DUE TO LARGE DRS MESSAGES
- PK35824: Extending PK32201 to batching of Invalidation events

The batch size of the replication data can now be configured using the following custom properties:

- com.ibm.ws.cache.CacheConfig.cachePercentageWindow

  This specifies a limit on the number of cache entries sent by the Data Replication Service in terms of the percentage of total cache in memory.

  - Default value: 2% of the number of entries in the cache
  - Scope: configurable per-cache instance

Archive

- com.ibm.ws.cache.CacheConfig.cacheEntryWindow

  This specifies a limit on the total number of cache entries sent by the Data Replication Service in terms of number of entries.

  - Default value: 50 entries
  - Scope: configurable per cache instance

- com.ibm.ws.CacheConfig.batchUpdateMilliseconds

  This specifies a batch update frequency in terms of milliseconds. Setting this property will result in Dynacache processing updates more frequently, and thus reducing the payload size. The default value is 1000 ms (1 s).

Tune in-memory cache size

The default cache size is 2000. This number represents the maximum number of cache entries that can be stored in memory. Multiplying your average page size with the cache size will give you the approximate heap size occupied by dynamic cache.

Start with the default cache size and monitor the heap utilization by cache. Gradually increase this number if there is sufficiently free heap space.

Filter out expiration and LRU-based invalidation events

The dynamic caching service sends invalidation notifications to all cluster members in all situations. In some cases, these invalidation events are unneeded and can become a performance drain.

Two custom JVM properties can be used to disable the sending of notifications in two situations: invalidation based on LRU eviction and invalidation based on time out.

WebSphere Commerce recommends that you set these custom JVM properties to TRUE:

- com.ibm.ws.cache.CacheConfig.filterLRUInvalidation
- com.ibm.ws.cache.CacheConfig.filterTimeOutInvalidation

These two new properties are available in WebSphere Application Server fix pack 6.0.2.13 and later. Fro more details, refer to:

http://www-1.ibm.com/support/docview.wss?rs=0&uid=swg24012317

Archived

13.4.7 Tune disk cache

When tuning and testing with disk cache, you should consider the parameters, disk cache size, offload to disk location, and other disk cache performance enhancements discussed in this section.

Control disk cache size

Do not let disk cache grow to an unlimited size. If you run out of disk space because the disk cache is too large, your server will become non-responsive.

Control disk cache size by setting this custom JVM property com.ibm.ws.cache.CacheConfig.diskCacheSizeInGB.

Start with a few GB and gradually increase that if the response time with writing to disk cache and retrieving from it is still within target. This needs to be monitored closely in your load test.

Configure offload location

Direct the offload to disk location to a separate file system that has sufficiently large and preferably dedicated space.

If the disk offload location is not specified, the default location ${WAS_TEMP_DIR}/node/server name/_dynacache/cache JNDI name is used.

If the disk offload location is specified, the node, server name, and cache instance name are appended. For example, ${USER_INSTALL_ROOT}/diskoffload generates the location as ${USER_INSTALL_ROOT}/diskoffload/node/server name/cache JNDI name. This value is ignored if disk offload is not enabled.

The default value of the ${WAS_TEMP_DIR} property is ${USER_INSTALL_ROOT}/temp. If you change the value of the ${WAS_TEMP_DIR} property after starting WebSphere Application Server, but do not move the disk cache contents to the new location:

- The application server creates a new disk cache file at the new disk offload location.
- If the Flush to disk setting is enabled, all of the disk cache content at the old location is lost when you restart the application server.

When you are specifying a directory, consider the following:

- If you expect to cache a large number of objects or large objects that will be around for some time, consider using a separate disk drive if you are using

Archived

Windows operating systems, or a separate file system if you are using UNIX platforms.

- If you use the default directory and the disk fills up, WebSphere Application Server could possibly stall if it needs to write messages to log files and there is no more space.
- If you specify a directory such as /tmp on UNIX platforms and that directory fills up, you may have trouble logging onto the system.
- Depending on the operating system, you may see disk full messages on the console.
- When specifying a non-default location on a UNIX system, ensure that proper read/write permission is set on the directory so the WebSphere Application Server instance owner (for example, wasuser) can write to/read from the offload to disk directory.

Additional disk cache tuning parameters

In this section we discuss additional disk cache tuning parameters.

com.ibm.ws.cache.CacheConfig.diskCachePerformanceLevel

This property indicates the performance level to tune the performance of the disk cache. Valid values are 0, 1, 2, or 3:

- 0: low performance and low memory usage

  A limited amount of metadata is kept in memory.

  Set htodCleanupFrequency in minutes (1 to 1440). A value of 0 indicates that cleanup runs only at midnight.

- 1: balanced performance and balanced memory usage

  Some metadata is kept in memory. Use the default settings to provide an optimal balance of performance and memory usage for most users.

  Set htodCleanupFrequency in minutes (1 to 1440). A value 0 indicates that cleanup runs only at midnight.

- 2: custom performance and custom memory usage

  This explicitly configures the memory that is consumed by the disk cache.

  - Set htodDelayOfIloadEntriesLimit (<=100).
  - Set htodDelayOffloadDepIdBuckets (>= 100).
  - Set htodDelayOffloadTemplateBuckets (>= 10).
  - Set htodCleanupFrequency in minutes (1 to 1440). A value of 0 indicates that cleanup runs only at midnight.

Archived

- 3: High performance and high memory usage

  All of the metadata, including dependency IDs and templates, are kept in memory.

  There is no need to perform a disk scan for expiration, for example, no cleanup frequency.

WebSphere Commerce recommends that you set it to 2 or 3.

Our studies have shown that there is minimal memory overhead using 3. It should definitely be experimented with during load testing. If the heap utilization is acceptable we recommend using 3 along with a random eviction policy.

We never recommend 0.

com.ibm.ws.cache.CacheConfig.htodDelayOffloadEntriesLimit

This property provides a way to limit the buffering of dependency and template information. This limit occurs by specifying an upper bound on the number of cache entries that any specific dependency can contain for buffering in memory. If there are more entries per dependency than this limit, the dependency and template information is written to the disk.

The default is 1000.

WebSphere Commerce recommends that you set it to 100000.

com.ibm.ws.cache.CacheConfig.htodCleanupFrequency

This property specifies the frequency at which the disk cache cleanup daemon removes expired entries from the disk cache.

The default value is zero (0), which means that cleanup is scheduled to run at midnight.

WebSphere Commerce recommends that you set this to 60 (1 hour).

If you have cache entries with small time-out values, you might need to reduce this cleanup frequency.

com.ibm.ws.cache.CacheConfig.htodDelayOffload

This property specifies whether extra memory buffers should be used for dependency IDs and templates to delay disk offload and to minimize input and output operations to the disk. This property is enabled by default.

WebSphere Commerce recommends that you explicitly set it to true if you had previously set it to false.

Archive

com.ibm.ws.cache.CacheConfig.disableTemplateInvalidation

This property disables the template-based invalidations during JSP reloads.

WebSphere Commerce recommends that you set it to true.

To configure the aforementioned custom properties:

1. In the administrative console, click **Servers** → **Application servers** → **server_name** → **Java and process management** → **Process definition** → **Java virtual machine** → **Custom properties** → **New**.
2. Type the name of custom properties. Include the full property path.
3. Type a valid value for the property in the Value field.
4. Save the property and restart WebSphere Application Server.

13.4.8 Instructions to set up cache replication

Follow the instructions below to set up cache replication in a clustered environment.

Create a replication domain

All components that need to share information must be in the same replication domain. There are two ways to create a replication domain.

1. Create a replication domain when you create the WebSphere Commerce cluster:
   a. When creating the WebSphere Commerce cluster, you can select the option **Create a replication domain for this cluster** and a replication domain with the same name as the cluster name will be created. This option is not selected by default.
   b. Finish creating the WebSphere Commerce cluster.
   c. Verify that a replication domain is created. From administrative console, click **Environment** → **Replication domains**.
   d. Select **Entire Domain**.
2. If you did not create a new replication domain when creating the WebSphere Commerce cluster, you can do so after the cluster is created. From the administrative console, click **Environment** → **Replication domains** → **New** → **<Fill in the Name field>** → **Select Entire Domain**.

When all the changes are made, ensure that you save the changes to the master configuration repository and synchronize to all nodes.

Click **OK** when the changes are saved and the nodes are synchronized.

Archived

Associate a server to replication domain

Now that the replication domain is created, you must also assign it to the individual servers that will participate in replication. Follow these steps:

1. From the administrative console, select the application server, click **Container Services** → **Dynamic Cache Service**, look under Consistency settings, and select **Enable cache replication**.
2. In the drop-down menu under Full group replication domain, select the replication domain that you created in the previous step.
3. For Replication type, select **Push only**.
4. Leave everything else as the default.
5. Click **OK**.
6. Repeat steps 1–5 for all cluster members, save all changes, and synchronize to all nodes.
7. Restart the WebSphere Commerce cluster.

Configure sharing policy in cachespec.xml

In some instances, you may want to overwrite the replication type configured in the administrative console. For example, to further reduce the overhead of sending too many DRS messages, you can choose to let each server generate its own cache and not share it with other cluster members. You can do so by setting the sharing-policy element to not-shared for the specific cache entry in cachespec.xml. Note that invalidation DRS messages are still replicated to all cluster members so the cache content on every server is valid. If this element is not present, then it defaults to the replication type configured in the administrative console.

This property does not affect distribution to Edge Side Include processors through the edge fragment caching property.

The sharing policy in the cachespec.xml maps to replication type defined in the administrative console as follows:

- not-shared (cachespec.xml) and Not Shared (console)

  Cache entries for this object are not shared among different application servers. These entries can contain non-serializable data. For example, a cached servlet can place non-serializable objects into the request attributes, if the <class> type supports it.

- shared-push (cachespec.xml) and Push only (console)

  Cache entries for this object are automatically distributed to the dynamic caches in other application servers or cooperating Java virtual machines

Archived

(JVMs). Each cache has a copy of the entry at the time it is created. These entries cannot store non-serializable data.

- shared-push-pull (cachespec.xml) and Both push and pull (console)

  Cache entries for this object are shared between application servers on demand. When an application server generates a cache entry, it broadcasts the cache ID of the created entry to all cooperating application servers. Each server then knows whether an entry exists for any given cache ID. On a given request for that entry, the application server knows whether to generate the entry or pull it from somewhere else. These entries cannot store non-serializable data.

The following example shows a sharing policy of not-shared:

```
<sharing-policy>not-shared</sharing-policy>
```

13.4.9 Other options to ensure cache content consistency across cluster

Cache replication is the recommended way to maintain cache content consistency in a clustered environment. However, due to its limitation on invalidation when a server is stopped, some clients chose other options to maintain cache content consistency on invalidation without using cache replication. Here are two examples: using a database and using a file system.

Invalidate cache entry per JVM using database

You can uniquely identify each JVM with this custom JVM property for WebSphere Commerce:

```
com.ibm.commerce.scheduler.SchedulerHostName
```

Set the value to NodeName.ServerName so that it is unique within the cell.

Next create a scheduled DynaCacheInvalidation job to run on a particular JVM. When calling the AddJob command to create the new schedule job, specify the host parameter and use the value that uniquely identifies the scheduler process. If you want to manually change an existing job, update the SCCHOST column in the SCHCONFIG table for the job that you want to run on a particular instance.

This option would eliminate the need for a replication domain. However, for each invalidation request, we need to create a SCHCONFIG record per JVM, one for each cluster member so they each run their individual DynaCacheInvalidation job. This introduces overhead on the database.

Archived

If you do not have too many cluster members (for example, 2), and the response time is acceptable when each server generates its own cache content without sharing, then you may want to explore this option.

Invalidate cache entry per JVM using file system

If you do not want to use the database to track the invalidation requests, you can implement customization so each JVM keeps track of its invalidations on the file system.

13.4.10 Monitor runtime cache

It is vital to monitor and analyze the health of the runtime cache during performance tests and in production. We recommend the following tools:

- Dynamic Cache Statistics Collector and Visualizer for IBM WebSphere Application Server

  http://www.alphaworks.ibm.com/tech/cacheviz

- IBM Extended Cache Monitor for IBM WebSphere Application Server

An Extended Cache Monitor exists that contains the following two enhancements:

- Display the contents of object cache instances
- Display the Dynamic Cache mbean statistics for cache instances

Details about this Extended Cache Monitor and the download package for it can be found here:

http://www-128.ibm.com/developerworks/websphere/downloads/cache_monitor.html

This is only available for WebSphere Application Server 6.0 (minimum of WebSphere Application Server 6.0.2.17) and WebSphere Application Server 6.1:

http://www.ibm.com/developerworks/websphere/downloads/cache_monitor.html

DRS Log Tool Analyzer:

https://cs.opensource.ibm.com/projects/welt/

13.4.11 Monitor ESI caching

The ESI processor's cache is monitored through the Dynamic Cache Monitor application. In order for the ESI processor's cache to be visible in the cache

monitor, the DynaCacheEsi application must be installed as described in “Install the DynaCacheEsi application” on page 271 and the esiInvalidationMonitor property must be set to true in the plugin-cfg.xml file, as described in “Reconfigure the Web servers” on page 281.

Point your browser to the Dynamic Cache Monitor application. To get this statistic you need to select **Edge Statistics** in the Cache monitor navigation bar.

Tip: During our tests, we found that the Edge statistics are only displayed in one of the Dynamic Cache Monitor applications on one of the application servers. So you will need to verify which one is the correct one.

Archived

The Edge Statistics could be confused with the Caching Proxy statistics, which is part of IBM WebSphere Edge Components. However, the term Edge Statistics in this case relates to the ESI cache statistics. The ESI cache statistics are shown in Figure 13-19. If you do not see Edge Statistics right away, try clicking the **Refresh Statistics** button.

Figure 13-19 ESI cache statistics

The following information is available:

- ESI Processes: This is the number of processes configured as edge caches.
- Number of Edge Cached Entries: This is the number of entries currently cached on all edge servers and processes.
- Cache Hits: This is the number of requests that match entries on edge servers.

- Cache Misses By URL: A cache policy does not exist on the edge server for the requested template.

  Note that the initial ESI request for a template that has a cache policy on a WebSphere Application Server results in a miss. Every request for a template that does not have a cache policy on the WebSphere Application Server will result in a miss by the URL on the Edge server.

- Cache Misses By Cache ID: In this case, the cache policy for the requested template exists on the edge server. The cache ID is created (based on the ID rules and the request attributes), but the cache entry for this ID does not exist.

  Note that if the policy exists on the edge server for the requested template, but a cache ID match is not found, the request is not treated as a cache miss.

- Cache Time Outs: The number of entries removed from the edge cache, based on the timeout value.

- Evictions: The number of entries removed from the edge cache, due to invalidations received from WebSphere Application Server.

If you click the **Contents** button, you can see the edge content for all processes or for a selected process number. You can see an example in Figure 13-20, showing the TopCategoriesDisplay URL, which we made edge cacheable in “Caching full pages” on page 285.

Figure 13-20 ESI cache contents

13.4.12 Summary

Replication is a service that transfers data, objects, and events among application servers. Data replication service (DRS) is the internal WebSphere Application Server component that replicates data.

Dynamic cache uses the data replication service to further improve performance by copying cache information across application servers in the cluster, preventing the need to repeatedly perform the same tasks and queries in different application servers.

In a clustered WebSphere Commerce environment, we recommend that you enable cache replication.

This chapter explained what cache replication is, how it works, how to tune it for, and how to configure it.

Archived

14

Profiling

Profiling is a technique that can be used at any phase of a project life cycle. During the development phase it can be used by developers to pinpoint areas in their code that might have performance implications. In this way performance issues can be addressed early on, resulting in incredible savings in time, effort, and money. Therefore, it is extremely important that the developers have the responsibility of profiling their own code in their development environment first, before any performance testing phases are started. Profiling can also be used during the performance load test phase to drill down on areas highlighted as having poor response times.

When assessing the performance of a single page, developers will find the following four exercises incredibly rewarding in quickly identifying the performance bottlenecks:

- Use IBM Page Detailer to get an overall idea of how fast the page loads and how long it takes to render each page element to the browser.
- Conduct SQL profiling to capture expensive or excessive database queries. Then either eliminate the unnecessary SQLs or tune the expensive SQLs to improve performance.
- Conduct Java code profiling to identify bottlenecks in the code and revise to improve performance.
- If necessary, apply techniques to map SQL statements to Java code in WebSphere Commerce.

© Copyright IBM Corp. 2008. All rights reserved.

14.1 SQL profiling

SQL profiling involves capturing queries made to the database from the application across a period of time. This measurement can be taken for a single request and also during load tests to determine queries that have high execution times, queries that are doing table scans, and queries that are executing too many times during the same request.

A tutorial about conducting SQL profiling on DB2 can be found at this link:

https://www.ibm.com/developerworks/websphere/library/tutorials/0804_clustering1/0804_clustering1.html

14.2 Java code profiling

Code profiling tools are used to capture all method invocations and their execution times during a request. This information can then be used to identify areas in the code that are causing bottlenecks and affecting performance. With this information a developer can then revisit a particular area of his code and rework it to solve the performance issue.

A tutorial on how to conduct Java code profiling with the Rational® profiling tool using WebSphere Commerce as an example is available at this link:

https://www.ibm.com/developerworks/websphere/library/tutorials/0804_clustering1/0804_clustering1.html

14.3 Mapping an SQL statement to Java code

In addition, to help you troubleshoot database-related problems with WebSphere Commerce, an article shows you how to find the Java code that executes a particular SQL statement. Follow this link to see how to map an SQL statement to Java code in WebSphere Commerce:

http://www.ibm.com/developerworks/websphere/library/techarticles/0802_doumbia/0802_doumbia.html

14.4 IBM Page Detailer

IBM Page Detailer is a browser-side tool to measure performance of a Web application. While the Profiler discussed in the previous sections supports analysis of the execution of the application on the server, the Page Detailer collects most of its useful data at the socket level to reveal the performance

details of items in the Web page, from the client's (browser's) perspective. It is also useful for measuring the incremental impact of changes in a Web application.

Page Detailer allows you to look at how and when each item is loaded in a Web page. Analyzing this data allows you to identify the areas where performance could be improved. The user's perception of performance is determined based on the time to display pages, so measuring and analyzing this data will provide insight into the user's experience of your application.

14.4.1 Overview

IBM Page Detailer is a graphical tool that enables Web site developers and editors to rapidly and accurately assess performance from the client's perspective. IBM Page Detailer provides details about the manner in which Web pages are delivered to Web browsers. These details include the timing, size, and identity of each item in a page. This information can help Web developers, designers, site operators, and IT specialists to isolate problems and improve performance and user satisfaction. Page Detailer can be used with any site that your browser can access.

Page Detailer is a separately downloadable product that can be obtained from IBM alphaWorks® at:

`http://www.alphaworks.ibm.com/tech/pagedetailer`

There are two versions available:

- Evaluation version: This version is for free but does not support all features.
- Pro version: This version must be licensed for a small fee and contains the following additional features:
  - Full support for HTTPS (SSL) traffic
  - Saving and restoration of captured data
  - Ability to add/edit one's own notes for captured pages and items
  - A find facility for working with text

The supported platforms for Page Detailer are Windows 2000 and Windows XP.

The Page Detailer can monitor all HTTP and HTTPS requests originating from the machine where it is running. Thus, it can be used to measure performance for Web applications running either locally or remotely, on WebSphere Application Server, IBM Rational Application Developer V6.0, or any other Web site or application. Hence, the Page Detailer may be used at any time during the project development cycle, from coding through to production support. Note that when analyzing the performance of non-production environments, differences in

Archived

the production environment topology and configuration could result in differences in the measured performance results.

IBM Page Detailer gathers the following information:

- Connection time
- Socks connection time and size
- SSL connection time and size
- Server response time and size
- Content delivery time and size
- Delays between transfers
- Request headers
- Post data
- Reply headers
- Content data
- Page totals, averages, minimums, and maximums

For each page that is accessed, a color-coded bar chart of the time taken to load the page items will be generated. The length of a particular bar gives a good idea of the relative time spent in loading that item, as compared to the entire page. You will see that in some cases, items of a page may be loaded in parallel. This will appear in the chart with bars that overlap. The information that is captured by the Page Detailer includes page size as well as sizes of all other items loaded by the browser.

Different colors in the bar indicate how the time was spent.

- Page Time (Purple)

  The time taken to load all the components of a page.

- Host Name Resolution (Cyan)

  The time spent to resolve the IP address of the host.

- Connection Attempt Failed (Brown)

  The time taken to receive an error when a connection attempt is made.

- Connection Setup Time (Yellow)

  The time taken to open a socket connection. If a SOCKS server is being used, this is the time to open a socket connection from the browser to the SOCKS server only.

Archived

- Socks Connection Time (Red)

  The time taken to open a connection from a SOCKS server to the remote site.

- SSL Connection Setup Time (Pink)

  This is the time taken to negotiate an encrypted connection between the browser and the remote site, once a normal socket connection has been established.

- Server Response Time (Blue)

  This is the time from the browser's request to the receipt of the initial reply, after all the communications setup has been completed. Large responses are broken down into smaller components (packets). The server response time only measures the time to receive the first one.

- Delivery Time (Green)

  The time taken to receive all additional data that was not included in the initial response.

An example of a chart produced with Page Detailer is provided in Figure 14-1.

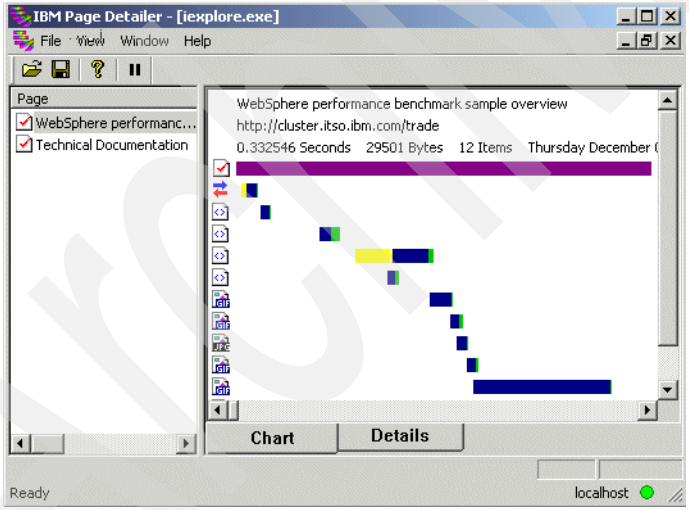


Figure 14-1 Page Detailer Chart view

Archived

To obtain more detailed information about a particular HTTP request, double-click the appropriate colored bar or the icon in the chart. This will display a text viewer as shown in Figure 14-2. It includes information about timings, sizes, header, and so on. You can add your own notes and save the file for comparison. This feature is available only in the Pro version.

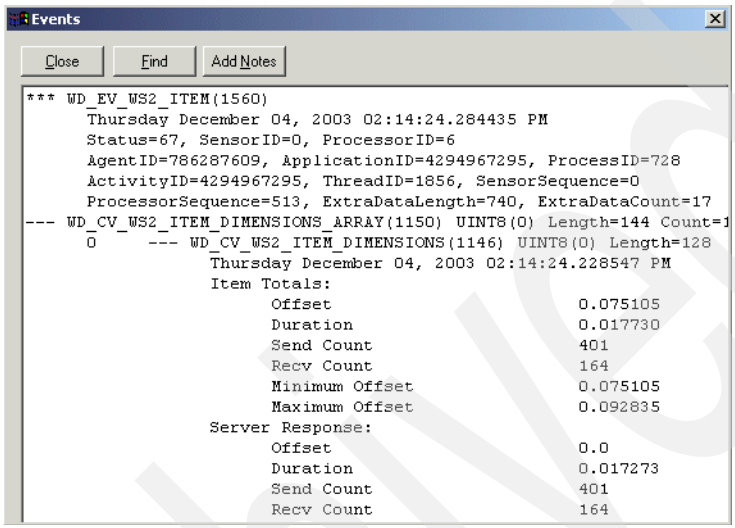


Figure 14-2 Page Detailer Events

Archive

In addition to the Chart view, it is also possible to view more details of all HTTP requests using the Details view. This can be seen by selecting the **Details** tab at the bottom of the window (see Figure 14-3).

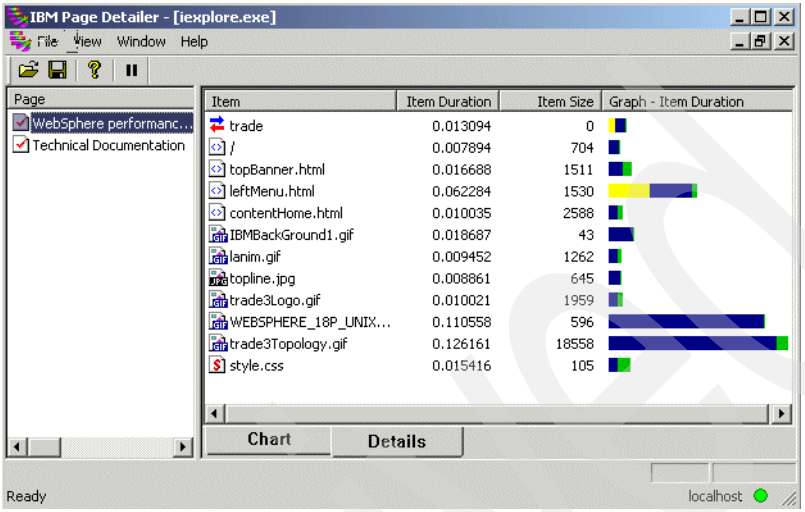


Figure 14-3 Page Detailer Details view

14.4.2 Important considerations

Some important considerations while taking measurements are:

- Impact of network delays

  Many problems may not be evident when accessing a server on a local network, but may become apparent when accessing the site remotely, particularly when using a modem connection. On the other hand, you can minimize the effect of external network delays by directly connecting on the Server's LAN. This will allow you to isolate the performance impact of a change made to the Web page.

- Browser cache

  Disabling the browser cache helps in getting repeatable results. However you could also check the performance from a user's perspective by enabling the browser cache and comparing both results.

- Packet loss

  Packet loss can happen and get corrected in the underlying TCP/IP layers. This is invisible to the Page Detailer. Packet loss manifests itself as inconsistent time measurements in Page Detailer. You can take a series of measurements at different times to factor it out.

Archived

14.4.3 Key factors

Some of the key factors that influence the time to load a Web page in a browser are:

- Page size
- Number, complexity, and size of items embedded in the page
- Number of servers that need to be accessed to retrieve all elements, and their location and network connectivity
- Use of SSL (This introduces an extra overhead.)

The Page Detailer will help you to identify when one of these problems is affecting some or all of your application. It will also help to identify problems such as broken links and server timeouts.

Some of the strategies that can be used to improve performance and resolve problems that you have identified include:

- Minimize the number of embedded objects. Avoid the excessive use of images in particular. In cases where there is a standard header, footer, or side menu on every screen, consider the use of frames so that common elements do not have to be downloaded every time.
- The browser will typically retrieve multiple items in parallel, in the order in which they appear on the HTML page that it receives. Hence, sequencing of the items so that downloads for larger objects are started early can reduce the total time required to display the page, and avoid the user having to wait for a long time for the last elements to be retrieved.
- Ensure that caching is being used effectively. Often the same images are used multiple times on the same page. If there are two references to the same image in close proximity to each other in the HTML source, the browser may encounter the second reference before the HTTP request that was initiated to download the first reference has been completed. In this case the browser may issue another request to retrieve the image again. This can be avoided by pre-loading frequently used images multiple times early, or by structuring the generated pages so that such URLs do not appear consecutively.
- Minimize the use of SSL where possible. For example, some content such as images may not need to be secured even though the application as a whole needs to be secure.
- Try to avoid switching the user to an alias server name during the page load. This will help the browser to reduce the lookup time and possibly avoid a new connection.

14.4.4 Tips for using Page Detailer

In this section we provide a few helpful tips to analyze the performance data shown by Page Detailer. A sample analysis, with comments, is shown in Figure 14-4.

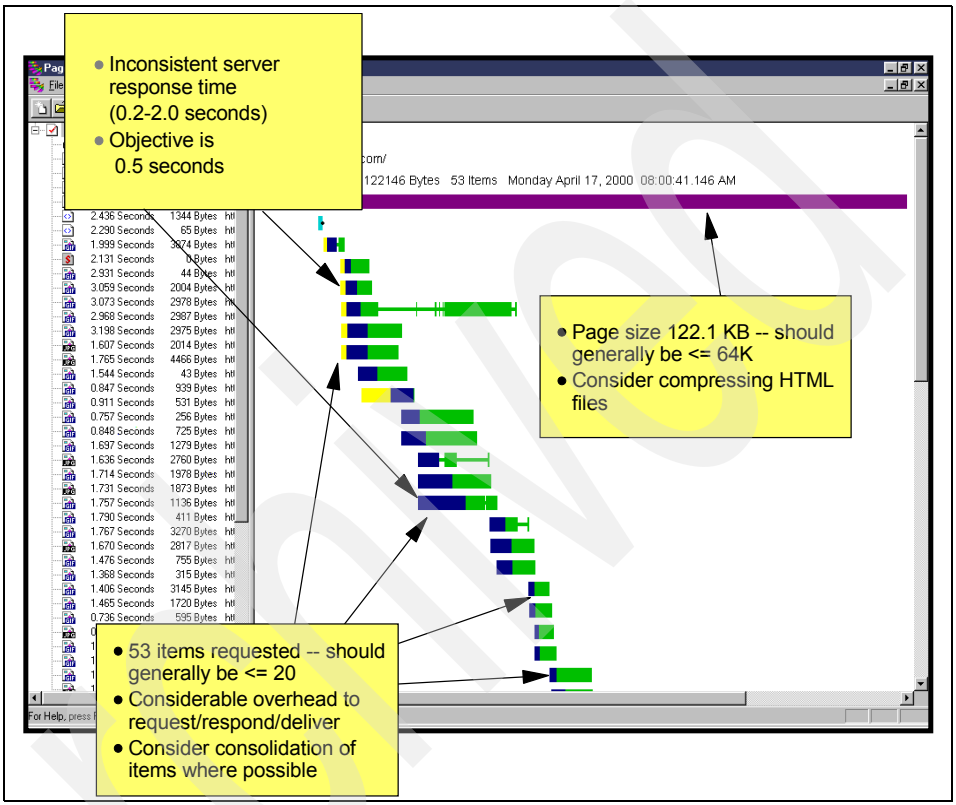


Figure 14-4 Page Detailer sample analysis

As per the analysis:

- A summary of the above information and the meaning of each icon and color can be obtained by using the menu **View** → **Legend**. This displays the window shown in Figure 14-5.

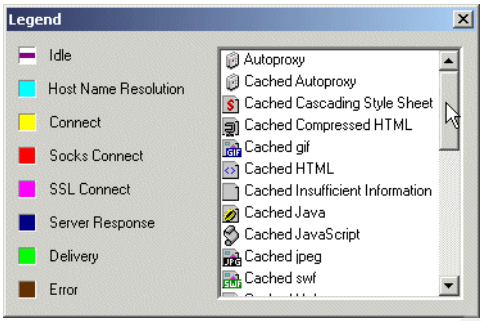


Figure 14-5 Page Detailer Legend

- Use a separate browser instance to collect data from related Web pages in one file. This allows for easy retrieval of performance data of related Web pages for comparison. Note that you can save your work only in the Pro version of the Page Detailer.
- You can select the columns to display in the Details view from the context menu (obtained by right-clicking). You can also choose to display the column as a graph if it contains a numerical value.

Archived

- You can move the vertical bar and make room to add new columns on the left hand side of the window, as shown in the Figure 14-6 and Figure 14-7.

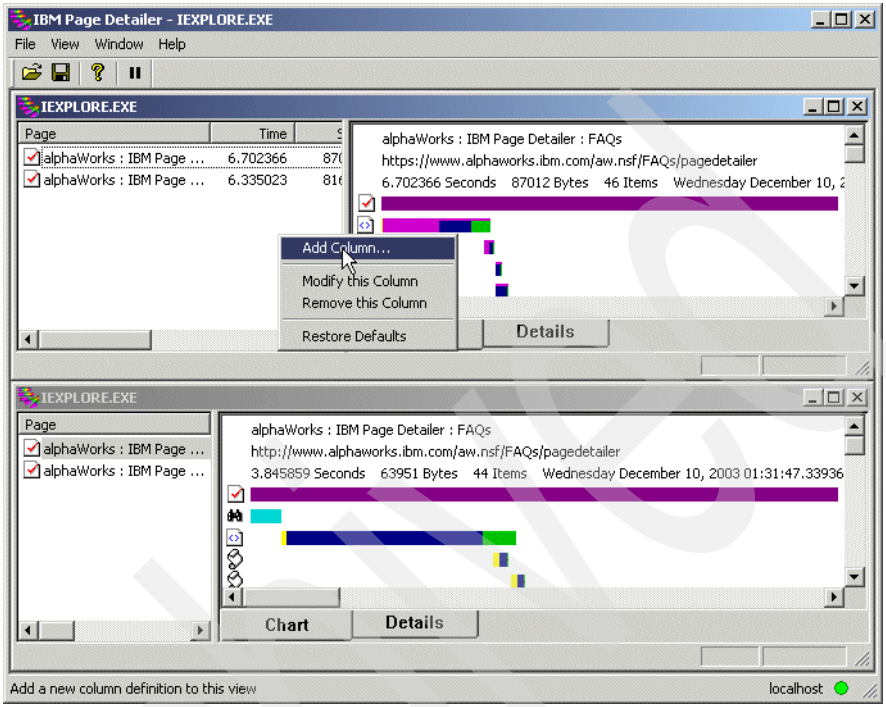


Figure 14-6 Moving the separator and adding a new column

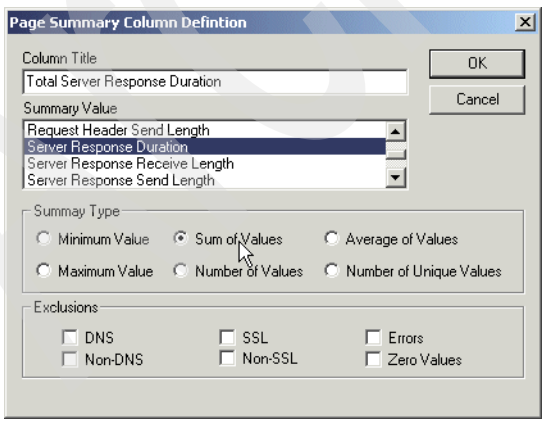


Figure 14-7 Column definition

Tip: If it seems that Page Detailer is not collecting your browser interactions, try to navigate a little bit slower (there is a delay between the browser showing the page and Page Detailer capturing its content). The other point that you must be aware of is that Page Detailer installs itself configured for Microsoft Internet Explorer and several versions of Netscape browsers. If you want to use another browser, you need to check the documentation to see how to set it up.

14.4.5 Reference

See the following references for further information:

- The article “Design for Performance: Analysis of Download Times for Page Elements Suggests Ways to Optimize” at:

  http://www.ibm.com/developerworks/websphere/library/techarticles/hvws/perform.html

- The Page Detailer Web site at IBM alphaWorks:

  http://www.alphaworks.ibm.com/tech/pagedetailer

Archived

Part 5

Monitoring and performance tuning

Monitoring and performance tuning are essential parts of any WebSphere Commerce site administration while in production. You must monitor the servers to ensure that they are running smoothly and troubleshoot problems as they occur.

More importantly, before launching into production, you must tune the performance of servers to achieve optimal performance based on the current system resources and expected traffic load.

Performance tuning is as much an art as it is a science. It is often done based on trial and error. You adjust the parameters on a server, monitor its performance over time, and measure the improvement achieved as a result of these updated settings. If the results are not as expected, you adjust the settings again. Avoid making any tuning changes directly on production servers. You should have a performance test environment that is similar in capacity and configuration to your production environment. After you test any tuning changes and are satisfied with the result on the performance test servers, you may make the same changes on the production servers.

© Copyright IBM Corp. 2008. All rights reserved.

Performance problems can be encountered almost anywhere. The problem can be network and hardware related or back-end system related. The problem can be actual product bugs, or quite often, application design issues.

Understanding the flow used to diagnose a problem helps to establish the monitoring that should be in place for your site to detect and correct performance problems. The first dimension is the *user* view,—the black box view of your Web site. This is an external perspective of how the overall Web site is performing from a user's point of view and identifies how long the response time is for an user. From this black box perspective, it is important to understand the load and response time on your site. To monitor at this level, many industry monitoring tools allow you to inject and monitor synthetic transactions, helping you identify when your Web site experiences a problem.

The second step is to understand the basic health of all the systems and networks that make an user request. This is the *external* view, which typically leverages tools and utilities provided with the systems and applications running. In this stage, it is of fundamental importance to understand the health of *every system* involved, including Web servers, application servers, databases, back-end systems, and so on. This dimension corresponds to the *what resource is constrained* portion of the problem diagnosis. To monitor at this level, you need to use product-specific tools and logs.

This section of the book covers the following:

- Various tools that are available to help you monitor servers at different tiers: operating system, database server, application server, Web server, and Load Balancer
- Key tuning parameters at various tiers

Archive

15

Operating system monitoring tools

An operating system is the infrastructure of all of the other software products, so it is critical to gracefully use the resource of an operating system. One of the most important responsibilities a system administrator has is monitoring her operating systems. As a system administrator you need the ability to see what is happening on your operating system at any given time, whether it is the percentage of a system's resources currently used or what commands are being run.

Basically, we need to pay more attention to three key resources in the operating system when a performance issue arises:

- CPU
- Memory
- Disk I/O

© Copyright IBM Corp. 2008. All rights reserved.

15.1 Operating system introduction

An operating system is a layer of software that takes care of technical aspects of a computer's operation. It shields the user of the machine from the low-level details of the machine's operation and provides frequently needed facilities.

Normally, the operating system has a number of key elements:

- A technical layer of software for driving the hardware of the computer, like disk drives, the keyboard, and the display.
- A file system that provides a way of organizing files logically.
- A simple command language that enables users to run their own programs and to manipulate their files in a simple way. Some operating systems also provide text editors, compilers, debuggers, and a variety of other tools.
- Legal entry points into its code for performing basic operations like writing to devices.

15.2 General utilities related with operating system monitoring

There are numbers of utilities in an operating system that can help to monitor the status of an operating system. Some of them are platforms specific. In this section, some general utilities are discussed. Be aware that most of the utilities described in this chapter are related to the AIX or UNIX platform.

15.2.1 nmon

The nmon tool is designed for AIX and Linux performance specialists to use for monitoring and analyzing performance data, including:

- CPU utilization
- Memory use
- Kernel statistics and run queue information
- Disks I/O rates, transfers, and read/write ratios
- Free space on file systems
- Disk adapters
- Network I/O rates, transfers, and read/write ratios
- Paging space and paging rates

Archived

- CPU and AIX specification
- Top processors
- IBM HTTP Web cache
- User-defined disk groups
- Machine details and resources
- Asynchronous I/O—AIX only
- Workload Manager (WLM)—AIX only
- Network File System (NFS)
- Dynamic LPAR (DLPAR) changes—only pSeries p5 and OpenPower® for either AIX or Linux

Benefit of nmon

The nmon tool is helpful in presenting all the important performance tuning information on one screen and dynamically updating it. The tool works on any dumb screen, Telnet session, or even dial-up line. In addition, the tool is very efficient. It does not consume many CPU cycles (usually below 2%). On newer machines, CPU usage is well below 1%.

Data is displayed on the screen and updated once every two seconds using a dumb screen. However, you can easily change this interval to a longer or shorter time period. If you display the data on X-Windows, VNC, putty, or similar, and stretch the window, nmon can output a great deal of information all in one place.

The nmon tool can also capture the same data to a text file for later analysis and graphing for reports. The output is in a spreadsheet format (.csv).

Installing and using the nmon tool

The tool is one stand-alone binary file (a different file for each AIX or Linux version) that you can install it in an easy way:

1. Copy the nmonXX.tar.Z file to the machine. If you download the file by using FTP, remember to use binary mode. Note that here, the XXX in this example will be replaced by the current version that you are using.
2. To uncompress the file run uncompress nmonXX.tar.Z.
3. To extract the files run tar xvf nmonXX.tar.
4. Read the README file.
5. Start nmon by typing the command **`nmon`**.

Archived

6. If you are the root user you may need to type `./nmon` to can get output as in Figure 15-1.

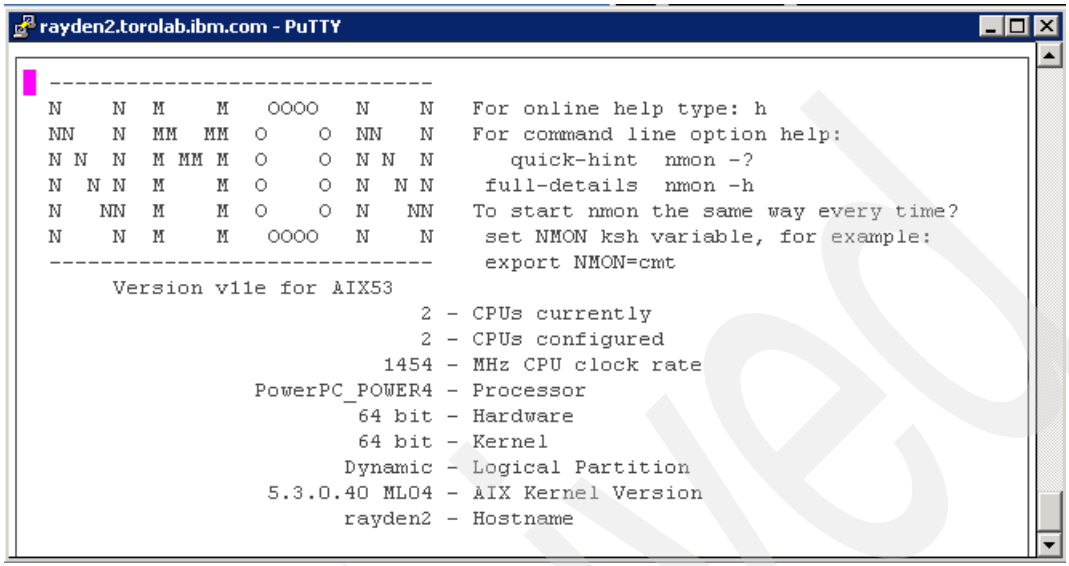


Figure 15-1 nmon interface

7. Press C and M to get the statistics data for CPU and memory, as shown in Figure 15-2.

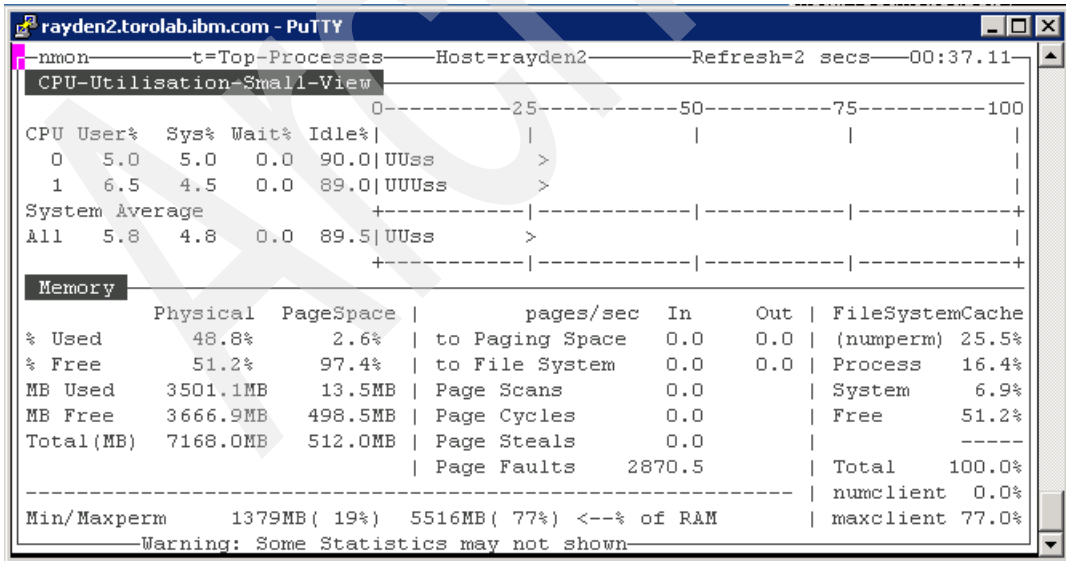


Figure 15-2 CPU and memory statistics data in nmon

Features of nmon are listed in Table 15-1.

Table 15-1 Features of nmon

New features	Description
Starting up	There is also now a small shell script called “nmon” that starts the correct nmon version. Place this script and nmon binaries in your $PATH and type `nmon`. This version is now only compiled in 32-bit mode. So, it runs on 32-bit and 64-bit hardware. The idea is to make it easier to install and run.
N = NFS	NFS is completely new for nmon 10.
p = Partitions	This is for shared CPU partition information—the big p5/AIX5.3 feature.
C = CPU	This is for machines with 32+ CPUs—up to 128 logical CPUs by demand.
c = CPU	This details your physical CPU use if you are on a POWER5™ with AIX 5.3 and in a shared CPU environment.
S = Subclass	This is for WLM subclasses—by request.
a = Disk adapters	This gives you details of the disk adapter—like its full type.
r = Resources	This includes your CPU speed in MHz.
k = Kernel	This gives some new fields.
L = Large pages	This gives you large-page stats, which are popular with high-performance people.
n = Network	This gives you information about your network adapters, MTU, and errors.
D = Disk	This gives you more information about your disks, disk type sizes, free, volume groups, adapter, and so forth.
m = Memory	This gives you more details on where your memory is going, system (kernel) and processes, and active virtual memory.
-B	This is a start-up option to remove the boxes.

Archive

15.2.2 Top

Top is a command widely used in the Linux operating system. This command displays a continually updating report of system resource usage.

The top portion of the report lists information such as the system time, uptime, CPU usage, physical and swap memory usage, and number of processes. Below that is a list of the processes sorted by CPU utilization.

You can modify the output of **top** while it is running. If you press an **i**, top will no longer display idle processes. Press i again to see the idle processes again. Pressing M will sort by memory usage, S will sort by how long the processes have been running, and P will sort by CPU usage again.

In addition to viewing options, you can also modify processes from within the **top** command. You can use u to view processes owned by a specific user, k to kill processes, and r to renice them.

For more in-depth information about processes, you can look in the /proc file system. In the /proc file system you will find a series of sub-directories with numeric names. These directories are associated with the process IDs of currently running processes. In each directory you will find a series of files containing information about the process.

Archive

Figure 15-3 is sample output for issuing the **top** command.

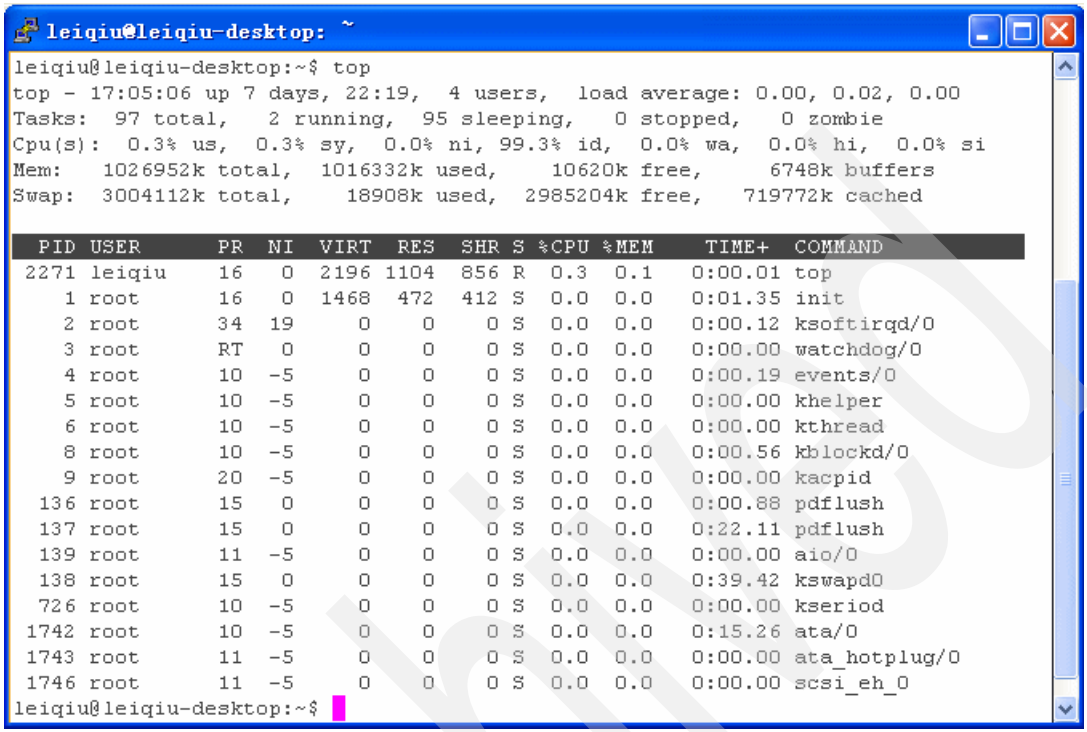


Figure 15-3 Output of command top

15.2.3 vmstat

The **vmstat** command reports statistics about kernel threads, virtual memory, disks, traps, and CPU activity. Reports generated by the **vmstat** command can be used to balance system load activity. These system-wide statistics (among all processors) are calculated as averages for values expressed as percentages, and as sums otherwise.

If the **vmstat** command is invoked without flags, the report contains a summary of the virtual memory activity since system startup. Basically, you can set the interval and count for this command. The Interval parameter specifies the amount of time in seconds between each report. If the interval parameter is not specified, the **vmstat** command generates a single report that contains statistics for the time since system startup and then exits. The count parameter can only be specified with the interval parameter.

After using the interval and count parameters, you get a result shown in Figure 15-4.

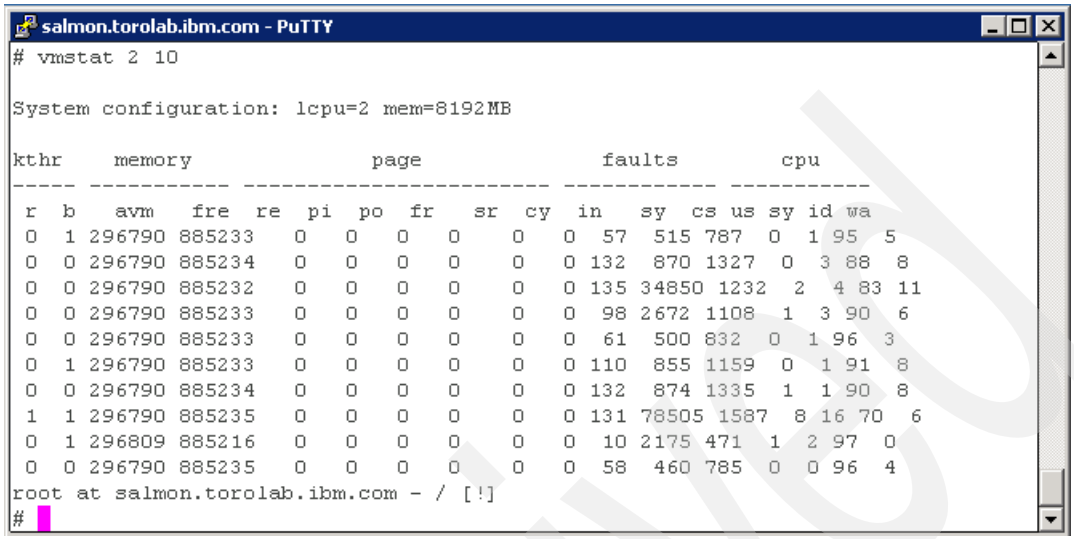


Figure 15-4 Output of command vmstat

For more details about this command, go to the IBM pSeries information center and search vmstat:

http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp

15.2.4 iostat

The **iostat** command is used for monitoring system input/output device loading by observing the time the physical disks are active in relation to their average transfer rates. The **iostat** command generates reports that can be used to change the system configuration to better balance the input/output load between physical disks and adapters.

All statistics are reported each time the **iostat** command is run. The report consists of a tty and CPU header row followed by a row of tty or asynchronous I/O and CPU statistics. On multiprocessor systems, CPU statistics are calculated system-wide as averages among all processors.

The **iostat** command generates four types of reports: the tty and CPU Utilization report, the Disk Utilization report, the System throughput report, and the Adapter throughput report.

Basically, you can set the interval and count for this command.The interval parameter specifies the amount of time in seconds between each report. If the interval parameter is not specified, the **iostat** command generates a single report containing statistics for the time since system startup (boot). The count parameter can be specified in conjunction with the interval parameter.

After using the interval and count parameters, you get the results shown in Figure 15-5.

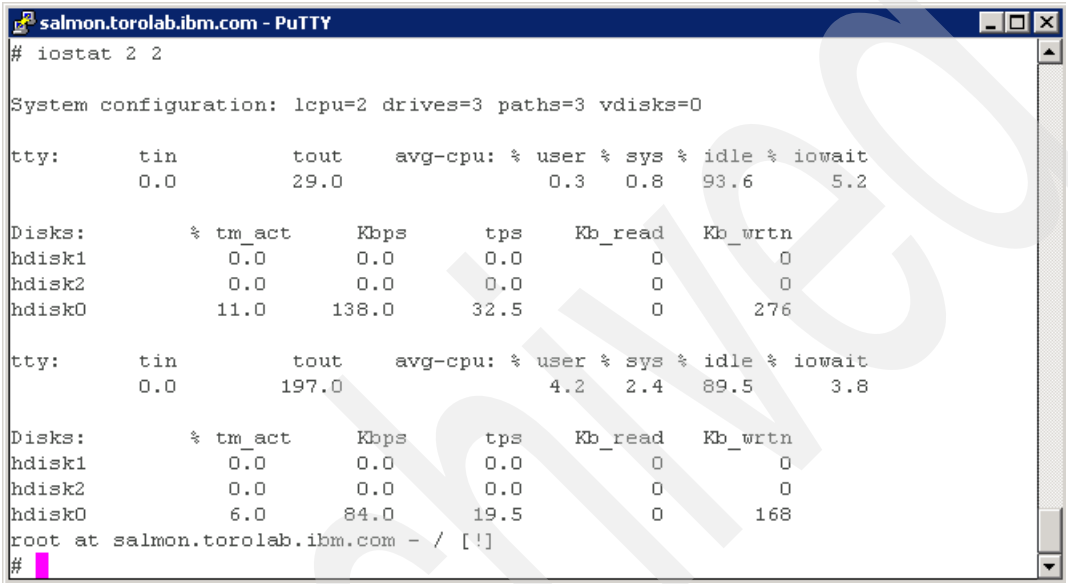


Figure 15-5 Output of command iostat

For more details about this command, go to the IBM pSeries information center:

http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.cmds/doc/aixcmds3/iostat.htm

15.2.5 ps

The **ps** (process status) command produces a list of processes on the system that can be used to determine how long a process has been running, how much CPU resource the processes are using, and whether processes are being penalized by the system. It also shows how much memory processes are using, how much I/O a process is performing, the priority and nice values for the process, and who created the process.

Archived

Figure 15-6 is sample output of the **ps** command.

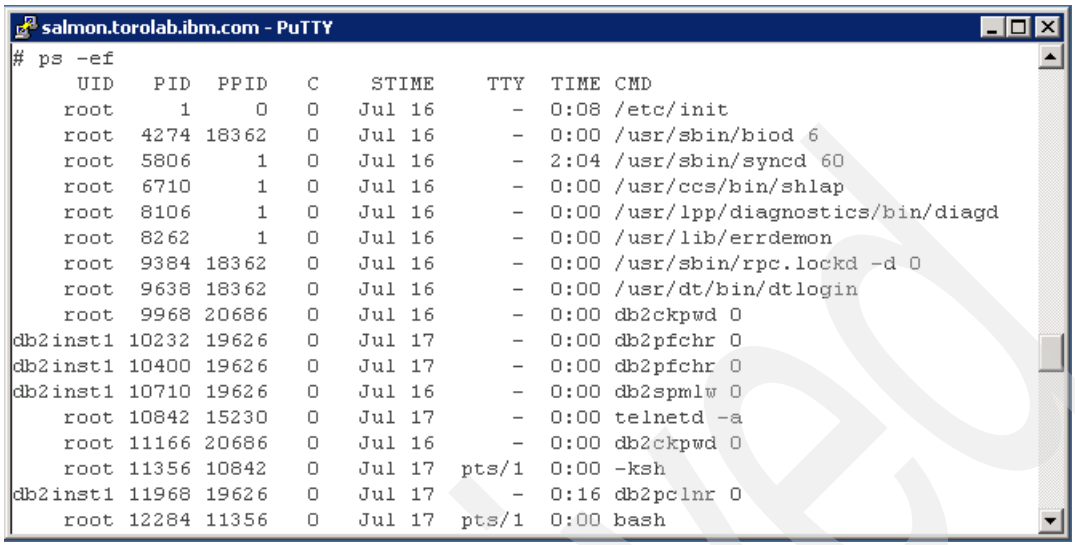


Figure 15-6 Output of command ps

The first column shows who owns the process. The second column is the process ID. The third column is the parent process ID. This is the process that generated, or started, the process. The fourth column is the CPU usage (in percent). The fifth column is the start time, or date if the process has been running long enough. The sixth column is the tty associated with the process, if applicable. The seventh column is the cumulative CPU usage (total amount of CPU time is has used while running). The eighth column is the command itself.

For more details about this command, go to the IBM pSeries information center:

http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.cmds/doc/aixcmds3/iostat.htm

15.2.6 svmon

This command is used to capture and analyze a snapshot of virtual memory.

The **svmon** command displays information about the current state of memory. The displayed information does not constitute a true snapshot of memory, because the **svmon** command runs at the user level with interrupts enabled.

The segment is a set of pages and is the basic object used to report the memory consumption, so the statistics reported by **svmon** are expressed in terms of pages.

A page is a block of virtual memory, while a frame is a block of real memory. Frames always have a size of 4 KB, whereas pages may have different sizes. The base page size is 4 KB. All pages inside a segment have the same size.

The memory consumption is reported using the inuse, free, pin, virtual, and paging space counters:

- The *inuse counter* represents the number of used frames.
- The *free counter* represents the number of free frames from all memory pools.
- The *pin counter* represents the number of pinned frames (that is, frames that cannot be swapped).
- The *virtual counter* represents the number of pages allocated in the system virtual space.
- The *paging space counter* represents the number of pages reserved or used on paging spaces.

Figure 15-7 is the sample output for this command.

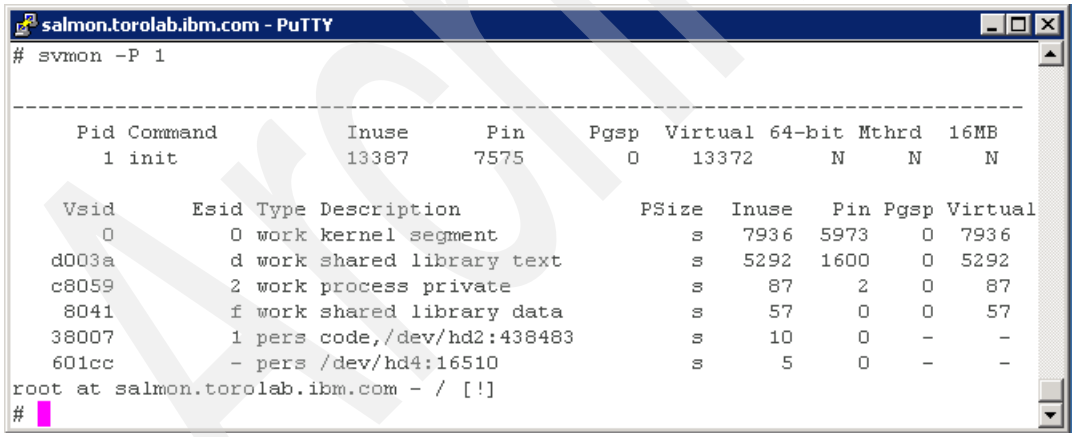


Figure 15-7 Output for command svmon

Archived

15.3 Best practices for AIX monitoring

In this section, we focus on AIX monitoring by using **nmon**. **nmon** can provide lots of information for testers and developers to use to analyze the status of the operating system. But it comes with a problem: how to use **nmon** and the information it generates efficiently.

How to get statistics data for operating system more efficiently

How do you get statistics data for the operating system? Specific to the version of **nmon** used for AIX53, we can use the data collection model to collect detailed status data about the operating system for further analysis. In Example 15-1, we collect the nmon performance results 180 times with an interval of 60 seconds. The **nmon** command exports its result to a file named hostname. Note that the t parameter is used to focus our attention on the status of the top processes.

```
nmon -f -t -r hostname -s 60 -c 180
```

A result file named hostname_timeSeries.nmon will be generated in the current directory. You might be able to get output as shown in Example 15-1.

Example 15-1 hostname_timeSeries.nmon snapshot

```
CPU01,T0002,62.6,5.5,15.8,16.1
CPU02,T0002,60.6,6.9,16.1,16.3
CPU_ALL,T0002,61.6,6.2,16.0,16.2,,2
CPU00,T0002,61.6,6.2,16.0,16.2
MEM,T0002,73.0,97.8,5234.5,500.5,7168.0,512.0
MEMNEW,T0002,13.3,7.3,6.4,73.0,5.9,17.0
MEMUSE,T0002,7.3,19.2,77.0,960,1088,0.0,77.0
PAGE,T0002,2369.3,80.2,72.2,0.0,0.0,0.0,0.0,0.0,0.0
PROC,T0002,2.86,1.77,2258,24013,455,108,11,10,14,59
FILE,T0002,0,937,4,1357820,246443,0,0,0
NET,T0002,342.7,0.0,472.6,0.0,
NETPACKET,T0002,887.7,0.6,852.0,0.6,
NETERROR,T0002,0.0,0.0,0.0,0.0,0.0,0.0,
IOADAPT,T0002,320.1,288.7,125.4,0.0,0.0,0.0
JFSFILE,T0002,16.4,49.9,79.9,66.8,3.7,6.1,5.5,21.6,8.5,3.7,3.8,43.9
JFSINODE,T0002,2.8,0.3,2.0,1.1,0.1,0.2,0.1,9.4,0.4,0.2,21.2,1.6
DISKBUSY,T0002,0.0,0.0,0.0,0.0,0.0,0.0
DISKREAD,T0002,320.1,0.0,0.0,0.0,0.0,0.0
DISKWRITE,T0002,288.7,0.0,0.0,0.0,0.0,0.0
DISKXFER,T0002,125.4,0.0,0.0,0.0,0.0,0.0
DISKBSIZE,T0002,4.9,0.0,0.0,0.0,0.0,0.0
TOP,0372916,T0002,19.29,19.17,0.12,1,6544,64,6480,987,0.357,7,db2sysc,Unclassif
ied
TOP,0680066,T0002,18.81,18.76,0.05,1,6532,64,6468,469,0.356,3,db2sysc,Unclassif
ied
```

```
TOP,0508142,T0002,18.15,18.09,0.06,1,10004,64,9944,1286,0.545,6,db2sysc,Unclass
ified
```

How to analyze the statistics data

After you collect the statistics data based on your requirement, you can use the tool named NMON_Analyser to analyze the result. NMON_Analyser is designed to complement NMON in analyzing and reporting performance problems. It is written in VBA for Excel® (2000 edition or later). NMON_Analyser produces graphs for virtually all sections of output produced using the *spreadsheet output* mode of NMON, as well as doing some additional analyses for ESS, EMC, and FAStT subsystems.

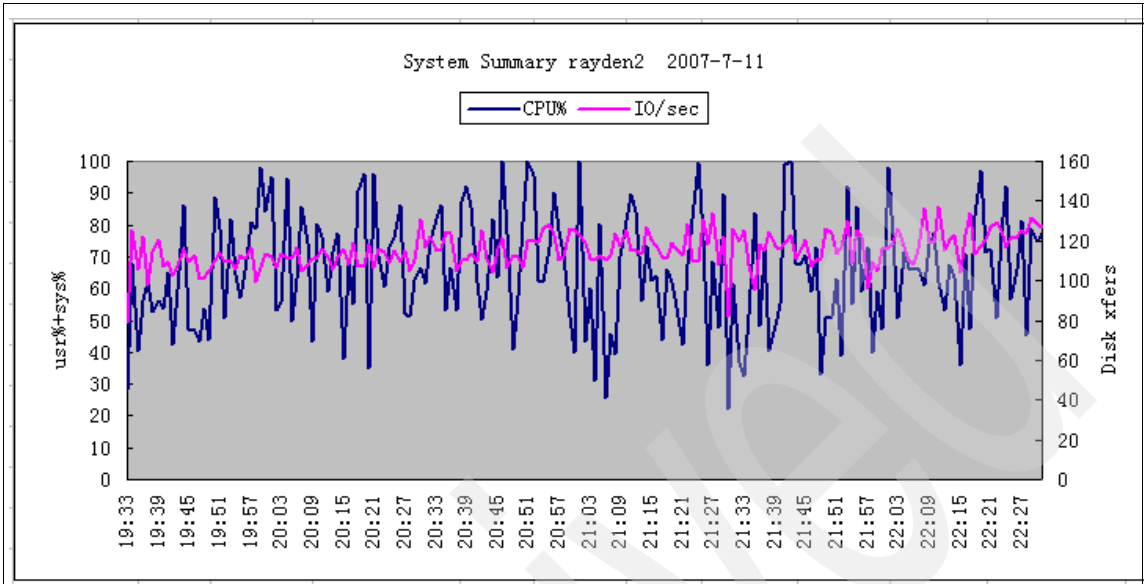
Based on the statistics data we get from the previous command issued, we can use NMON_Analyser to generate the analysis result. Generally, we can get an overall system status report. Besides that, it gives different views for different resources in this operating system, which contains:

- Whole system status
- CPU status
- Memory status
- Disk status
- Network status

In WebSphere Commerce development and test, based on previous experience, CPU, memory, disk I/O, and the network should potentially be the root causes of performance issues, so let us focus on the following five aspects.

Archive

Figure 15-8 provides a system summary view.



Total System I/O Statistics		CPU:	Usr	Sys	Wait	Idle
Avg tps during an i	115.3	Avg	59.4	6.3	20.1	14.2
Max tps during an i	136.9	Max	93.8	22.0	44.8	58.7
Max tps interval ti	22:11:49					
Total number of Kby	36,494.2					
Total number of Kby	73,870.7					
Read/Write Ratio:	0.5					

Figure 15-8 System summary view

Archive

Figure 15-9 provides a CPU view.

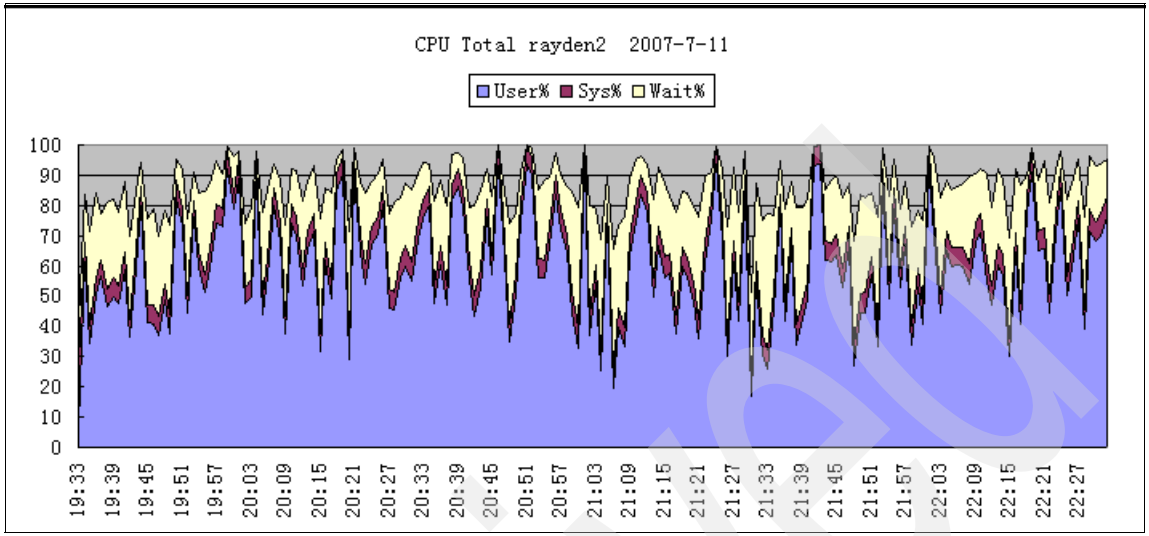


Figure 15-9 CPU total view

Figure 15-10, Figure 15-11 on page 338, and Figure 15-12 on page 338 provide disk I/O views.

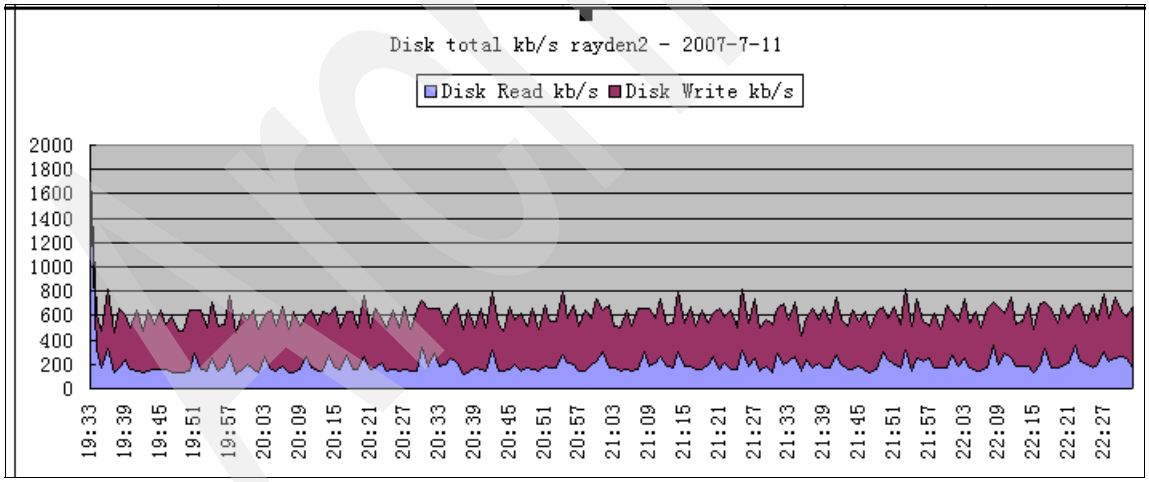


Figure 15-10 Disk summary view

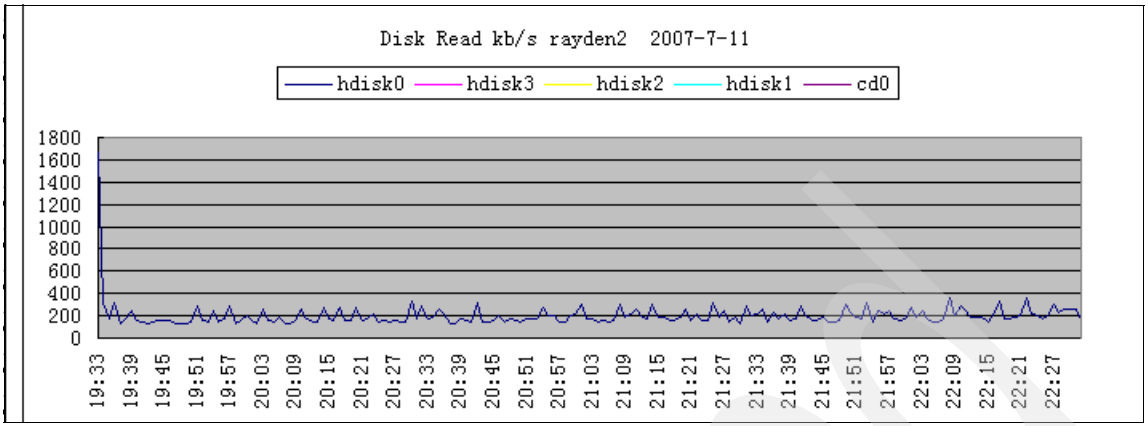


Figure 15-11 Disk read view

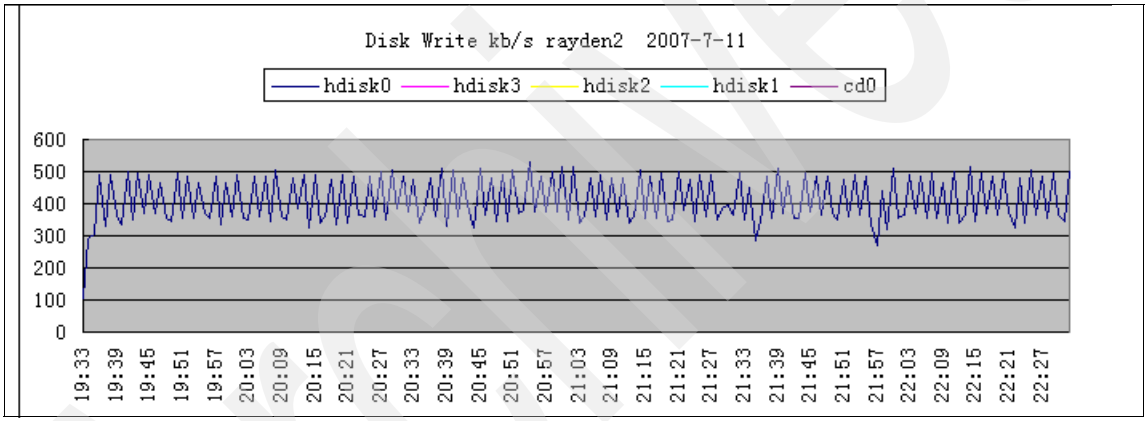


Figure 15-12 Disk write view

Archived

Figure 15-13 and Figure 15-14 provide memory views.

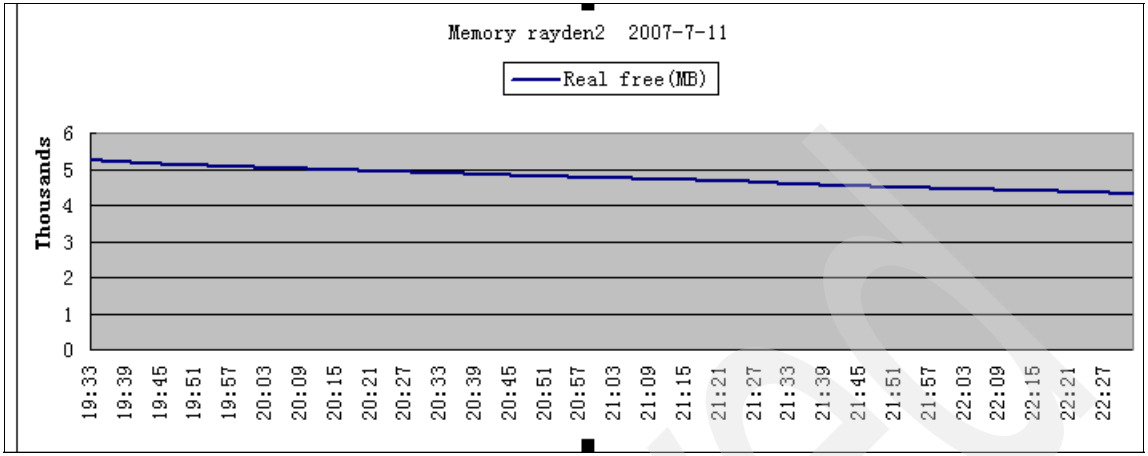


Figure 15-13 Memory summary view

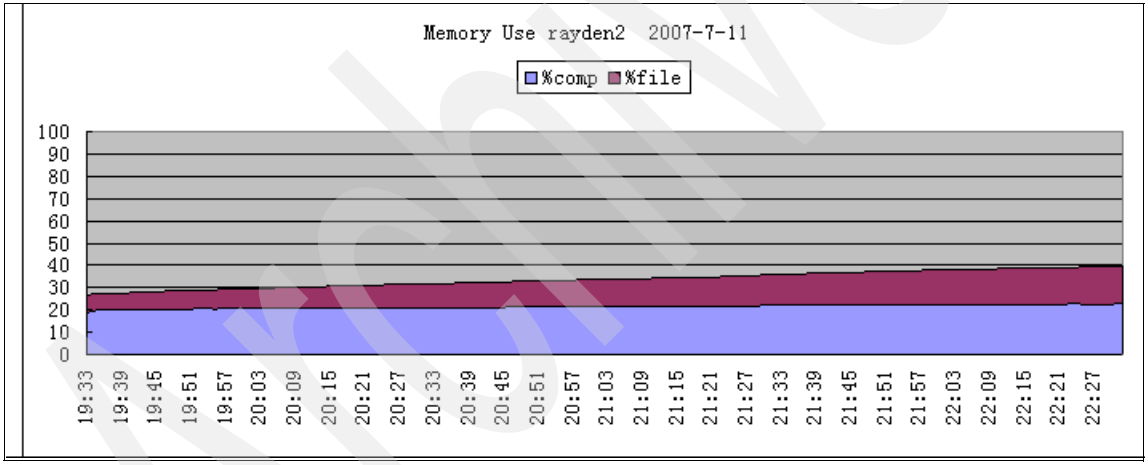


Figure 15-14 Memory use view

Archive

Figure 15-15 provides a network view.

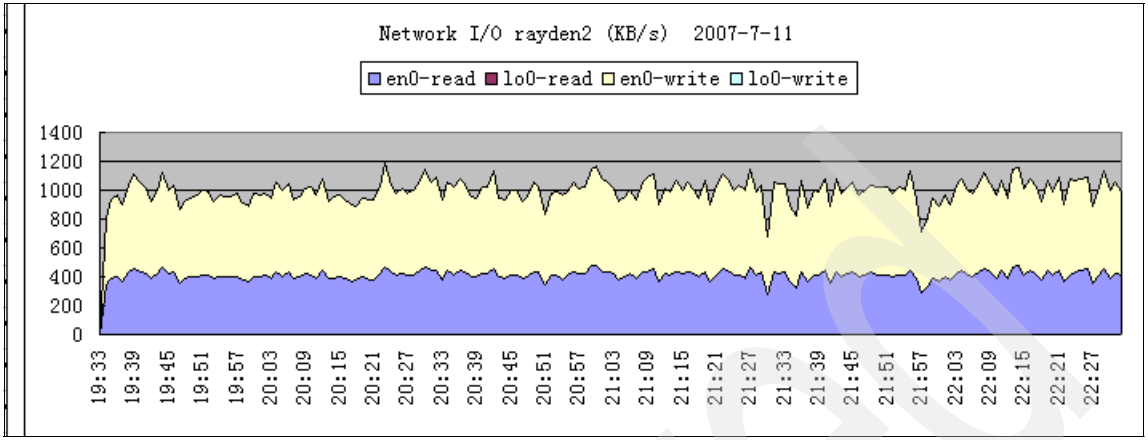


Figure 15-15 Network status view

The best practice for AIX performance analysis

On an AIX operating system, do the following general performance analysis to identify performance issues that might occur in your environment:

1. Identify the level of your AIX operating system.

   In this step, you can use the command **`oslevel -r`** to make sure that you are using the correct operating system level. Also, you should make sure that all of the filesets used for those products exist on the system.

2. Identify the workload of your system.

   You can use the command **`uptime`** to indicate the number of programs that are being executed at the same time, and how long your system is running. A sample output for this command is shown in Figure 15-16.

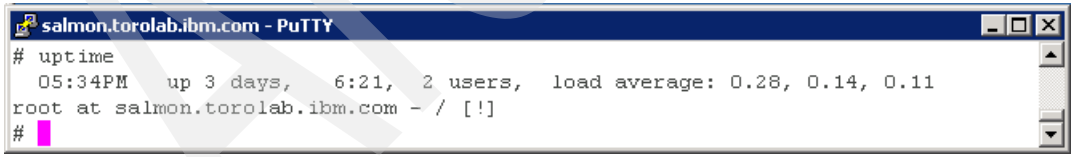


Figure 15-16 Output for command uptime

3. Identify the workload of CPU.

   By using the command **`vmstat`** (or the nmon and nmon analyzer tool to generate the analysis result), you can get an indication of how busy your CPU is, and whether there are any free CPU resources to handle specific and additional requirements. If the result indicates that the CPU is near 100%

utilization, you can safely assume that the CPU has become a performance issue. Specific to the WebSphere Commerce scenario, let us state that there is a significant number of users accessing and shopping on the Commerce server concurrently. For example, there are greater than 20 concurrent users. If the server is not powerful enough to manage the activities above and beyond the 20 concurrent users, the CPU of the server can easily be driven to 100% utilization. Therefore, you must reconfigure your server to have a CPU or multiple CPUs that can support the number of concurrent users and reduce the utilization of the CPUs to a more acceptable utilization rate and where performance is satisfactory.

4. Identify the memory and swap status in your system.

   The primary memory is the most important resource a computer has. Since CPUs are only made with instructions for reading and writing to memory, no programs would be able to run without it. If we identify that the CPU is working in a stable/reasonable status, it is the time to consider whether there is a problem in your memory. You can use the command **vmstat** or **nmon** to analyze the status of memory.

   WebSphere Commerce is a complex eBusiness solution based on J2EE techniques. Generally, a large number of Java objects will be created and loaded into JVM in Commerce runtime, which is really memory consumed in the Commerce infrastructure. Although you want to enlarge the JVM to support Commerce server runtime to contain more capacity to handle business requirements, sometimes this does not work, since it is up to the status of your memory. If memory is already 100% consumed for JVM and system usage, there should be a bottleneck in the system memory.

5. Identify the disk I/O status in your system.

   Disk I/O may be another bottleneck in your system when it is under peak transactions. Sometimes we can see that the status of CPU and memory is good, but the system suffers serious and long response times. Or we can see that the rate of CPU idle is high. In that case, we should convert to disk I/O. From the result from **iostat** or **nmon** tooling, we can identify the frequency of I/O read and I/O write. If there is always high I/O access to the disk, this might be a problem of disk bottleneck in the system.

6. Match the symptom with one of the monitoring results that you got from previous steps, then locate the root cause of this problem.

7. If you cannot find the root cause from the previous steps, it should be some type of product-specific problem. You can continue reading this book to learn more about how to monitor an individual product in the WebSphere Commerce environment separately.

15.4 Summary

In this chapter, we introduced some tools for operating system monitoring (some of which are operating system specific). Be aware that, as the basic infrastructure for WebSphere Commerce, operating system monitoring is always the basic monitoring information that we have to collect to make sure that the entire system is working with a stable status.

Archived

16

IBM DB2 Universal Database

The database is usually one of the potential areas for bottlenecks that makes WebSphere Commerce unable to scale and perform well. It is therefore crucial that the database performs appropriately and efficiently for your implementation.

In Chapter 5, “Database tier High Availability” on page 39, and Chapter 9, “High Availability solution for IBM DB2 Universal Database” on page 145, we describe the existing solutions to build up a High Availability runtime environment in WebSphere Commerce topology and how to set up such an environment. In this chapter, we discuss the following topics:

- DB2 performance considerations in WebSphere Commerce
- DB2 monitoring
- DB2 tuning
- Best practices in WebSphere Commerce

© Copyright IBM Corp. 2008. All rights reserved.

16.1 DB2 performance considerations

Before attempting to tune the database, it is important to realize that a WebSphere Commerce implementation is considered an On-Line Transaction Processing (OLTP) application. Therefore, the database should be tuned for an OLTP application. Typical characteristics of an OLTP application are:

- It has frequent insert or update activity.
- There are a high number of users accessing the database.
- SQL executing against the database is quick and relatively simple.

The following sections cover some of the specifics related to DB2 UDB V8.1 for distributed platforms. A more detailed document on DB2 for distributed platforms and WebSphere Commerce can be found at the WebSphere Commerce Developer Domain site at:

`http://www.ibm.com/websphere/developer/zones/commerce`

This section identifies and reiterates some of the key considerations.

16.1.1 Physical environment considerations

Considerations for the physical environment are related to how the data is actually spread among the disks and how the memory is managed for the databases, so that I/O and memory always attract most of the developers' and testers' concentration in implementing WebSphere Commerce.

Efficient I/O management

Reading from the database and writing back to the database (disk I/O) may become a bottleneck for any application accessing a database. Proper database layout can help reduce the potential for this bottleneck. It is a significant effort to change the physical layout of the database once it has been created. Hence, proper planning at the initial stages is important.

The first consideration is to ensure that the DB2 transaction logs reside on their own physical disk. Every update issued against the database is written to the logs (in addition to being updated in memory). Hence, there will be a lot of disk I/O in the location where the DB2 transaction logs reside. It is a good practice to try to ensure that all read/write activity on the disk is related only to the transaction logs, thus eliminating any I/O contention with other processes that may access the disk.

To set the location for the DB2 transaction logs, issue the following command:

```
db2 update db cfg for <dbalias> using NEWLOGPATH <path>
```

Archived

You will need to deactivate the database (disconnect all connected sessions or issue the **db2 deactivate** command) before the new location for the logs will be used.

Configuration parameters related to the log files can be found in 16.3.2, “Parameters related to transaction logs” on page 358.

The second consideration in terms of disk layout is to determine how to manage the tablespaces efficiently. One performance principle in the management of Relational Database Management Systems (RDBMs) is to separate the database table data and database index data onto different physical disks. This enables better query performance, since index scans can execute in parallel with data fetch operations because they are on different physical disks.

In DB2, two types of tablespaces can be defined:

- System Managed Storage (SMS) tablespaces, which is the default
- Database Managed Storage (DMS) tablespaces

The separation of table and index data can only be specified if database-managed storage tablespaces are defined. So should you always use DMS tablespaces? Not necessarily. In many installations, you may not have the luxury of many physical disks, and hence will not be able to separate data and index. Additionally, SMS tablespaces are easier to administer as compared to DMS tablespaces, so you may wish to trade off some performance for ease of use. A lot of up-front planning is required for DMS tablespaces, since the space is pre-allocated at creation time. SMS tablespaces allocate space as it is required by the database.

If you choose to stick with the default and use SMS tablespaces, there is a utility that you can run to improve write performance. Issue the following command:

```
db2empfa <dbalias>
```

After this command has been issued, as new space is required by the database, it will be allocated one extent at a time, as opposed to one page at a time.

If you choose to go with DMS tablespaces, in addition to redefining the three user tablespaces (USERPACE1, TAB8K, TAB16K), you will also need to define an additional 4 K tablespace for index data. You then have to modify the table definition in the schema creation file to point to the new tablespace. For example, if you chose to name the additional tablespace IND4K, then you would need to access the file:

WC_Install_Dir/schema/db2/wcs.schema.sql

You would replace the INDEX IN clause for every table definition to use IND4K. For example, consider the following coding:

```
CREATE TABLE acacgpdesc (
       acactgrp_id          INTEGER NOT NULL,
       displayname          VARCHAR(254) NOT NULL,
       description          VARCHAR(254),
       language_id          INTEGER NOT NULL
)
  IN USERSPACE1
  INDEX IN USERSPACE1;
```

It changes as follows:

```
CREATE TABLE acacgpdesc (
       acactgrp_id          INTEGER NOT NULL,
       displayname          VARCHAR(254) NOT NULL,
       description          VARCHAR(254),
       language_id          INTEGER NOT NULL
)
  IN USERSPACE1
  INDEX IN IND4K;
```

This must be done prior to your WebSphere Commerce instance creation.

Note: Although you have different page sizes for your tablespaces for table data, you only need one size for the tablespace for index data, because all indexes created are less than 4 K in length.

On the same subject of achieving better disk I/O, read performance can be improved by spreading the tablespaces across many physical disks. When the database manager has to read from disk, if the data is stored across multiple disks, it can be read in parallel, yielding better read performance. This can be done in one of two ways:

- Specifying multiple containers in the tablespace definition with each container specifying a file system that resides on different physical disks.
- Specifying a single container, but the container is a file system that spans several physical disks. You normally would rely on the operating system or a volume manager to help define the file system.

A utility such as iostat on UNIX platforms can be used to identify disk I/O bottlenecks. On an AIX platform, you can install the nmon, which is a useful tool to analyze disk I/O.

Configuration parameters related to disk I/O can be found in 16.3.3, “Parameters related to disk I/O” on page 359.

Archived

Memory usage

DB2 associates memory for the database through the use of bufferpool objects. A bufferpool has a page size associated with it and is linked to one or more tablespaces. Thus, if tablespaces of different page sizes are created, then bufferpools corresponding to the different page sizes are required.

While you can create multiple bufferpools having the same page size, we recommend that only one bufferpool per page size be created, for the most efficient usage of memory on the database server.

The question is always how much memory to assign to the bufferpools. For DB2 32-bit implementations, there is a limit, based on the operating system, that can be available for bufferpools. This ranges from a maximum of 1.5 GB on AIX platforms to 3.3 GB on Solaris.

Assuming a dedicated database server, a general rule of thumb is to allocate a large proportion of memory available on the server, about 75% to 80%, but not exceed the platform limits.

Note that for 64-bit implementations of DB2, the limits are significantly increased. In this case, the bufferpool hit ratio would need to be monitored to determine the optimal setting for the bufferpools. You can also monitor the hit ratio for 32-bit implementation using database snapshots using the following command:

```
db2 get snapshot for database on <dbalias>
```

The output generated will contain some statistics on bufferpool logical and physical reads:

```
Buffer pool data logical reads              = DLR
Buffer pool data physical reads             = DPR
...
Buffer pool index logical reads             = ILR
Buffer pool index physical reads            = IPR
```

In this output, DLR, DPR, ILR, and IPR will have actual values. The hit ratio can be computed using the following formula:

```
(1 - (( DPR + IPR) / (DLR + ILR))) * 100%
```

The size of the bufferpool can be changed using the ALTER BUFFERPOOL command, or the BUFFPAGE parameter if the size of the bufferpool is set to -1.

16.1.2 DB2 objects management

This section describes how DB2 manages the objects in it, which contain tables, indexes, tablespaces, and agents.

Table management

After many changes to table data, logically sequential data may be on non-sequential physical data pages so that the database manager must perform additional read operations to access data. Additional read operations are also required if a significant number of rows have been deleted. In such a case, you might consider reorganizing the table to match the index and to reclaim space. You can reorganize the system catalog tables as well as database tables.

Note: Because reorganizing a table usually takes more time than running statistics, you might execute RUNSTATS to refresh the current statistics for your data and rebind your applications. If refreshed statistics do not improve performance, reorganization might help. For detailed information about the options and behavior of the REORG TABLE utility, refer to its command reference.

Consider the following factors, which might indicate that you should reorganize a table:

- A high volume of insert, update, and delete activity on tables accessed by queries.
- Significant changes in the performance of queries that use an index with a high cluster ratio.
- Executing RUNSTATS to refresh statistical information does not improve performance.
- The REORGCHK command indicates a need to reorganize your table.
- The tradeoff between the cost of increasing degradation of query performance and the cost of reorganizing your table, which includes the CPU time, the duration time, and the reduced concurrency resulting from the REORG utility locking the table until the reorganization is complete (performance overhead).

Efficient table management can improve the performance of the entire database system, but before that, we recommended analyzing and determining how to performance reorganize tables, and when it is necessary to perform a table reorganization. The best approach for DB2 is to examine the statistics collected by RUNSTATS. From the information RUNSTATS gets, the statistics show the data distribution within tables, the number of used and empty pages, and RIDs marked deleted in index leaf pages. The statistics also provide information about prefetch efficiency. If you run RUNSTATS regularly and analyze the statistics over a period of time, you can get more indication about the performance trend of the database system.

Index management

If there is no index defined in a table, a table scan must be performed for each each table referenced in a database query. When the database system is running for a long time, such as table orders in the Commerce database, it will become larger and larger. The result is that the larger the table, the longer a table scan takes since a table scan requires each table row to be accessed sequentially. In most cases, for a SQL query that might return only some rows, an index scan is much faster than a table scan, which will give the DB2 database system more capacity to handle other transactions' requests at a consolidated duration.

Note: Commerce already has many indexes defined. Therefore, users should not need to create them everywhere, but carefully consider the implications that are listed below.

Although indexes can reduce access time significantly, they can also have adverse effects on performance. Before you create indexes, consider the effects of multiple indexes on disk space and processing time:

- Each index requires storage or disk space. The exact amount depends on the size of the table and the size and number of columns in the index.
- Each INSERT or DELETE operation performed on a table requires additional updating of each index on that table. This is also true for each UPDATE operation that changes the value of an index key.
- The LOAD utility rebuilds or appends to any existing indexes.
- The indexfreespace MODIFIED BY parameter can be specified on the LOAD command to override the index PCTFREE used when the index was created.
- Each index potentially adds an alternative access path for a query for the optimizer to consider, which increases the compilation time.

Choose indexes carefully to address the needs of the application program. Some general index planning tips are:

- To avoid some sorts, define primary keys and unique keys.
- To access tables with a small size efficiently, try to use index to optimize frequent queries to create an index on any column that you will use when joining tables is necessary.
- To search efficiently, decide between ascending and descending ordering of keys depending on the order that will be used most often.
- We recommend using the SQL Explain facility to determine whether creating an index on specific columns is necessary.

Archived

- To plan indexes, you can use the db2adv tool to get advice about indexes that might be used by one or more SQL statements.

Tablespace management

If you want to use Database Managed Storage devices as containers for tablespaces, you should consider following factors to make sure to effectively manage DMS:

- How to perform file system caching

  File system caching is performed as follows:

  - For DMS device container tablespaces, the operating system does not cache pages in the file system cache.
  - For a DMS file container, the operating system might cache pages in the file system cache.

- How to buffer data read

  A database buffer pool can store the data read from disk casually. But considering that there is a limitation to the buffer pool space, some data pages might be deleted from the buffer pool before the application actually uses this page. In this case, you might be able to increase the size of the database buffer pool to achieve more buffered data, which eventually improves the performance of the database. But on the other hand, it consumes more memory at the same time, so that you should identify whether there is buffer memory that can be used for the database buffer pool.

- Using LOB or LONG data

  When an application retrieves either LOB or LONG data, the database manager does not cache the data in its buffers. Each time an application needs one of these pages, the database manager must retrieve it from disk. However, if LOB or LONG data is stored in SMS or DMS file containers, file system caching might provide buffering and, as a result, we can get better performance.

Agent management

For each database that an application accesses, various processes or threads start to perform the various application tasks. These tasks include logging, communication, and prefetching.

There are four types of agents defined in the DB2 system:

- Idle agents
- Inactive agents
- Active coordinator agents
- Subagents

Archived

Most applications establish a one-to-one relationship between the number of connected applications and the number of application requests that can be processed by the database. However, it may be that your work environment is such that you require a many-to-one relationship between the number of connected applications and the number of application requests that can be processed.

The ability to control these factors separately is provided by two database manager configuration parameters, max_connections and max_coordagents. Before applying these two parameters to modify your agent working mechanism, we highly recommend thoroughly analyzing the database system based on your requirement, to prevent reaching the upper limitation of available database system resources.

16.2 DB2 monitoring

The normal approach to identifying a database performance issue is to utilize the results acquired from the database monitoring utility.

16.2.1 Introduction

DB2 should collect information from the database manager, its databases, and any connected applications. With these functionalities, we can achieve:

- Problem determination

  Problem determination requires a sense of what is happening now, as it is necessary to see what is causing, or has recently caused, the problem. With some performance management and trending, you can avoid most problems.

- Performance management

  Performance management allows you to use system resources optimally, and helps ensure that some problems are avoided. By using performance management information and techniques, you can try to avoid some time on problem determination and increase overall user satisfaction.

- Trend analysis

  Trend analysis takes performance management to another level, where historical data is kept and used to determine growth and trends in usage. Trends help you identify changes in overall system activity and plan hardware upgrades if they are needed.

Archive

In DB2, there are two primary tools with which you can access system monitor information, each serving a different purpose:

- The *Snapshot™ monitor* enables you to capture a picture of the state of database activity at a particular point in time (the moment the snapshot is taken).
- The *Event monitor* logs data as specified database events occur.

The system monitor provides multiple means of presenting monitor data to you. For both Snapshot and event monitors you have the option of storing monitor information in files or SQL tables, viewing it on screen (directing it to standard-out), or processing it with a client application.

Collecting system monitor data introduces processing overhead for the database manager, which not only consumes more CPU resources, but also increases memory consumption to store the collected data.

In order to minimize the overhead involved in maintaining monitoring information, monitor switches control of the collection of potentially expensive data by the database manager. Each switch has only two settings: ON or OFF. If a monitor switch is OFF, the monitor elements under that switch's control do not collect any information. There is a considerable amount of basic monitoring data that is not under switch control, and will always be collected regardless of switch settings.

The typical information that you can get from the DB2 system monitor can be classified as listed in Table 16-1.

Table 16-1 DB2 monitoring system information classification

Monitor switches	**Information collected**
Buffer Pool	Buffer pool usage statistics
Lock Info	The number of locks that have occurred and deadlocks
Sort Info	Sort overflows, number of sorts
Statement	Seeing what SQL statements are currently running on the DB2 server
Timestamp Info	Timestamp information
Unit of Work	Statistics for units of work included in start and stop time and status

You can use the GET SNAPSHOT command to get snapshot information once you have turned on the switches.

To check the status of the monitor switches, you can use the get monitor switches command. Figure 16-1 shows the output of this command.

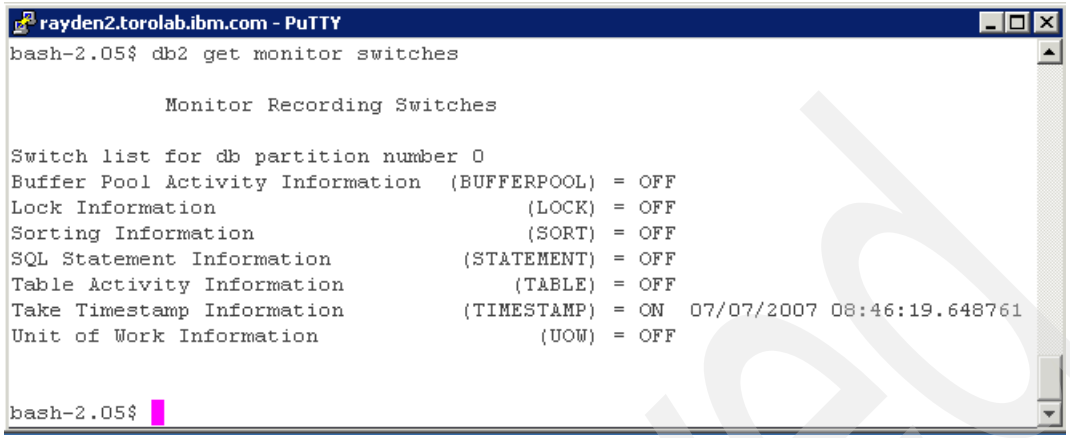


Figure 16-1 Status of monitor switches

16.2.2 Snapshot monitor

After you turn on the switch for the specific object that you want to monitor, you can use the GET SNAPSHOT command to get the a snapshot. Table 16-2 shows the available commands that you can use for viewing the status of the database system.

Table 16-2 Commands list for viewing DB2 snapshots

Snapshot	Command
Buffer Pool	db2 get snapshot for bufferpools on dbname
Locks	db2 get snapshot for locks on dbname
Dynamic SQL	db2 get snapshot for dynamic sql on dbname
Table Activity	db2 get snapshot for tables on dbname
Applications	db2 get snapshot for applications on dbname
Tablespace	db2 get snapshot for tablespaces on dbname
Database	db2 get snapshot for database on dbname
Database Manager	db2 get snapshot for DBM

Archived

From the DB2 command line, issue the commands listed in Table 16-2 on page 353 to get a snapshot like Example 16-1.

Example 16-1 DB2 DBM snapshot

```
bash-2.05$ db2 get snapshot for DBM

            Database Manager Snapshot

Node type                                      = Enterprise Server
Edition with local and remote clients
Instance name                                  = db2inst8
Number of database partitions in DB2 instance  = 1
Database manager status                        = Active

Product name                                   = DB2 v8.1.1.96
Service level                                  = s050811 (U803920)

Private Sort heap allocated                    = 0
Private Sort heap high water mark              = 865
Post threshold sorts                           = Not Collected
Piped sorts requested                          = 108505
Piped sorts accepted                           = 108505

Start Database Manager timestamp               = 07/07/2007
08:46:19.648761
Last reset timestamp                           =
Snapshot timestamp                             = 07/09/2007
16:08:39.221033

Remote connections to db manager               = 19
Remote connections executing in db manager     = 0
Local connections                              = 2
Local connections executing in db manager      = 0
Active local databases                         = 1

High water mark for agents registered          = 27
High water mark for agents waiting for a token = 0
Agents registered                              = 27
Agents waiting for a token                     = 0
Idle agents                                    = 3

Committed private Memory (Bytes)               = 5603328

Switch list for db partition number 0
```

Archived

```
Buffer Pool Activity Information  (BUFFERPOOL) = OFF
Lock Information                        (LOCK) = ON  07/07/2007
08:50:02.428191
Sorting Information                     (SORT) = OFF
SQL Statement Information          (STATEMENT) = OFF
Table Activity Information             (TABLE) = OFF
Take Timestamp Information         (TIMESTAMP) = ON  07/07/2007
08:46:19.648761
Unit of Work Information                 (UOW) = OFF

Agents assigned from pool                      = 4406
Agents created from empty pool                 = 36
Agents stolen from another application         = 0
High water mark for coordinating agents        = 27
Max agents overflow                            = 0
Hash joins after heap threshold exceeded       = 0

Total number of gateway connections            = 0
Current number of gateway connections          = 0
Gateway connections waiting for host reply     = 0
Gateway connections waiting for client request = 0
Gateway connection pool agents stolen          = 0

Memory usage for database manager:

    Memory Pool Type                           = Database Monitor Heap
       Current size (bytes)                    = 180224
       High water mark (bytes)                 = 180224
       Configured size (bytes)                 = 376832

    Memory Pool Type                           = Other Memory
       Current size (bytes)                    = 2801664
       High water mark (bytes)                 = 2834432
       Configured size (bytes)                 = 18661376
```

Archived

16.2.3 Event monitor

To start your SQL event monitoring:

1. Open a new DB2 command-line processor session and execute the following DB2 UDB commands:

```
db2 connect to dbname user username using password
db2 update monitor switches using statement on
db2 create event monitor SampleMon for statements write to file
'/tmp/sample' maxfiles 10 maxfilesize 10000
db2 set event monitor SampleMon state=1
```

   Note: Keep this session open until the database activities are complete. Make sure that the /temp/sample directory is sufficiently large to hold the trace files. The /tmp/sample directory is chosen, as all the users have access to this directory. However, this directory could be replaced with any other directory.

2. Perform the normal database activities.

   You will set the criteria for those regular database activities as well as potential issues that you seek to track during a specific period to be monitored. During monitoring, you will see a group of files in the /tmp/sample directory with an .evt extension. The size of the files is determined by the parameters and duration of time set by you. During monitoring, you should be able to see a group of files in the /tmp/sample directory with an .evt extension, and the sizes of these files are up to what information you set to be monitored and how long it lasts.

3. Go to the session that you opened in step 1 and issue the statement:

```
db2 set event monitor SampleMon state=0
db2 terminate
```

4. Execute the following command from a normal command-line session:

```
$ db2evmon -path /temp/sample > small.statement
```

   All the captured SQL statements and their details will be captured in the single file small.statement.

Since abundant SQL statements are generated into the file mall.statement, it is important for you to identify what exactly you want. Example 16-2 is a sample output for an SQL statement.

Example 16-2 Sample output of SQL statement monitoring

```
409) Statement Event ...
  Appl Handle: 1373
```

```
  Appl Id: P7957356.D9C4.044423183000
  Appl Seq number: 0057
  Record is the result of a flush: FALSE
  ------------------------------------------
  Type     : Dynamic
  Operation: Close
  Section  : 37
  Creator  : NULLID
  Package  : SYSSH200
  Consistency Token  : SYSLVL01
  Package Version ID  :
  Cursor   : SQL_CURSH200C37
  Cursor was blocking: TRUE
  Text     : SELECT T1.XATTRIBUTE_ID, T1.LANGUAGE_ID, T1.ENUM, T1.NAME,
T1.DESCRIPTION, T1.DISPLAYLABEL, T1.XGLOSSARY_ID, T1.ISSTORESPECIFIC,
T1.LASTUPDATE, T1.CONTENTSOURCE, T1.USAGE, T1.FIELD1, T1.FIELD2,
T1.OPTCOUNTER FROM XATTRIBUTE  T1 WHERE T1.DESCRIPTION=?
  ------------------------------------------
  Start Time: 07/08/2007 12:50:04.372305
  Stop Time:  07/08/2007 12:50:04.375307
  Exec Time:  0.003002 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: 230
  Rows written: 0
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
  Bufferpool data logical reads: 1123
  Bufferpool data physical reads: 0
  Bufferpool temporary data logical reads: 0
  Bufferpool temporary data physical reads: 0
  Bufferpool index logical reads: 0
  Bufferpool index physical reads: 0
  Bufferpool temporary index logical reads: 0
  Bufferpool temporary index physical reads: 0
  SQLCA:
  sqlcode: 100
  sqlstate: 02000
```

Archived

Generally, four types of SQL statements need to be identified carefully:

- SQL statements consume the most execution time.
- SQL statements consume the most sort time.
- SQL statements consume the most CPU resource.
- SQL statements run most frequently.

16.3 DB2 tuning in WebSphere Commerce

There are many parameters to consider for performance. This section describes a subset of these that are considered important for WebSphere Commerce implementations. To set the values for the parameters, the following command can be used:

```
db2 update db cfg for <dbalias> using <paramname> <paramvalue>
```

16.3.1 Parameters related to memory

The database heap (DBHEAP) contains control block information for database objects (tables, indexes, and bufferpools), as well as the pool of memory from which the log buffer size (LOGBUFSZ) and catalog cache size (CATALOGCACHE_SZ) are allocated. Its setting is dependent on the number of objects in the database and the size of the two parameters mentioned.

In general, the following formula can be used to estimate the size of the database heap:

```
DBHEAP=LOGBUFSZ + CATALOGCACHE_SZ + (SUM(# PAGES in each bufferpool) * 3%)
```

The log buffer is allocated from the database heap, and is used to buffer writes to the transaction logs for more efficient I/O. The default size of this setting one hundred and twenty-eight 4 K pages. A recommended starting point for the log buffer size (LOGBUFSZ) in WebSphere Commerce implementations is 256.

16.3.2 Parameters related to transaction logs

When considering values for the transaction log file size (LOGFILSZ) and the number of primary (LOGPRIMARY) and secondary (LOGSECOND) logs, some generalizations for OLTP applications can be applied. A high number of short transactions are typical in OLTP systems. Hence, the size of the log file should be relatively large. Otherwise, more processing time will be spent managing log files and writing to the transaction logs. A good starting point for the size of the log file in WebSphere Commerce implementations is to set the value to 10000.

Archived

Primary log files are allocated when the database is activated, or on the first connect. If a long-running transaction fills up all the primary logs, then secondary logs will be allocated as needed until the LOGSECOND limit is reached. The allocation of a secondary log file is a significant performance hit, and should be minimized if it cannot be avoided.

To determine the correct settings for these parameters, you need to monitor the database and see whether secondary log files are being allocated. If they are, then you need to increase the number of primary log files. You can monitor this by taking a database snapshot and looking for the following two lines:

```
Maximum secondary log space used (Bytes)   = 0
Secondary logs allocated currently         = 0
```

A good starting point for the number of primary log files (LOGPRIMARY) is anywhere from 6 to 10.

16.3.3 Parameters related to disk I/O

In addition to physical disk layout, several tuning parameters can be manipulated to affect disk I/O. Two key parameters are NUM_IOSERVERS and NUM_IOCLEANERS.

- NUM_IOSERVERS specifies the number of processes that are launched to prefetch data from disk to the bufferpool pages. To maximize read parallelism, this parameter should be set to the number of physical disks that are being used by the database, to enable reading from each disk in parallel.
- NUM_IOCLEANERS specifies the number of processes that are launched to flush dirty bufferpool pages to disk. To maximize usage of system resources, this parameter should be set to the number of CPUs on the system.

The frequency of how often dirty bufferpool pages are flushed to disk can be influenced by the CHNGPGS_THRESH parameter. Its value represents the limit, in the form of a percentage, that a bufferpool page can be dirty before a flush to disk is forced. For OLTP applications, we recommend a lower value. For WebSphere Commerce implementations, the value should be set to 40.

One final parameter to consider in this section is MAXFILOP. It represents the maximum number of files DB2 can have open at any given time. If this value is set too low, valuable processor resources will be taken up to open and close files. This parameter needs to be monitored to be set to the correct value, but a good starting point is to set this value to 128. You can monitor this by taking a database snapshot and looking at the following line:

```
Database files closed                      = 0
```

Archived

If the value monitored is greater than zero, then the value for this parameter should be increased.

16.3.4 Parameters related to locking

Reducing locking contention is key to performance. Several parameters exist to influence locking behavior. The total amount of memory available to the database for locks is defined by the LOCKLIST parameter. The MAXLOCKS parameter defines the maximum amount of memory available for each connection to the database. It is represented as a percentage of the LOCKLIST.

Both of these parameters need to be sized appropriately in order to avoid lock escalations. A lock escalation occurs when all of the memory available to a connection is used, and multiple row locks on a table are exchanged for a single table lock. The amount of memory used for the first lock on an object is 72 bytes, and each additional lock on the same object is 36 bytes.

A good starting value for LOCKLIST can be approximated by assuming that a connection requires about 512 locks at any given time. The following formula can be used:

```
LOCKLIST = (512 locks/conn * 72 bytes/lock * # of database connections)
/ 4096 bytes/page
```

MAXLOCKS can be set to between 10 and 20 to start. Further monitoring will be necessary to adjust both of these values. In the database snapshot output, look for the following lines:

```
Lock list memory in use (Bytes)             = 432
Lock escalations                            = 0
Exclusive lock escalations                  = 0
```

If lock escalations occur (value higher than 0), increase the locklist to minimize the escalations or increase the MAXLOCKS value to increase the limit of how much of the LOCKLIST a connection can use.

16.3.5 Parameters related to agents management

The following database manager configuration parameters determine how DB2 database agents can be created and managed:

- Maximum number of agents (maxagents): The number of agents that can be working at any one time. This value applies to the total number of agents that are working on all applications, including coordinator agents, subagents, inactive agents, and idle agents.

- Agent pool size (num_poolagents): The total number of agents, including active agents and agents in the agent pool, that are kept available in the system. The default value for this parameter is half the number specified for maxagents.
- Initial number of agents in pool (num_initagents): When the database manager is started, a pool of worker agents is created based on this value. This speeds up performance for initial queries. The worker agents all begin as idle agents.
- Maximum number of connections (max_connections): This specifies the maximum number of connections allowed to the database manager system on each partition.
- Maximum number of coordinating agents (max_coordagents): For partitioned database environments and environments with intra-partition parallelism enabled when the connection coordinator is enabled, this value limits the number of coordinating agents.
- Maximum number of concurrent agents (maxcagents): This value controls the number of tokens permitted by the database manager.

In some peak usage system, where the resources for memory, CPU, and disk are almost 100% utilized, insufficient configuration might cause performance degradation for peak load periods. You can modify some of these parameters to control the load and avoid performance degradation.

16.3.6 Best practices

In this section we describe some of the most common best practices for any IBM DB2 UDB implementation.

Reorganizing data in tablespaces

When a high number of inserts, updates, or deletes have been issued against a table in the database, the physical placement of the rows and related indexes may not be optimal. DB2 provides a utility to reorganize data for a table:

```
db2 REORG TABLE <tabschema>.<tabname>;
```

DB2 also provides a utility to check whether a table or index data needs to be organized. While connected to a database, the following command can be issued:

```
db2 REORGCHK
```

This command checks all tables in the database and produces a listing, first by table and second by index. In the listing, an asterisk (*) in any of the last three columns implies that the table or index requires a REORG.

Archived

Collecting statistics

Each SQL statement submitted to the database is parsed, optimized, and a statement access plan is created for execution. To create this access plan, the optimizer relies on table and index statistics. In order for the optimizer to generate the best access plan, up-to-date statistics are required. Collecting statistics frequently (or at least when a significant amount of data changes) is a good practice.

To collect statistics for a table, the following command can be issued:

```
db2 RUNSTATS ON table <tabschema>.<tabname> WITH DISTRIBUTION AND
DETAILED INDEXES ALL;
```

Statistics on the catalog tables should also be collected.

16.4 Utilities in database tier for WebSphere Commerce

In WebSphere Commerce, the site administrator plays a critical role in perfuming system administration tasks. Besides the basic tasks, includes install, configure, and maintain WebSphere Commerce and the associated software and hardware. Another key responsibility of the site administrator is to manage the Commerce database to support High Availability online stores operating 24 hours a day, 365 days of the year.

WebSphere Commerce has already delivered some useful database utilities to achieve High Availability and high performance, which include:

- Massload
- Staging server
- DBClean

16.4.1 Massload

The WebSphere Commerce Loader package consists primarily of command utilities for preparing and loading data into a WebSphere Commerce database. You can use the Loader package to load large amounts of data and to update data in your WebSphere Commerce database.

The Loader command utility in this package uses valid and well-formed XML as input to load data into the database. Elements of the XML document map to table names in the database, and element attributes map to columns.

The Loader command utilities allow you to do the following:

- Transform data between a character-delimited variable format and an XML data format.
- Transform XML data into alternate XML formats.
- Generate a DTD based on the target database.
- Generate identifiers for XML elements.
- Load data into the WebSphere Commerce database.
- Extract data from a database as an XML document.

Table 16-3 lists all of the utilities included in the Commerce Loader package.

Table 16-3 Loader package utilities

Utility	Description
Text Transformer	The Text Transformer transforms data between a character-delimited variable format and an XML data format.
XML Transformer	The XML Transformer changes, aggregates, and remaps the data in an XML document to alternate XML formats for use by other users or systems as needed.
DTD Generator	The DTD Generator generates a DTD based on the target database to which your data must conform. This DTD will be used throughout the load process.
ID Resolver	The ID Resolver is a Loader package command utility that generates identifiers for XML elements with their associated identifiers.
Loader	The Loader is responsible for populating and updating the WebSphere Commerce database. The Loader is the most common means of loading data into a system.
Extractor	The Extractor extracts selected subsets of data from a database in the form of XML files.
Logger	The Loader package command utilities log messages to indicate success, failure, and errors, as well as to provide program trace information.

How massload works

The massload utility can help the WebSphere Commerce site administrator load an XML input file into a target database. Loading the XML file populates and updates the WebSphere Commerce database. The massload utility allows

Archived

column-level updates to a table. It also allows you to delete data from a database.

Besides the functionalities listed above, the massload utility also includes the features below:

- Error reporter

  The massload utility includes an error reporter that generates an exception document if an error occurred during massload execution.

- Product Advisor search-space synchronization

  If you enable the loading utility's Product Advisor search-space synchronization feature, you can maintain near real-time synchronicity of Product Advisor search spaces and WebSphere Commerce catalog tables being updated by the massload utility.

In the next section we provide the steps for loading data to the WebSphere Commerce database.

Step1: Configure the loading utilities

Ensure that you have finished the necessary configuration for running this utility:

1. Configure the loading utilities.

   Before you can use the loading utilities, you might have to update some of the environment variables used by the utilities. The environment variables are set by the following scripts (for AIX) that are called by the loading utilities and other WebSphere Commerce utilities:

   – Setenv script: *WC_Install_Dir*/bin/setenv.sh
   – Setenv.db2 script: *WC_Install_Dir*/bin/setenv.db2.sh

2. Configure tracing and logging for the loading utilities.

   In addition to the individual logs for some of the loading utilities, you can configure an additional message log file and a trace file for the loading utilities. You can configure the location of the loading utilities message log and trace files and their contents by editing the WCALoggerConfig.xml file. To configure additional tracing and logging for the loading utilities:

   a. Open the following in a text editor:

      WC_Install_Dir/*xml/loader/WCALoggerConfig.xml*

   b. Modify the contents of the file as follows:

      i. Define the parameters for tracing.
      ii. Define the parameters for logging.

   c. Save your changes.

d. Ensure that the full path to WCALoggerConfig.xml is defined as part of the CLASSPATH environment variable before running any of the loading utilities.

3. Configure the massload utility.

   a. Before using the massload utility you might want to change the following to suit your environment.

      By default, the maximum amount of memory allocated to the JVM heap is 64 MB. If this is not increased, the JVM can eventually run out of memory during the load process. The maximum amount of memory allocated to the Java heap can be varied by using the JVM -mx option in the Java command. If you are loading files that are more than 500 MB, then increase the JVM heap size to 512 MB or 1024 MB.

      To modify the Java Virtual Machine (JVM) heap size used for the massload utility:

      i. Open the following massload utility file in a text editor:

         WC_Install_Dir/bin/massload.sh

      ii. Change the JVM heap size to 1024 MB by specifying the -Xms and -Xmx options of the Java command.

         If the -Xms and -Xmx parameters are already specified in the file, change the values to the heap size that you want. Ensure that you update all occurrences of -Xms and -Xmx.

         If the -Xms and -Xmx parameters are not specified in the file, change all occurrences of *%JAVA_HOME%\bin\java* in the utility command file to %JAVA_HOME%\bin\java -Xms1024M -Xmx1024M.

      iii. Save the changes.

   b. (Optional) Change the directory for the massload utility error log.

   c. (Optional) Configure the MassLoadCustomizer.properties file.

Note: In addition to the trace log and message log for loading utilities, this utility produces the following log file for further problem analysis and determination:

- (For DB2): *WC_Install_Dir*/logs/massload.db2.log
- (For Oracle): *WC_Install_Dir*/logs/massload.db2.log

Step 2: Generate a DTD and schema for using loading utilities

The dtdgen utility creates a document type definition (DTD) file to use with the loading utilities. The dtdgen utility uses an input text file containing a list of

database table names and generates a DTD file describing the database, depending on how you invoke the dtdgen utility.

The dtdgen utility can create a DTD file based on the WebSphere Commerce database schema. If you use the DTD files provided with the starter store archives and you do not modify the database schema, you do not need to generate a DTD file using the dtdgen utility.

To generate a DTD file, or a DTD file and XML schema definition (XSD) file:

1. If necessary, configure the loading utilities first. (We discussed it this in “Step1: Configure the loading utilities” on page 364).
2. Create an input text file that contains database table names, one on each line, as shown in Example 16-3.

Example 16-3 Input text file for generating DTD file

```
MEMBER
ADDRBOOK
ADDRESS
```

3. Save the file as a .txt file.
4. Run the dtagen utility. If you have configured the dtdgen utility, make sure that you specify the new file name as the value of the customizer parameter of the DTD Generator command. You can get more information about the Commerce dtagen utility from this link:

   http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp?topic=/com.ibm.commerce.data.doc/refs/rml_dtdgen.htm

Note: If you are loading data for a store archive and created the XML file using the DTDs provided with the store archive, this step is not necessary.

Step 3: Resolve identifiers for records that share identifiers with existing data

If you are loading new data into a WebSphere Commerce database that shares identifiers with data that already exists in the WebSphere Commerce database, you must include the existing data in the XML file that you want to load. The existing data must appear before the new data in the XML file.

An example of this situation is adding a new language to an existing catalog. For example, you have a master catalog in English and you want to add French information to the catalog.

Archived

The idresgen utility generates new identifiers for the new data instead of referencing the existing data for either of the following situations:

- You do not include the existing data in your XML file.
- You include the existing data, but the existing data appears after the new data.

By including the existing data before the new data, the idresgen utility is correctly able to resolve the identifiers and relate them to existing identifiers.

Step 4: Execute the massload utility

Make sure that you have configured correctly, as listed in previous steps, and then we can start running this utility as the non-root WebSphere Commerce user ID. Do not run this command as root. The syntax of the command **massload** is as Figure 16-2.

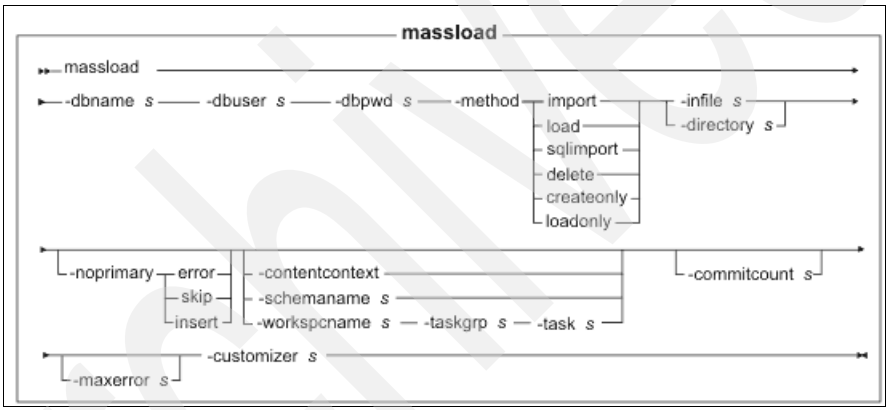


Figure 16-2 Syntax for massload command

Preformation consideration and practice

Loading utilities is sometime time-consuming and resource-consuming, especially when you want to load a huge volume of the content data related to Commerce stores into database.

Below are several considerations that might result in performance degradation with running the massload utility:

- Trace is enabled.

  To disable the tracing if it is enabled:

  a. Log on the WebSphere Application Server admin console:

     `http://hostname:port/ibm/console`

b. Click **Servers** → **Application Servers**.

c. From the Application Servers list, select the server name (for example, WC_demo) for your implementation.

d. Click **Logging and Tracing** → **Diagnostic Trace**. The diagnostic trace service panel is shown, as in Figure 16-3.

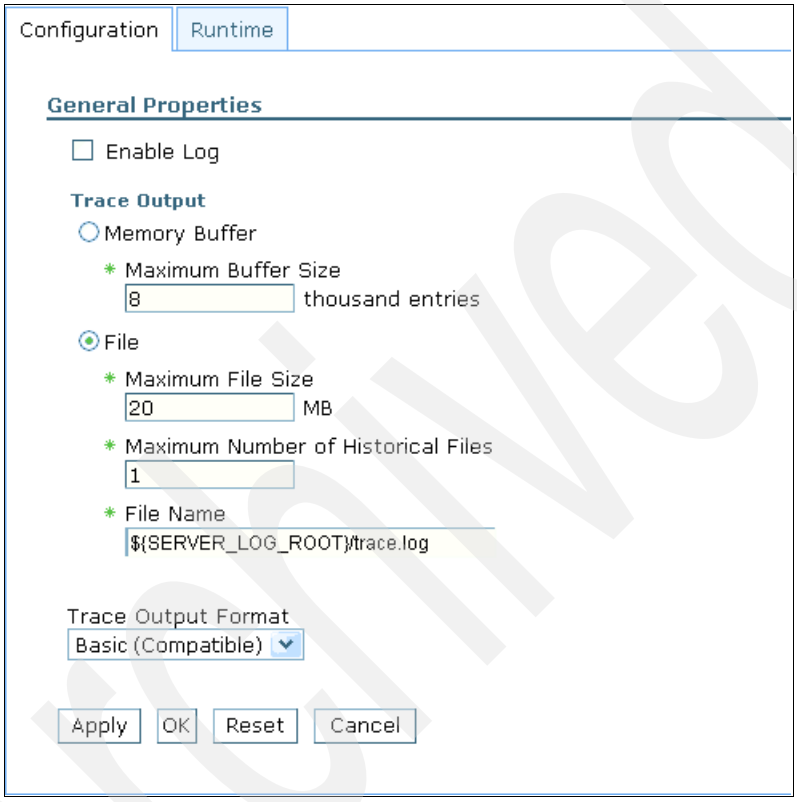


Figure 16-3 Diagnostic trace panel

e. Make sure that the Enable Log check box is not selected.

f. After changing the configuration, click **Apply** → **Save** to save the change.

- Not enough memory

  Make sure that you have applied enough paging for loading huge volumes of data.

  If massload is running during heavy workload on the server, locking issues may be causing the problem. This can sometimes be eliminated by disabling the cache triggers.

Archived

Note: For more information about enabling and disabling the cache triggers, refer to the Enabling and disabling caching task in the IBM WebSphere Commerce online help found at the following Web site:

http://www-306.ibm.com/software/genservers/commerce/wcpe/library/lit-tech-general-en.html

16.4.2 Staging server

Most online stores operate 24 hours a day, 365 days of the year, making it difficult to perform maintenance or test changes to the system. The WebSphere Commerce staging server allows the site administrator to update the data on the staging server and test the changes, and then propagate the change to the production server. This is useful for testing updates to the product catalog, but it is also important for testing new shopping process commands.

Staging server introduction

The WebSphere Commerce server application server environment on a staging server is very similar to the production server. The staging server requires exactly the same hardware and software and operating system configuration as the production server. Both the staging and production environments include the WebSphere Application Server, WebSphere Commerce Server, a database server, and a database. The versions of the software and operating systems must also be the same.

- The staging server contains a complete WebSphere Commerce Server instance, just like the production server. Normally, the staging server instance is created on a dedicated machine on your network.
- The staging server environment does not have to have the same number of nodes as the production environment. For example, a production environment may have the database on one physical machine, the WebSphere Commerce Server on another physical machine, and the HTTP server on a third machine. The staging environment can have these three elements on one physical machine.
- From the application content's point of view, the staging server has the same EAR file, the same WAR file, and the same database schema as the production server.
- From the data content's point of view, the staging server has the same JSP pages, HTML pages, Java files, and the same data in the database as the production server.

Archived

- From the function's point of view, the staging server has the same function as the production server. Users can perform the same actions on the staging server as they would on the production server. For example, a user can launch the WebSphere Commerce Accelerator, perform a shopping flow, load the database using the loading utilities, or publish a store.

These similarities allow a user to test changes on the staging server as though it were the production server. These tests reassure users that changes that run correctly on the staging server will run correctly on the production server. This is the basic philosophy of the staging server.

The staging server consists of the following components:

- A WebSphere Commerce instance

  Tests and modifies your data.

- Database schema scripts

  Creates the staging tables and triggers for the staging database. The staging database contains the same schema and tables as the production database, plus a set of triggers to log changes made in the staging database. The staging database schema scripts add triggers to the database. Changes are logged to the STAGLOG table (a staging table) using database triggers. Whenever you change a database table record in the staging database, the STAGLOG table records this change.

- The stage copy utility

  Allows an administrator to copy data from the production database to the staging database. You can copy the data into site-related tables, merchant-related tables, or individual tables. The stage copy utility should only be used in specific administrative situations, such as setting up a new staging server or recovery from a corrupt staging server database. An administrator should not make day-to-day changes on the production server or routinely use stage copy to copy the data to the staging server.

- The stage propagate utility

  Allows an administrator to propagate changes from the staging database to the production database. The information in the STAGLOG table identifies the records in the staging database that must be inserted, updated, or deleted in the production database. The identified records are then updated in the production database. Processed records are indicated in the STAGLOG table by a 1 in the STGPROCESSED column.

- The stage check utility

  Allows an administrator to check for potential unique index key conflicts between two tables on a staging server and a production server.

Archived

How staging server works

The staging server uses the following four utilities.

Staging copy utility

The stagingcopy utility allows an administrator to copy data from the production database to the production-ready data. You can copy the data into site-related tables, merchant-related tables, or individual tables. After running the stagingcopy utility, the staging database data is aligned with that of a production database, so that any changes made to the staging database afterwards can be logged and later published to the production database.

Staging check utility

The stagingcheck utility command checks for unique index conflicts between the tables in the production-ready data and the production database. A unique index conflict can occur when content data is changed in the production database instead of the production-ready data (for example, if you create a product on the staging server, then create a different product on the production server, and the two different products have the same product ID).

Note: If you do not use the staging check utility, key conflicts found when running the stagingprop utility are indicated in the stagingprop log file.

We recommend running the stagingcheck utility before publishing data to the production server using the stagingprop utility.

Staging prop utility

The stagingprop utility publishes the delta database data from the production-ready data to the production server. It processes each record in the STAGLOG table. The stagingprop utility is the most frequently used of the staging utilities. For example, you may use the stagingprop utility nightly to publish the delta from the production-ready data to the production database.

Fileprop utility

The fileprop utility publishes managed files from the production-ready data to the WebSphere Commerce EAR file on the production server. Managed files are not copied to the database on the production server.

The general approach to create a staging server is described below. You can use it as instructions along with the WebSphere Commerce Installation Guide to create a staging server:

1. Install WebSphere Commerce and its supporting software using the custom installation option of the WebSphere Commerce installation wizard.

2. Prepare the staging server to connect to the production database:
    a. Install a database client suitable for communication with your production database.
    b. Catalog the remote production database so that is accessible from your staging server.
3. Create a new WebSphere Commerce instance:
    a. Start the WebSphere Commerce Instance Creation wizard.
    b. Complete the pages of the wizard.
    c. On the Staging page of the wizard, ensure that you select **Use staging server**. If you do not select this check box, the resulting WebSphere Commerce instance will be a production WebSphere Commerce instance.
    d. Ensure that caching is not enabled in the Cache page. When the instance creation process complete, you will have a staging server instance.
4. Configure the staging database and the production database for use with the staging utilities.
5. If you have any custom tables that you want to enable for staging, create triggers for the custom tables.

Note: The staging server should be run on a separate system or machine partition from your production server.

Performance consideration and practice

The staging utility provides the ability to propagate all changes for the staged data to the production data in one session. But in some cases, customers might have several stores in their production server, if they only want to propagate the changes in a certain store within one propagation session. In a normal scenario of a staging server, it is not working, which results in wasting time and losing concentration. In some extreme situation, if it is necessary to propagate updated staging data to the production server, more staging data always means more performance impact to the production server.

The staging utility in WebSphere Commerce V6.0 introduced a feature called *filtering*, which provides the infrastructure to achieve the filtered propagation according to business needs. Basically, it does not provide specific logic to filter according to specific business goals, like filter by store, by contract, or by promotion. However, it provides the infrastructure on top of which those goals can be implemented.

In this filtering solution for staging server, performance impact can be reduced by introducing a marking for each change. The marking can be any natural number

stored in a new column called STGFILTER in the STAGLOG table. All of the changes related to the same business goal can be assigned the same marking to be recognized by the staging utilities. For example, since the latest propagation process, all of the changes related with store 501 have the same marking in the column STGFILTER.

You can use the following command to propagate all changes with a new parameter named -filter:

```
stagingprop ... -filter 501 ...
```

The new parameter -filter can tell the stagingprop utility that only the changes with a marking 501 will be propagated in this time. For more details about filtered propagation, you can refer to this paper in developerWorks® from this link:

http://www.ibm.com/developerworks/websphere/library/techarticles/0702_jiang/0702_jiang.html

16.4.3 DBClean

Most of the database performance defects or PMRs opened with WebSphere Commerce are a result of poor database maintenance. Over time the database is filled with expired orders information, expired guest user information, user traffic logs, stale coupons, and promotion information. If we do nothing to the database, the result is frequent occurrences of deadlocks, transaction time outs, throughput degradation, and time-consuming response times.

The DBClean, combined together with other well known DB2 maintenance utilities, consist of an entire WebSphere Commerce DB2 database maintenance solution. Refer to Chapter 25, “Database maintenance” on page 563, for more about DB2 maintenance.

Archived

16.5 Conclusions

In this chapter, we talked about database performance consideration, database performance monitoring, and performance tuning. These are not separate from each other. They closely rely on each other. For example, without database monitoring utilities, you cannot know the real status of the database system. A High Availability and high performance database solution should be based on these three aspects, which means that the High Availability solution and the high performance solution are the output of the interactive and continued cycle, as shown in Figure 16-4.

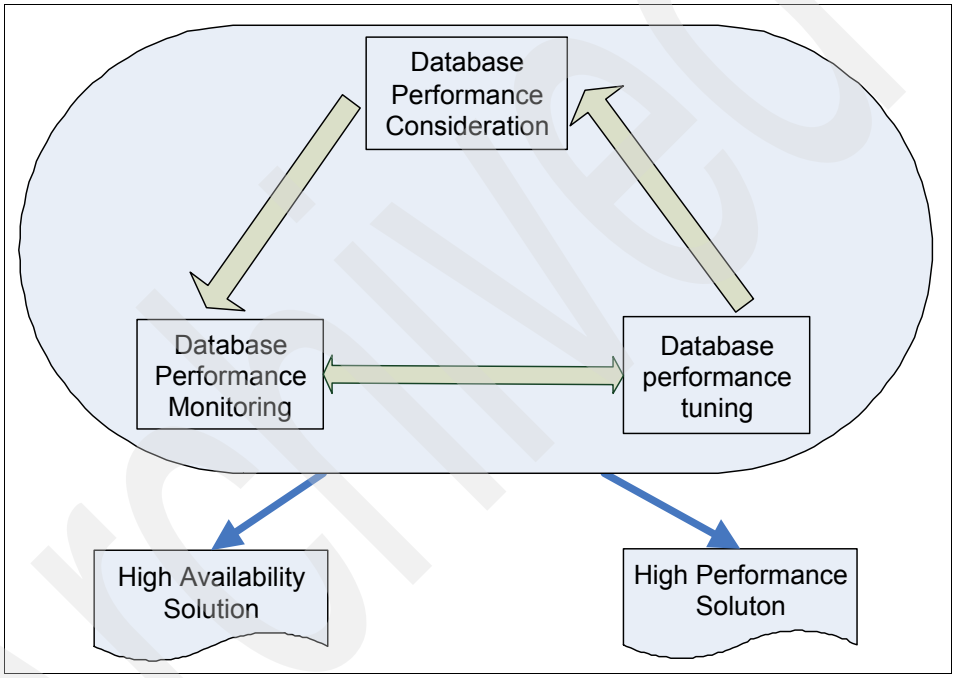


Figure 16-4 DB2 performance improvement cycle

Archived

17

Monitor and tune WebSphere Application Server for WebSphere Commerce

In general, WebSphere Commerce follows the funnelling methodology of tuning most WebSphere applications.

More processing is required on a Web site's front-end servers than on its back-end. Hence, a higher number of requests-serving threads is required at tier one (usually the Web server tier), and is then reduced for each subsequent tier (application server tier and database server tier). This configuration creates a funnel effect in which thread requirements are reduced at each tier.

This chapter introduces some key tuning parameters in WebSphere Application Server, and recommendations from WebSphere Commerce on how to use these parameters. Settings and values configured for some parameters are unique to WebSphere Commerce.

© Copyright IBM Corp. 2008. All rights reserved.

17.1 Web container thread connection pool

Unlike the threads used by the Web server, which requires many of them to handle large quantities of relatively simple requests in a short period of time (that is, serve a .jpg file), the application server's requests are fewer in number, more complex, and take longer to process.

WebSphere Application Server allows users to customize both the minimum and maximum size of the Web container thread pool. The belief that having a high number of threads improves both throughput and the ability to handle a high number of concurrent users is simply not true.

In reality, with too many Web container threads running, each processor (CPU) on the server could potentially be dealing with a large number of requests at any given time. This may result in the operating system having to spend a significant amount of time managing and switching between threads instead of processing the actual requests. Also, each additional thread consumes resources such as memory and database connections in WebSphere Application Server. These resources from extraneous threads can be used more valuably if freed up.

Of course, if the Web container threads value is set too low, the WebSphere Application Server will not have enough workload, so the CPU will be underutilized.

A good rule of thumb is to set the maximum value to at least 10 threads. If you have multiple processors, start with five threads for each CPU when configuring the application server. For example, in a 4-way system, 20 Web container threads should be used. Increase the maximum number of threads only when you believe that CPU utilization could be higher by having additional concurrent workload.

WebSphere Commerce recommends setting the minimum and maximum values of Web container threads to the same value. This avoids thread close and spawn costs. The value should be tuned to throttle the application server to a point where it would continue to perform efficiently.

Tip: Web container thread connection pool's default is 10–50 connections. If the machine has multiple processors, start with 10 threads per processor and increase if the processors seem underutilized.

To set Web container threads, open the WebSphere Application Server administrative console and follow these steps:

1. Expand **Servers**.

2. Click **Application Servers**.
3. Click your_Commerce_Application_Name.
4. Click **Web Container**.
5. Click **Thread Pool**.
6. Set the minimum size and maximum size values.
7. Click **Apply** and **OK**.

17.2 Database connection pool

Unlike the recommendation in WebSphere Application Server's InfoCenter, which states that the database connection pool should be smaller than the Web container thread pool, for WebSphere Commerce configurations, each WebSphere Web container thread needs at least one database connection to match. In addition, when Commerce Scheduler is being used, an additional database connection needs to be reserved to ensure that the WebSphere Commerce scheduler threads function properly.

Tip: The database connection pool's default is 5–65 connections. In general, you should have at least one database connection per Web container thread. If you are using the Commerce Scheduler, one additional connection is necessary.

To set the data source connections, open the WebSphere Application Server administrative console and follow these steps:

1. Select **Resources**.
2. Click **JDBC™ Providers**.
3. Click **Data Sources**.
4. Select your_commerce_data_source.
5. Go to **Connection Pool**.
6. Set the minimum connections and maximum connections for the database pool.

17.3 Prepared statement cache

According to the WebSphere Application Server Information Center, a 10% to 20% performance improvement can be observed with the use of this parameter. To determine the correct setting, the Tivoli Performance Viewer is often used to observe the behavior and minimize cache discards. However, for WebSphere

Commerce, because of the high number of SQL statements executed internally, it is not possible to cache all prepared SQL statements in the application server.

The only reliable way to determine the optimal setting for the prepared statement cache is through actual performance measurement in a controlled environment with repeatable workloads. The recommended initial value for the prepared statement cache is 150, and it should be increased in intervals of 50, until no more performance increase is observed.

Tip: The prepared statement cache's default is 50 statements/connections. There is a 10–20% performance increase when tuning this parameter. Test performance in a repeatable manner in a controlled environment.

To set the prepared statement cache size, open the WebSphere Application Server administrative console and follow these steps:

1. Select **Resources**.
2. Click **JDBC Providers**.
3. Click your_commerce_data_source.
4. Set the value for statement cache size.

17.4 Dynamic caching

Caching dynamic content is one of the most important aspects of improving WebSphere Commerce performance. It improves both response time and throughput while reducing system loads. As a result, the site has better performance, and infrastructure costs can be reduced.

Since the page layout design and the access pattern of each Web site is so different, Application Server's Dynamic Caching configuration file needs to be customized and configured differently in order to maximize the benefit of dynamic caching as much as possible.

All our sample stores come with a default dynamic cache configuration file that consists of some basic rules to cache some obvious pages (that is, product display). These rules are a good starting point to understand the potential of dynamic cache, but are not sufficient to make the store operate in the most efficient way.

The same cache tuning parameters are discussed in Chapter 13, "Caching" on page 265.

17.5 Java Virtual Machine heap management

When a Java Virtual Machine (JVM) is started, it obtains a large area of memory from the native operating system. This area is called the heap, and Java performs its own memory management by allocating areas of the heap as memory is needed by the process.

To ensure stability and achieve optimal performance, it is vital to understand how the heap is utilized and also continuously monitor and tune it.

To manage JVM settings, open the WebSphere Application Server administrative console and follow these steps:

1. Open the WebSphere Application Server administration site.
2. Select **Servers** → **Application servers** → **server1** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.

17.5.1 Heap expansion

Heap expansion occurs after garbage collection while exclusive access of the virtual machine is still held. The active part of the heap is expanded up to the maximum if one of the following is true:

- The garbage collector did not free enough storage to satisfy the allocation request.
- Free space is less than the minimum free space, which you can set by using the -Xminf parameter. The default is 30%.
- More than 13% of the time is being spent in garbage collection.

The amount to expand the heap is calculated as follows:

- If the heap is being expanded because less than -Xminf (default 30%) free space is available, the garbage collector calculates how much the heap needs to expand to get -Xminf free space.

  If this is greater than the maximum expansion amount, which you can set with the -Xmaxe parameter (default of 0, which means no maximum expansion), the calculation is reduced to -Xmaxe.

  If this is less than the minimum expansion amount, which you can set with the -Xmine parameter (default of 1 MB), it is increased to -Xmine.
- If the heap is expanding and the JVM is spending more than 13% for any other reason, the garbage collector calculates how much expansion is needed to expand the heap by 17% free space. This is adjusted as above, depending on -Xmaxe and -Xmine.

Archived

- Finally, the garbage collector ensures that the heap is expanded by at least the allocation request if garbage collection did not free enough storage.

All calculated expansion amounts are rounded up to a 256-byte boundary (512 bytes if concurrent mark is used) on 32-bit architecture, or a 1024 byte boundary on 64-bit architecture.

17.5.2 Heap shrinkage

Heap shrinkage occurs after garbage collection while exclusive access of the virtual machine is still held. Shrinkage does not occur if any of the following are true:

- The garbage collector did not free enough space to satisfy the allocation request.
- The maximum free space, which can be set by the -Xmaxf parameter (default is 60%) is set to 100%.
- The heap has been expanded in the last three garbage collections.
- This is a System.gc(), and the amount of free space at the beginning of the garbage collection was less than -Xminf (default is 30%) of the live part of the heap.
- If none of the above is true and more than -Xmaxf free space exists, the garbage collector must calculate how much to shrink the heap to get it to -Xmaxf free space, without going below the initial (-Xms) value. This figure is rounded down to a 256-byte boundary (512 bytes if concurrent mark is used) on 32-bit architecture, or a 1024 byte boundary on 64-bit architecture.

A compaction occurs before the shrink if all the following are true:

- A compaction was not done on this garbage collection cycle.
- No free chunk is at the end of the heap, or the size of the free chunk that is at the end of the heap is less than 10% of the required shrinkage amount.
- The garbage collector did not shrink and compact on the last garbage collection cycle.

Note that, on initialization, the JVM allocates the entire heap in a single contiguous area of virtual storage. The amount that is allocated is determined by the setting of the -Xmx parameter. No virtual space from the heap is ever freed back to the native operating system. When the heap shrinks, it shrinks inside the original virtual space.

Archived

17.5.3 Tuning the JVM heap size

There are many objects stored in the JVM. A typical WebSphere Commerce JVM heap is shown in Figure 17-1.

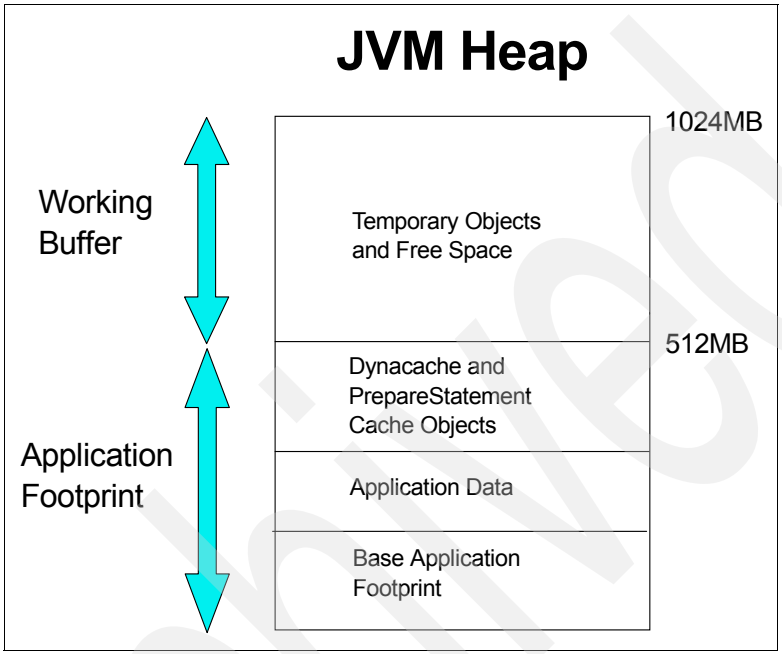


Figure 17-1 JVM heap size and application footprint

WebSphere Commerce recommends that you begin with a minimum heap size of 256 MB and maximum of 1024 MB.

To change the minimum and maximum JVM heap size:

1. Open the WebSphere Application Server administrative console, http://hostname:port/ibm/console, and log in.
2. Expand **Servers** → **Application servers** → **server1** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.
3. Change the min and max heap size values.
4. Click **Apply** and click **Save** at the top of this page.
5. Restart WebSphere Application Server.

Archived

17.5.4 Monitoring JVM memory and garbage collection

Using Verbose Garbage Collection (GC) is one of most efficient ways to understand the memory utilization and the garbage collection behavior within the Java Virtual Machine (JVM). This feature adds detailed statements to the JVM error log file of the application server about the amount of available and in-use memory.

Enable verbose garbage collection

To set up verbose garbage collection, open the WebSphere Application Server administrative console and follow these steps:

1. Open the WebSphere Application Server administration site.
2. Select **Servers** → **Application servers** → **server1** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.
3. Check the box beside Verbose garbage collection.
4. Click **Apply** and click **Save** at the top of this page.
5. Restart WebSphere Application Server.

Here is an example of verbose garbage collection output:

```
<AF[2]: Allocation Failure. need 6920 bytes, 44962 ms since last AF>
<AF[2]: managing allocation failure, action=1 (0/533723648)
(3145728/3145728)>
<GC(2): GC cycle started Mon Oct 13 14:34:35 2003
<GC(2): freed 396872056 bytes, 74% free (400017784/536869376), in 411
ms>
  <GC(2): mark: 342 ms, sweep: 69 ms, compact: 0 ms>
  <GC(2): refs: soft 0 (age >= 32), weak 2, final 6655, phantom 0>
<AF[2]: completed in 414 ms>
```

17.5.5 Heap fragmentation due to pinned and dosed objects

Heap fragmentation is seen when the JVM fails to allocate an object when there should be sufficient space on the heap to do so. This occurs because the JVM cannot find a large enough contiguous space on the heap to allocate the requested object.

Archived

Fragmentation is caused by the interaction between objects on the heap that cannot be moved and the fact that objects need to be allocated into a single contiguous area on the heap. These unmovable objects fall into two types:

- Pinned objects

  These are objects that are referenced from a location not within the Java heap, either by variables in JNI™ native code or by some part of the JVM's internal structure. These fall into two more sub categories:

  – Class objects: These are referenced from within the native component of the classloaders and the JIT.

  – Other pinned objects.

  These types of objects tend to be long lived, so once allocated they will stay in that location on the heap.

- Dosed objects

  These are transiently pinned objects. These objects cannot be moved during the current GC cycle because references to them are held on the stack of a currently executing thread. This means that they are either temporary local variables to the methods in the call stack or they are parameters that have been passed between methods within the call stack.

  To reduce fragmentation caused by pinned and dosed objects, a solution is to group these unmovable objects together into pools. In Java SDK 1.4.2, the GC allocates a kCluster as the first object at the bottom of the heap. A kCluster is an area of storage that is used exclusively for class blocks.

  We recommend setting the kCluster size to be 10% more than the number of classes observed at peak. To find the number of pinned and dosed objects, add the -Dibm.dg.trc.print=st_verify parameter to the Generic JVM Argument field and you will see the number printed out in the GC log.

17.5.6 Heap fragmentation due to large objects

Often heap fragmentation is caused by large Java objects. Prior to launch into production, developers should monitor garbage collection and try to avoid creating large objects because of inefficient code.

Identify Java stack that creates large objects

An environment variable ALLOCATION_THRESHOLD enables a user to identify the Java stack of a thread making an allocation request of larger than the value of this environment variable. This is a very heavy trace. Avoid using it on a production environment. The output is:

```
Allocation request for <allocation request> bytes <java stack>
```

If there is no Java stack, <java stack> becomes No Java Stack.

If you set this option to a value nnn (bytes), whenever an allocation request is made for an object size >= nnn (bytes), the Java stack trace corresponding to the thread requesting the allocation is printed into the standard error log.

Consider the following test case:

```
import java.io.*;
public class largeobj {
static int limit = 20;
static int size1 = 1000000;
static int size2 =2*size1;
public static void main(String []args) throws IOException {
for (int index0=0; true; index0++) {
if (0 == index0 % 100) System.out.println(index0);
ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(bos);
oos.writeObject(String.class);
oos.close();
Object   array1 = null;
for(int i1=0; i1<limit; i1++) {
System.out.println("" + i1);
array1 = new Object[size1] ;
for (int i2=0; i2<limit; i2++) {
array1 = new Object[size2];
}
}
array1=null;
}

}

}
```

If you set the option arbitrarily as:

```
export ALLOCATION_THRESHOLD=5000000
```

You will get messages in the following format during any allocation request larger than or equal to the threshold value:

```
Allocation request for 8000016 bytes
at largeobj.main(largeobj.java:18)
```

To set ALLOCATION_THRESHOLD, navigate in the administrative console to **Application servers → server_name → (Expand Java and Process Management) → Process Definition → Custom Properties**.

Add the following name/value pairs:

Name ALLOCATION_THRESHOLD
Value value

Make sure that you save your changes to the master configuration and restart the application server.

Tune heap to accommodate more large objects

It is not always possible to avoid creating large objects.

IBM Sovereign 1.4.2 SDK SR1 and later (build date of 20050209 and later) supports the configuration of large object area (LOA) to reserve the Java heap for allocating large objects (>=64 KB).

With a new option -Xloratio specified, certain parts of the Java heap are reserved for large objects (>=64 KB). This part of the Java heap will never be used for small object allocation (<64 KB). For example:

```
-Xloratio0.2
```

This command reserves 20% of the active Java heap (not 20% of -Xmx, but 20% of the current size of the Java heap) to the allocation of large objects (>= 64 KB) only. When an allocation request for an object not less than 64 KB arrives, the process first tries to allocate from the remaining 80% of the heap. If it is unable to allocate, it then tries to allocate in the exclusively reserved area for large objects.

-Xmx should be changed to make sure that you do not reduce the size of the small object area, by using the following formula:

```
[New Xmx] = [Current Xmx] / (1 - [loratio])
```

For example, we need at least 1462 MB for -Xmx to use -Xloratio0.3 with current -Xmx1024 MB:

```
1024 MB / (1 - 0.3) = 1462 MB
```

When to use LOA

Generally, thorough analysis of the verbosegc trace is needed to decide whether to configure the -Xloratio. WebSphere Commerce recommends that you start with -Xloraio0.1.

If this is not enough, increase LOA with the following information as a general guideline. If you see fragmentation of the Java heap because of large objects (>= 64 KB) and there is a significant number of allocation failures due to these objects, then you can enable Xloratio with 0.2 or 0.3.

Archived

From the verbosegc, if you see that 0.2 or 0.3 is being used up and the Java heap is still fragmented because of large objects, then consider increasing Xloratio to 0.4 or 0.5.

How to read LOA utilization

Consider this sample GC trace:

```
<AF[2163]: Allocation Failure. need 17112 bytes, 18162 ms since last
AF>
<AF[2163]: managing allocation failure, action=1 (259024/472958976)
(86602816/118239744)>
<GC(2163): mark stack overflow[1849]>
<GC(2163): GC cycle started Fri Mar 14 00:11:06 2008
<GC(2163): heap layout: (180404344/472958976) (108889384/118239744) /0>
<GC(2163): freed 202431888 bytes, 48% free (289293728/591198720), in
342 ms>
<GC(2163): mark: 310 ms, sweep: 32 ms, compact: 0 ms>
<GC(2163): refs: soft 0 (age >= 32), weak 12, final 968, phantom 3>
<AF[2163]: completed in 344 ms>
```

LOA utilization is displayed in the second bracket on the lines of `managing allocation failure` and `heap layout:`.

In the line of
`<AF[2163]: managing allocation failure, action=1 (259024/472958976) (86602816/118239744)>`
The second bracket displays (86602816/118239744), which indicates 86602816 bytes (86MB) out of 118239744 bytes (118MB) is free in the Large Object Area. Subtract 86MB from 118MB, we get 32MB in LOA being utilized.

After the GC cleanup, in the line of:

`<GC(2163): heap layout: (180404344/472958976) (108889384/118239744)`
we see 108889384 bytes (108MB) out of 118239744 bytes (118MB) is free in LOA, which means only 10MB of LOA is used.

In this example, the user's LOA is set to 0.2, which is more than enough. There is no need to increase LOA. The user may even want to lower it back to 0.1 and continue monitoring.

How to configure LOA

Add the -Xloratio to Generic JVM Arguments in the administrative console:

Archived

On WebSphere Application Server V6.0 select **Servers** → **Application Servers** → **server_name** → **Java and Process Management** → **Process definition** → **Java Virtual Machine** → **Generic JVM Arguments**.

17.6 Monitoring

In this section, we describe how WebSphere Application Server should be monitored, the types of data available for monitoring, and the tools and logs available within WebSphere Application Server to display that data.

17.6.1 Performance Monitoring Infrastructure (PMI)

The Performance Monitoring Infrastructure provides a set of APIs to obtain performance data for system resources, WebSphere Application Server queues, and actual customer application code.

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

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

PMI predefined statistic sets

In IBM WebSphere Application Server V6, PMI provides four predefined statistic sets that can be used to enable a set of statistics. These four predefined statistic sets are:

- None
- Basic
- Extended
- All

You can also use the Custom setting to define your own statistic set. Table 17-1 provides the details on these options.

Table 17-1 Predefined statistic sets

Statistic set	Description
None	All statistics are disabled.
Basic	Statistics specified in J2EE 1.4, as well as top statistics like CPU usage and live HTTP sessions, are enabled. This set is enabled by default and provides basic performance data about runtime and application components.
Extended	Basic set plus key statistics from various WebSphere Application Server components like WLM and Dynamic caching are enabled. This set provides detailed performance data about various runtime and application components.
All	All statistics are enabled.
Custom	Enable or disable statistics individually.

If Extended or All statistic set is selected, PMI can have a very large memory footprint on the JVM heap. WebSphere Commerce is very dependent on caching inside the JVM to achieve optimal performance. Leaving PMI statistics collection at a high level can severely impact stability and performance.

Most of what PMI can provide is either not necessary for WebSphere Commerce or can be obtained from logs. For example, runtime JVM statistics can be obtained from a verbose GC log. Dynamic caching statistics can be viewed from a cache monitor.

Currently, we recommend that you use PMI only for monitoring the Web container thread pool, data connection thread pool, and prepared statement cache discard count.

For details on how PMI works, see *WebSphere Application ServerWebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392.

17.6.2 Trace and logging

WebSphere Commerce recommends that you monitor and regularly rotate the following logs in production: SystemOut.log and verbose GC log.

17.7 Tools and reference

Use the tools and references listed in this section when troubleshooting problems with WebSphere Application Server.

IBM Support Assistant

This client software integrates many of the JVM problem determination tools such as:

- ThreadAnalyzer to analyzer java thread dumps
- MDD4J to analyze heap dumps
- PMAT to analyze verbose GC trace

It can be downloaded from:

http://www-306.ibm.com/software/support/isa/

JVM Diagnostic Guide:

http://www-128.ibm.com/developerworks/java/jdk/diagnosis/

WebSphere Application Server MustGathers for debugging JVM Hang/Crash/OOM:

http://www-1.ibm.com/support/docview.wss?uid=swg21145599

GCCollector to analyze GC logs:

http://www.alphaworks.ibm.com/tech/gcdiag

Other heapdump analysis tools:

- Heap Analyzer

  http://www.alphaworks.ibm.com/tech/heapanalyzer
- HeapRoots

  http://www.alphaworks.ibm.com/tech/heaproots

Archived

17.8 Performance fixes

We recommend that you stay up to date on performance fixes that are released on the product support Web sites:

- Recommended fixes and settings for WebSphere Commerce

  http://www-1.ibm.com/support/docview.wss?uid=swg21261296

- WebSphere Commerce support Web site

  http://www-306.ibm.com/software/genservers/commerce/wcbe/support/

- WebSphere Application Server support Web site

  http://www-306.ibm.com/software/webservers/appserv/was/support/

Archived

18

Monitor and tune Web servers

Our topology addresses Web server performance in two ways:

- Separate the Web server tier from the application server tier for security and performance reasons.

  The Web server machines are typically placed in a DMZ, with one network interface connected to the outbound firewall and a second network interface connected to the inbound firewall and the application server tier behind it (see topology overview in Figure 3-1 on page 28). The application server tier is better protected against direct malicious attacks from the Internet. Another reason for separating the Web tier is to permit independent scaling and tuning of Web (I/O intensive) and application (CPU intensive) layers. Unlike the sample topology used in this book, often fewer Web servers than application servers are required to support a given Commerce workload.

  By separating Web servers and application servers, the workload is distributed to separate machines, yielding increased performance compared to a single tier installation.

- Run multiple Web servers for High Availability and performance reasons. Run enough Web servers so that there is enough capacity for managing peak workloads even when one Web server experiences a planned or unplanned outage.

© Copyright IBM Corp. 2008. All rights reserved.

In this chapter, we describe how to monitor and tune the performance of each Web server to support maximum end-to-end throughput and capacity for your WebSphere Commerce application.

We also describe how the IBM HTTP Server Plug-in for WebSphere Application Server can be tuned.

Archived

18.1 Monitor

In addition to OS level monitoring such as CPU and memory utilization of the Web servers, you may use the IBM HTTP Server built-in status page feature. You may also want to analyze the workload management being performed by the IBM HTTP Server Plug-in and configure IBM HTTP Server logging to provide some useful information in the access log for subsequent (offline) analysis.

If you use Load Balancer, your Web servers are also monitored by Load Balancer's advisors.

18.1.1 IBM HTTP Server status page

The IBM HTTP Server server-status page is available on all supported IBM HTTP Server platforms. It shows performance data on a Web page in HTML format.

To activate the server-status page:

1. Open the IBM HTTP Server file httpd.conf in an editor.
2. Remove the comment character (#) from the following lines:

```
#LoadModule status_module modules/mod_status.so

#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>
```

3. Customize .example.com in the sample configuration to match your source workstation reverse DNS entry so that you are allowed to access the page. In our example this must be changed to:

```
Allow from .raleigh.ibm.com
```

4. Save the changes and restart the IBM HTTP Server.
5. Open the URL `http://<webserver>/server-status` in a Web browser, and click **Refresh** to update the status.

Archived

If your browser supports refresh, you can also use the URL `http://<webserver>/server-status?refresh=5` to refresh every 5 seconds. As shown in Figure 18-1, you can see the number of requests currently being processed and the number of idle servers.

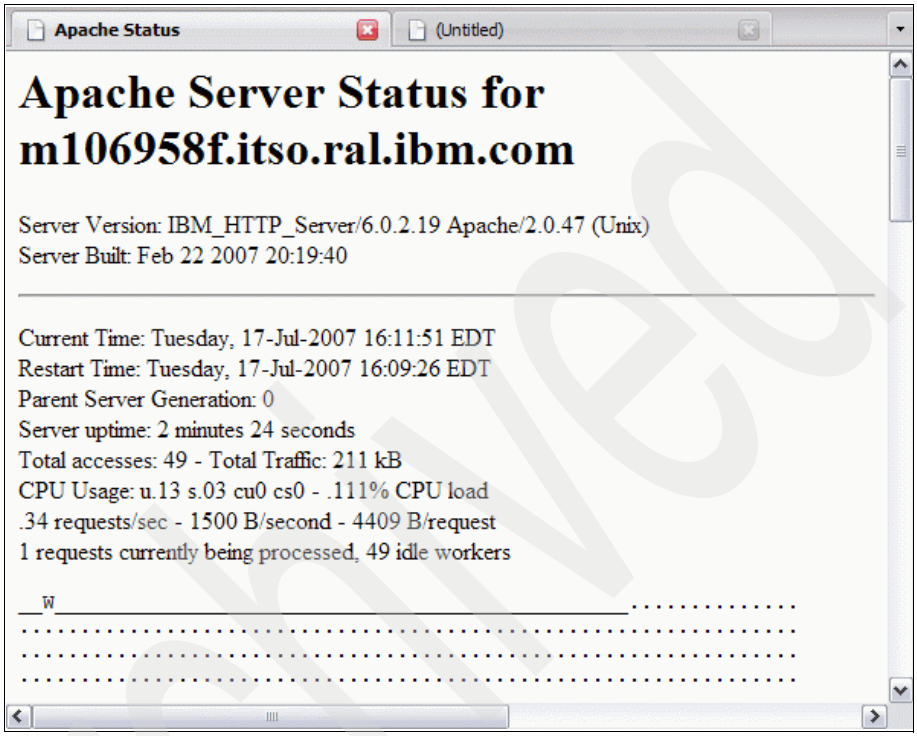


Figure 18-1 IBM HTTP Server status page

18.1.2 Access log

The Web server access log typically contains every request received by the Web server. The access log for the IBM HTTP Server is maintained in the directory given by the CustomLog directive in the httpd.conf file. In “Verify the Web server configuration file” on page 174, we verified the following setting for the WebSphere Commerce instance:

```
CustomLog "WC_Install_Dir/instances/Instance_Name/httplogs/
access_log" common
```

In that line, **`common`** denotes the log format. There are several predefined log formats, and you can define custom log formats using the LogFormat directive in order to provide the metrics that you want to monitor.

Refer to the Apache HTTP Server documentation for instructions on how to define custom log formats:

http://httpd.apache.org/docs/2.0/mod/mod_log_config.html#formats

For instance, you can include the time taken to serve each request. This information can be useful for analyzing DynaCache configurations (see 13.1.1, “Dynamic caching” on page 266).

18.1.3 Monitoring performed by Load Balancer

Load Balancer monitoring can be used to detect Web server outages or problems. Load Balancer monitoring is described in 19.1, “Monitor” on page 418. You can use reports or the graphical port monitor to detect Web server issues.

If Load Balancer assigns a negative weight to a server for a balanced port, the advisor for that port is not able to reach the balanced server. If the weight is negative on all other balanced ports, too, the server could be down.

If the weight for a balanced server is low, this indicates that the server is busier than other balanced servers.

Whenever Load Balancer marks a balanced server down, it looks for the serverDown script in the *LoadBalancer_Install_Dir*/bin directory (see “serverDown” on page 242). If the script exists, it is executed. You can modify this script to send an alert e-mail to an administrator, or any other action that you can script.

18.1.4 IBM HTTP Server Plug-in

IBM HTTP Server Plug-in intercepts IBM HTTP Server request processing and forwards requests having a certain URI pattern to the application servers. For example, for the WebSphere Commerce store front, the default URI pattern is webapp/wcs/stores/*.

There are several ways of checking how the plug-in works. You can enable tracing for the plug-in and use the WebSphere Application Server sample applications to test the workload behavior. Refer to “WebSphere plug-in behavior” in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, for information about how to trace how IBM HTTP Server Plug-in distributes workload among the application servers.

Archived

18.2 Tuning parameters

There are several parameters for IBM HTTP Server that affect Web server performance. As IBM HTTP Server V6.0 is based on Apache HTTP Server V2.0, all parameters documented for the Apache server can be used for the IBM HTTP Server.

First of all, in WebSphere Commerce scenarios, IBM HTTP Server is used to serve static content only. All dynamic processing and HTTP session management take place in the application server tier. Keep this in mind when adjusting parameters for your Web servers.

18.2.1 Operating system settings

The following OS level settings are relevant for Web server performance. Refer to the documentation available for your operating system for instructions on how to change them:

- TCP/IP timeout on sockets

  Do not use a too large a time-out value for TCP connections, as the server may run out of connections when receiving a high load.

- Maximum number of file descriptors

  This should be increased as necessary for serving a potentially high number of static content files like image, Javascript, and CSS files.

18.2.2 httpd.conf settings

Most configuration settings are made in the httpd.conf file.

The default httpd.conf file that comes with IBM HTTP Server contains a number of directives with comments that you can use. The WebSphere Commerce instance creation process uses the default httpd.conf as a template for the httpd.conf file that is used by the instances.

In our scenario, the httpd.conf file resides in the *WC_Install_Dir*/instances/*Instance_Name*/httpconf directory on each Web server.

Keepalive

HTTP is a connectionless protocol, and by default each request opens and closes a socket. We recommend using the **`keepalive`** directive to keep sockets open for a certain time and reuse them for multiple requests. This is useful

because each Web page typically requires multiple requests for embedded objects.

Web pages served by rich Internet applications (RIAs, such as the WebSphere Commerce Web 2.0 starter store, which uses the Dojo Javascript library for AJAX processing) typically make extensive use of Javascript to dynamically and transparently reload content in the background after loading an initial version of a page. If the initially opened socket is kept open for these kinds of pages, it will be reused multiple times.

The directives to use are:

- Keepalive ON

  This allows clients to send multiple requests down a socket connection. As the overhead needed to establish a connection is much larger than that needed to send data packets, it is nearly always necessary to set this to ON in order to achieve high throughput.

- KeepAliveTimeout

  This is the number of seconds to wait for the next request from a client on any single keepalive connection. Three to five seconds should work well here as a starting point. Do not use too large of an amount of time here, as some Web servers do not close connections properly. This setting forces the browsers to discard connections and reopen new ones.

- MaxKeepAliveRequests

  This controls the maximum number of requests that are allowed on a keepalive connection before it is closed. Typically, 100 will suffice. This setting will force even very active users to reopen connections at some point, so the value should not be too small.

Threads

Apache 2.0 is a thread-based Web server. (Although Apache 1.3 is thread-based on the Windows platform, it is a process-based Web server on all UNIX and Linux platforms. This means that it implements the *multi-process, single-thread* process model: for each incoming request, a new child process is created or requested from a pool to handle it.)

However, Apache 2.0, and therefore IBM HTTP Server V6.0 as well, is now a fully thread-based Web server on all platforms. This gives you the following advantages:

- Each request does not requires its *own* httpd process anymore, and less memory is needed.

Archived

- Overall performance improves because in most cases new httpd processes do not need to be created.
- The plug-in load-balancing and failover algorithms work more efficiently in a multi-threaded HTTP server. If one plug-in thread marks an application server unavailable, all other connection threads of the plug-in within the same process will share that knowledge, and will not try to connect to this particular application server again before the RetryInterval has elapsed (see “Retry interval” on page 406).

Note: On the UNIX platform, Apache 2.0 also allows you to configure more than one process to be started. This means that the thread model is then changed to multi-process, multi-thread.

Multi-Processing Modules (MPM) architecture

Apache 2.0 achieves efficient support of different operating systems by implementing a Multi-Processing Modules (MPM) architecture, allowing it, for example, to use native networking features instead of going through an emulation layer in Version 1.3. For detailed information about MPM refer to the Apache HTTP Server documentation found at:

`http://httpd.apache.org/docs/2.0/mpm.html`

MPMs are chosen at compile time and differ for each operating system, which implies that the Windows version uses a different MPM module from the AIX or Linux version. The default MPM for Windows is mpm_winnt, whereas the default module for AIX is mpm_worker. For a complete list of available MPMs, refer to the Apache MPM documentation URL above. To identify which MPM compiled into an Apache 2.0 Web server, run the **`apachectl -l`** command, which prints out the module names. Look for a module name *worker*, or a name starting with the *mpm* prefix (see Example 18-1).

Example 18-1 Listing of compiled in modules for IBM HTTP Server V6 on AIX

```
# ./apachectl -l
Compiled in modules:
  core.c
  worker.c
  http_core.c
  mod_suexec.c
  mod_so.c
```

The modules are:

- mpm_winnt module

  This Multi-Processing Module is the default for the Windows operating systems. It uses a single control process that launches a single child process that in turn creates all the threads to handle requests.

- mpm_worker module

  This Multi-Processing Module implements a hybrid multi-process multi-threaded server. This is the default module for AIX. By using threads to serve requests, it is able to serve a large number of requests with less system resources than a process-based server. Yet it retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.

  The mpm_worker module has proven efficient in many configurations.

MPM parameters

This section gives you configuration tips for the UNIX platforms and provides a good starting point for Web server tuning for WebSphere Commerce. Keep in mind that every system and every site has different requirements, so make sure to adapt these settings to your needs.

- ThreadsPerChild

  Each child process creates a fixed number of threads as specified in the ThreadsPerChild directive. The child creates these threads at startup and never creates more. If using an MPM like mpm_winnt, where there is only one child process, this number should be high enough to handle the entire load of the server. If using an MPM like mpm_worker, where there are multiple child processes, the total number of threads should be high enough to handle the common load on the server.

- ThreadLimit

  This directive sets the maximum configured value for ThreadsPerChild for the lifetime of the Apache process. ThreadsPerChild can be modified during a restart up to the value of this directive.

- MaxRequestsPerChild

  This directive controls after how many requests a child server process is recycled and a new one is launched. This was once added to the Apache server to work around memory leaks that have long been fixed. We therefore recommend setting this value to zero (which is the default).

- MaxClients

  This controls the maximum total number of threads that may be launched. This should equal ServerLimit x ThreadsPerChild.

Archive

It a Web server is used only to route requests to an application server, the maximum number of threads needs to be only slightly greater than the number of threads in the application server. The default configuration for WebSphere Commerce, however, requires the Web server to also serve all the static content for your WebSphere Commerce instance. For high performance, we therefore need more threads, as there are typically multiple static requests for each dynamic request.

Note that the parameter name is misleading, as a client (for example, a browser) typically opens multiple connections so that more than one thread is used per client.

- StartServers

  The number of processes that will initially be launched is set by the StartServers directive.

- MinSpareThreads and MaxSpareThreads

  During operation, the total number of idle threads in all processes will be monitored, and kept within the boundaries specified by MinSpareThreads and MaxSpareThreads.

- ServerLimit

  The maximum number of processes that can be launched is set by the ServerLimit directive.

Attention: Using a ThreadsPerChild value greater than 512 is not recommended on the Linux and Solaris platform. If 1024 threads are needed, the recommended solution is to increase the ServerLimit value to 2 to launch two server processes with 512 threads each. One or few server processes are best, except Solaris. The ThreadsPerChild should not exceed 500 for Linux, but only 25–100 for Solaris.

Threads and Keepalive

Keepalive sockets require threads to block on them during their lifetime. Therefore, you need to carefully balance the number of sockets, the socket time outs, and the number of threads on each Web server. Too many threads might lead to memory thrashing, too few threads could lead to all threads blocking on idle keepalive connections, forcing new connections to queue waiting for a thread to become available.

SSL

In WebSphere Commerce systems, Web servers are typically SSL endpoints. SSL handshakes, encryption, and decryption therefore add significant overhead over IBM HTTP Server's processing.

Archived

As WebSphere Commerce makes extensive use of SSL, we recommend using multiple remote Web servers on dedicated machines. Also, in WebSphere Commerce, once a user has signed in or added anything to his shopping cart, many pages will be requested using HTTPS, so limiting encrypted content is not an option.

WebSphere Commerce instance creation configures IBM HTTP Server to use the module mod_ibm_ssl for SSL processing. The following two directives are relevant for performance:

- SSLV2Timeout and SSLV3Timeout

  The SSL sessions should live slightly longer than the typical user session, to avoid too many change cipher suite handshakes. Keepalive settings have no effect here. The length of the typical user session can be determined by analyzing the Web server access log.

- SSLCipherSpec

  Another tuning knob is the encryption algorithm used by SSL. The client browser and IBM HTTP Server server negotiate the algorithm to be used by attempting to use the strongest algorithm in its list, then walking down the list until the algorithms match. Different encryption routines have widely varying rates of CPU consumption, for example, Triple DES uses vastly more cycles than RC4. To save processing time, you could therefore set the crypto list in the server to favor RC4 with MD5, as shown in Example 18-2.

Example 18-2 Setting encryption algorithm in httpd.conf for SSL-enabled virtual hosts

```
<VirtualHost ...>
...
SSLEnable
...
# RC4 with MD5 for SSL V2
SSLCipherSpec 21
# RC4 with MD5 for SSL V3
SSLCipherSpec 34
...
</VirtualHost>
```

You can find more information about the mod_ibm_ssl module at:

http://www-1.ibm.com/support/docview.wss?rs=177&uid=swg21179559

Compression

The mod_deflate module allows supporting browsers to request that content be compressed before delivery. It provides the *deflate* output filter that lets output

Archived

from the server be compressed before it is sent to the client over the network. Some of the most important benefits of using the mod_deflate module are:

- Saves network bandwidth during data transmission
- Shortens data transmission time
- Generally improves overall performance

Detailed information about configuring and using mod_deflate can be found at:

http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

Access logs

All incoming HTTP requests are logged here. Logging degrades performance because of the (possibly significant) I/O overhead.

To turn logging on or off, search for a line with the text CustomLog in httpd.conf. Comment out this line, then save and close the httpd.conf file. Stop and restart the IBM HTTP Server. By default, logging is enabled, but for better performance we recommend that you disable the access logs, or at least keep the log format not too verbose, in order to avoid to much hard disk I/O. If you decide to do access logging, make sure to turn reverse host name lookup off using the HostnameLookups Off directive, or else each client request results in at least one lookup request to DNS.

Expiration of static content

The module mod_expires can be used to enable expiration times for static content. This is typically done for static content that is likely to be updated. For example, a downloadable PDF file with terms and conditions for using your online store is updated every time the terms and conditions change.

To prevent the client browser from loading the file from its cache after an update, you can set expiration times at file, directory, virtual host, or server level. You can also set default expiration times for file types. The expiration time is passed back as an HTTP header with the response to a request for a static file, causing the client browser to reload the file from the server after the expiration time has passed even if the file is still being cached.

Redbooks

You should not use short expiration times on static files that are usually not changing, for example, product images. Only use expiration time on files that are going to be updated. Example 18-3 shows how to load and activate the module, how to configure images in the nocache directory for no caching (using an expiration time of 1 second after client access time), and how to configure PDF files in the same directory to expire one week after the file modification time.

Example 18-3 Static content expiration

```
LoadModule expires_module     libexec/mod_expires.so
...
ExpiresActive On
...
<Directory
/usr/IBM/WebSphere/AppServer/profiles/demo/installedApps/WC_demo_cell/W
C_demo.ear/Stores.war/ConsumerDirect/images/nocache>
   <FilesMatch "\.(gif|jpg|png)">
      order allow,deny
      allow from all
# expire one second after client access time
      ExpiresDefault A1
   </FilesMatch>
   <FilesMatch "\.pdf">
      order allow,deny
      allow from all
# expire one week after modification time
      ExpiresDefault M604800
   </FilesMatch>
</Directory>
```

More information about using mod_expires can be found at:

http://httpd.apache.org/docs/2.0/mod/mod_expires.html

18.2.3 IBM HTTP Server Plug-in

In this section we describe settings affecting the performance of the IBM HTTP Server Plug-in. The plug-in is introduced in 6.3, “Web container clustering and failover (Web server plugin)” on page 66. Performance tuning options are:

- You can change the workload distribution policy in the configuration file. See “Workload management policies” on page 404.
- You can change the retry interval for connecting to a cluster member marked as down. See “Retry interval” on page 406.

Archived

- You can change connection time-out settings to bypass the operating system time out. See “Connection timeout” on page 408.
- You can divide the servers into a primary server list and a backup server list. This is a feature available since WebSphere V5, also called two-level failover support. See “Primary and backup servers” on page 410 for information.
- You can change the maximum number of connections that will be allowed to a server from a given plug-in. If this attribute is set to either zero or -1, there is no limit to the number of pending connections to the application servers. The default value is -1. See “Maximum number of connections” on page 412.
- You can change the refresh interval for the reloading of the plug-in configuration file. See “Refresh interval” on page 415.

Workload management policies

Since WebSphere Application Server V5, the plug-in has two options for the load distributing algorithm:

- Round-robin with weighting
- Random

Note: The weighting for the round-robin approach can be turned off by giving all application servers in a cluster equal weights.

Archive

The default value is Round Robin. It can be changed by selecting **Servers** → **Web servers** → ***WebServer_Name*** → **Plug-in properties** → **Request Routing**, as shown in Figure 18-2.

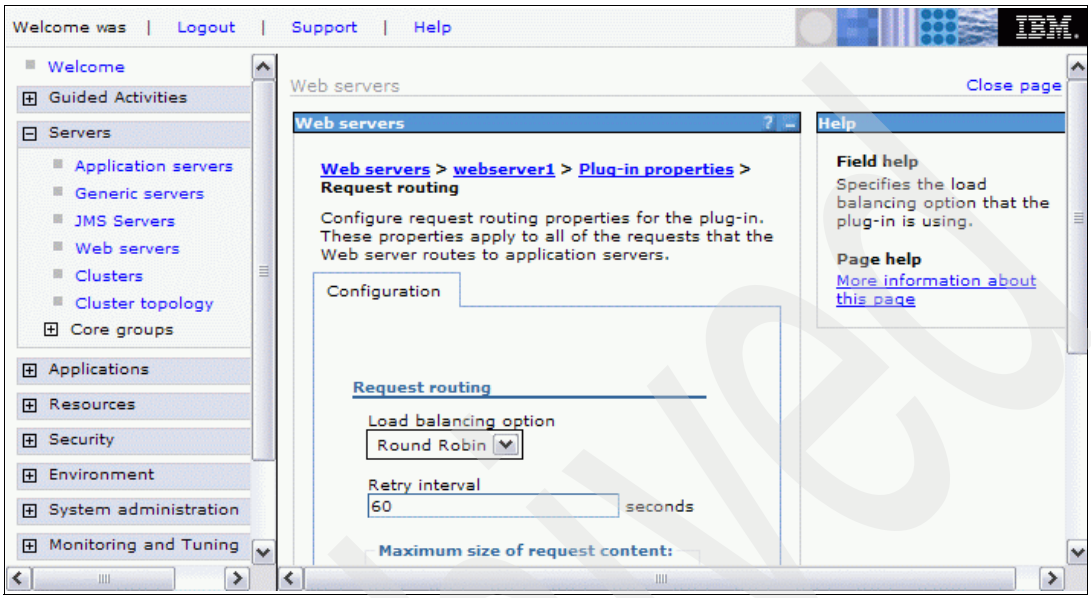


Figure 18-2 Plug-in load balancing options

There is also a feature in WebSphere Application Server V6 for the plug-in called ClusterAddress that can be used to suppress load balancing. However, this is normally not desirable. As in the sample topology, a *low-level* Load Balancer (software-based or hardware-based) normally precedes the Web servers and not the application servers. See “Cluster Address” on page 271, in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, for more details (this setting cannot be made through the administrative console).

Weighted round robin

When using this algorithm, the plug-in selects a cluster member at random from which to start. The first successful browser request is routed to this cluster member and then its weight is decremented by 1. New browser requests are then sent round robin to the other application servers and subsequently the weight for each application server is decremented by 1. The spreading of the load is equal between application servers until one application server reaches a weight of 0. From then on, only application servers with a weight higher than 0 will have requests routed to them. The only exception to this pattern is when a cluster member is added or restarted or when session affinity comes into play.

Random

Requests are passed to cluster members randomly. Weights are not taken into account as with round robin. The only time the application servers are not chosen randomly is when there are requests with sessions associated with them. When the random setting is used, cluster member selection does not take into account where the last request was handled. This means that a new request could be handled by the same cluster member as the last request.

Retry interval

There is a setting in the plug-in configuration file that allows you to specify how long to wait before retrying a server that is marked as down. This is useful in avoiding unnecessary attempts when you know that server is unavailable. The default is 60 seconds.

This setting is specified in the configuration of each Web server, as shown in Figure 18-2 on page 405, on the Retry interval field. This default setting means that if a cluster member was marked as down, the plug-in would not retry it for 60 seconds. To change this value, go to **Servers** → **Web servers** → ***WebServer_Name*** → **Plug-in properties** → **Request Routing**.

Finding the correct setting

There is no way to recommend one specific value. The value chosen depends on your environment, for example, on the number of cluster members in your configuration.

Setting the retry interval to a small value allows an application server that becomes available to quickly begin serving requests. However, too small of a value can cause serious performance degradation, or even cause your plug-in to appear to stop serving requests, particularly in a machine outage situation. For example, if you have numerous cluster members, and one cluster member being unavailable does not affect the performance of your application, then you can safely set the value to a very high number.

Alternatively, if your optimum load has been calculated assuming all cluster members to be available or if you do not have very many, then you will want your cluster members to be retried more often to maintain the load.

Also, take into consideration the time it takes to restart your server. If a server takes a long time to boot up and load applications, then you will need a longer retry interval.

Another factor to consider for finding the correct retry interval for your environment is the operating system TCP/IP timeout value. To explain the relationship between these two values, let us look at an example configuration with two machines, which we call A and B. Each of these machines is running

two clustered application servers (CM1 and CM2 on A, CM3 and CM4 on B). The HTTP server and plug-in are running on AIX with a TCP timeout of 75 seconds, the retry interval is set to 60 seconds, and the routing algorithm is weighted round robin. If machine A fails, either as expected or unexpectedly, the following process occurs when a request comes in to the plug-in:

1. The plug-in accepts the request from the HTTP server and determines the server cluster.
2. The plug-in determines that the request should be routed to cluster member CM1 on system A.
3. The plug-in attempts to connect to CM1 on machine A. Because the physical machine is down, the plug-in waits 75 seconds for the operating system TCP/IP timeout interval before determining that CM1 is unavailable.
4. The plug-in attempts to route the same request to the next cluster member in its routing algorithm, CM2 on machine A. Because machine A is still down, the plug-in must again wait 75 seconds for the operating system TCP/IP timeout interval before determining that CM2 is also unavailable.
5. The plug-in attempts to route the same request to the next cluster member in its routing algorithm, CM3 on system B. This application server successfully returns a response to the client, about 150 seconds after the request was first submitted.
6. While the plug-in was waiting for the response from CM2 on system A, the 60-second retry interval for CM1 on system A expired, and the cluster member is added back into the routing algorithm. A new request is routed to this cluster member, which is still unavailable, and this lengthy waiting process will begin again.

There are two options to avoid this problem:

- The recommended approach is to configure your application servers to use a non-blocking connection. This eliminates the impact of the operating system TCP/IP timeout. See “Connection timeout” on page 408 for information.
- An alternative is to set the retry interval to a more conservative value than the default of 60 seconds, related to the number of cluster members in your configuration. A good starting point is 10 seconds + (number of cluster members * TCP timeout). This ensures that the plug-in does not get stuck in a situation of constantly trying to route requests to the failed members. In the scenario described before, this setting would cause the two cluster members on system B to exclusively service requests for 235 seconds before the cluster members on system A are retried, resulting in another 150-second wait.

Connection timeout

When a cluster member exists on a machine that is removed from the network (because its network cable is unplugged or it has been powered off, for example), the plug-in, by default, cannot determine the cluster member's status until the operating system TCP/IP timeout expires. Only then will the plug-in be able to forward the request to another available cluster member.

It is not possible to change the operating system timeout value without unpredictable side effects. For instance, it might make sense to change this value to a low setting so that the plug-in can fail over quickly.

However, the timeout value on some of the operating systems is not only used for outgoing traffic (from Web server to application server) but also for incoming traffic. This means that any changes to this value will also change the time it takes for clients to connect to your Web server. If clients are using dial-up or slow connections, and you set this value too low, they will not be able to connect.

To overcome this problem, WebSphere Application Server V6 offers an option within the plug-in configuration file that allows you to bypass the operating system timeout.

Archived

It is possible to change the connection timeout between the plug-in and each application server, which makes the plug-in use a non-blocking connect, as shown in Figure 18-3. To configure this setting, go to **Servers** → **Application servers** → ***AppServer_Name*** → **Web server plug-in properties**.

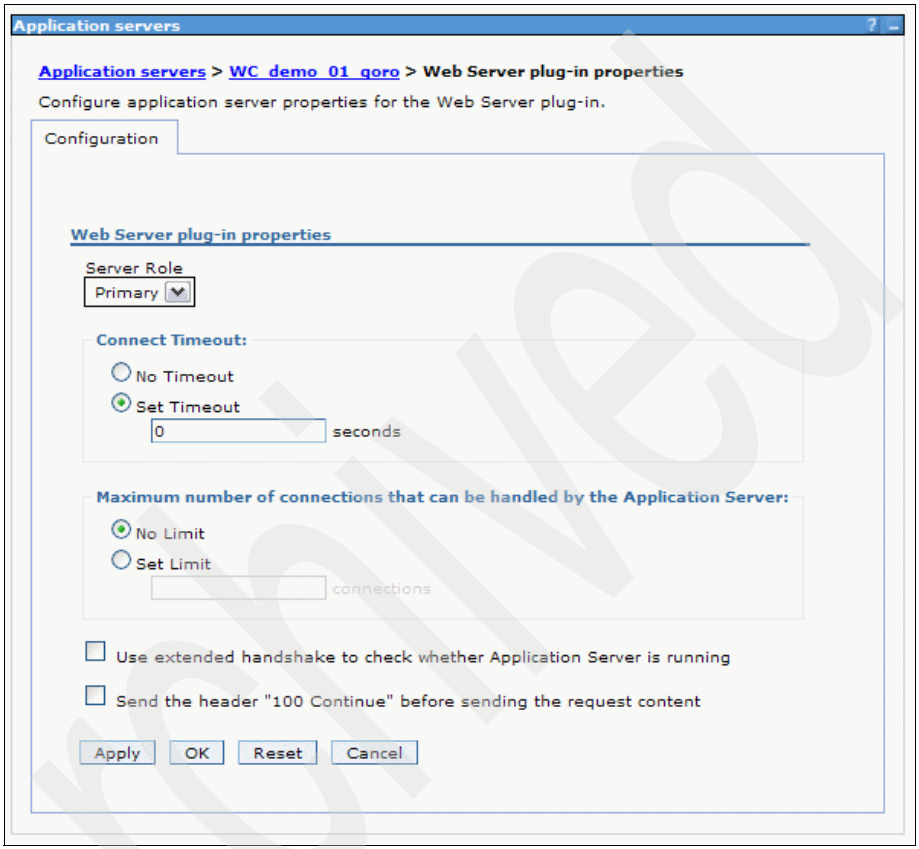


Figure 18-3 Plug-in connection timeout settings

Setting the connect timeout attribute for a server to a value of zero (default) is equal to selecting the No Timeout option, that is, the plug-in performs a blocking connect and waits until the operating system times out. Set this attribute to an integer value greater than zero to determine how long the plug-in should wait for a response when attempting to connect to a server. A setting of 10 means that the plug-in waits for 10 seconds to time out.

Finding the correct setting

To determine what setting should be used, you need to take into consideration how fast your network and servers are. Complete some testing to see how fast your network is, and take into account peak network traffic and peak server

Archived

usage. If the server cannot respond before the connection timeout, the plug-in will mark it as down.

Since this setting is determined on the each application server, you can set it for each individual cluster member. For instance, you have a system with four cluster members, two of which are on a remote node. The remote node is on another subnet and it sometimes takes longer for the network traffic to reach it. You might want to set up your cluster in this case with different connection timeout values.

If a non-blocking connect is used, you will see a slightly different trace output. Example 18-4 shows what you see in the plug-in trace if a non-blocking connect is successful.

Example 18-4 Plug-in trace when ConnectTimeout is set

```
...
TRACE: ws_common: websphereGetStream: Have a connect timeout of 10;
Setting socket to not block for the connect
TRACE: errno 55
TRACE: RET 1
TRACE: READ SET 0
TRACE: WRITE SET 32
TRACE: EXCEPT SET 0
TRACE: ws_common: websphereGetStream: Reseting socket to block
...
```

Primary and backup servers

Starting with V5, WebSphere Application Server implements a feature called *primary* and *backup* servers. When the plugin-cfg.xml is generated, all servers are initially listed under the PrimaryServers tag, which is an ordered list of servers to which the plug-in can send requests.

There is also an optional tag called BackupServers. This is an ordered list of servers to which requests should only be sent if all servers specified in the primary servers list are unavailable.

Within the primary servers, the plug-in routes traffic according to server weight or session affinity. The Web server plug-in does not route requests to any server in the backup server list as long as there are application servers available from the primary server list. When all servers in the primary server list are unavailable, the plug-in will then route traffic to the first available server in the backup server list. If the first server in the backup server list is not available, the request is routed to the next server in the backup server list until no servers are left in the list or until a request is successfully sent and a response received from an application

server. Weighted round-robin routing is not performed for the servers in the backup server list.

Important: In WebSphere V6, the primary and backup server lists are only used when the new partition ID logic is not used. In other words, when partition ID comes into play, then primary/backup server logic does not apply any longer. To learn about partition ID, refer to *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392.

You can change a cluster member role (primary or backup) using the administrative console. Select **Servers** → **Application servers** → ***<AppServer_Name>*** → **Web server plug-in properties** and select the appropriate value from the Server Role pull-down field (see Figure 18-3 on page 409).

All application server details in the plugin-cfg.xml file are listed under the ServerCluster tag. This includes the PrimaryServers and BackupServers tags, as illustrated in Example 18-5.

Example 18-5 ServerCluster element depicting primary and backup servers

```
...
<ServerCluster>
...
</Server>
   <PrimaryServers>
      <Server Name="wasna01_wasmember01"/>
      <Server Name="wasna02_wasmember03"/>
   </PrimaryServers>
   <BackupServers>
      <Server Name="wasna01_wasmember02"/>
      <Server Name="wasna02_wasmember04"/>
   </BackupServers>
</ServerCluster>
...
```

The backup server list is only used when all primary servers are down.

Archive

Maximum number of connections

All requests to the application servers flow through the HTTP Server plug-in. The application server selection logic in the plug-in has been enhanced so that it takes into account the number of pending connections to the application server. The maximum number of connections attribute is used to specify the maximum number of pending connections to an application server that can be flowing through a Web server process at any point in time.

Each application server can have a maximum number of pending connections coming from Web server plug-ins, as shown in Figure 18-3 on page 409. To change this setting, go to **Servers** → **Application servers** → ***AppServer_Name*** → **Web server plug-in properties**.

The default setting is No Limit, which is the same as though the value is set to -1 or zero. The attribute can be set to any arbitrary value. For example, let the two application servers be fronted by two nodes running IBM HTTP Server. If the MaxConnections attribute is set to 10, then each application server could potentially get up to 20 pending connections.

If the number of pending connections reaches the maximum limit of the application server, then it is not selected to handle the current request. If no other application server is available to serve the request, HTTP response code 503 (service unavailable) is returned to the user.

To monitor the behavior of the plug-in when a cluster member has too many requests, use a load testing tool (such as ApacheBench or JMeter), the plug-in log, the HTTP servers' access log, and Tivoli Performance Viewer.

For our test we have configured wasmember05 to have a maximum connection setting of 3 and wasmember06 of 4. We have one HTTP server fronting the application servers. Running a load test with 30 concurrent users against this cluster eventually results in both members having reached the maximum connections value. At this point, the user gets Error 503. See Example 18-6 (plug-in log) and Example 18-7 on page 413 (HTTP server access log).

Example 18-6 Plug-in log file errors when no server can serve requests (MaxConnections)

```
...
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of wasna01_wasmember05,
ignoreWeights 0, markedDown 0, retryNow 0, wlbAllows 17
reachedMaxConnectionsLimit 1
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - WARNING:
ws_server_group: serverGroupCheckServerStatus: Server
```

```
wasna01_wasmember05 has reached maximmum connections and is not
selected
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of wasna02_wasmember06,
ignoreWeights 0, markedDown 0, retryNow 0, wlbAllows 14
reachedMaxConnectionsLimit 1
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - WARNING:
ws_server_group: serverGroupCheckServerStatus: Server
wasna02_wasmember06 has reached maximmum connections and is not
selected
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - ERROR: ws_server_group:
serverGroupNextRoundRobinServer: Failed to find a server; all could be
down or have reached the maximimum connections limit
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - WARNING: ws_common:
websphereFindServer: Application servers have reached maximum
connection limit
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - ERROR: ws_common:
websphereWriteRequestReadResponse: Failed to find a server
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - ERROR: ESI: getResponse:
failed to get response: rc = 8
[Wed May 11 07:26:23 2005] 00007ab5 aaff9bb0 - ERROR: ws_common:
websphereHandleRequest: Failed to handle request
...
```

When the plug-in detects that there are no application servers available to satisfy the request, HTTP response code 503 (service unavailable) is returned. This response code appears in the Web server access log, as shown in Example 18-7.

Example 18-7 HTTP Server access log example

```
[11/May/2005:07:26:23 -0500] "GET /wlm/BeenThere HTTP/1.1" 503 431
[11/May/2005:07:26:23 -0500] "GET /wlm/BeenThere HTTP/1.1" 503 431
```

Further down in the plug-in log you can see that eventually both servers respond to requests again when they have reduced their backlog. This is shown in Example 18-8.

Example 18-8 Maximum connections - application servers pick up work again

```
...
#### wasmember05 worked off the pending requests and is "back in
business", wasmember06 is still in “reachedMaxConnectionsLimit” status:
```

```
[Wed May 11 07:26:23 2005] 00007ab9 b640bbb0 - STATS: ws_server:
serverSetFailoverStatus: Server wasna01_wasmember05 :
pendingConnections 0 failedConnections 0 affinityConnections 0
totalConnections 1.
[Wed May 11 07:26:23 2005] 00007ab9 aa5f8bb0 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of wasna01_wasmember05,
ignoreWeights 0, markedDown 0, retryNow 0, wlbAllows 17
reachedMaxConnectionsLimit 0
[Wed May 11 07:26:23 2005] 00007ab9 aa5f8bb0 - STATS: ws_server:
serverSetFailoverStatus: Server wasna01_wasmember05 :
pendingConnections 0 failedConnections 0 affinityConnections 0
totalConnections 2.
[Wed May 11 07:26:23 2005] 00007ab9 a9bf7bb0 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of wasna02_wasmember06,
ignoreWeights 0, markedDown 0, retryNow 0, wlbAllows 14
reachedMaxConnectionsLimit 1
[Wed May 11 07:26:23 2005] 00007ab9 a9bf7bb0 - WARNING:
ws_server_group: serverGroupCheckServerStatus: Server
wasna02_wasmember06 has reached maximmum connections and is not
selected
...
#### wasmember06 is working off the the requests. Once it reduced the
number of pending connections to below 4 (which is the maximum) it can
then also serve requests again. Both servers are handling user requests
now:
...
[Wed May 11 07:26:23 2005] 00007ab5 b4608bb0 - STATS: ws_server:
serverSetFailoverStatus: Server wasna02_wasmember06 :
pendingConnections 3 failedConnections 0 affinityConnections 0
totalConnections 4.
[Wed May 11 07:26:23 2005] 00007ab9 b5009bb0 - STATS: ws_server:
serverSetFailoverStatus: Server wasna02_wasmember06 :
pendingConnections 2 failedConnections 0 affinityConnections 0
totalConnections 4.
[Wed May 11 07:26:23 2005] 00007ab9 b3c07bb0 - STATS: ws_server:
serverSetFailoverStatus: Server wasna02_wasmember06 :
pendingConnections 1 failedConnections 0 affinityConnections 0
totalConnections 4.
[Wed May 11 07:26:23 2005] 00007ab9 b5a0abb0 - STATS: ws_server:
serverSetFailoverStatus: Server wasna02_wasmember06 :
pendingConnections 0 failedConnections 0 affinityConnections 0
totalConnections 4.
[Wed May 11 07:26:24 2005] 00007ab9 b3206bb0 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of wasna02_wasmember06,
```

```
ignoreWeights 0, markedDown 0, retryNow 0, wlbAllows 14
reachedMaxConnectionsLimit 0
...
```

This feature helps you to better load balance the application servers fronted by the plug-in. If application servers are overloaded, the plug-in skips these application servers automatically and tries the next available application server.

However, a better solution is to have an environment that can handle the load that you are expecting and to have it configured correctly. This includes setting weights that correspond to the system capabilities, having the correct balance of cluster members and Web servers, and setting up the queues for requests and connections.

Refresh interval

The refresh interval defines how often the plug-in will check to see whether the plug-in configuration file (plugin-cfg.xml) has changed. In production, preferably use a higher value than the default 60 seconds because updates to the configuration will not occur so often. If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in configuration file successfully reloads. If you are not seeing the changes that you made to your plug-in configuration, check the plug-in log file for indications of the problem.

To change the refresh interval and specify the log file location and the log level, go to **Servers** → **Web servers** → ***WebServer_Name*** → **Plug-in properties** and change the Refresh configuration interval, the Logfile name, and the Log level fields. See Figure 10-3 on page 177.

Archive

Archived

19

Monitor and tune Load Balancer

We use IBM WebSphere Edge Components Load Balancer to improve performance of our Web server tier and at the same time to make the Web server tier highly available. In this chapter we describe how to monitor and tune Load Balancer.

We also discuss server affinity as a means for increasing performance and facilitating Web server log analysis.

© Copyright IBM Corp. 2008. All rights reserved.

19.1 Monitor

In addition to OS level monitoring such as CPU and memory utilization of the active Load Balancer machine itself, you may use reports provided by Load Balancer's Dispatcher component to monitor load balancing behavior. You can see whether the Web servers that make up the cluster are active and sending responses to the advisors. You can also see whether the traffic is being balanced using the server monitor on the GUI.

Furthermore, you can use the binary logging feature, which allows server information to be stored in binary files. These files can then be processed to analyze the server information that has been gathered over time.

19.1.1 Reports

Example 19-1 shows the output of the **dscontrol manager report** command. The first table lists the servers being load balanced and their status. The second table lists the servers by port, weight, number of active and new connections, and load values.

The last table shows the advisors that were started, the port, and the timeout value attributed to it.

Example 19-1 Manager report

```
------------------------------------------------------------------
|       SERVER                    |    IP ADDRESS   |   STATUS   |
------------------------------------------------------------------
|           srvb504.torolab.ibm.com |    9.26.127.157 |     ACTIVE |
|         m106958f.itso.ral.ibm.com |     9.42.171.83 |     ACTIVE |
|           srvb501.torolab.ibm.com |    9.26.126.120 |     ACTIVE |
------------------------------------------------------------------
-----------------------------
|   MANAGER REPORT LEGEND   |
-----------------------------
| ACTV | Active Connections |
| NEWC | New Connections    |
| SYS  | System Metric      |
| NOW  | Current Weight     |
| NEW  | New Weight         |
| WT   | Weight             |
| CONN | Connections        |
-----------------------------
------------------------------------------------------------------
```

Archived

```
| nat1.torolab.ibm.com                                                  |
| 9.26.52.154        | WEIGHT |  ACTV  |  NEWC  |  PORT  |   SYS  |
|     PORT:    80    |NOW  NEW|   49%  |   50%  |    1%  |    0%  |
-------------------------------------------------------------------------
| srvb504.torolab.ibm.com                                               |
|                    | 10  10 |      0 |      0 |    109 |      0 |
| m106958f.itso.ral.ibm.com                                             |
|                    |  9  9  |      0 |      0 |    211 |      0 |
| srvb501.torolab.ibm.com                                               |
|                    | 10  10 |      0 |      0 |     13 |      0 |
-------------------------------------------------------------------------
-------------------------------------------------------------------------
| nat1.torolab.ibm.com                                                  |
|     9.26.52.154    | WEIGHT |  ACTV  |  NEWC  |  PORT  |   SYS  |
|     PORT:   443    |NOW  NEW|   49%  |   50%  |    1%  |    0%  |
-------------------------------------------------------------------------
| srvb504.torolab.ibm.com                                               |
|                    |  4  4  |      4 |      6 |     56 |      0 |
| m106958f.itso.ral.ibm.com                                             |
|                    | 13  13 |      2 |      2 |    241 |      0 |
| srvb501.torolab.ibm.com                                               |
|                    | 11  11 |      3 |      0 |     76 |      0 |
-------------------------------------------------------------------------
-------------------------------------------------
|    ADVISOR    |    CLUSTER:PORT     |  TIMEOUT  |
-------------------------------------------------
|          http |                  80 | unlimited |
|           ssl |                 443 | unlimited |
-------------------------------------------------
```

You can check whether packets are being forwarded to the cluster by issuing the **dscontrol executor report** command, which produces a report of the packet traffic on the Executor component of Dispatcher, as shown in Example 19-2.

Example 19-2 Executor report

```
Executor Report:
---------------
Version level ................................ 06.00.02.58 -
20070126-105431 [wsbld47]
Total packets received since starting ......... 714,873
Packets sent to nonforwarding address ......... 464,998
Packets processed locally on this machine ..... 0
Packets sent to collocated server ............. 0
Packets forwarded to any cluster .............. 91,223
```

Archived

```
Packets not addressed to active cluster/port .. 74
KBytes transferred per second ................. 62
Connections per second ........................ 2
Packets discarded - headers too short ......... 0
Packets discarded - no port or servers ........ 0
Packets discarded - network adapter failure ... 0
Packets with forwarding errors................. 0
```

19.1.2 Graphical server monitor

We can use the server monitor on the GUI to graphically view the load being distributed among the servers. It is available per port and per server. If you right-click a desired port or a specific server, you can select the Monitor option.

The Monitor tool provides the same information that you can view in the Manager report, but it dynamically updates the data and shows it in a chart.

If you choose to monitor a port, there will be one bar or line for each server configured on that port. If you choose to monitor a specific server then there will be only one bar or line for the chosen server. By default, the chart presents the weight of the servers.

Archived

Figure 19-1 shows a snapshot of the server weights for our NAT forwarding scenario, with the Load Balancer and two Web servers, srvb501 and srvb504, on the same subnet Toronto, and one Web server, m106958f, in Raleigh, while running a Borland SilkPerformer test script simulating five concurrent clients, making an HTTPS request every five seconds. You can also select Port load (the load value provided by the advisor), System load (the load value provided by the metric server), Active connections, and New connections.

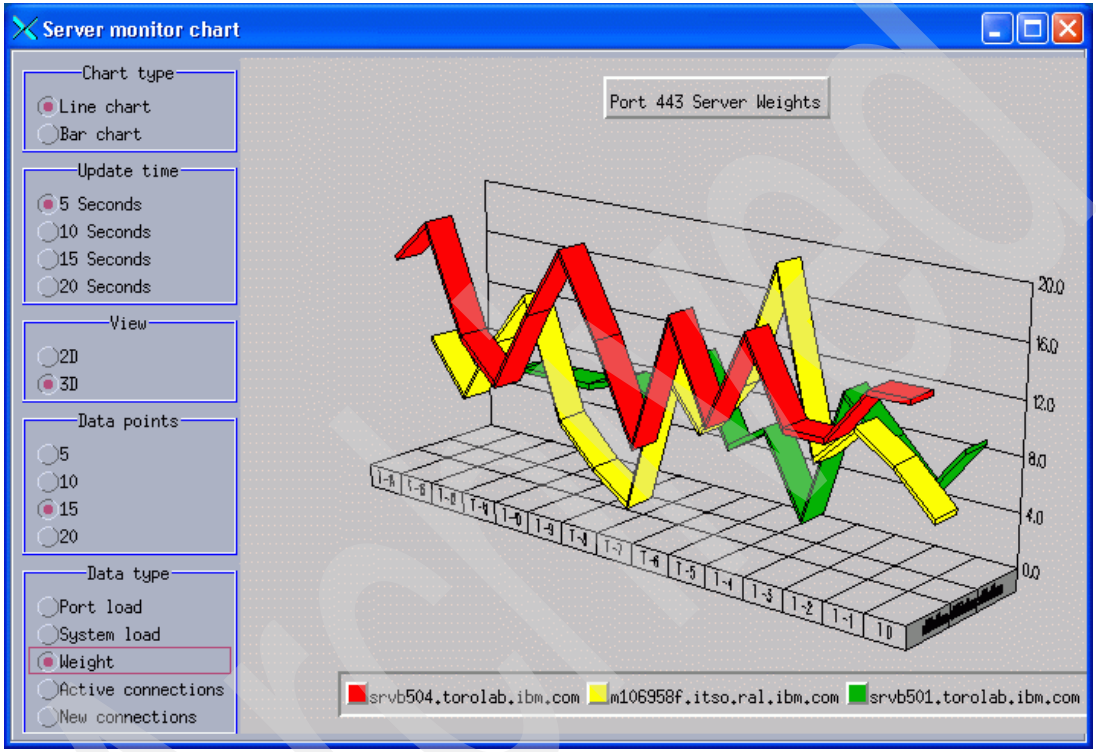


Figure 19-1 Monitor tool

In order to monitor the behavior of the cluster, you can try repeatedly selecting a clustered page using the browser, or you can use an HTTP benchmarking tool to send requests to the cluster. In order to monitor the behavior of the cluster, you can try repeatedly selecting a clustered page using the browser, or you can use an HTTP benchmarking tool to send requests to the cluster. In our lab, we used Borland SilkPerformer.

Using the Monitor tool, we can see the distribution of the load by selecting **New connections** in the Data Type box. This is shown in Figure 19-2 for a case in which one of the servers (or the network between Toronto and Raleigh) is experiencing problems, not accepting any new connections.

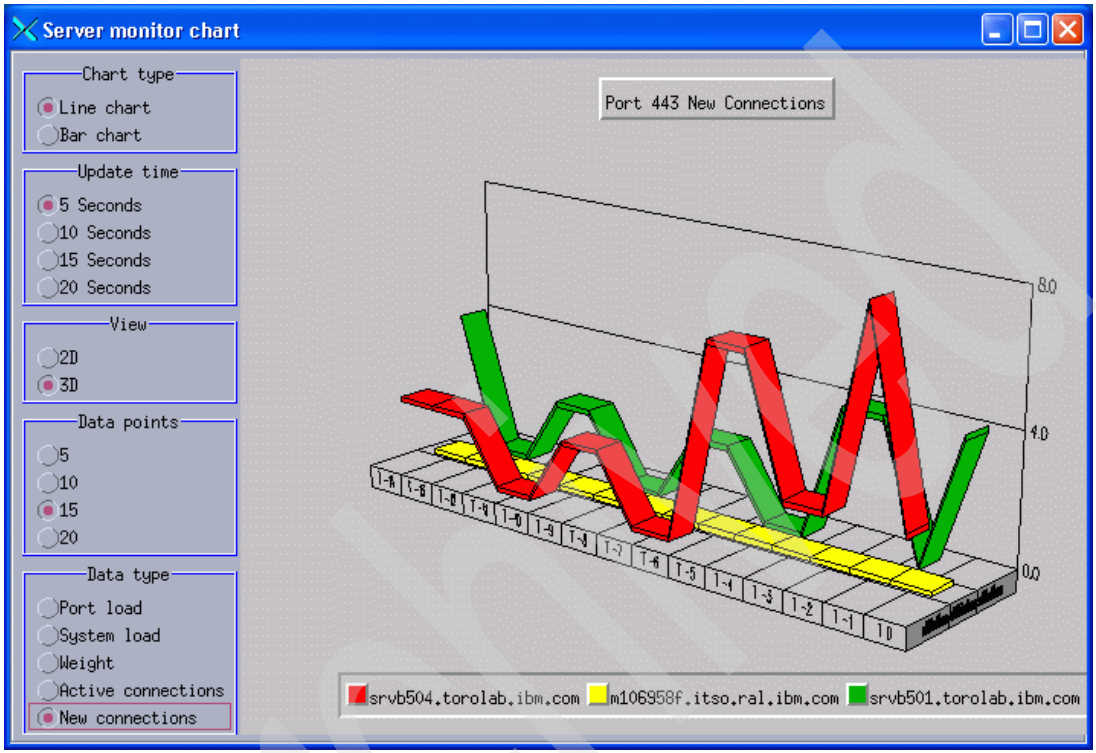


Figure 19-2 Load distributions (new connections) with one server down

Looking at the current number of active connections by selecting **Active connections**, as shown in Figure 19-3, we can see that the active connections of m106958f (the Raleigh machine) have timed out.

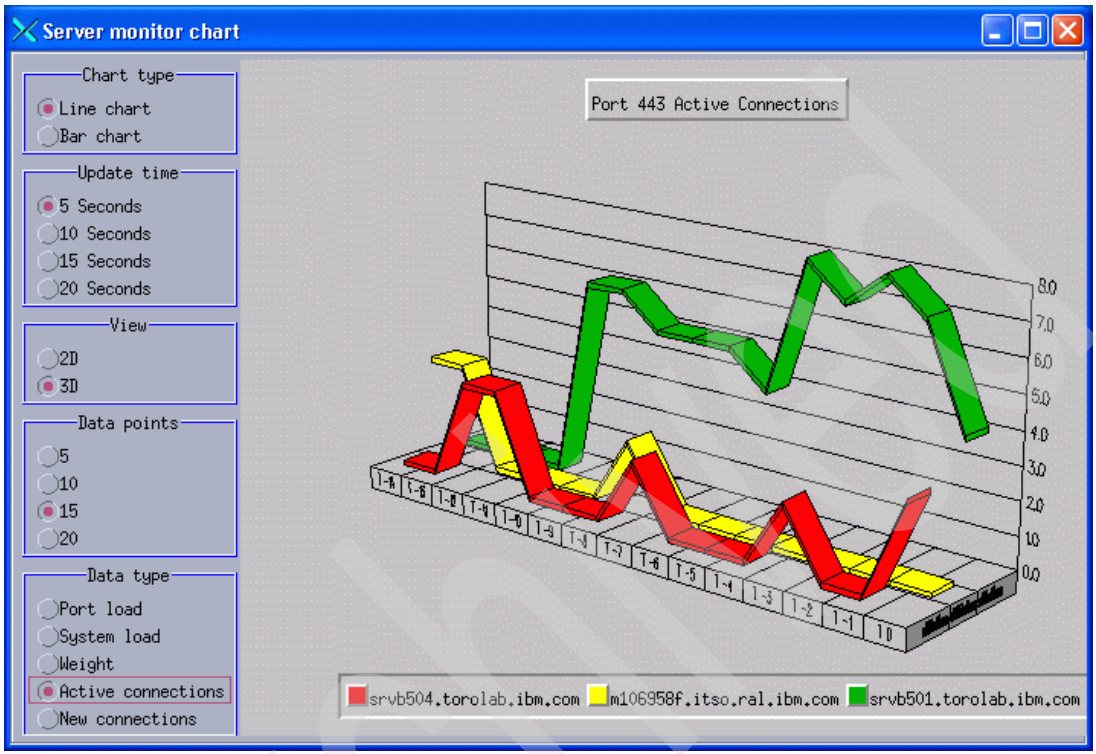


Figure 19-3 Active connections

Archive

The information about the Raleigh server being down is also available in port info of the **dscontrol manager report** output, as shown in Example 19-3.

Example 19-3 Manager report - failure of m106958f server

```
------------------------------------------------------------------
| nat1.torolab.ibm.com                                           |
|     9.26.52.154    | WEIGHT |  ACTV  |  NEWC  |  PORT  |   SYS  |
|     PORT:    80    |NOW  NEW|   49%  |   50%  |    1%  |    0%  |
------------------------------------------------------------------
| srvb504.torolab.ibm.com                                        |
|                     |  9  9  |      3 |      2 |     20 |      0 |
| m106958f.itso.ral.ibm.com                                      |
|                     |  0  0  |      0 |      0 |     -1 |      0 |
| srvb501.torolab.ibm.com                                        |
|                     | 10  10 |      4 |      4 |     22 |      0 |
------------------------------------------------------------------
```

Note that the PORT column shows the value -1 for the m106958f server. This means that the advisor is getting no response from this server. That makes the weight of this server set to zero (see column WEIGHT).

19.1.3 Binary logging

You may switch on binary logging for analyzing server information that has been gathered over time.

To start binary logging, right-click **Manager** and select **Start Binary Logging**. To stop binary logging, right-click **Manager** and select **Stop Binary Logging**. You may configure binary logging by clicking **Advisor:** ***Protocol Port*** and selecting **Configurations settings** in the right pane (see 19.2.7, “Advisor” on page 436).

The following information is stored in the binary log for each server defined in the configuration:

- Cluster address
- Port number
- ServerID
- Server address
- Server weight
- Server total connections
- Server active connections
- Server port load
- Server system load

Archived

Some of this information is retrieved from the executor as part of the manager cycle. Therefore, the manager must be running in order for the information to be logged to the binary logs.

Refer to “Using binary logging to analyze server statistics” in *Load Balancer Administration Guide,* GC31-6858, for more information about binary logging.

19.2 Tuning Load Balancer parameters

From monitoring the load balancing behavior, you can draw conclusions for tuning both Web servers and Load Balancer. In Chapter 18, “Monitor and tune Web servers” on page 391, we describe how each single Web server can be configure for good performance for WebSphere Commerce. In this section, we describe how Load Balancer can be configured to distribute requests among the Web servers in a way yielding good performance and scalability.

Most important, you should configure Load Balancer’s Dispatcher component to use the manager and advisor subcomponents to provide server weights to the executor component for load balancing. Optionally, you may use metric server, which needs to be installed on the balanced servers to send OS level statistics back to the manager component. While the use of metric server is beyond the scope of this book, manager and advisors are used in all the scenarios as described in 7.1.2, “IBM WebSphere Edge Components Load Balancer” on page 82, and, more specifically, in 11.2, “Configure Load Balancer” on page 193.

Dispatcher can be tuned in many ways. For each component (executor, manager, advisor) and each managed object (host, cluster, port, server), there are several parameters. We list some of them here with regard to optimizations for WebSphere Commerce. Refer to *Load Balancer Administration Guide,* GC31-6858, for detailed information.

Archived

To make the settings described below in the GUI of either the MAC forwarding or the NAT forwarding scenario, log on to your Load Balancer node, run **dsserver** (if necessary) and **lbadmin**, and connect to your host as described in step 1 on page 193 to 4 on page 195.

Note: For some objects, default values for contained objects can be defined. For example, you can set the default port stale time out at the cluster level for new ports in the cluster. We describe the settings under the component to which they apply. For example, port stale time out is explained at the port level.

Furthermore, all settings applying to server affinity (for example, port sticky time) are explained in 19.3, “Server affinity” on page 437.

19.2.1 Host

To display executor stings, select **Host:** ***DispatcherHostName*** in the tree view, then click **Configuration settings** in the right pane. This displays the Host settings, as shown in Figure 19-4.

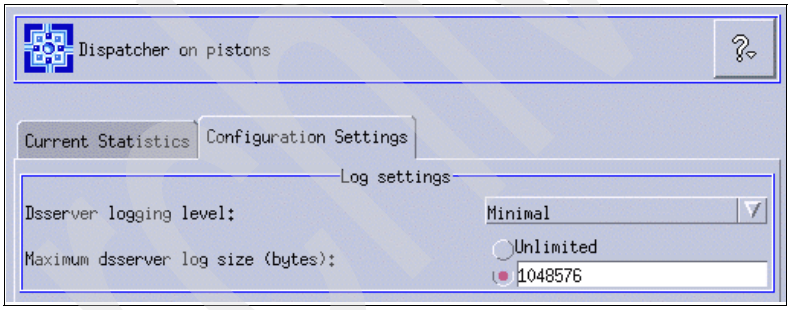


Figure 19-4 Host settings

The following parameters have an impact on performance: dsserver logging level and maximum dsserver log size. In your production system, keep the logging level low and limit the log size to minimize runtime file system access. See “Maintain Load Balancer logs” on page 596, for information about maintaining log files.

19.2.2 Executor

To display executor settings, select **Executor: *IP address*** in the tree view, then click **Configuration settings** in the right pane. This displays the executor settings, as shown in Figure 19-5.

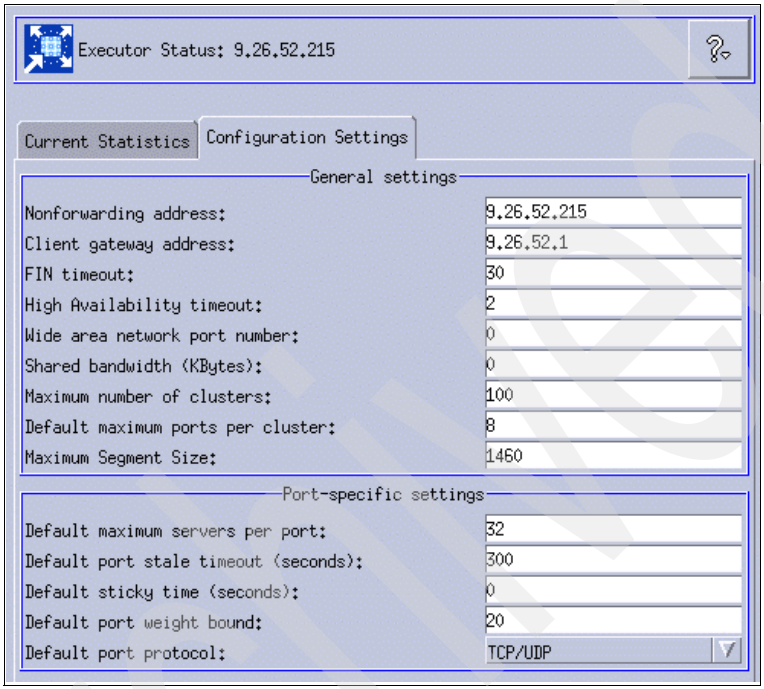


Figure 19-5 Executor settings

The following parameters have an impact on performance:

- High Availability time out: The number of seconds that the executor uses to time out High Availability heartbeats. The default value is 2. Decrease this if you want to force faster takeover when the Load Balancer is in standby mode. To avoid too many takeovers, do not choose a value that is too small.
- FIN timeout: A client sends a FIN packet after it has sent all its packets so that the server will know that the transaction is finished. When Dispatcher receives the FIN packet, it marks the transaction from active state to FIN state. When a transaction is marked FIN, the memory reserved for the connection can be cleared.

  To improve the performance of connection record allocation and reuse, set FIN time out to control the period during which Dispatcher should keep connections in the FIN state active in the Dispatcher tables and accepting

traffic. Once a connection in the FIN state exceeds the FIN time out, it will be removed from the Dispatcher tables and ready for reuse.

Once a connection in the FIN state exceeds fintimeout, it will be removed from the Dispatcher tables and ready for reuse.

- Maximum segment size: Dispatcher does not support MTU negotiation for Dispatcher's NAT (or CBR, see 19.3.3, “Configure CBR and SSL session ID affinity” on page 444) forwarding method because it is actively involved as an endpoint for TCP connections. For NAT and CBR forwarding, Dispatcher defaults the MTU value to 1500. This value is the typical MTU size for standard Ethernet, so you will probably not need to adjust this setting.

  When using Dispatcher's NAT or CBR forwarding method, if you have a router to the local segment that has a lower MTU, you must set the MTU on the Dispatcher machine to match the lower MTU.

19.2.3 Cluster

To display cluster settings, select **Cluster:** ***ClusterName*** in the tree view, then click **Configuration settings in the right pane**. This shows the cluster settings, as shown in Figure 19-6.

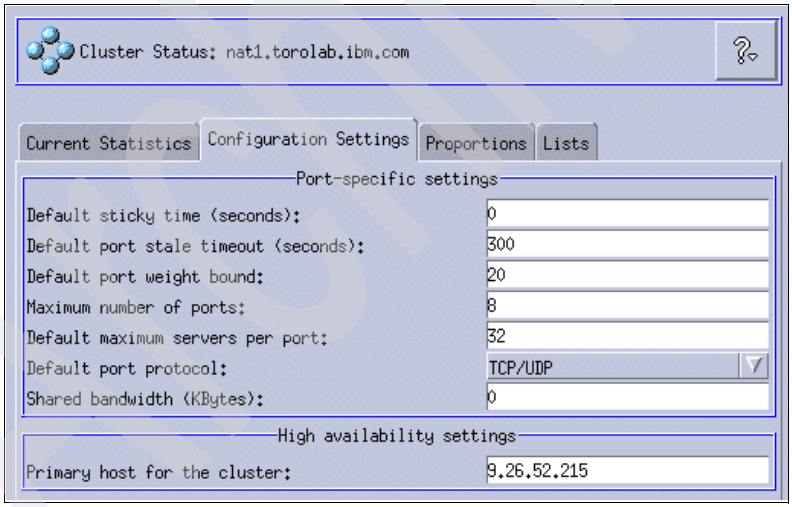


Figure 19-6 Cluster settings

There are no settings at the cluster level on this panel other than default settings for contained ports and servers, which we identified as relevant for WebSphere Commerce.

Click **Proportions** in the right pane. Here we can make more settings for generating server weights for balancing, and thus influence performance. The configuration panel is shown in Figure 19-7.

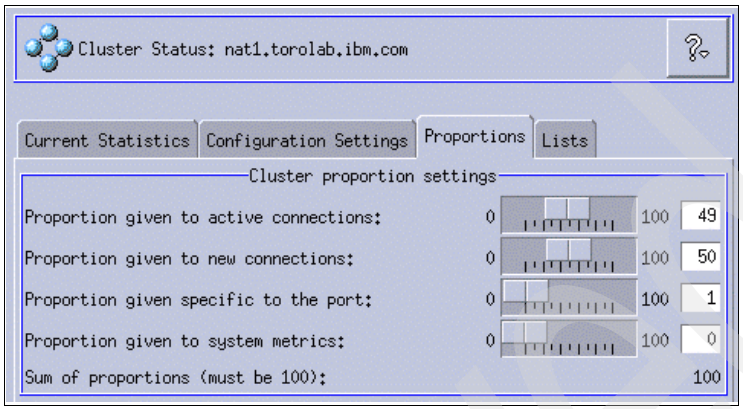


Figure 19-7 Cluster proportions setting

We can configure the importance for active connections (active), new connections (new), information from any advisors (port), and information from a system monitoring program such as metric server (system) that are used by the manager to set server weights. Each of these values, described below, is expressed as a percentage of the total and they therefore always total 100.

When an advisor is started and if the port proportion is 0, Load Balancer automatically sets this value to 1 in order for the manager to use the advisor information as input for calculating server weight.

19.2.4 Port

The most important parameter for port performance is the forwarding method, which is defined when adding the port to the cluster. MAC forwarding is the fastest method, but you can only use it if all your load-balanced Web servers are in the same IP network. NAT forwarding is a little slower, as it requires that Load Balancer also processes the Web server responses. (With MAC forwarding, the IP source IP address information is not modified, so the Web servers can send their responses directly back to the client.) Content-based routing is generally the most expensive method, depending on the routing algorithm and the complexity of the rules in use (see “Rule-based load balancing” on page 431).

You could increase balancing performance by working without server weights, for example, without manager and advisors. However, we do not recommend

this, as you would not be able to automatically detect Web server outages and you might risk server overloading.

Once a port is created, its forwarding method cannot be changed, but there are several configuration options that can be changed at runtime.

To display port settings, select **Port: *PortNumber*** in the tree view, then click **Configuration settings** in the right pane. This displays the port settings, as shown in Figure 19-8.

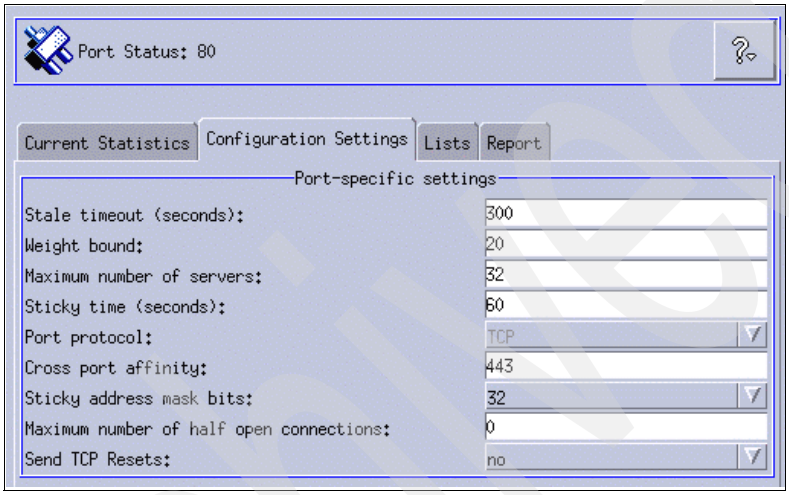


Figure 19-8 Port settings

The following parameters have an impact on performance:

- Stale time out: For Load Balancer, connections are considered stale when there has been no activity on that connection for the number of seconds specified in stale time out. When the number of seconds has been exceeded with no activity, Load Balancer will remove that connection record from its tables, and subsequent traffic for that connection will be discarded.

  Some well-defined ports have different default stale time-out values. For example, the Telnet port 23 has a default of 259,200 seconds. Some services may also have staletimeout values of their own.

  Connectivity problems can occur when Load Balancer's stale timeout value is smaller than the service's timeout value. Clients may then believe that they have connections to the server after the timeout value, but Load Balancer will discard all connections in that case.

  You can use stale timeout and FIN timeout (at the executor level) to control the cleanup of connection records.

- Weight bound: This specifies the maximum weight boundary that any server can have. Weights are set by the manager function based upon internal counters in the executor, feedback from the advisors, and feedback from a system-monitoring program, such as metric server. As you increase this number, the difference in how servers can be weighted is increased. At a maximum weightbound of 2, one server could get twice as many requests as another. At a maximum weightbound of 10, one server could get 10 times as many requests as another. The default maximum weightbound is 20.
- Sticky time: See 19.3, “Server affinity” on page 437.
- Port protocol: This is only changeable for MAC forwarding. Depending on the protocol that you are balancing, this can be set to TCP, UDP, or both. In case of HTTP, it must be either TCP or both.
- Cross port affinity: See 19.3, “Server affinity” on page 437.
- Sticky address mask bits: See 19.3, “Server affinity” on page 437.
- Maximum number of half-open connections: The threshold for the maximum half-open connections. Use this parameter to detect possible denial of service attacks that result in a large number of half-opened TCP connections on servers.

  A positive value indicates that a check will be made to determine whether the current half-open connections exceed the threshold. If the current value is above the threshold, a call to an alert script is made. See “Denial of service attack detection” in *Load Balancer Administration Guide,* GC31-6858, for more information about how to set this up.
- Send TCP resets: If TCP reset is activated, Dispatcher will send a TCP reset to the client when the client has a connection to a server whose weight is 0. A server's weight may be 0 if it is configured as 0 or if an advisor marks it down. A TCP reset will cause the connection to be immediately closed. This feature is useful for long-lived connections where it hastens the client's ability to renegotiate a failed connection.

  A useful feature to configure, in conjunction with TCP reset, is advisor retry. With this feature, an advisor has the ability to retry a connection before marking a server down. This would help prevent the advisor from marking the server down prematurely, which could lead to connection-reset problems. That is, just because the advisor failed on the first attempt does not necessarily mean that the existing connections are also failing. See 19.2.7, “Advisor” on page 436.

Rule-based load balancing

You can configure rule-based load balancing on a port to fine-tune when and why packets are sent to which servers. To do that, right-click **Port: *PortNumber***, then click **Add rule** and select the type of rule. This displays a pop-up window for

configuring a rule of the chosen type). While there are a number of different rules available for MAC and NAT forwarding, we do not recommend using them for WebSphere Commerce scenarios like ours for the following reasons:

- All of our Web servers are capable of serving the same content.
- The Web server utilizes only static content. All dynamic content is generated by the application tier.
- Differences in capacity of the Web servers are managed by manager and the advisors (and, if you use them, the metric servers).
- Rule evaluation is generally expensive. When using content rules, Load Balancer also needs to collect all IP packets of a request before it can forward the request.

Refer to *Load Balancer Administration Guide,* GC31-6858, for detailed information about rule-based load balancing.

19.2.5 Server

To display server settings, select **Server: *Hostname*** in the tree view, then click **Configuration settings** in the right pane. This displays the server settings, as shown in Figure 19-9.

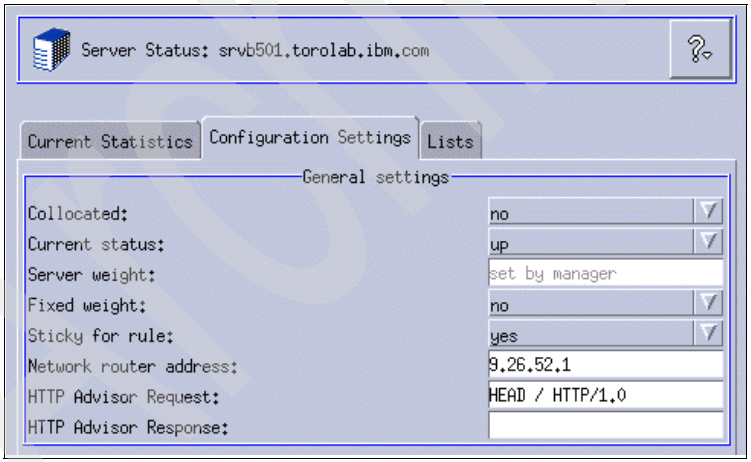


Figure 19-9 Server settings

Archived

The following parameters have an impact on performance:

- Collocated: See step 15 on page 208 and step 7 on page 233.
- Current status: Use this to manually exclude a server from load balancing, for example, in order to apply changes to it. See Chapter 27, “Maintain and update Web servers” on page 579.
- Server weight: A number from 0–100 (but not to exceed the specified port's weightbound value) representing the weight for this server. Setting the weight to zero will prevent any new requests from being sent to the server, but will not end any currently active connections to that server. The default is one-half the specified port's maximum weightbound value. If the manager is running, this setting will be quickly overwritten.
- Fixed weight: The fixedweight option allows you to specify whether you want the manager to modify the server weight. If you set the fixedweight value to yes, when the manager runs it will not be allowed to modify the server weight.
- HTTP Advisor Request and Response: You may configure which requests an advisor should send to this server in order to check the server’s status, and which response the advisor should expect (if any). For instance, you could have the advisor try to access your store homepage. See “Configuring the HTTP or HTTPS advisor using the request/response (URL) option” in *Load Balancer Administration Guide,* GC31-6858, for details.

Archived

19.2.6 Manager

To display manager settings, select **Manager** in the tree view, then click **Configuration settings** in the right pane. This displays the manager settings, as shown in Figure 19-10.

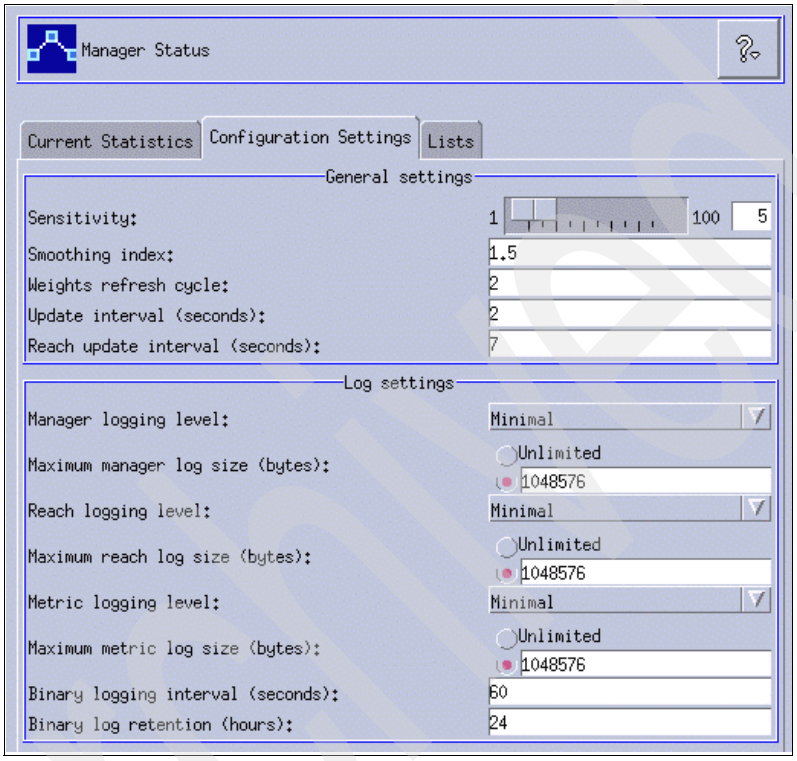


Figure 19-10 Manager settings

The following parameters have an impact on performance:

- Sensitivity: To work at top speed, updates to the weights for the servers are only made if the weights have changed significantly. Constantly updating the weights when there is little or no change in the server status would create an unnecessary overhead. When the percentage weight change for the total weight for all servers on a port is greater than the sensitivity threshold, the manager updates the weights used by the executor to distribute connections. Consider, for example, that the total weight changes from 100 to 105. The change is 5%. With the default sensitivity threshold of 5, the manager will not update the weights used by the executor, because the percentage change is not above the threshold. If, however, the total weight changes from 100 to 106, the manager will update the weights.

- Smoothing index: The manager calculates the server weights dynamically. As a result, an updated weight can be very different from the previous one. Under most circumstances, this will not be a problem. Occasionally, however, it may cause an oscillating effect in the way the requests are load balanced. For example, one server can end up receiving most of the requests due to a high weight. The manager will see that the server has a high number of active connections and that the server is responding slowly. It will then shift the weight over to the free servers and the same effect will occur there, too, creating an inefficient use of resources. To alleviate this problem, the manager uses a smoothing index. The smoothing index limits the amount that a server's weight can change, effectively smoothing the change in the distribution of requests. A higher smoothing index will cause the server weights to change less drastically. A lower index will cause the server weights to change more drastically. The default value for the smoothing index is 1.5. At 1.5, the server weights can be rather dynamic. An index of 4 or 5 will cause the weights to be more stable.
- Weights refresh cycle: The manager refresh cycle specifies how often the manager will ask the executor for status information. The refresh cycle is based on the interval time. For example, if you set this to three, the manager will wait for three intervals before asking the executor for status.
- Update interval: This is the base for the weights refresh cycle setting.
- Reach update interval: This sets the update interval for the reach advisor, which you can use to check whether certain reach targets are up and responding. The advisor is also used by the High Availability component of Load Balancer to check the reach targets (see 11.3.2, “Adding reach targets” on page 235).
- Manager, Reach, and Metric logging level: In your production system, keep the logging level low and limit the log size to minimize runtime file system access. See “Maintain Load Balancer logs” on page 596, for information about maintaining log files.
- Binary logging interval: The interval option controls how often information is written to the logs. The manager will send server information to the log server every manager interval. The information will be written to the logs only if the specified log interval seconds have elapsed since the last record was written to the log.
- Binary log retention: The retention option controls how long binary log files are kept. Log files older than the retention hours specified will be deleted by the log server. This will only occur if the log server is being called by the manager, so stopping the manager will cause old log files *not* to be deleted.

Archive

19.2.7 Advisor

To display advisor settings, select **Advisor: *Protocol Port*** in the tree view, then click **Configuration settings** in the right pane. This displays the advisor settings, as shown in Figure 19-11.

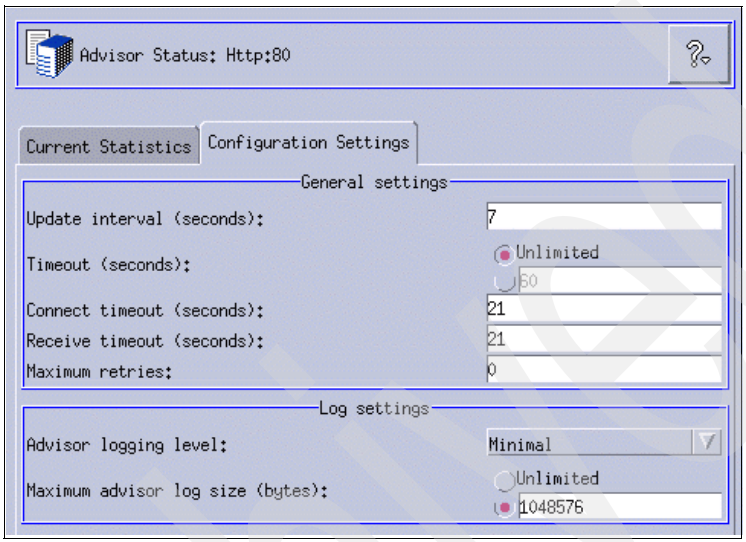


Figure 19-11 Advisor settings

The following parameters have an impact on performance:

- Update interval: Set this value to control how often the advisor will query the servers for information.
- Timeout: This is the timeout in seconds at which the advisor waits before reporting that a receive from a server fails.
- Connect timeout: Set how long an advisor waits before reporting that a connect to a server for a particular port on a server (a service) fails.

  To obtain the fastest failed-server detection, set the advisor connect and receive time outs to the smallest value (one second), and set the advisor and manager interval time to the smallest value (one second). However, if your environment experiences a moderate to high volume of traffic such that server response time increases, be careful not to set the connect timeout and receive timeout values too small, or the advisor may prematurely mark a busy server as failed.
- Receive timeout: Set this to control how long an advisor waits before reporting that a receive from a particular port on a server (a service) fails.

Archived

- Maximum retries: Advisors have the ability to retry a connection before marking a server down. The advisor will not mark a server down until the server query has failed the number of retries plus 1. While we recommend in *Load Balancer Administration Guide,* GC31-6858, that the retry value should be no larger than three, we recommend setting it to one or even to zero, if your network is stable. If you choose to set it to zero, you should not send TCP resets for the port (see 19.2.4, “Port” on page 429).
- Logging level and maximum log size: In your production system, keep the logging level low and limit the log size to minimize runtime file system access. See “Maintain Load Balancer logs” on page 596, for information about maintaining log files.

19.3 Server affinity

Server affinity is a technique that enables the Load Balancer to remember which balanced server was chosen for a certain client at its initial request. Subsequent requests are then directed to the same server again.

If the affinity feature is disabled when a new TCP/IP connection is received from a client, Load Balancer chooses the correct server at that moment and forwards the packet to it. If a subsequent connection comes in from the same client, Load Balancer treats it as an unrelated connection, and again chooses the most appropriate server at that moment.

Server affinity allows load balancing for those applications that need to preserve state across distinct connections from a client at the Web server tier.

WebSphere Commerce does not store client state at the Web server tier. Web servers are only used for serving static content. All dynamic processing is done in the application server tier, where client state is preserved in the form of (HTTP) session objects (which might even be stored persistently in a database). When processing a request in the Web container, cookies are added to the response, containing WebSphere Commerce session information. The client browser resends these cookies with every new request, so that the application server can identify the user based on the cookie values.

While WebSphere Commerce requests may be load balanced to any Web server, server affinity can still be useful for WebSphere Commerce sites:

- For HTTPS, server affinity can be used to avoid repeated SSL handshake processing between the client and different Web servers.
- As a complete Web page is typically retrieved by multiple requests (for example, one for the HTML markup and several subsequent requests for

embedded images and other media content), HTTP compression can be used more efficiently if all requests for one page are routed to the same Web server.

- Finally, if all requests within a browser session (or even all requests of the same user across multiple sessions) are processed by only one Web server, access log analysis is easier, as all the information is written to the same log file.

19.3.1 Types of server affinity

Some options available to maintain application state based on server affinity are:

- Stickyness to source IP address
- Cross-port affinity
- Passive cookie affinity
- Active-cookie affinity
- URI affinity
- SSL session ID

When using Load Balancer, the passive cookie, active cookie, and URI affinity options are rule based. They depend on the content of the client requests.

Stickyness to source IP address

This affinity feature is enabled by configuring the clustered port to be sticky. Configuring a cluster port to be sticky allows subsequent client requests to be directed to the same server. This is done by setting the sticky time to a positive number. The feature can be disabled by setting the sticky time to zero.

The sticky time value represents the timeout of the affinity counter. The affinity counter is reset every time Load Balancer receives a client request. If this counter exceeds sticky time, new connections from this client may be forwarded to a different back-end server.

In Dispatcher and CBR components, you can set the sticky time in three elements of the Load Balancer configuration:

- Executor: Setting the sticky time for the executor makes this value valid for all clusters and ports in the configuration.
- Cluster: You can set a specific sticky time value for each cluster.
- Port: You can set a specific sticky time value for each port.

Archived

Important: Setting affinity at the different levels means that any subsequent lower-level objects inherit this setting by default (when they are added). In fact, the only true value that is used for sticky time is what is set at the port level. So, if you set the sticky time for the executor to 60, then add a cluster and port, these also have a sticky time of 60.

However, if you set a different sticky time for the cluster or the port, for example, you set it to 30, then this value overrides the Executor sticky time.

This feature applies to all forwarding methods. We describe how to set up source IP affinity for MAC and NAT forwarding in 19.3.2, “Configure source IP affinity for MAC and NAT forwarding” on page 442.

Note: This affinity strategy has some drawbacks. Some ISPs use proxies that collapse many client connections into a small number of source IP addresses. A large number of users who are not part of the session will be connected to the same server. Other proxies use a pool of user IP addresses chosen at random, even for connections from the same user, invalidating the affinity.

Cross-port affinity

Cross-port affinity is the sticky feature that has been expanded to cover multiple ports. For example, if a client request is first received on one port and the next request is received on another port, cross port affinity allows Dispatcher to send the client requests to the same server.

For example, a user browses products and adds them to his shopping cart using port 80 (HTTP). For viewing the shopping cart and for the checkout process, HTTPS is used, which will encrypt all communication between the browser and the server. Cross-port affinity enables Dispatcher to forward this user’s requests for both ports 80 and 443 to the same server.

In order to use this feature, the ports must:

- Share the same cluster address.
- Share the same servers.
- Have the same sticky time value (not zero).
- Have the same sticky mask value.

More than one port can link to the same cross port. When subsequent connections come in from the same client on the same port or a shared port, the same server will be accessed.

Cross-port affinity applies to the MAC and NAT forwarding methods of the Dispatcher component. We describe how to set up cross port affinity for MAC and NAT forwarding in 19.3.2, “Configure source IP affinity for MAC and NAT forwarding” on page 442.

Note: The same drawbacks apply as for source IP affinity (see above).

Passive cookie affinity

Passive cookie affinity is based on the content of cookies (name/value) generated by the HTTP server or by the application server. You must specify a cookie name to be monitored by Load Balancer in order to distinguish which server the request is to be sent to.

If the cookie value in the client request is not found or does not match any of the cookie values of the servers, the most appropriate server at that moment will be chosen by Load Balancer.

This feature requires that CBR forwarding is used (see 19.3.3, “Configure CBR and SSL session ID affinity” on page 444) and works only on HTTP ports, not on HTTPS ports, because Dispatcher cannot decrypt SSL content. For content-based routing of HTTPS traffic you would need to use the Load Balancer CBR *component* in conjunction with IBM WebSphere Edge Components Caching Proxy. This configuration is beyond the scope of this book. Refer to *Concepts, Planning, and Installation for Edge Components Version 6.0,* GC31-6855, for more information.

Although looking at cookies could help to overcome the drawbacks of source IP affinity (see above), there are reasons why this type of affinity does *not* work well with WebSphere Commerce generated cookies:

- The cookies WC_USERACTIVITY and WC_AUTHENTICATION are only generated when a generic user logs in or adds an item to the shopping cart. From these points, the users are not generic users anymore. They are registered or guest users identified by a user ID. WC_USERACTIVITY and WC_AUTHENTICATION cookies carry the user ID as part of their names, *not* values. Load Balancer cannot be configured to look for cookies with names consisting of a static and a dynamic part.
- The cookie JSESSIONID identifies the HTTP session and would be a good candidate for routing requests to the same Web server for a user *session*. This cookie’s name is static, so Load Balancer can identify it in HTTP requests. But Load Balancer does not keep track of the Web server used for the first request made with a new cookie value (it does not hold the cookie value). Rather than doing that, it requires that each server configured for a port is assigned a fixed value, which is compared to the cookie value at every

request. If the server value is contained in the cookie value, the server is chosen to handle the request. JSESSIONID carries as part of its value the clone ID of the application server that handled the first request of the session, so JSESSIONID could only be used to link Web servers and application servers, which we do not recommend for WebSphere Commerce.

Active-cookie affinity

Active-cookie affinity enables load balancing Web traffic with affinity to the same server based on cookies generated by the Load Balancer. This function is enabled by setting the sticky time of a rule to a positive number, and setting the affinity to *cookie*. The generated cookie contains:

- The cluster, port, and rule
- The server that was load balanced to
- A timeout time stamp for when the affinity is no longer valid

Active cookie affinity formats the cluster/port/server/time information into a key value in the format of IBMCBR#####, so the IP and configuration information is not visible to the client browser.

The active cookie affinity feature applies only to the CBR *component*, which requires IBM WebSphere Edge Components Caching Proxy. This configuration is beyond the scope of this book. Refer to *Concepts, Planning, and Installation for Edge Components Version 6.0,* GC31-6855, for more information.

URI affinity

URI affinity allows you to load balance Web traffic to caching proxy servers, which allow unique content to be cached on each individual server. As a result, you will effectively increase the capacity of your site's cache by eliminating redundant caching of content on multiple machines. You can configure URI affinity at the rule level, and once it is enabled and the servers are running, then the Load Balancer will forward new incoming requests with the same URI to the same server.

Dispatcher can use URI affinity only with HTTP. For HTTPS you would need the CBR *component* in conjunction with IBM WebSphere Edge Components Caching Proxy. This configuration and load balancing to caching proxy servers are beyond the scope of this book. Refer to *Concepts, Planning, and Installation for Edge Components Version 6.0,* GC31-6855, for more information.

Archived

SSL session ID

During establishment of an SSL encrypted session, a handshake protocol is used to negotiate a session ID. This handshaking phase consumes a good deal of CPU power, so directing subsequent HTTPS requests to the same server, using the already established SSL session, saves processing time and increases the overall performance of the Web server.

Load Balancer watches the packets during the handshake phase and holds information about the session ID if SSL session negotiation is detected. To avoid repeated SSL handshakes between a client and different Web servers, this affinity type may be configured for WebSphere Commerce.

The forwarding method used to configure SSL session ID affinity is the Dispatcher's CBR forwarding method. We describe how to configure the CBR forwarding method and SSL session ID affinity in 19.3.3, "Configure CBR and SSL session ID affinity" on page 444.

Server affinity conclusion

When using IBM WebSphere Edge Components Load Balancer, the types of server affinity that we recommend to use for WebSphere Commerce are source IP affinity and cross-port affinity.

While using SSL, session ID affinity is possible for HTTPS traffic using the CBR forwarding method. Source IP affinity and cross-port affinity are not configurable at the same time for HTTP ports (only basic rule-based affinity is possible).

Passive cookie affinity cannot be used with WebSphere Commerce cookies.

Active cookie and URI affinity methods require advanced configurations. This configuration requiring IBM WebSphere Edge Components Caching Proxy is not covered in this book. Another reason for not covering it here is that the Caching Proxy is also deprecated as of IBM WebSphere Application Server Network Deployment V6.1.

19.3.2 Configure source IP affinity for MAC and NAT forwarding

With MAC and NAT forwarding, source IP affinity can be achieved for a port by configuring the port to be sticky.

GUI configuration

To configure sticky ports for both your primary and your standby Load Balancer server, using the GUI:

1. Open the Load Balancer GUI by running **lbadmin** and connect to your server as described in steps 3 on page 194 through 4 on page 195 in 11.2, “Configure Load Balancer” on page 193.
2. Expand the tree view to see the ports.
3. In our scenario we set the sticky time for ports 80 and 443 to 60 seconds. In the tree view, select **Port:80**. In the right pane of the GUI, select the **Configuration settings** tab. Enter a value of 60 into the Sticky time field. Then click **Update configuration**. Repeat for port 443 or any other port that needs to be sticky. (Figure 19-12 shows this setting for port 80.)

   WebSphere Commerce switches between HTTP and HTTPS within one session. For example, when users have finished shopping and proceed to the checkout, or when they navigate to their user profile, they will be directed to a HTTPS site, which will encrypt all communication between the browser and the server. Cross-port affinity enables Dispatcher to forward this user’s requests for both ports 80 and 443 to the same server.
4. To set up cross-port affinity 80 to 443, select **Port:80** in the tree view, then select the **Configuration settings** tab in the right pane. Enter the value 443 in the Cross port field, as shown in Figure 19-12. Do not forget to click **Update configuration**.

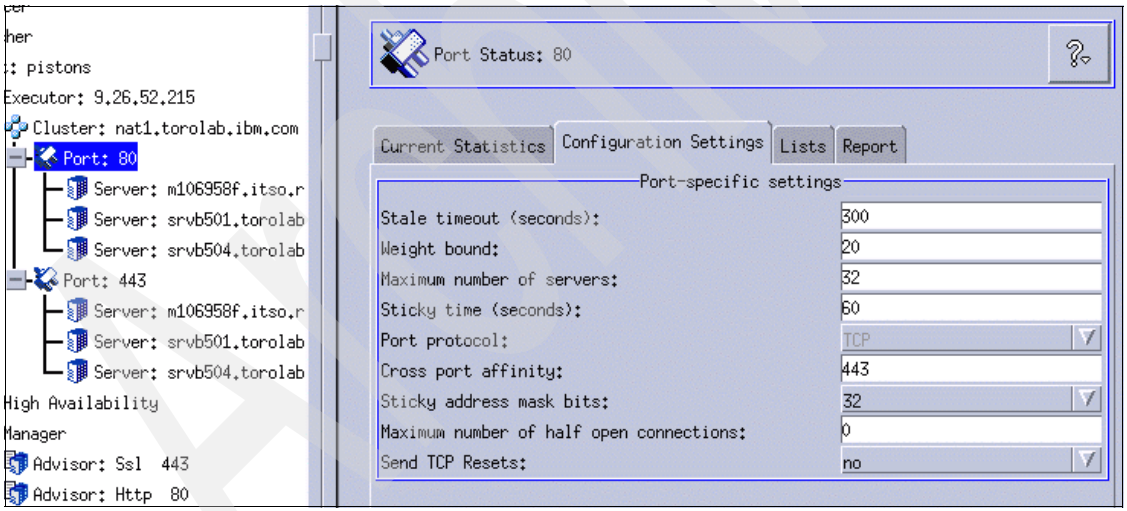


Figure 19-12 Setting sticky time and cross-port affinity for port 80

5. Save the configuration as explained in step 15 on page 208 in 11.2, “Configure Load Balancer” on page 193.

Command-line configuration

Port stickyness can also be configured using **dscontrol** from the command line. Example 19-4 shows the additional configuration for both the primary and the standby server.

Example 19-4 Setting up source IP stickyness and cross port affinity

```
dscontrol port set nat1.torolab.ibm.com:80 stickytime 60
dscontrol port set nat1.torolab.ibm.com:443 stickytime 60
dscontrol port set nat1.torolab.ibm.com:80 crossport 443
```

19.3.3 Configure CBR and SSL session ID affinity

If you decide that you do not need source IP stickyness for HTTP, but still want to reduce SSL overhead, you may use Dispatcher's content-based routing (CBR) forwarding method.

To set up CBR forwarding, you have two options: using the GUI or changing the NAT configuration files and reloading them using the GUI.

Configure CBR using the GUI

Follow the steps below:

1. Use the GUI and follow the instructions for NAT forwarding setup, for example, steps 1 on page 211 through 11 on page 218.
2. Only when adding ports (see step 4 on page 214), set the forwarding method to content-based routing, as shown in Figure 19-13.

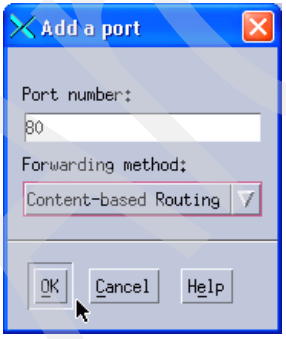


Figure 19-13 Adding a port for CBR

Archived

3. After clicking **OK**, if Content-based Routing is selected, another dialog box is displayed for choosing the protocol to be forwarded, as shown in Figure 19-14.

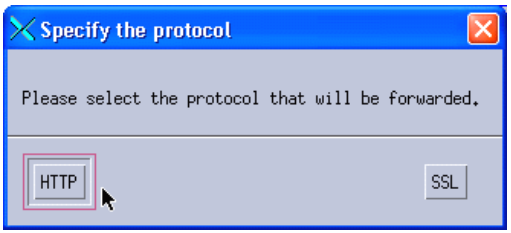


Figure 19-14 Selecting the port protocol for CBR

4. Click **HTTP** when adding port 80, and **SSL** when adding port 443. Also choose **SSL** for WebSphere Commerce administrative ports such as 8000, 8002, and so on.
5. After this, continue with the setup as described for the NAT forwarding method.
6. SSL session ID affinity is activated by setting the sticky time for the HTTPS port. Expand the tree view on the left until you see the ports, then click **Port: 443**, and in the right pane click **Configuration settings**.
7. Change the Sticky time (seconds) field to your desired sticky time, for example, 60, as shown in Figure 19-15. Note that there is no field for cross-port affinity when using CBR forwarding.

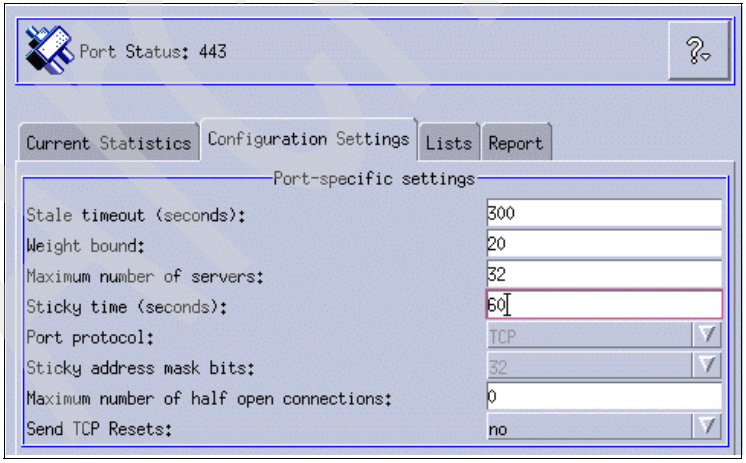


Figure 19-15 Setting sticky time for port 443

8. Click **Update Configuration** at the bottom to activate the change.

Configure CBR by modifying the NAT configuration files

If you have configured NAT forwarding and High Availability (but not source IP affinity or cross-port affinity), you may change the relevant lines in the configuration files for the primary Load Balancer and the Standby Load Balancer and then load the new configuration files.

1. The lines in the NAT configuration files that need to be changed are shown in Example 19-5.

   Example 19-5 NAT configuration file lines to be changed for CBR (two lines)

   ```
   dscontrol port add nat1.torolab.ibm.com:80 method nat reset no
   dscontrol port add nat1.torolab.ibm.com:443 method nat reset no
   ```

2. Change these lines and add a line for port 443 sticky time, as shown in Example 19-6.

   Example 19-6 New CBR configuration file lines for adding the ports (two lines)

   ```
   dscontrol port add nat1.torolab.ibm.com:80 method cbr protocol http
   reset no
   dscontrol port add nat1.torolab.ibm.com:443 method cbr protocol ssl
   reset no
   dscontrol port set nat1.torolab.ibm.com:443 stickytime 60
   ```

3. Save the new files to the configuration directory (*LoadBalancer_Install_Dir*/servers/configurations/dispatcher on Linux, AIX, and Solaris) on the primary Load Balancer and the Standby Load Balancer, then follow the remaining steps on each machine.

   We now need to load the new files using the GUI on each machine.

4. Open the Load Balancer GUI by running **lbadmin** and connect to your server as described in steps 3 on page 194 through 4 on page 195 in 11.2, “Configure Load Balancer” on page 193.

Archive

5. Right-click **Host: *Hostname*** and select **Load New Configuration**, as shown in Figure 19-16.

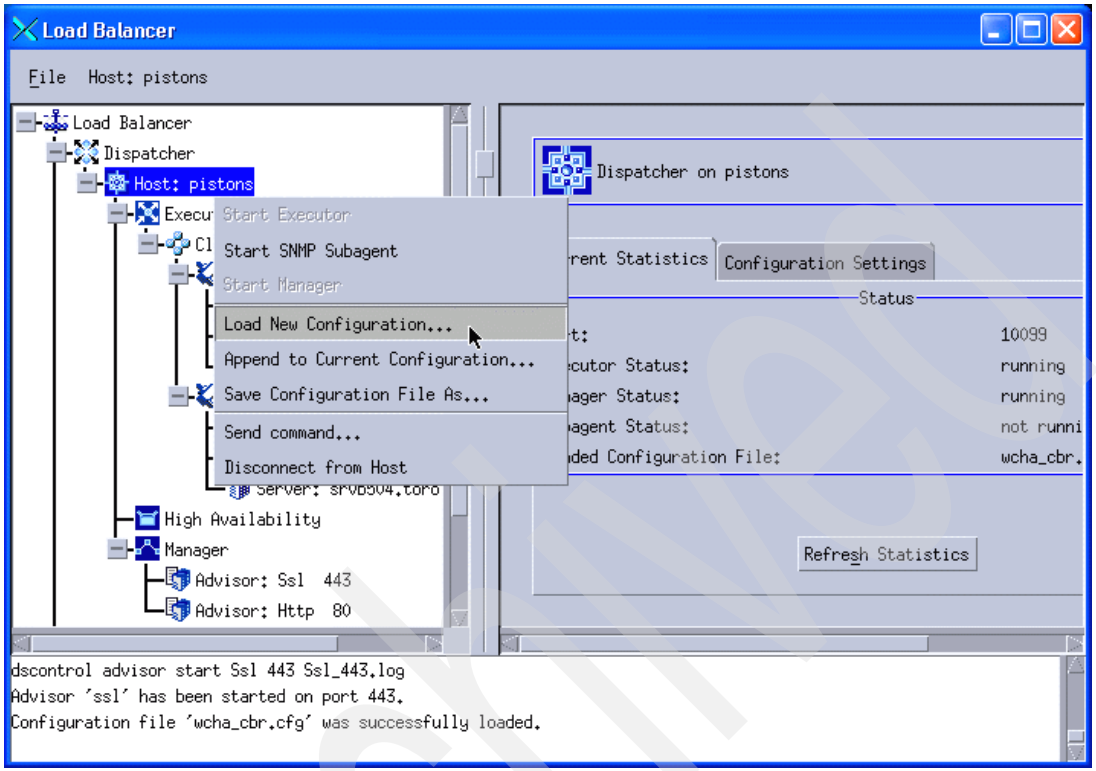


Figure 19-16 Loading a new configuration

6. A dialog is displayed showing the files in the configuration directory, as shown in Figure 19-17.

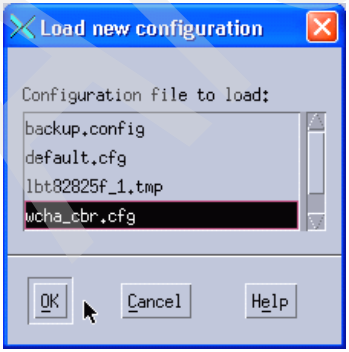


Figure 19-17 Choosing the configuration file from the configuration directory

7. Click **OK**.

The new configuration is now being loaded and activated.

SSL traffic is now sticky and the SSL session ID is saved as long as the sticky time. If a client does not make another request during that time, the ID is discarded and a new server is picked upon the next request for that client.

When using CBR with the Dispatcher component, you may use an additional rule type (content) for HTTP ports by right-clicking **Port: 80**, selecting **Add Rule**, and then selecting **Content**, as shown in Figure 19-18.

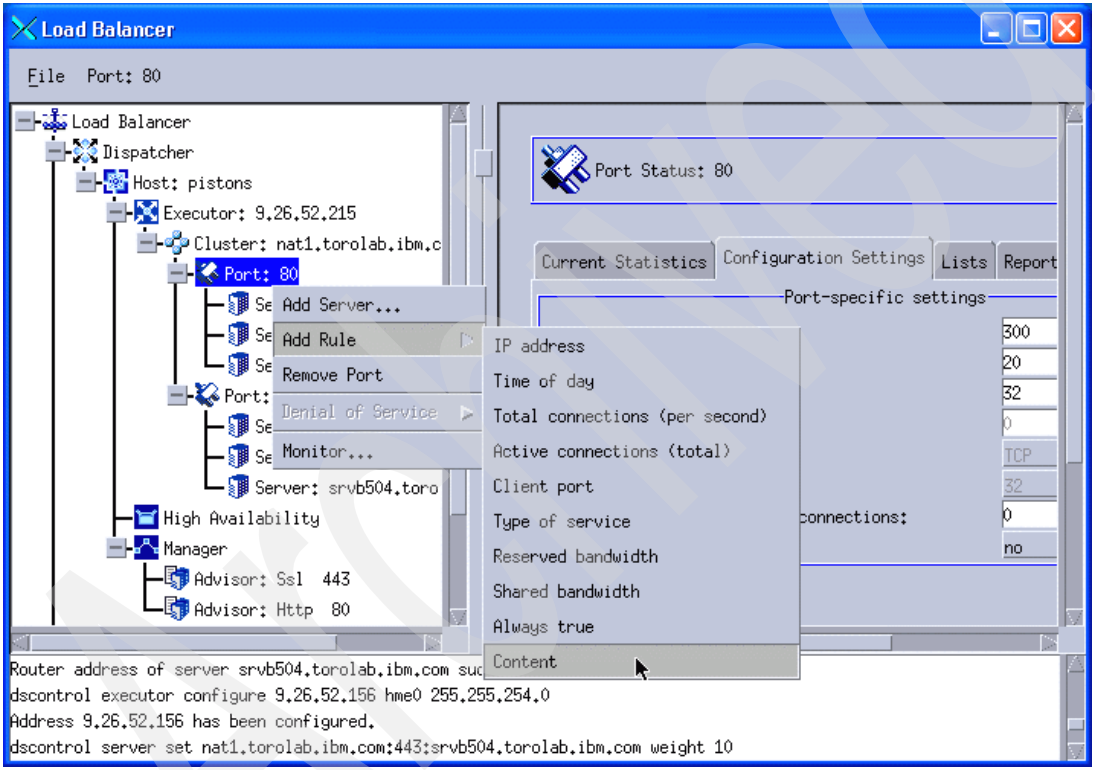


Figure 19-18 New rule type content for HTTP ports when using CBR forwarding

Refer to *Load Balancer Administration Guide,* GC31-6858, for detailed information about how to configure content-based rules.

Note, however, that, as mentioned above, passive cookie affinity does not work for any rule type with WebSphere Commerce cookies.

19.3.4 Testing server affinity

After having set up source IP and cross-port affinity for MAC or NAT forwarding, or SSL session ID affinity for CBR forwarding, you can verify that server affinity works by right-clicking the port that you want to monitor and selecting **Monitor**. This will display the Server monitor chart window.

Now quickly make repeated requests using the same machine (for IP affinity), or the same browser instance (for SSL session ID affinity, respectively). For example, quickly navigate through your WebSphere Commerce store front.

You will see that new connections from your browser are routed to only one Web server (Figure 19-19).

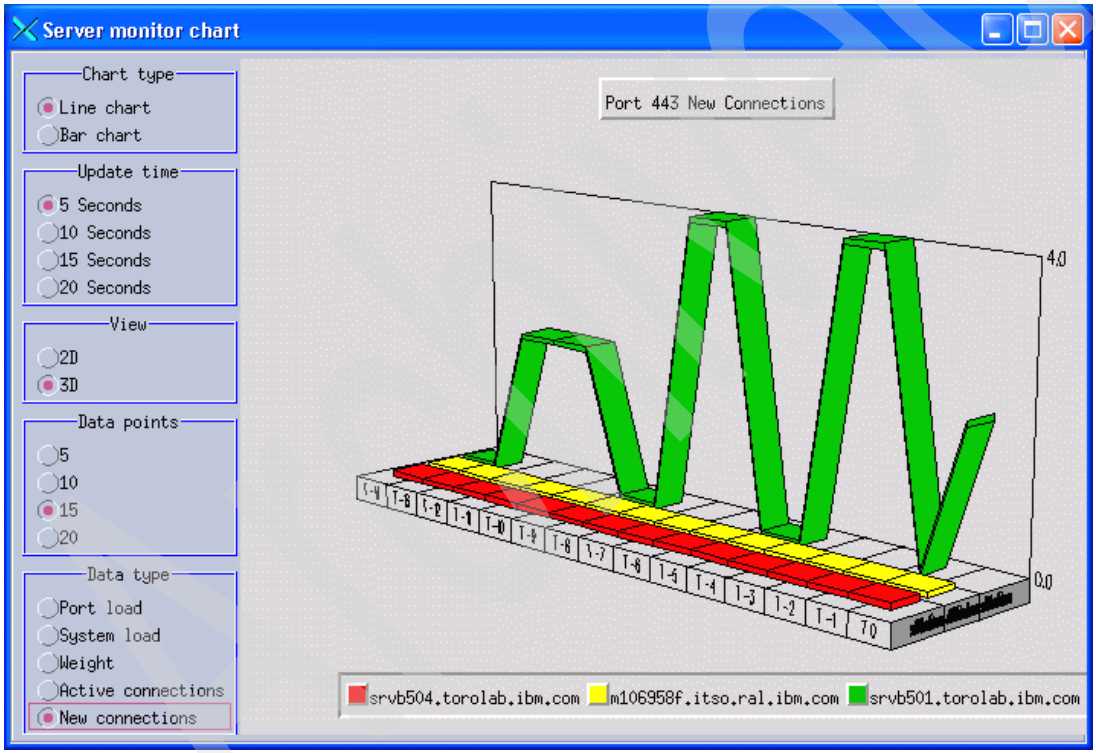


Figure 19-19 New connections with server affinity

Active connections on the other servers are being closed for the other Web servers, while the number of active connections to the server that has initially been chosen is increasing, as shown in Figure 19-20.

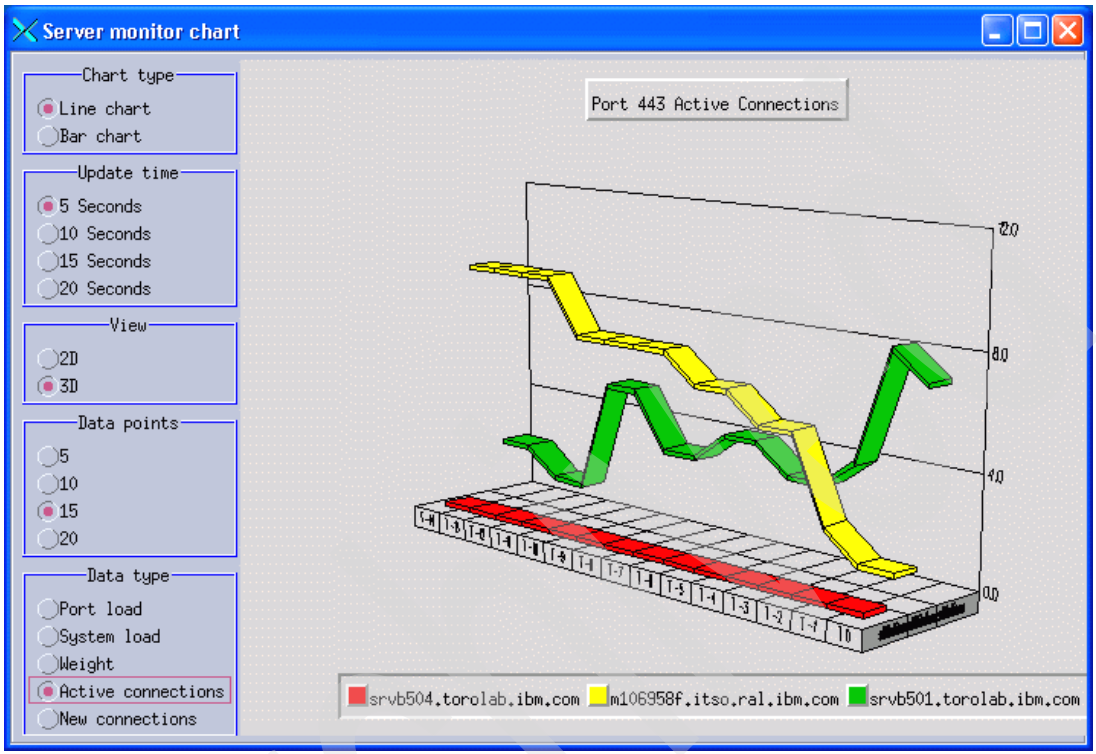


Figure 19-20 Active connections with server affinity

Part 6

Performance test

Considering that the speed, scalability, stability, and confidence are the majority concerns for a business application, performance testing should be well defined and well planned to proactively detect performance issue before deploying to the production site. In this part, we cover five areas:

- Brief introduction to performance testing in WebSphere Commerce
- Design performance test plan
- Performance test tools
- Applying performance testing to WebSphere Commerce
- Analyzing test results and solving performance problems

© Copyright IBM Corp. 2008. All rights reserved.

Archived

20

Introduction to performance testing

Performance testing, also known as load testing, is testing of the responsiveness of a system. For example, performance testing can be used to ascertain how quickly can a system respond, how many responses a system provide can in a given unit of time, how long a certain level of responsiveness can be maintained, at what cost, and so on.

The objective of performance testing is not to test the proper functioning of a given component or product in a single user test scenario. This aspect of testing should be covered by *function testing*. We assume that the single user function testing has already passed successfully, prior to performance testing that function.

All the responses received during performance testing should be error free or the number of errors should be within an acceptable margin of error. During performance testing these errors could be due to concurrent users accessing the system or due to other performance defects. If a given system has good performance but the number of defective responses (due to functional, performance, or other defects) are above the margin of error, then such a performance test will be deemed to have failed. A failed test case is not always undesirable. For example, it can be useful in ascertaining capacity limits during stress testing.

© Copyright IBM Corp. 2008. All rights reserved.

The aim of performance testing is to:

- Establish the performance characteristics of the application, such as transaction response time, transaction rates, and CPU utilization. Profiling can be used on those slow requests to help determine the root cause of their poor response times. Fore more information about profiling refer to Chapter 14, “Profiling” on page 309.
- Identify system bottlenecks.
- Determine system capacity.
- Demonstrate system scalability by:
  - Measuring CPU and throughput at various loads
  - Employing techniques to fully utilize available hardware
  - Driving machines to >90% utilization
  - Validating that hardware and software configuration is optimal

Methodical performance testing and recording of results reveals system capacity and provides insight into system scalability. A performance reference model, obtained from the execution of a controlled test plan, provides scaling predictability. Thus, when a variable, such as the number of users, is increased, the reference model may flag the need for additional hardware capacity.

The performance test plan is a methodical testing approach that starts with acquisition of baseline information, and then adds capacity to determine how the system scales for all components. By controlled workload escalation, the defined test procedures expose constrained resources, or *bottlenecks*, which cause the overall system to begin displaying non-linear performance characteristics. The bottleneck then becomes the focus of *drill down* testing and analysis to alleviate the constraint, after which the process is repeated. Commonly restrained resources include thread pools, connection pools, and available memory.

Performance test scenarios are similar to, and sometimes overlap, other types of testing, but with the focus on developing a deep understanding of system performance. There are typically two desired results of a performance test:

- Gaining knowledge of how a system performs
- Using that knowledge to improve the performance of the system by tuning the components, or perhaps by adding additional hardware capacity

Abridged

20.1 Why is it complex

It is sometimes inaccurately assumed that performance testing is just as complex as some other test types, and thus any tester without any performance test experience can easily conduct performance testing.

Although each type of testing has its unique challenges, the level of complexity that a performance test requires tends to be much higher. Here are some specific reasons why performance testing can be complex and challenging:

- Performance testing generally tends to be towards the end of the site development and release cycle, as shown in Figure 20-1. Delays in earlier stages of the development cycle result in less time for conducting performance testing, thus making the task even more challenging. In addition to allocating sufficient time and a *contingency buffer*, every attempt should be made to start performance testing as early in the development cycle as possible.
- Performance testing may not be possible on the actual production hardware and equivalent requirements (or targets for the test results) may be unavailable or unclear for new or custom features. In such a case a correlation formula will need to be developed by some trial and error method to translate results obtained from the test environment to the production environment.
- Performance testing requires a diverse technical skill set. Performance testers should posses system administration skills for various platforms, software products, and technologies.
- Performance testing requires troubleshooting skills. Defects found during performance testing are often severe, yet difficult to isolate and fix. Often such defects will be difficult to reproduce as well.
- Performance testing usually requires building automation scripts and generating large amounts of realistic data before any testing may be started. You may even need to create utilities for generating test data.
- The amount of skills one requires to perform a system test takes time to acquire.

Archive

20.2 Why it is important

Even though performance testing can be complex, the investment of both sufficient time and resources in this effort is well worth the expense to:

- Avoid losing revenue and customers.

  Performance defects tend to be runtime defects. Runtime defects that are discovered on live production environments impact the revenue generation capability of the site and, as such, have a direct impact on the bottom line of a business. In addition to impacting immediate revenue generation, a runtime defect may impact future revenue generation capability by discouraging customers from revisiting the site and repeating their business.

  Performance runtime issues are generally not trivial to address, so you do not want to discover them on your live production environment. Runtime defects also tend to be complex—taking a long time to be fixed and impacting a large number of users or shoppers.

- Drive out defects.

  Performance testing often reveals many serious defects in the areas of database deadlocks, code deadlocks, code limitations, corruption of data, and the unavailability of service.

- Ensure concurrency.

  Often there are problems that occur when two threads or processes are trying to do something at the same time. These problems, referred to as timing windows, are often difficult to reproduce with normal load on the site. Stress testing helps to find such problems, as it increases the chance for encountering such timing windows.

- Ensure future quality.

  Since software development is an iterative process, understanding the load limitations of the current product allows for the improvement of load capabilities for the future. This is very important, since it often finds, in advance, scaling problems that usually lead to expensive rework and redesign of the code. Understanding the problems encountered allows ample time for the planning of such redesigns.

- Do an iterative comparison.

  This allows us to understand how various pieces of functionality affect concurrency and throughput. WebSphere Commerce applications can be complex and, as you add more complexity to them, the performance problems can be more elusive. By starting from simpler scenarios that use a minimal set of functions and adding more functionality with each iteration or test

Archive

scenario, we can understand how each new function affects the overall performance.

20.3 Overall site development life cycle

When you set out to develop your site, or an enhancement to it, a number of process are engaged, depending on your business requirements.

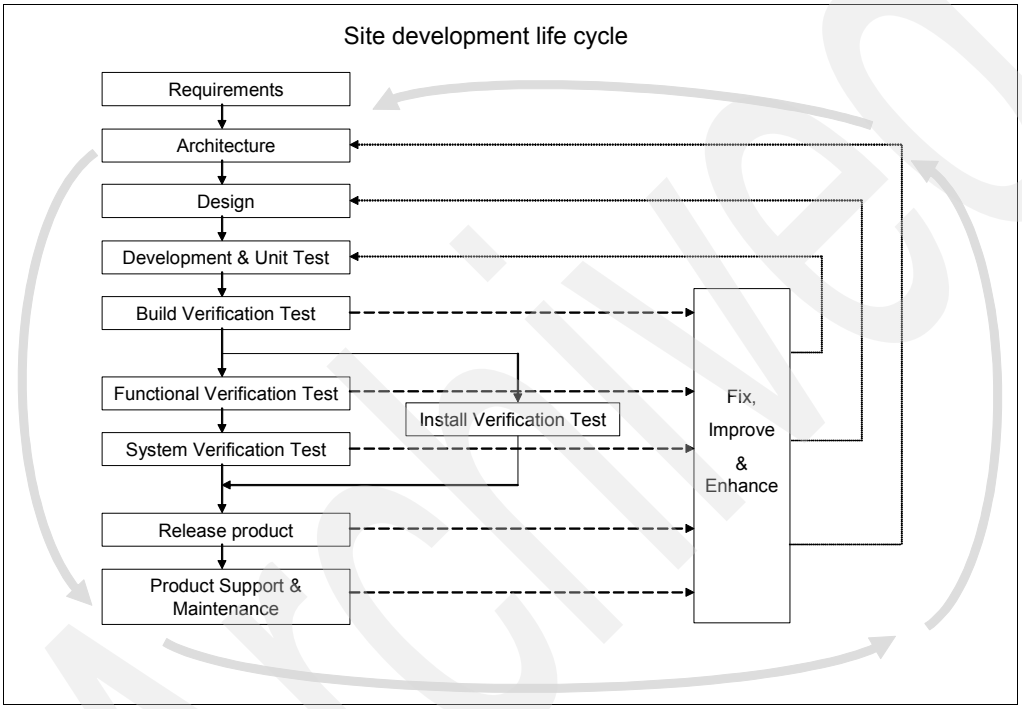


Figure 20-1 Site development life cycle

Figure 20-1 is self-explanatory. The key points that we want to highlight here are:

- Many of the activities are listed in a sequential format. However, there are still many possible ways to introduce parallel development for individual components. For example, independent components (or custom features) may be function and system (performance) tested as soon as they are available, instead of waiting for the development of the entire site to finish.
- Performance testing is a type of system test. As discussed above, the objective of performance testing is not to ensure that the function works accurately. Function testing should be done prior to performance testing. Using performance testing to find functional defects can be very expensive,

Archived

since setting up a performance test environment and executing a performance test are much more involved activities then running a single user functional test.

We recommend that you test performance of relatively independent components individually. However, if you take that approach, then ensure that you plan on some level of integration or end-to-end testing so that these features are tested concurrently in a more realistic end-to-end scenario, prior to taking them live.

Performance test life cycle

Performance testing tasks usually involve, but are not limited to, the cycle of activities shown in Figure 20-2.

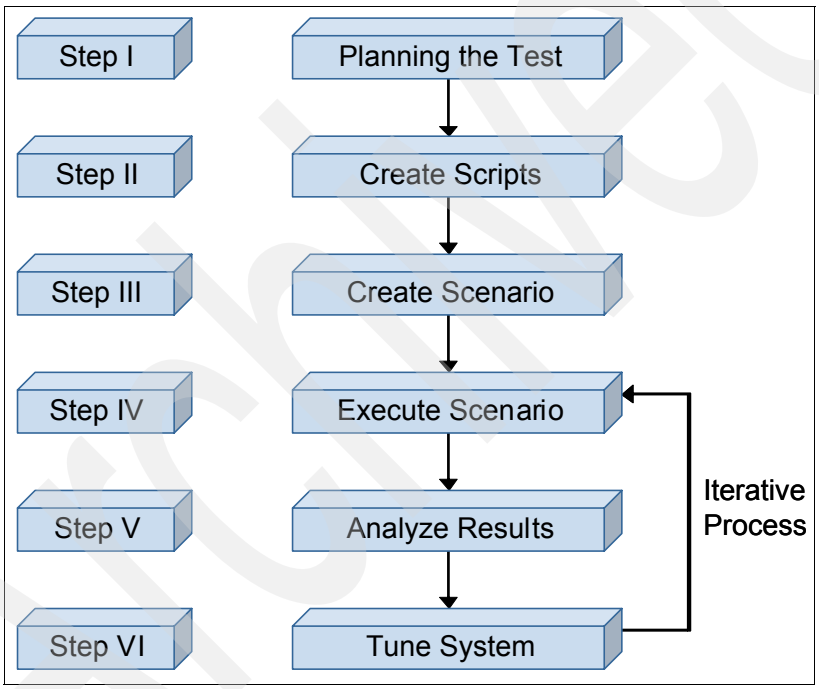


Figure 20-2 Performance test life cycle

Here is brief description of what is involved at each stage of performance testing:

1. Plan the test.
   a. Review and gather business requirements.
   b. Review design documents:

Archived

Ensure that the design is in line with the business requirements and contains all the necessary information that may be required to create test scenarios.

c. Write test plan.

d. Have the test plan viewed and approved by the appropriate stakeholders, such as business analysts or business owners.

2. Create scripts or automation tools.

   Develop, customize, or maintain automation tools (for example, simulators) or automation scripts (for example, that simulate test cases).

3. Create scenarios.

   Create scenarios as per the approved test plan.

4. Execute scenario:

   a. Prepare a test environment, if it was not done already for you:

      - Set up the WebSphere Commerce environment against which the test will be executed.
      - Set up the test environment. This will simulate the test client environment,on which the test scripts will be executed.

   b. Execute the test.

   c. Collect the logs and test the results for analysis as well as safe keeping for future reference.

5. Analyze and report the results.

   a. Analyze the test results.

   b. Analyze all defects or performance issues and concerns.

6. (Optional) Performance tune.

   If the test results are not acceptable then we performance tune the system and re-execute the scenario. For example, the execution of the scenarios during the performance test will show which requests have poor response times. Although not pictured in Figure 20-2, profiling fits into this iterative process. Profiling the slow requests can help determine the root cause of their poor response times. Fore more information about profiling refer to Chapter 14, “Profiling” on page 309.

Archived

20.4 Typical performance characteristics of a WebSphere Commerce site

Figure 20-3 shows the performance characteristics of a typical WebSphere Commerce site. At a high level, the trends depicted here can arguably apply to most well-performing eCommerce sites, although the scales of these graphs and some other details will differ.

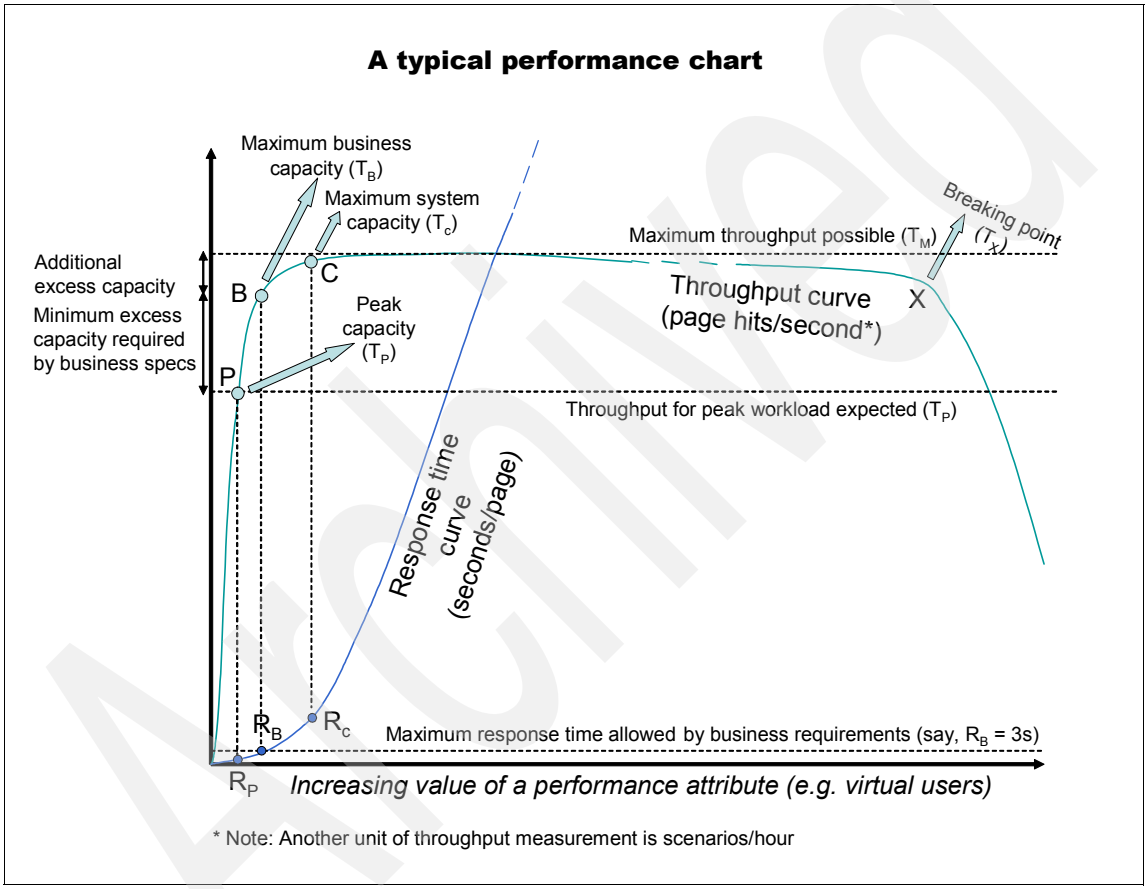


Figure 20-3 A typical performance chart for WebSphere Commerce site (not to scale)

For the sake of our discussion, we assume the workload to be WebSphere Commerce application tier intensive and that the other tiers are able to handle it with relative ease. For example, we assume that the database server gets only a fraction of the overall workload due to a high Dynamic Cache hit ratio and that it is able to handle increasing workload without consuming system resources, such

Archived

as CPU, with as much intensity as WebSphere Commerce tier would. For such a system, the typical performance behavior of a site should be as depicted in Figure 20-3. Figure 20-3 will be referred throughout the remaining portion of this chapter.

WebSphere Commerce benefits from the robust underlying scalability provided by WebSphere Application Server. In our testing we find that once the site reaches its maximum throughput (T_M), it is able to maintain that throughput even with many-folds increase in workload, with only slight degradation to throughput. That is, the sites held well under stress. However, eventually, at a certain breaking-point, T_X, it is not able to maintain maximum throughput, and then throughput comes down and suddenly we start seeing all sorts of concurrency and time-out errors.

Although maximum throughput (T_M) may be maintained for a much larger volume of the workload, maximum business capacity (T_B) may not. A system's maximum business capacity is defined by its business requirements. Specifically, in Figure 20-3, any excess capacity (for example, throughput) that may be available past the limit defined by maximum business requirements (for example, maximum response time allowed, which is R_B in our example) cannot be counted towards business capacity available.

Building your site capacity to account for High Availability

Usually, the more the excess capacity, the higher the availability. The excess capacity would usually translate to how many redundant components may be allowed to fail without causing any noticeable drop in user experience and thus still meeting the business capacity requirements. As always, cost of redundant components required to maintain excess capacity and business requirements dictate what level of High Availability could be built into a site.

Expected peak capacity versus maximum business capacity

The maximum business capacity (T_B) should also be higher than the peak capacity (T_P) expected. The expected peak capacity is based on the peak workload expected by the site. Any excess capacity, for example, C_{B-P} (for example, excess throughput in the above graph, $T_B - T_P$), over the peak capacity contributes towards High Availability of the site.

Usually, if your peak workload drives your available Web server CPU to >=50%, your available database CPU >=60%, and the following day is a public holiday in in a country and CPU >= 70% of its capacity, then it is time to scale your system. That is, your business capacity requirements for CPU consumption are 50%, 60%, and 70% for your Web server, database, machines where the following day is a public holiday in a country, respectively.

Maximum business capacity versus maximum system capacity

Notice that the theoretical maximum system capacity is different from the business capacity that is defined by the business requirements. These two terms are often confused with one another. The maximum business capacity should not equate the maximum system capacity. Otherwise, any good news (for example, a business underestimating the site workload) may turn into a nightmare.

Theoretical maximum system capacity does *not* occur at maximum throughput (T_M). Instead, it occurs for that value of throughput and response time that correspond to the maximum differential of the throughput to response time:

$$C = \left(\frac{dT}{dR}\right)_{max}$$

The maximum business capacity is a special case of this equation. It can be derived by applying the constraints as defined by business requirements. For example, in our case it is the maximum response time allowed of (for example, $R = R_B$):

$$C' = \left(\frac{dT}{dR}\right)_{R = R_B}$$

The difference between the maximum system capacity (C) and the business capacity required (C') (for example, (C - C')), is the second buffer or level of contingency over the excess capacity as defined in the previous section (C_{B-P}).

Maximum system capacity (C), just like the required business capacity (C'), is usually calculated experimentally, by a trial-and-error process of running various load tests and observing where the change in slope of (dT/dR) occurs.

20.5 Types of performance tests for WebSphere Commerce

In the most simple of terms, the difference between various types of load and performance testing is the workload applied and the duration of the test. Here we discuss how different types of workloads and different test durations can be used to inspect a system's performance characteristics.

20.5.1 Stress testing

Stress testing is a form of performance or load testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, usually at or beyond the peak workload expected, often to a breaking point, in order to observe the results, and to find breaking points or bottlenecks. For example, a Web server may be stress tested using scripts, bots, and various denial-of-service tools to observe the performance of a Web site during peak loads.

The goal is to incorporate various workloads that will place abnormal burdens or specific areas of concern. These areas include CPU, memory, MQ Queues, I/O, threads, connections, JVM, and content.

Another aspect of stress testing is *recovery testing*. Here the goal is to verify the ability of the system to recover from varying degrees of failure. Such testing is conducted to evaluate a system or component at or beyond the limits of its specified business requirements. Stress testing methodology includes processes and technologies for testing system architecture to determine the maximum sustainable load for the site's hardware and software applications. The process defines how a software system is put under heavy load and demanding conditions in an attempt to make it fail. Such testing attempts to cause failures involving how the system behaves under extreme but valid conditions (for example, extreme utilization, insufficient memory, inadequate hardware, and dependency on over-utilized shared resources).

Stress testing determines how the system degrades and eventually fails as conditions become extreme (for example, the number of simultaneous users increases, queries that return the entire contents of a database, queries with an extreme number of restrictions, and an entry at the maximum amount of data in a field). This technique measures whether the application environment is properly configured to handle expected, or potentially unexpected, high transaction volumes. Ideally, stress testing emulates the maximum system capacity that the application can support before causing a system outage or disruption.

20.5.2 Scalability testing

As mentioned earlier, the goal of stress testing is to identify peak load and stress conditions at which the program will fail to handle processing loads within required timespans. Whereas stress testing attempts to identify abnormal behaviors when the system is under extreme load conditions, the goal of scalability testing is to identify if the system scales as workload, data, or store complexity increases. That is, the goal is to build a graph (for example, data size versus response time) that will allow us to predict with reasonable accuracy the

performance impact of site growth and possibly predict capacity milestones of peak capacity (P) and business capacity (B), as shown in Figure 20-3.

Examples of abnormal and non-scalable behavior are:

- Non-linear increase in response times
- Non-linear decrease in throughput
- Data corruption
- Database or code deadlocks
- Cookie resets
- System crashes

Scalability testing includes validation of site scalability from the following aspects:

- Interface (for example, GUI) scalability

  This allows us to determine that size of data does not impact negatively the look and feel of the product. If you do not have access to a tool to test interface scalability then you can consider taking running a 100+1 test, as explained at 20.5.5, “100% + 1 testing” on page 467.

- Performance impact of data growth

  As time goes on, WebSphere Commerce users will create more and more data and the existing data will *age* (for example, data that needs to be archived or cleaned/removed from the database). We need to determine the impact that this will have on the performance of their system.

- Capacity scalability

  Another major component of scalability is capacity scalability. We need to be able to understand and predict how different or additional hardware and topologies influence throughput and response times when using large sets of data.

Scalability testing is achieved by primarily scaling the workload, data scalability, and store complexity, as discussed in the following sections.

Data scalability

Data scalability testing is defined as any testing that scales any part of the system that grows as the customer uses the system. This is primarily the WebSphere Commerce database (for example, the number of users, number of orders, and so on) and the various log files on your system. Since the plays a significant role towards WebSphere Commerce site performance, it is crucial to identifying the impact of increasing or aging data on site performance by running different scenarios for increasing data sets.

Archived

Workload scalability

Workload scalability refers to increasing the number of operations leading to increased workload to the WebSphere Commerce application. The increase in number of operations can be in the total number of operations or operations of a certain kind. For example, this may refer to increasing the number of virtual users while keeping the workload distribution untouched, or this may refer to keeping the same number of virtual users but increasing the number operation of certain kinds, say, changing the browse::buy ratio from 95::5 to 75::25.

Store complexity scalability

Usually, these types of test cases would not apply to all sites. Nonetheless, this is an important contributor to the site performance.

Store complexity scalability refers to increasing the complexity of store pages, for example, increasing the size of your JSPs significantly, adding complexity to your JSPs, adding a lot of eSpots, or increasing the number of JSPs (for example, having thousands of eSites that do not inherit JSPs from a presentation asset store), and so on. Note that the last eSites example is just for the purpose of elucidating the point. If there is such an eSite model then it would be, most likely, a bad design.

Special consideration should be given to testing Web2.0-based stores as the increased usability of Web 2.0 widgets comes at the cost of increased complexity of the underlying store page.

20.5.3 Soak, endurance, or reliability testing

A soak, endurance, reliability test is a test that runs for a long time (usually for days) under heavy load. It is generally used to detect memory leaks, which may be very slow and take days to impact the system as well as concurrency errors that may otherwise elude us, for example:

- Any (overnight) WebSphere Commerce related jobs, for example, WebSphere Commerce scheduler jobs, data exports or imports, and so on
- Any (overnight) non-WebSphere Commerce scheduled jobs, for example, security checks or audits, backups, and so on

Archived

Reliability testing is also a good test vehicle to test end-to-end performance of a fully integrated system. WebSphere Commerce sites can be very extensive or complex depending on your business requirements, integration of sub-components, and integration with other products. Many of these sub-components and products may have been tested separately and by different testers within or outside of your organization. Thus, it is very important for us to test the reliability of the entire system, as the interaction in this complex environment could be the source of many reliability problems.

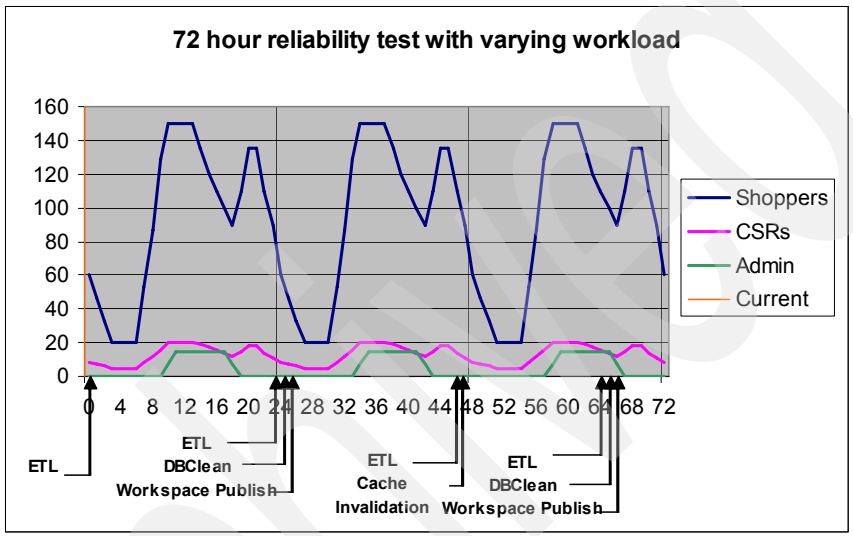


Figure 20-4 72 hour (3-day) reliability test with varying workload

In Figure 20-4, the reliability workload chart, we can see daily repetitive peak periods and WebSphere Commerce system maintenance activities. ETL refers to the *extraction, transformation, and load utilities*, for example, for your offline data mining activities. Cache invalidation may be required to clear the cache so that cache can be rebuilt based on the changes loaded into the database. Workspace publish refers to data published as a result of stage propagation or a publish request from the WebSphere Commerce Authoring environment.

20.5.4 Stress-endurance test

The main variable in the case of reliability testing is the period of time (the duration) for which the test is executed. However, we find that running the stress test for a very long duration at levels much higher than maximum system capacity (refer to 20.4, “Typical performance characteristics of a WebSphere Commerce site” on page 460) but below the breaking-point is able to show small and otherwise elusive memory leaks as well as any slow throughput degradation

much quicker than a reliability test. It is almost like looking at these bottlenecks through a magnifying glass. However, the trade-off is that the response times of such a test cannot be very useful as a reliability test since the stress endurance test will be run at the maximum system capacity, beyond the maximum business capacity.

20.5.5 100% + 1 testing

The 100% + 1 test strategy is useful when an automated test solution that tests the client GUI directly does not exist, but there is an automation tool that can simulate the client's interactions with the server.

The 100% refers to running the 100% automation bucket that runs against the server directly simulating the GUI interactions but without GUI (since the assumption here is that you do not have a tool to test the GUI in an automated mechanism). The ” refers to the 1 (or more) testers manually running the test case using the client GUI.

The key benefit of this approach versus the single user functional test is that you test the client GUI behavior while the server is under load. Using this methodology:

- Any server-side defect could be found since the 100% automation bucket is still being executed.
- Any client GUI-side defect that always occurs (either due to basic functional issues or load on the site) could be found since the manual test execution will always encounter it.
- Any client GUI-side defect that does not always occur (say, it occurs on 10% of the times) may not always be found since the manual test execution would need to be done 10 times to detect the defect once, and this may not be enough to generate a detectable error rate, even when that happens.

Although this process is not foolproof, it is probably the next best alternative to using an automation tool that works with the client GUI.

20.5.6 Capacity testing

The objective of capacity testing is to determine the maximum business capacity (C_B) that a system can sustain without exceeding performance requirements as defined by the business. For example, there may be a limit to the maximum response time and maximum CPU utilization limits, and so on.

During capacity testing, the system is tested under increasing but realistic load to find the point where a system resource fails to meet business requirements, for

example, the point where it causes unacceptable response time, CPU utilization, or failures. Sufficient performance data at various loads allows the prediction of application performance under different operating conditions, as well as prediction of the maximum number of concurrent users. In this sense capacity testing is closely related to the scalability testing.

Capacity testing can highlight scalability issues involving distribution and load-balancing mechanisms. Increasing the workload will identify the maximum threshold of linear operation. Furthermore, capacity testing provides insight into how components interact with each other.

For a more detailed description about the various ways of calculating capacity of your site refer to 20.4, “Typical performance characteristics of a WebSphere Commerce site” on page 460.

Saturation point testing

The objective of saturation point testing is to determine the maximum system capacity (C) as defined in “Maximum business capacity versus maximum system capacity” on page 462.

20.5.7 Performance regression testing

In order to ensure viable release-to-release comparison, the tests from the previous release should be executed on the new release, but on the same hardware as in previous releases. When this is not possible, for example, if the hardware is different, then a new baseline will need to be established by executing the tests using the new hardware with the previous release.

20.5.8 High Availability testing

High Availability testing is a unique performance testing. Sometimes we can say that it is a combination of most types of performance testing, but they are customized performance tests. Based on the requirements about what the components with High Availability features enabled need to be tested, different types of performance testing can be driven to evaluate the functionalities/usabilities of High Availability components. To WebSphere Commerce site testing, the major purpose of the High Availability testing can be summarized as following:

- To evaluate whether the High Availability utilities are working and stable in a WebSphere Commerce environment
- To evaluate the performance impact from High Availability enabled utilities to normal operations of the WebSphere Commerce site. Sometimes it might

result in performance degradation. Generally, the impact can be classified into different terms, such as throughput degradation, CPU utilization, memory utilization, network latency, I/O wait, data synchronization, and so on.

- To evaluate the capability of High Availability in the WebSphere Commerce site to handle different types of planned and unplanned outage. Different types of outages can be designed, mostly based on the requirements from customers, such as unplanned power outage, planned software maintenance/upgrade, unplanned process outage, unplanned network outage, and so on. Most of times, such types of outage should be simulated by automatic script or manually. It is up to which type of High Availability you want to achieve in the WebSphere Commerce Web site, and the frequency with which the outage will happen in a realistic production site.
- To evaluate the High Availability feeling from the customers' point of view. With High Availability utilities plugged into a WebSphere Commerce site, if any outage planned or unplanned happens, there should be no significant performance impact to the site, at least, from the customers' view, the outage should not be noticeable, or it should be eliminated as much as possibility.

High Availability is a topic that becomes more common than before, so that High Availability testing is important for customers to select an appropriate solution to provide High Availability to their Commerce site to achieve high customer satisfaction and stable revenue growth.

Archive

Archived

21

Designing a test plan

For any testing, we should create a test plan. Creating a test plan for performance testing is extremely critical. This is due to the fact that in a performance test there are many *knobs* that may be tweaked or that may need to be tweaked. These knobs are all the variables that go into creating a test scenario. This may include your environment settings, database or application server confirmation, scenario flow, and so on.

All these knobs must be clearly documented and communicated to the appropriate stakeholders of your business, including the other testers and administrators, architects, business analysts, business owners, and so on. If these knobs are not set to correct values then it will very likely invalidate your test case execution.

In this chapter, we deal with many of the key knobs, and how to track them as well as how they may be communicated to your test plan reviewers and stakeholders.

© Copyright IBM Corp. 2008. All rights reserved.

21.1 Define scope and requirements of new design

It is important to gather the business requirements and their scope not only to ensure that we test them but also to review the design and ensure that the design is in accordance with the requirements. That is, the job of performance testers should not start after the site, component, or feature is developed, but at a time while it is being designed.

Identify the scope of the business requirements deriving the new design or a change to the design. For example:

- Is there an additional feature that is being developed or a new site?
- Who is the consumer of the this new feature (for example, administrators or shoppers, and so on)?
- Which environment will host this feature—the staging environment, production environment, or is it a just a productivity tool for the development environment?
- When will this feature be used in relation to the existing peak workload?
- What workload will this feature be a part of (for example, is this part of *browse* or *buy*)?
- What is the peak workload and, if this is an additional feature, then what additional workload will this feature generate?
- What are the throughput and response time requirements for the new workload?
- What is the data retention policy and purging frequency, and so on?

21.2 Define target environment

The ideal test environment is the one that mimics the production environment. However, it is not always financially viable, and as such your test environment may be different from your production environment. In such a case you need to convert your requirements to be applicable to your target test environment, and provide justification for the validity of your conversion factor or algorithm. This conversion factor or algorithm may be non-trivial, depending on the complexity of your site.

We recommend that you test with a database copy of your production environment. If, however, that is not feasible, or if you are developing a new site, then you also need to consider the amount of data that you require for executing

Archived

your test. For example, what should the catalog size be? The number of products, items, attributes, the number of users, and so on.

21.3 Define scenario and workload distribution

Depending on the similarity of your target test site to the production environment, you would need to scale the workload to suit your test environment. In addition to defining the easier aspects of the workload, such as the number of users (shoppers, administrators, and so on), scheduled jobs, and response time and throughput expectations, you also need to define scenarios based on your site flow, their relative weight, such as the browse-buy ratio, and their relative length, such as how many products a shopper will go through or how many search queries will be executed before placing an order. An accurate representation of the site flow, scenario weight, and length is important, as it impacts any performance results.

Relative weights can be defined as a percentage of total site traffic. One common and easy to make mistake is to not have the total traffic add up to 100%.

Archive

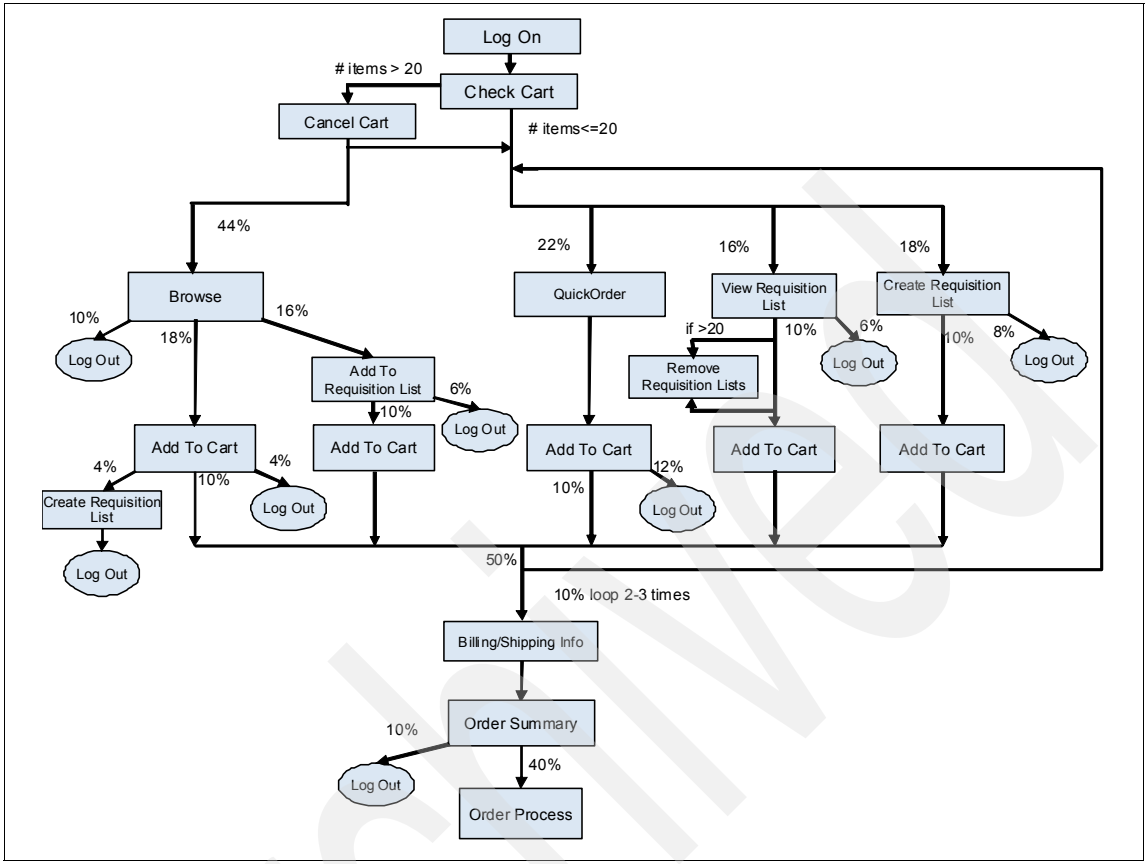


Figure 21-1 A sample B2B scenario with workload distribution

In addition to the workload generated by human intervention, be that from the shoppers or from the various administrators, we also need to define any other modes of content update such as the backend updates, scheduled jobs, and so on.

Archive

21.4 Define test cases

Throughout this book we have referred to *scenario* or *test scenario* without distinguishing it from a test case. As defined in 1.2.3, “Scenario” on page 8, a scenario is a summary of a sequence of events, whereas test cases are built from given scenarios, and they are the actual sequence of steps executed as part of the test. Thus, a test case includes details such as the data, objects, tools, user interface, clickable actions, user information, and so on.

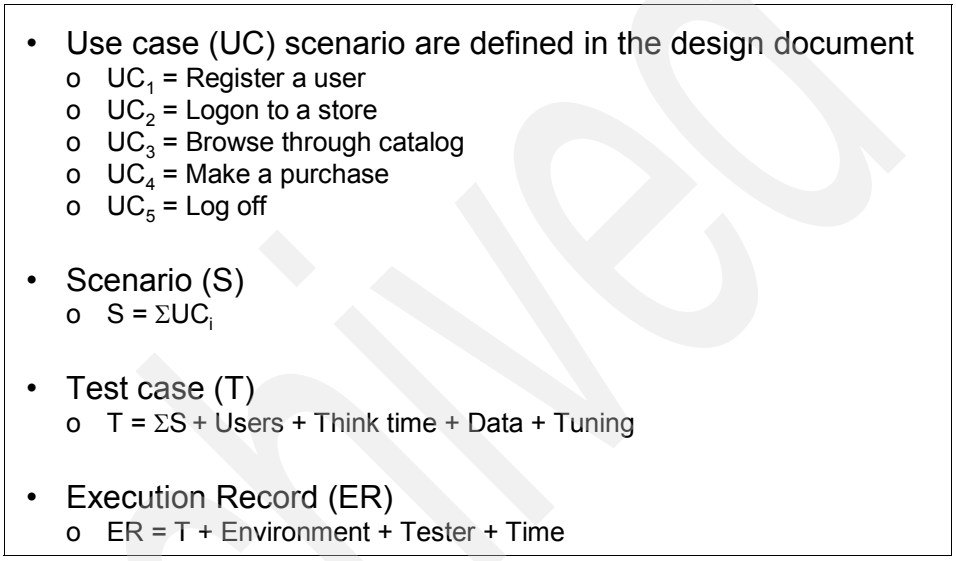


Figure 21-2 Scenario, test case, and execution record defined

In a test plan, we build scenarios from the *use cases* defined in a design document, and from scenarios we build test cases.

A scenario in itself is just a scenario. It is not a *functional* or *performance* scenario. On the contrary, a scenario must be *functional* and free from defects (in a single user functional test case) before it can be performance tested.

A test case, on the other hand, can be a functional or performance test case. A test case can further be a stress test case, soak test case, and so on.

A test case includes all the test attributes and control attributes (as defined in Chapter 23, “Applying performance testing to WebSphere Commerce” on page 507), except for the hardware.

An instance of a test case is an execution record. The hardware on which a test case is run, the tester who executes it, and at what time the test case is executed

Archived

are all part of this record. Although the time of the test may sound trivial at first glance, your network traffic may have a different pattern depending on what time of day and day of week it is run.

21.5 Maintaining a well-defined test plan

The primary purpose of maintaining a test plan document is to ensure that there are no miscommunications as to the objective and substance of testing. This document also serves as a historic record upon which future testing can be based, such as for regression testing purposes and cross-reference purposes.

All the information discussed in this section belongs to a test plan. If any of the information is not directly available, then valid assumptions can be made as long as the accompanying justifications are documented as well. Along the same lines, any exclusions from the test plan should also be documented along with both the accompanying justifications and the expected risk of such an exclusion to the overall performance quality of the site.

Like any other project documentation, a lot of other useful information should go in to this document as well (for example, a list of dependencies for performance testing to be conducted and successful, as well as entry/exit criterion).

Entry and exit criterion refer to conditions for starting and concluding testing. They assume key significance if performance testing of various components or features happens in parallel, since this would allow efficient utilization of test resources by testing those components first, which meet the entry criterion first.

Last but not least, it is also very useful to include your test strategy as part of the test plan or as a separate document. The test strategy should include any information that would be critical in allowing the test plan reviewers to ascertain whether the testing will be valid and realistic. For example, is your the tooling capable of emulating a realistic site traffic and user experience, or is there additional configuration or work required to ensure that the testing would be realistic?

After the test plan is reviewed and approved by the appropriate stakeholders, it should be put under change control, and any subsequent change to this document should go to the appropriate level of the review process.

Archive

22

Performance test tools

Rather than manually stressing an application, it is more reasonable to use test tools to test an application efficiently. In this chapter, we provide an overview of a small array of performance test tools, which includes:

- Rational Performance Tester
- SilkPerformer
- Page detailer

Archived

© Copyright IBM Corp. 2008. All rights reserved.

22.1 Test tools introduction

Deploying applications that perform and scale in an acceptable manner is not an accidental occurrence. Producing high performance software requires that you include several rounds of stress testing during the development cycle. You can use one or more of the many open source or commercial stress testing tools to automate the execution of your stress tests.

The primary purpose of stress testing tools is to discover under what conditions your application's performance becomes unacceptable. You do this by changing the application inputs to place a heavier and heavier load on the application and measuring how performance changes with the variation in those inputs. This activity is also called load testing. However, load testing usually describes a very specific type of stress testing: increasing the number of users to stress test your application.

The simplest way to stress test an application is to manually vary the inputs (for example, the number of clients, size of requests, frequency of requests, mix of requests) and then chart how the performance varies. If you have many inputs, or a large range of values over which to vary those inputs, you probably need an automated stress testing tool. Moreover, you will want test automation to repeat test runs following environmental or application-specific changes once you uncover an issue.

If you are testing manually, it can be difficult to accurately reproduce an identical set of tests across multiple test executions. When it comes to having multiple users testing your application, it is almost impossible to run manual tests consistently, and it can be very difficult to scale up the number of users testing the application.

Today, there is no generic, one-size-fits-all stress testing tool. Every application differs in what inputs it takes and how it executes them. Java and WebSphere-based Web applications generally receive requests from clients via the HTTP protocol. There are many stress testing tools that can simulate user activity over HTTP in a controlled and reproducible manner.

Archive

22.1.1 How to select test tool

With so many stress testing tools available today, how can you choose the one that is most appropriate for your application? Some of the points to consider when evaluating stress testing tools include:

- Client interaction

  The stress testing tool must be able to handle the features and protocols that your application uses.

- Simulation of multiple clients

  This is the most basic functionality of a stress testing tool.

- Scripted execution with the ability to edit scripts

  If you cannot script the interaction between the client and the server, then you cannot handle anything except the most simple client requests. The ability to edit the scripts is essential. Minor changes should not require you to go through the process of regenerating a script.

- Session support

  If a stress testing tool does not support sessions or cookies, it is not very useful, and may not be able to stress test Java and WebSphere applications.

- Configurable numbers of users

  The stress testing tool should let you specify how many simulated users are running each script or set of tasks, including allowing you to vary the number of simulated users over time. Many stress testing tools enable you to start with a small number of users and ramp up slowly to a higher numbers of users.

- Reporting: success, errors, and failures

  The tool that you choose must have a defined way to identify a successful interaction, as well as failure and error conditions. An error might be getting no Web page back at all, whereas a failure might be getting the wrong data back on the page.

- Page display and playback

  A useful feature in many stress testing tools is the capability to inspect some of the pages that are being sent to the simulated users or to replay entire test scripts. You can then be confident that the stress test is functioning as you expect.

- Exporting test results

  After running a stress test, you may want to be able to analyze the test results using various tools that are external to the stress testing tool, including spreadsheets and custom analysis scripts. Most stress testing tools include

Archived

extensive built-in analysis functions, but being able to export the data gives you more flexibility to analyze and catalog the data in arbitrary ways.

- Think time

  Real-world users do not request one Web page immediately after another. There are generally delays between viewing one Web page and the next. The term *think time* is the standard way of expressing the addition of a delay into a test script to more realistically simulate user behavior. Many stress testing tools support randomly generated think times based on a statistical distribution.

- Variable data

  Live users do not work with the same set of data on each interaction with your application. During a stress test, this should also be true of your simulated users. It is easier to make your simulated users appear to be working with varied data if the stress testing tool supports data input from lists, files, or a database.

- Script recording

  Rather than writing scripts, it is much easier to manually run through a session with your browser and have that session recorded for later editing. Most stress testing tools include provisions for capturing manual interaction with your application.

- Analysis tools

  Measuring performance is only half the story. The other, and perhaps more important, half of stress testing is analyzing the performance data. The type of analysis tools and degree of detailed analysis you can perform depend directly on what analysis tools are supported by the tool that you select. Therefore, evaluate this support in the tools that you are considering carefully.

- Load distribution

  Your deployed application may well need to support hundreds of concurrent users once in production. How can you simulate this level of traffic in a stress testing environment? A typical workstation running a stress testing tool will likely begin bottlenecking once approximately 200 virtual users are running. To simulate a greater number of users, you can distribute the stress testing load across multiple workstations. Many of the available stress testing tools support distribution of load, and you will certainly want this feature for large-scale stress testing.

- Measuring server-side statistics

  The basic stress testing tool measurement is client-based response times from client/server interactions. However, you may also want to gather other statistics, such as the CPU utilization or page faulting rates. With this

server-side data, you can then do useful things like view client response times in the context of server load and throughput statistics.

22.1.2 Performance test tools classification

We can separate the major performance test tools into three categories:

- Commercial test tools
  - IBM Rational Performance Tester
  - Seague SilkPerformer
  - Mercury Interactive Loadrunner
  - Radview WebLoad
  - CompuWare QALoad
- Free test tools
  - Microsoft Web Application Stress Tool
  - Microsoft Application Center Test
- Open source test tools
  - OpenSTA
  - Jmeter
  - Grinder
  - Eclipse Test Performance and Tools Platform (TPTP)

22.2 IBM Rational Performance Tester

IBM Rational Performance Tester 6.1 (RPT 6.1) is a multi-user system performance test product hosted in the Eclipse shell with a Java-based execution engine. The focus of IBM Rational Performance Tester 6.1 is multi-user testing of Web applications.

IBM Rational Performance Tester is a load and performance testing solution for teams concerned about the scalability of their Web-based applications. Combining ease-of-use features with flexibility, Rational Performance Tester simplifies the test creation, execution, and data analysis to help teams ensure the ability of their applications to accommodate required user loads before the applications are deployed.

Archived

22.2.1 Architecture of Rational Performance Tester

Figure 22-1 shows the relationship between Rational Performance Tester and the open source test solution driven by IBM and some other major sponsors.

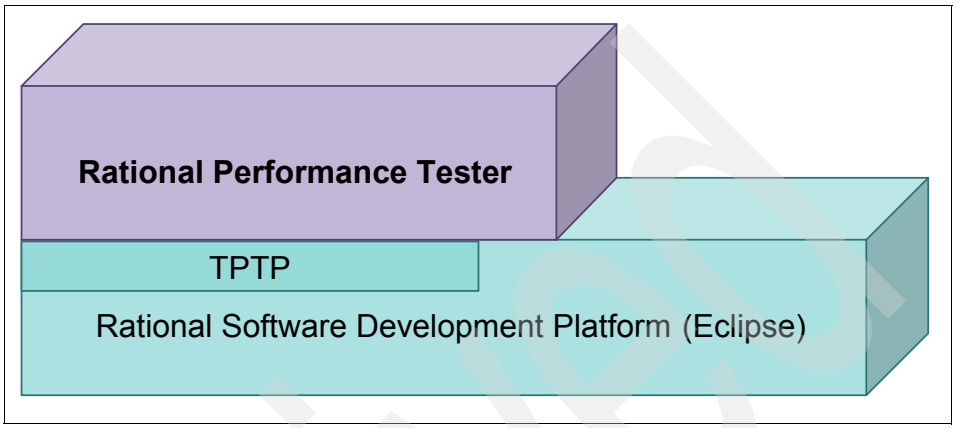


Figure 22-1 Rational Performance Tester Architecture

Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools, and runtimes for building, deploying, and managing software across the life cycle. The Eclipse Foundation is a not-for-profit, member-supported corporation that hosts the Eclipse projects and helps cultivate both an open source community and an ecosystem of complementary products and services.

TPTP is a project in the Eclipse community that provides powerful frameworks and services for an open platform upon which developers build unique test and performance tools—both open source and commercial—that easily integrate with Eclipse and other tools and address the entire test and performance life cycle, from developer testing through production monitoring.

The key points about the architecture of Rational Performance Tester are:

- Rational Performance Tester is built as a plug-in to the Rational Software Development Platform (Eclipse/workbench), which is hosted in the Eclipse shell.
- Rational Performance Tester uses Java-based tests and an execution engine.
- Rational Performance Tester uses many of the TPTP components, such as the HTTP proxy recorder, execution engine, IBM Rational Agent Controller, and so on.
- In RPT, some of the components from TPTP have been customized. Some components are hidden.

For more detailed information about Eclipse or TPTP, you can check out the official Web site for TPTP:

http://www.eclipse.org/tptp/

22.2.2 Features of RPT

The basic features of Rational Performance Tester can be summarized as following:

- HTTP and HTTPS protocol support/capture
- Built-in Verification Points
- Automatic Data Correlation
- Data substitution with Datapools
- Programming extensibility with Java custom code
- Report customization and export capability

22.2.3 Procedure to use RPT to run performance test

The entire life cycle to perform a test by using Rational Performance Tester can be divided to five steps, as discussed in this section.

Create a performance test

In this step, you should create a performance test project and record a HTTP test.

1. Create a performance test project.

   To create a performance test project:

   a. In the test perspective, select **File** → **New** → **Performance Test Project**. The New Performance Test Project dialog box opens.

   b. In the Project Name field, type a name for the project.

   c. In Project contents, select **Use default location**.

   d. Click **Finish**. The performance test project is created and the Create New Test from Recording window appears so that you can record a test now.

   e. Click **Next** to start recording a test, or click **Cancel** to record the test later.

2. Record an HTTP test.

The test creation wizard starts when you record a test. This wizard combines these actions: recording a session with a Web application, generating a test from the recording, and opening the test in the test editor. You can record a test from Internet Explorer or from another browser.

Archived

For recording a test:

1. Click **File** → **New** → **Test from Recording**.
2. In the New window, expand **Test**, click **Test From Recording**, and then click **Next**.
3. In the Create New Test from Recording window, select **Create Test from New Recording**, and then click **Next**.
4. In the list of projects, click the one in which to store this test and related files.

   Note: If you have not yet created a project, the Project Name field displays a default name of testproj, which you can change.

5. In the Recording file name field, type a name for the test, and click **Finish**. The standard Navigator and the Java Package Explorer use this name, with extensions. You will also see this name in the Test Navigator. A progress window opens while your browser starts.
6. In the address box for your browser, type the address of the Web-based application to test and activate the link.

   Note: If you enter the address of a secure Web site (one that starts with https:), your browser might display a security alert. Depending on the security certificate for the site, you might be required to accept a security risk to proceed with the recording.

7. Perform the user tasks that you want to test. While you are recording, follow these guidelines:
   - Wait for each page to load completely. This waiting will not affect performance results, because you can remove extra *waiting time* (think time) when you play back the test.
   - Do not change browser preferences.
8. After you finish performing the user tasks, stop recording. To stop recording, close your browser or click the right side of the Recorder Control view bar. A progress window opens while the test is generated.

Archive

On completion, the Recorder Control view displays the message `Test generation completed`, the Test Navigator lists your test, and the test opens for you to inspect or edit. Figure 22-2 depicts the flow to generate the performance test in RPT.

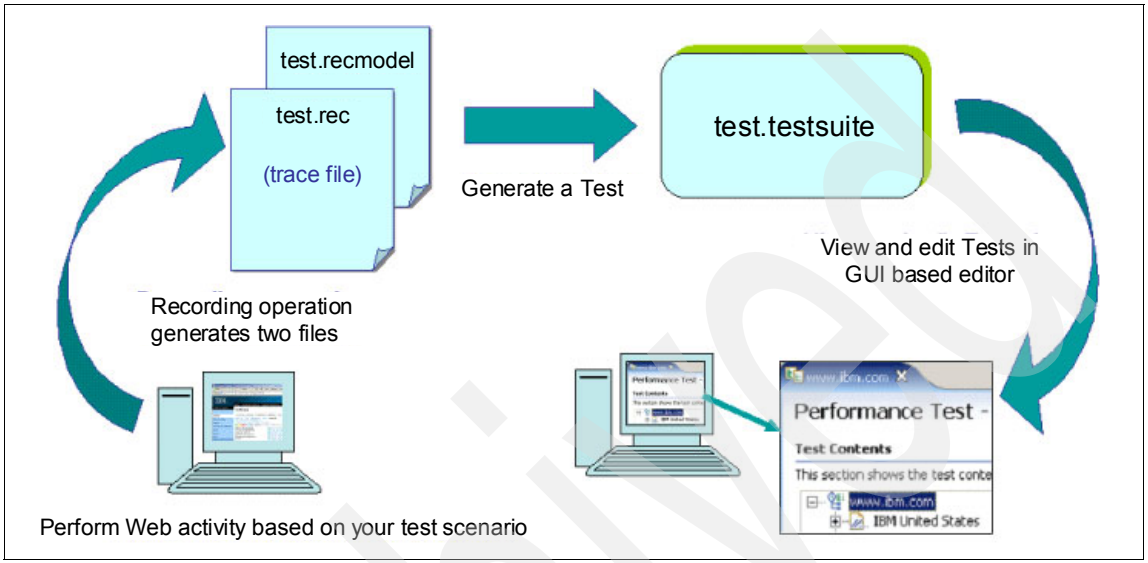


Figure 22-2 Procedure to record and generate a test

Edit a performance test

After you record a test, you can edit it to include datapools (to provide variable data rather than the data that you recorded), verification points (to confirm that the test runs as expected), and correlate (to ensure that returned data is appropriate for the corresponding request). You can also add protocol-specific elements to a test.

With the test editor, you can inspect or customize a test that you recorded.

The test editor lists the HTTP pages for a test, by title. There are two main areas in the test editor window. The area on the left, Test Contents, displays the hierarchy of the HTTP pages for the test. The area on the right, the HTTP tab in Test Element Details, displays specific information about the selected item. The Timeout action and Timeout value settings apply to each page in the test.

Archived

When you expand a test page, you see a list of the requests for the page, in separate folders, with names that are the full Web address minus the initial http://. Some requests are highlighted in yellow. This highlighting indicates that these requests contain one or both of the following types of information:

- A datapool candidate: This is a value, usually one specified by the tester during recording, that the test generator determined is likely to be replaced by values in a datapool.
- Correlated data: These are values in a test, usually one of them in a response and the other in a subsequent request, that the test generator determined needed to be associated in order to ensure correct test playback.

When you expand a request, you see the response data for the request. As shown in the following example, requests can also contain connection data. Because the response is selected in the test contents area, the test element details area displays the response data for this request.

Emulate workloads in a performance test

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

- Group tests under user groups to emulate the actions of different types of users.
- Set the order in which tests run: sequentially, randomly, or in a weighted order.
- Set the number of times each test runs.
- Run tests at a certain rate.
- Run user groups at remote locations.

After you have created a schedule that describes the behavior for your system, you can run this schedule using successive builds of the application being tested or using an increasing number of virtual users. You then analyze the results that are reported.

For creating a schedule:

1. Right-click the project, and then click **New** → **Performance Schedule**.

Archive

2. In the Performance Schedule wizard, type the name of the schedule, and then click **Finish**. A new schedule is displayed with one user group. You can add user groups, tests, and other items to the schedule to emulate a workload.Figure 22-3 is a sample view of the schedule in RPT, which has been added by the test created in previous steps.

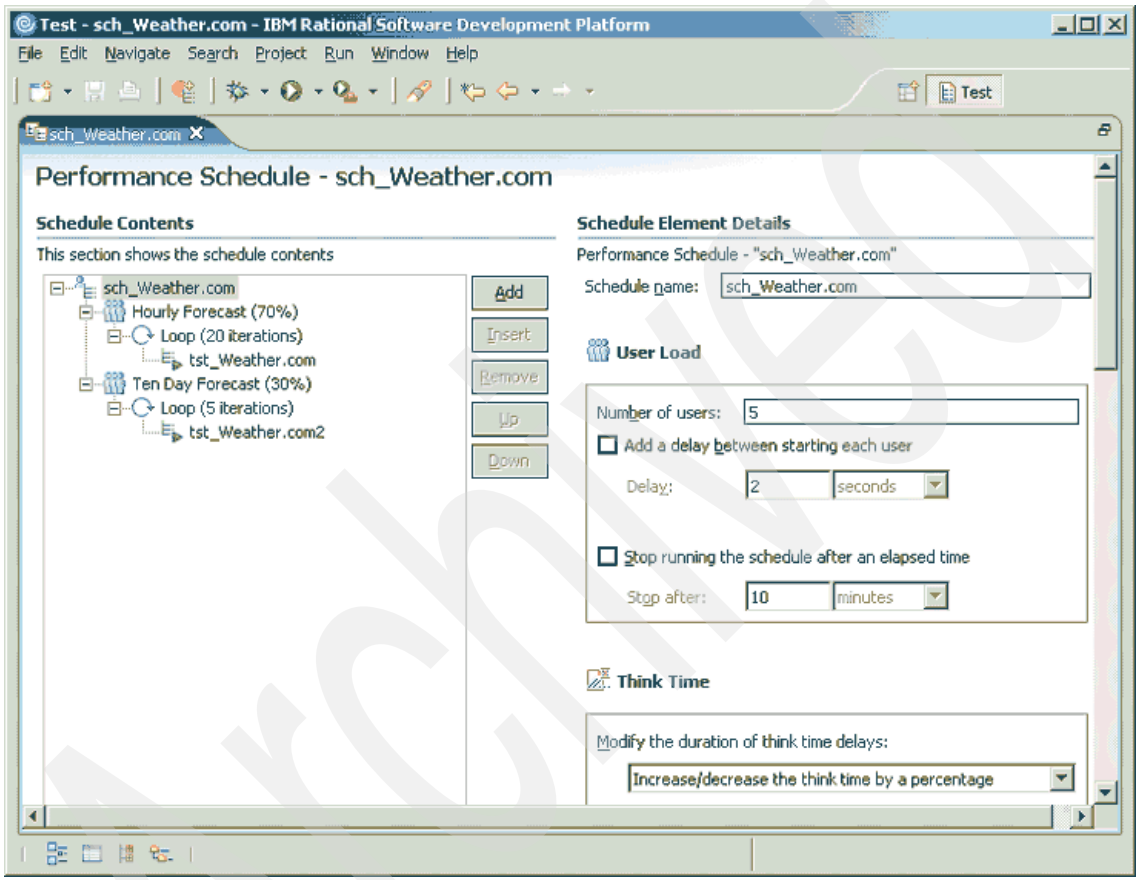


Figure 22-3 RPT schedule view

After creating a schedule, you can implement following operations to the schedule:

- Add elements to a schedule.
- Set think time behavior.
- Run a test at a set rate.
- Run tests in random order.
- Set the number of users that start a run.
- Start users at different times.

- Run a user group at a remote location.
- Emulate network traffic from multiple hosts.
- Set the duration of a run.
- Set the data that the test log collects.
- Control how a schedule stops.
- Set the problem determination level.
- Set the statistics displayed during a run.

Execute a test or schedule

After you have added the user groups, tests, and other items to a schedule, and you are satisfied that it represents a realistic workload, you run the schedule or test.

You can run a test locally with one user or a schedule with a default launch configuration.

For running a schedule or test locally:

1. In the Test Navigator, expand the project until you locate the schedule or test.
2. Right-click the schedule or test, and then click **Run** → **Run Performance Schedule** or **Run** → **Run Performance Test**.

When you run a schedule or test in this way, Performance Tester automatically sets up a simple launch configuration. A test runs on the local computer, with one user. A schedule runs with the user groups and the locations that you have set. However, the execution results have a default name (the same as the schedule or test, with a suffix) and are stored in a default location.

Note: You can configure a schedule or test. A typical reason for setting up a configuration is to control where the execution results are stored.

Analyze the test result

You can evaluate the results that are generated dynamically during a run. You can also regenerate the results for viewing and analysis after a run.

Reports are displayed automatically during a run. When you close a report, it is not saved. However, you can display it again.

For displaying a report after a run has been completed:

1. In the Performance Test Runs view, expand the project until you locate the run. Each run begins with the name of the schedule or test, and ends with the date of the run in brackets.

Archived

2. Right-click the test run, and then click one of the following options:
   - Display Default Report

     Displays the report that you previously set as the default. To change this default, click **Window** → **Preferences** → **Test** → **Performance Test Report**.
   - Display report-name Report

     Displays the system-supplied reports. Note that the percentile report is available only after a run.
   - Display Report

     Displays a list of reports to select from. This list includes user-defined reports.

Besides displaying reports after a run, RPT can also help testers to more sufficiently manage and customize the result by:

- Comparing two reports

  You can simultaneously display reports from different runs for comparison.
- Customizing reports

  You can customize reports to specifically investigate a performance problem in more detail than what is provided in the default reports.
- Exporting results to a CSV file

  You can export the entire results of a run or specific parts of the results to a CSV file for further analysis.
- Exporting reports to HTML format

  You can export an entire report, or a tab on a report, to HTML format. You can then e-mail the report or post it on a Web server. The exported report can be displayed and printed from any browser. To further analyze the data, paste the exported report into a spreadsheet program.
- Viewing the test logs

  To see a record of all the events that occurred during a test run or a schedule run, open the test log for that run.
- Inspecting test log details in the Protocol Data view

  To verify that a test is performing as you intended, use the Protocol Data view, which displays the HTML details that were generated during a schedule run. If problems occur in a test run, you can also compare the data retrieved during the run with the recorded data.

Archived

- Transferring test log data in real time

  By default, test log data is transferred at the end of a run. To transfer data during a run, change the -DrptRealTimeHistory setting in the location file. Changing this setting lets you use the test log as a real-time progress monitor.

- Exporting test logs in XML format

  You can export test logs in XML format for further analysis.

Figure 22-4, Figure 22-5 on page 491, and Figure 22-6 on page 492 are some sample reports after schedule execution in RPT.

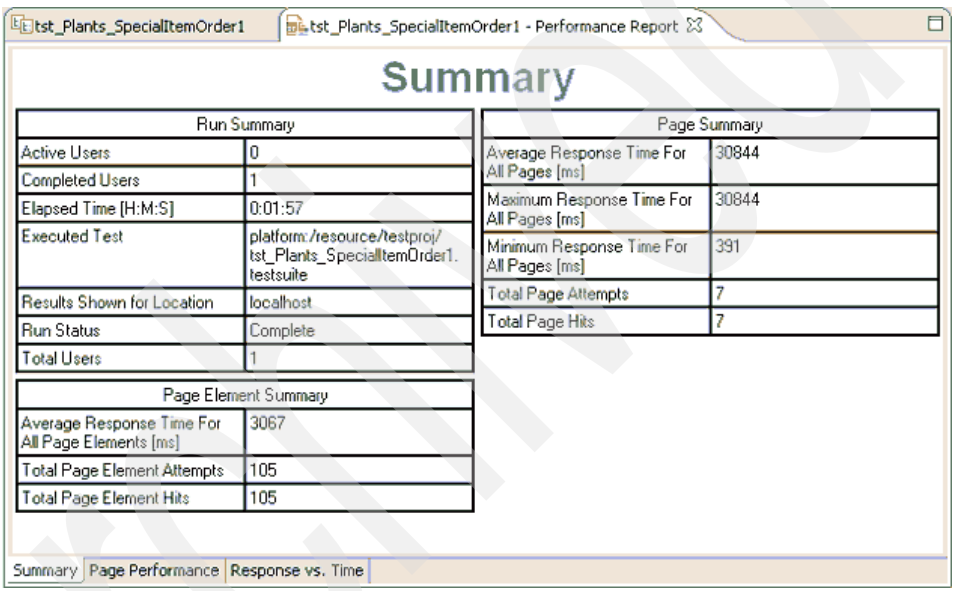
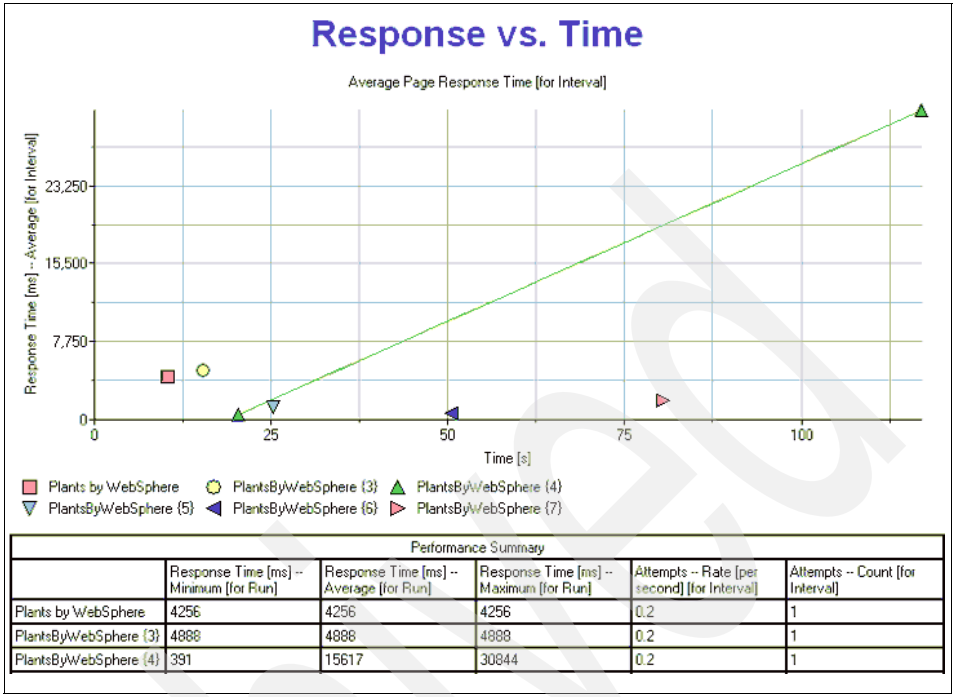


Figure 22-4 RPT result - summary

Archived

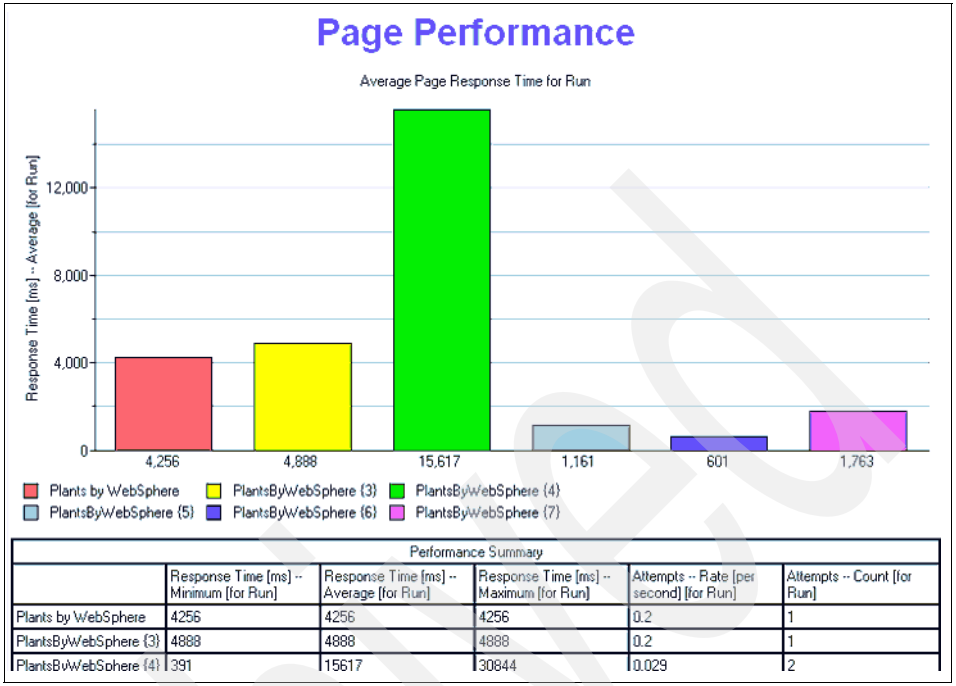


Performance Summary

	Response Time [ms] -- Minimum [for Run]	Response Time [ms] -- Average [for Run]	Response Time [ms] -- Maximum [for Run]	Attempts -- Rate [per second] [for Interval]	Attempts -- Count [for Interval]
Plants by WebSphere	4256	4256	4256	0.2	1
PlantsByWebSphere {3}	4888	4888	4888	0.2	1
PlantsByWebSphere {4}	391	15617	30844	0.2	1

Figure 22-5 RPT result - response time

Archived



Performance Summary

	Response Time [ms] -- Minimum [for Run]	Response Time [ms] -- Average [for Run]	Response Time [ms] -- Maximum [for Run]	Attempts -- Rate [per second] [for Run]	Attempts -- Count [for Run]
Plants by WebSphere	4256	4256	4256	0.2	1
PlantsByWebSphere {3}	4888	4888	4888	0.2	1
PlantsByWebSphere {4}	391	15617	30844	0.029	2

Figure 22-6 RPT result - page performance

22.3 Seague SilkPerformer

SilkPerformer is the load-testing tool that leads the industry today. It is used to assess the performance of Internet servers, database servers, distributed applications, and middleware, before and after they are fully developed. SilkPerformer can help you to quickly and cost-effectively produce reliable, high-quality application solutions.

22.3.1 What SilkPerformer can do

SilkPerformer creates highly realistic and fully customizable load tests. It does this through the use of virtual users that automatically submit transactions to the system being tested, in the same way that real users would. Using a minimum of hardware resources, you can generate tests that simulate many hundreds or thousands of concurrent users. You can use SilkPerformer's powerful reporting tools both during and after a load test to analyze the performance of your server and to locate bottlenecks so that you can maximize the potential of your system.

Archived

Generally, SilkPerformer can help you answer the following questions about your system under test:

- How many simultaneous users can my server support?
- What response times will my users experience during peak hours?
- Which hardware and software products do I need to ensure optimum performance from my server?
- Which components are the bottlenecks on my server?
- What is the impact on the performance of my system if I employ security technology?
- Which areas of my application perform well and which have bottlenecks? You can investigate your business transactions, objects, and operations.
- Which factors affect performance? What kinds of effect do they have? And at what point do those factors start impacting on service levels?

22.3.2 Procedure to use SilkPerformer to run performance test

Figure 22-7 shows the main interface to SilkPerformer.

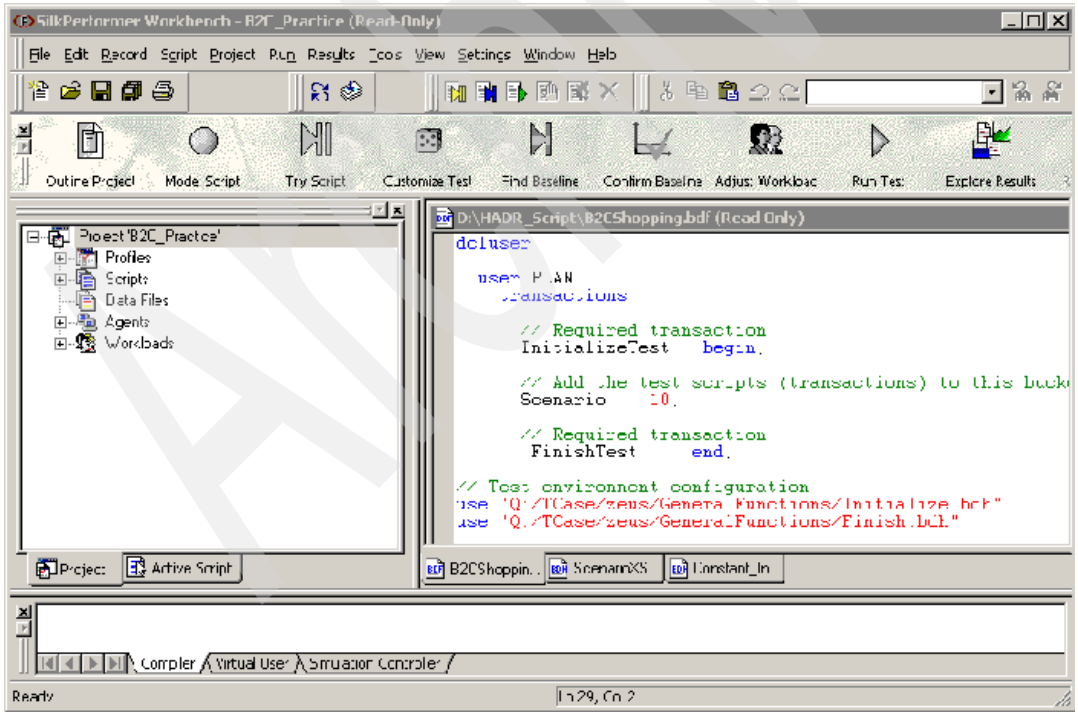


Figure 22-7 Main interface of SilkPerformer

For conducting a SilkPerformer performance test,:

1. Define a project outline.

   The first step in conducting a SilkPerformer load test is to define the basic settings for the load-testing project. The project is given a name and, optionally, a brief description can be added.

   The settings that are specified are associated with a particular load-testing project. It is easy to switch between different projects, to edit projects, and to save projects so that they can later be modified and reused.

   In general, a project contains all the resources needed to complete a load test.

2. Create a test script.

   The second step in conducting a SilkPerformer load test is to create the test script that will prescribe the actions of the simulated users who will be run during the test. The script is written in SilkPerformer's proprietary scripting language, the Benchmark Description Language (BDL).

   Scripts can be created in two ways. The easiest way is to use the SilkPerformer Recorder, which can automate much of the process for you. Alternatively, you can create a test script manually.

   – Test script generation with the SilkPerformer Recorder

     The standard method of creating a test script is to use the SilkPerformer Recorder to first capture and then record a representative amount of real traffic of the kind that you need to simulate in your test. The SilkPerformer Recorder then automatically generates the BDL test script from the recorded traffic.

   – Manual test script generation

     A second method of creating a test script is to manually create a new script from scratch in BDL.

   – Sample script reuse

     A variant on the second method of creating a test script is to base the new script on one of the sample BDL scripts provided with SilkPerformer.

3. Try-out test script.

   The third step in conducting a SilkPerformer load test is to do a trial run of the test script that was created in the previous step. The object of the trial run is to ensure that the test script is free from errors, and that it will reproduce accurately the interaction needed between the client application and the server. Normally, this will be the traffic that was recorded by the SilkPerformer Recorder in the previous step.

For this try-out of the test script, options are automatically selected so that you can see a live display of the actual data downloaded. Log files and a report file are created so that you can also check later that the script is working properly. Only one user is run, and the stress test option is enabled so that there is no think time and no delay between transactions.

SilkPerformer's TrueLog Explorer helps you to find replay errors quickly, to customize session handling, and to add verifications to the script. Figure 22-8 shows a sample display of TrueLog Explorer.

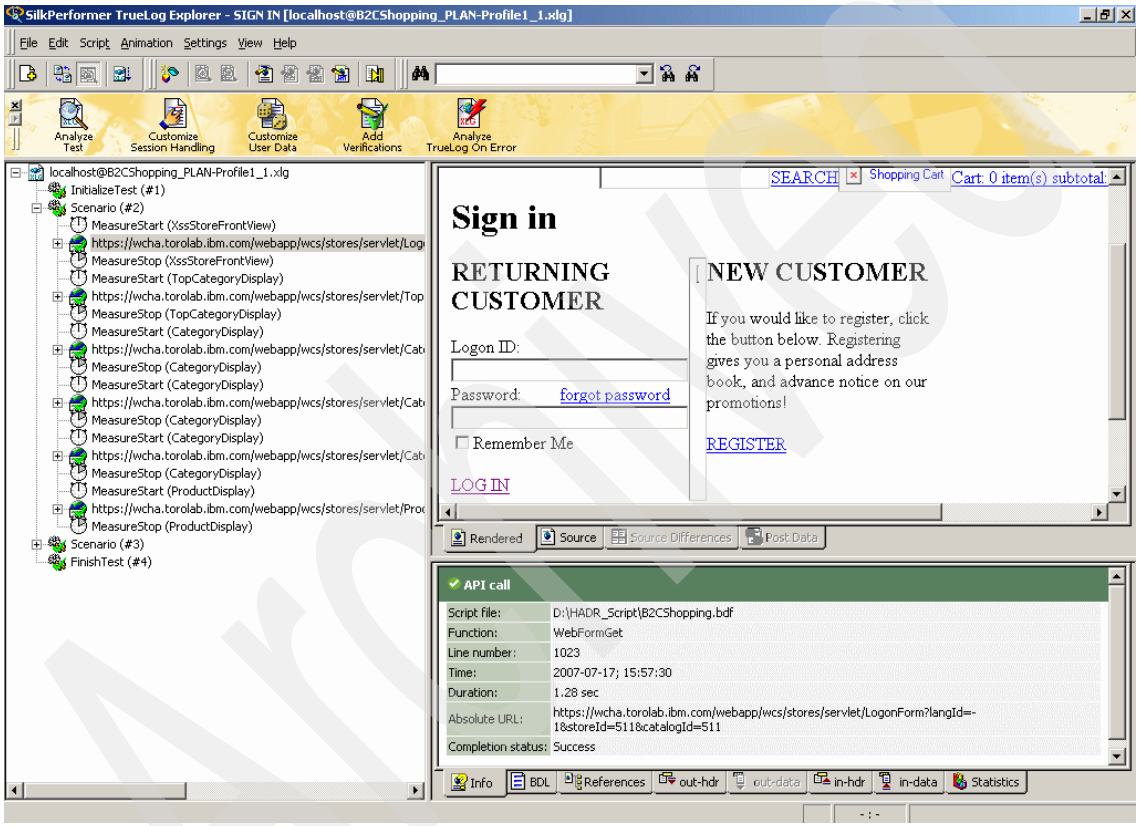


Figure 22-8 SilkPerformer TrueLog Explorer

4. Test customization.

   The fourth step in conducting a SilkPerformer load test is to customize the test with random data. In order to run the script without any replay errors, Segue's TrueLog Explorer helps you to customize the session handling and to add verifications to the script.

If a test script is used that has not been customized, the load tests will repeatedly reproduce the behavior recorded on one particular occasion, and this will not provide very accurate results. To realistically emulate the varied traffic of a large number of different users, the transactions in the test script need to be modified and, where necessary, randomized.

5. Establish baseline.

   The fifth step in conducting a SilkPerformer load test is to ascertain the baseline performance, that is, the basic, ideal performance of the application being tested. The now-customized test is run with just one user per user type, and the results from this unstressed performance of the application form the basis for calculating the number of concurrent users and the setting of thresholds for page and transaction times. The measurements typical for a real load test (of response times and throughput, for example) are performed.

   A secondary reason for determining the baseline test is to serve as a trial run of the customized test that was created in the previous step. Here the objective is to ensure that the customization has not introduced new errors into the script, and that the script will accurately and fully reproduce the interaction that is intended between the client application and the server.

   After that, we should confirm that the test baseline established by the test in the sixth step actually reflects the desired performance of the application under test. This is done by inspecting the results from that test in a baseline report.

Archive

6. Specify the performance test workload.

   Prior to the execution of a load test, you must select the workload model that you want to use for your load test.

   You can choose between the following six different workload models, as shown in Figure 22-9:

   – Increasing
   – Steady State
   – Dynamic
   – All Day
   – Queuing

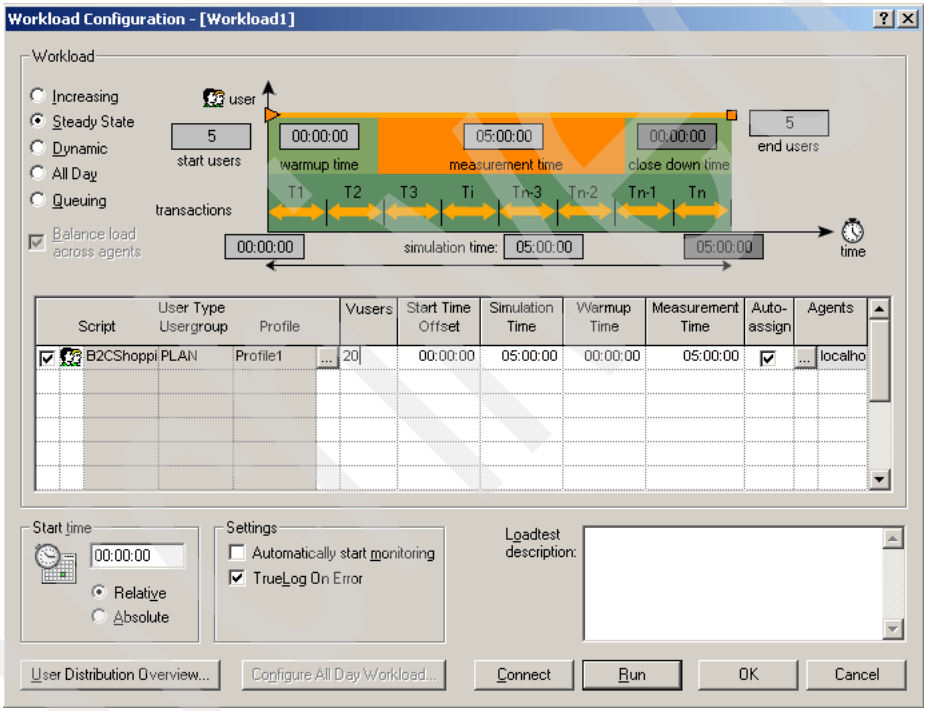


Figure 22-9 Workload setting for performance test

Note: Additionally, the number of concurrent virtual users per user type, the duration, and the involved agents must be adjusted accordingly.

7. Execute the performance test.

   To start the execution of the performance test:

   a. Click the **Run Test** button in the SilkPerformer Workflow bar. The Workload Configuration dialog appears.

   b. Check and, if necessary, change the workload configuration for your load test. Then run the test.

   c. While the test is in progress, you can monitor its progress and also monitor activity on the server by viewing the tabular monitor view of SilkPerformer and the graphical monitor view of the Performance Explorer.

Before the execution, make sure that you have enabled the generation of the kinds of test results that will be needed to assess the performance of the server. You can configure this information from the project profile. Note that each project has a specific profile to work with itself. Figure 22-10 shows is the profile configuration interface.

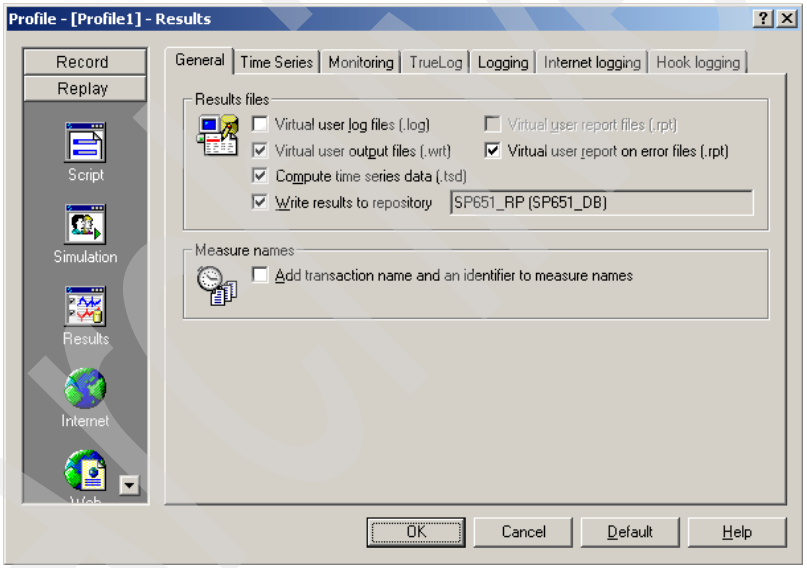


Figure 22-10 Project profile configuration

8. Evaluate and analyze the test result.

   After the test execution, first of all, you can see a summarized result for the test from SilkPerformer's baseline report, as shown in Figure 22-11.

Test Type :	Stress Test
Number of Users :	20
Test Duration :	03:00:06
Total OK Scenarios :	22682
Throughput :	7,556 (scenarios/hr)
Number of Errors :	0
Scenario Error Rate :	0.000%
Errors Per Hour :	0.000/hr
Number of WebInteractions :	215820
WebInteractions Error Rate :	0.0000%
Max Avg Response Time :	18.245 seconds
SVT Status :	Passed

Figure 22-11 SilkPerformer Baseline report for performance test

At the same time, SilkPerformer provides an abundant statistics chart about the execution result. Testers and developers can get a more comprehensive view of the behavior of the system under test with:

- *Transaction Status Report*, which depicts the number of SilkPerformer transactions per second, as shown in Figure 22-12.

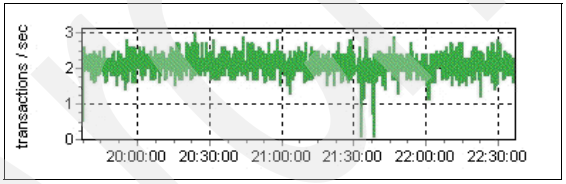


Figure 22-12 Transaction report in SilkPerformer

Archived

- *Throughput Report*, as shown in Figure 22-13.The amount of data sent to and received from the server. This includes header and body content information, all TCP/IP-related traffic (HTTP, native TCP/IP, IIOP, POP3, SMTP, FTP, LDAP, and WAP), and secure traffic over SSL/TLS.

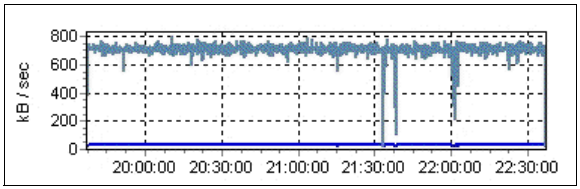


Figure 22-13 Throughput report in SilkPerformer

- *Response Time Report*, as shown in Figure 22-14. The response time of successful transactions, excluding the think times within those transactions. A transaction response time is reported in this type of measurement if all API function calls within the transaction succeed.

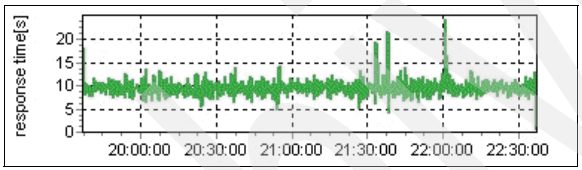


Figure 22-14 Response time report in SilkPerformer

- *Error Report*, as shown in Figure 22-15. This chart shows the number of API errors per second, including Internet, database, and middleware APIs. A problem is considered an error if its severity is defined as an *error* or worse, that is, of higher severity (*transaction exit* or *process exit*). A problem is ignored if its severity is defined as *informational* or *warning*.

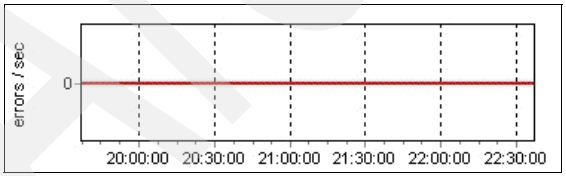


Figure 22-15 Error report in SilkPerformer

22.4 Page Detailer

IBM Page Detailer is a browser-side tool to measure performance of a Web application. While the Profiler discussed in the previous sections supports

analysis of the execution of the application on the server, the Page Detailer collects most of its useful data at the socket level to reveal the performance details of items in the Web page, from the client's (browser's) perspective. It is also useful for measuring the incremental impact of changes in a Web application.

Page Detailer allows you to look at how and when each item is loaded in a Web page. Analyzing this data allows you to identify the areas where performance could be improved. The user's perception of performance is determined based on the time to display pages, so measuring and analyzing this data will provide insight into the user's experience of your application.

22.4.1 Overview

IBM Page Detailer is a graphical tool that enables Web site developers and editors to rapidly and accurately assess performance from the client's perspective. IBM Page Detailer provides details about the manner in which Web pages are delivered to Web browsers. These details include the timing, size, and identity of each item in a page. This information can help Web developers, designers, site operators, and IT specialists isolate problems and improve performance and user satisfaction. Page Detailer can be used with any site that your browser can access.

Page Detailer is a separately downloadable product that can be obtained from IBM alphaWorks at:

http://www.alphaworks.ibm.com/tech/pagedetailer

IBM Page Detailer gathers the following information:

- Connection time
- Socks connection time and size
- SSL connection time and size
- Server response time and size
- Content delivery time and size
- Delays between transfers
- Request headers
- Post data
- Reply headers
- Content data
- Page totals, averages, minimums, and maximums

For each page that is accessed, a color-coded bar chart of the time taken to load the page items will be generated. The length of a particular bar gives a good idea of the relative time spent in loading that item, as compared to the entire page. You will see that in some cases, items of a page may be loaded in parallel. This

will appear in the chart with bars that overlap. The information that is captured by the Page Detailer includes page size as well as sizes of all other items loaded by the browser.

Different colors in the bar indicate how the time was spent:

- Page Time (Purple)

  The time taken to load all the components of a page.

- Host Name Resolution (Cyan)

  The time spent to resolve the IP address of the host.

- Connection Attempt Failed (Brown)

  The time taken to receive an error when a connection attempt is made.

- Connection Setup Time (Yellow)

  The time taken to open a socket connection. If a SOCKS server is being used, this is the time to open a socket connection from the browser to the SOCKS server only.

- Socks Connection Time (Red)

  The time taken to open a connection from a SOCKS server to the remote site.

- SSL Connection Setup Time (Pink)

  This is the time taken to negotiate an encrypted connection between the browser and the remote site, once a normal socket connection has been established.

- Server Response Time (Blue)

  This is the time from the browser's request to the receipt of the initial reply, after all the communications setup has been completed. Large responses are broken down into smaller components (packets). The server response time only measures the time to receive the first one.

- Delivery Time (Green)

  The time taken to receive all additional data that was not included in the initial response.

Archived

An example of a chart produced with Page Detailer is provided in Figure 22-16.

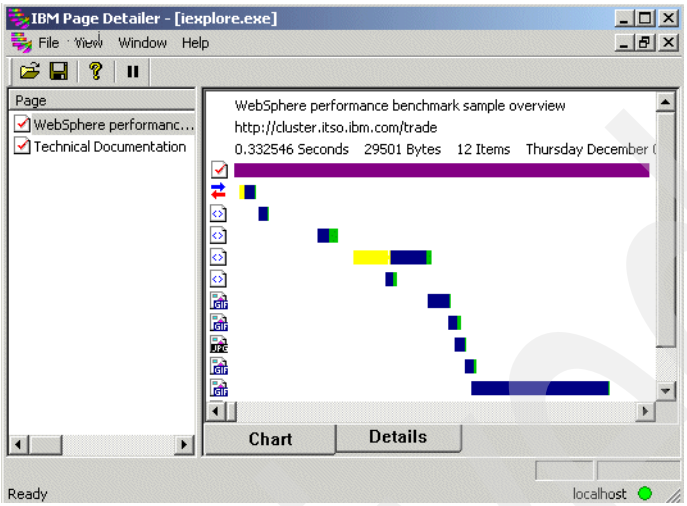


Figure 22-16 Page Detailer Chart view

To obtain more detailed information about a particular HTTP request, double-click the appropriate colored bar or the icon in the chart. This will display a text viewer

22.4.2 Important considerations

Some important considerations while taking measurements are:

- Impact of network delays

  Many problems may not be evident when accessing a server on a local network, but may become apparent when accessing the site remotely, particularly when using a modem connection. On the other hand, you can minimize the effect of external network delays by directly connecting on the Server's LAN. This will allow you to isolate the performance impact of a change made to the Web page.

- Browser cache

  Disabling the browser cache helps in getting repeatable results. However, you could also check the performance from a user's perspective by enabling the browser cache and comparing both results.

- Packet loss

  Packet loss can happen and get corrected in the underlying TCP/IP layers. This is invisible to the Page Detailer. Packet loss manifests itself as

Archive

inconsistent time measurements in Page Detailer. You can take a series of measurements at different times to factor it out.

22.4.3 Key factors

Some of the key factors that influence the time to load a Web page in a browser are:

- Page size
- Number, complexity, and size of items embedded in the page
- Number of servers that need to be accessed to retrieve all elements, and their location and network connectivity
- Use of SSL (This introduces an extra overhead.)

The Page Detailer will help you to identify when one of these problems is affecting some or all of your application. It will also help to identify problems such as broken links and server timeouts.

Some of the strategies that can be used to improve performance and resolve problems you have identified include:

- Minimize the number of embedded objects. Avoid the excessive use of images in particular. In cases where there is a standard header, footer, or side menu on every screen, consider the use of frames so that common elements do not have to be downloaded every time.
- The browser will typically retrieve multiple items in parallel, in the order in which they appear in the HTML page that it receives. Hence, sequencing of the items so that downloads for larger objects are started early can reduce the total time required to display the page, and avoid the user having to wait for a long time for the last elements to be retrieved.
- Ensure that caching is being used effectively. Often the same images are used multiple times on the same page. If there are two references to the same image in close proximity to each other in the HTML source, the browser may encounter the second reference before the HTTP request that was initiated to download the first reference has been completed. In this case the browser may issue another request to retrieve the image again. This can be avoided by pre-loading frequently used images multiple times early, or by structuring the generated pages so that such URLs do not appear consecutively.
- Minimize the use of SSL where possible. For example, some content such as images may not need to be secured even though the application as a whole needs to be secure.

Archived

- Try to avoid switching the user to an alias server name during the page load. This will help the browser to reduce the lookup time and possibly avoid a new connection.

22.5 Other performance test tools

There are many kinds of open-source and commercial products and services related to Web application testing. Below is a list of some of the commercial and free tools available. They are merely provided as an alternative source of testing tools if ApacheBench, OpenSTA, or Rational Performance Tester are not used.

- JMeter, Open Source software available from the Apache Software Foundation

  http://jakarta.apache.org/jmeter/

- TestMaker and TestNetwork, from PushToTest

  http://www.pushtotest.com/

- Grinder

  http://grinder.sourceforge.net

- LoadRunner from Mercury Interactive

  http://www.mercury.com/us/products/performance-center/loadrunner/

- WebLOAD from Radview

  http://www.radview.com/products/WebLOAD.asp

- WebStone from Mindcraft

  http://www.mindcraft.com/webstone/

- OpenLoad from Opendemand Systems

  http://www.opendemand.com/openload/

22.6 Trend of performance test tools

With the development of the IT industry, more and more new techniques have been involved in the software system, for example, Web2.0 and Ajax, so that performance testing becomes more and more complex and brings more challenges to performance testers.

Archived

In the area of performance testing tools, there are three major characteristics:

- More utilities related to performance test have been integrated into test tools, which include test recording, test editing, test execution, test result analysis, problem determination, and so on.
- Customization in test tools becomes more and more popular. For various testing purposes, customization of the testing tools always plays an important role in performance testing. The trend for this requirement is that more and more utilities in test tools can be customized for customers' specific requirements, which surely brings more benefits to users.
- The fast development of Open source test tools. More and more software engineers dedicate themselves to the open source community, where they are investigating the mechanisms of the test tools in the code level, and collaborating with others. That is the reason why the open source tooling can achieve significant improvement in the future.

Archived

23

Applying performance testing to WebSphere Commerce

Now that we have gone through various performance test types let us see how they apply to WebSphere Commerce.

There are many variables when it comes to running performance testing and many possible bottlenecks. This makes an already complex task even more daunting. However, applying performance testing to a certain type of business (for example, eCommerce, such as in the case of WebSphere Commerce) does narrow the problem domain.

© Copyright IBM Corp. 2008. All rights reserved.

23.1 Key attributes of a performance test

Before we delve into different types of performance tests we define the features that distinguish one performance test from another.

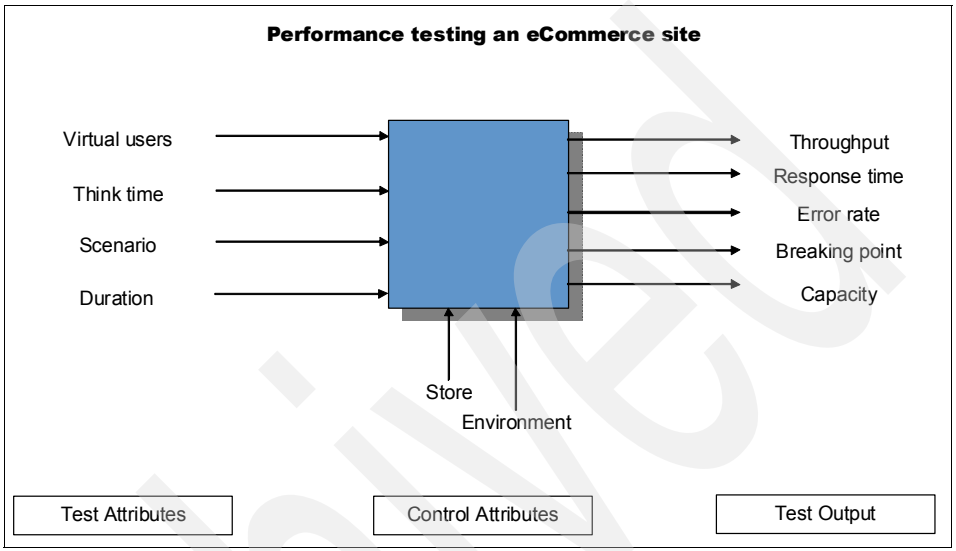


Figure 23-1 Performance testing an eCommerce site, such as a WebSphere Commerce site

In the case of eCommerce, the test attributes that dictate workload include:

- Number and type of virtual users

  Simply put, virtual users are the concurrent, simulated users. A test run with 100 virtual users means that there are 100 concurrent, simulated users who are using the site at any given time during the test.

  These users can be configured to use the same session or a different session for every new scenario execution. Usually, you would need to start a new scenario execution with a new session.

  Also, if 100 virtual users includes registered users then it does not mean that the same logon IDs will be used over and over again. On the contrary, almost always you would want to use random users. The only exception to this rule is in the case when your unique business requirements specifically call out for such a scenario. In our testing we noticed that if we keep using the same test IDs over and over again, then the stress-endurance test or the reliability test show gradual throughput degradation.

Archived

Virtual users can be store administrators, shoppers, marketing managers, and so on. Depending on your scenario, you may need to further define the users based on certain demographic requirements of your business. This information should be available from your business analyst.

- Thinktime

  Thinktime is decided by the distribution and type of incoming user traffic. Again, your business analyst should be able provide thinktime specifications (or requirements) for your testing.

- Scenario

  Scenario includes the series of actions that virtual users execute while interacting with the WebSphere Commerce site. Inherently, this also includes different interfaces or utilities what might be used to interact with the WebSphere Commerce site. The scenario is decided by the use cases for your site design. Use cases should be available in your site design documents.

- Duration

  The duration of a test is dictated by the intention of the test case. If the intention of the test case is to find deadlocks, memory leaks, or gradual throughput degradation (GTD), then the test should be run for longer—running into hours or days, as is the case with soak tests. If, however, you are testing scalability of the system, then the test can be run for a much shorter duration—running into an hour or a few.

The control attributes are related to the site setup and configuration:

- Store

  Store refers to both the file and data assets that constitute your store. As discussed in “Store complexity scalability” on page 465, the complexity of your store pages and database customizations design impact the performance of your site. The amount of data could be another factor that could influence your site's performance.

  It would be ideal to test your site’s performance with a backup copy of your production database. Although your initial tendency may be to focus on the performance considerations of your database customizations, you should also consider purging unwanted data from your database using the WebSphere Commerce dbclean utility. We recommend that you run the WebSphere Commerce dbclean utility periodically. For more information about WebSphere Commerce dbclean, refer to WebSphere Commerce InfoCenter.

Archived

- Environment

  Environment includes both the hardware and software, along with their configuration settings.

  - Hardware and site topology

    The hardware that you employ for your site impacts its performance (for example, CPU speed, amount of memory, disk size and speed, disk controllers, disk cache, and so on). The topology that you have for your site also has a significant impact on your site's performance. For example, having WebSphere Commerce database on the same machine as the WebSphere Commerce application will cause both of them to compete for the same hardware resources. While deciding on topology, you also need to consider the level of clustering that you need for your site, the active/active or active/passive support that you require, security or firewall options, and so on.

  - Hardware and software configuration

    This refers to all the various hardware and software configurations possible to performance tune your site, as discussed in this book. For example, setting up a 32-bit or 64-bit database, WebSphere Application Server, WebSphere Commerce configuration, and so on.

The test results provide a rich set of indicators. Most of the small to medium sites test for throughput and response time, whereas breaking point and capacity testing is left for major upgrades to the site. For a additional output parameters refer to 24.1, "Test results to be collected and verified" on page 522

- Throughput

  The number of client interactions with WebSphere Commerce. The unit of interactions can be defined at different levels of granularity. For more details refer to 1.2.4, "Throughput" on page 9.

- Response time

  Elapsed time between client request and server response. For more details refer to 1.2.5, "Response time" on page 9.

- Error rate

  Error rate should be defined in similar terms as the throughput. For the reasons discussed in 1.2.4, "Throughput" on page 9 we recommend that you define throughput in terms of scenarios completed. In such a case the error rate would be defined as:

  ```
  Scenario Error Rate = Total # of failed scenarios / Total # of
  attempted scenarios
  ```

Archive

However, if the throughput is defined in terms of transactions then the error rate would be defined as:

```
Transaction Error Rate = Total # of failed scenarios / Total # of
attempted scenarios
```

- Capacity

  Capacity can be maximum system capacity, required business capacity, or expected peak capacity. For more details refer to 20.4, “Typical performance characteristics of a WebSphere Commerce site” on page 460.

- Breaking point

  The point of meltdown where the site performance degrades severely and unpredictably. For more details refer to 20.4, “Typical performance characteristics of a WebSphere Commerce site” on page 460.

The key idea of performance testing a WebSphere Commerce site is that, for a given set of scenarios and control attributes, you would tweak the virtual users and thinktime to put the system under stress.

23.2 Common test execution steps

Common performance test execution steps include:

1. Set up the test environment.

   Setting up the test environment includes setting up everything that is required to run a performance test. This includes setting up a test WebSphere Commerce environment to mimick or simulate your target WebSphere Commerce production environment, as well as the test client software.

   This also assumes that you have developed the automation scripts that will be executed using your client software. If you are building a new site you will also need to pre-create test data for your scripts to run on.

2. Tune up the test environment.

   Throughout this book there are various performance tuning considerations that should be applied to optimize your site’s performance.

3. Warm up the test environment.

   Before you can run the formal target performance test case you must warm up the environment.

   Warm-up is required to initialize that various caches in your system, especially Dynamic Cache, as well as to allow your database to build any access plans that may be required. For convenience we call out these two

key aspects in the flow chart below and call them as application server warm-up and database warm-up, respectively.

To warm-up your test environment start your automation test with a single user and run up to a few users while monitoring your cache status. Once the hit ratio stabilizes, your site has been warmed up.

After you have done this a few times you should know the rules of thumb (for example, the number of users required to run a warm-up test for a given duration) to warm up your site, and you may not need to constantly monitor your site's operation to confirm whether warm-up was done successfully.

4. Reorganize the database and optimize database statistics.

   Your test generates data and, as such, you should run the DB2 RUNSTATS and REORG utilities to optimize your database.

5. Back up the test environment.

   Before continuing any further we recommend that you back up your environment, including your database. This would allow you to return to a *baseline* environment to compare the impact of various performance tuning or fixes that you may apply later on during the performance testing.

6. Execute the performance test.

   If you backed up your environment, as mentioned in the previous step, or if you are restarting your test from a backed up environment, then you need to warm up your site a little bit to load the Dynamic Cache and the file system cache, and so on. You can do this with even a single user test for a few iterations of your test scenarios.

   In the remainder of this chapter we discuss the execution of various performance tests.

7. Analyze the test results.

   This subject is discussed in detail in Chapter 24, "Analyzing test results and solving performance problems" on page 521.

Note: Warming-up your test environment with the same users as the users included in your automation test can skew your performance test, depending on the duration of your warm-up, the duration of your test, and the amount of test data in the database. As such, we recommend that users used for warming up your test environment not be included in the target performance test case.

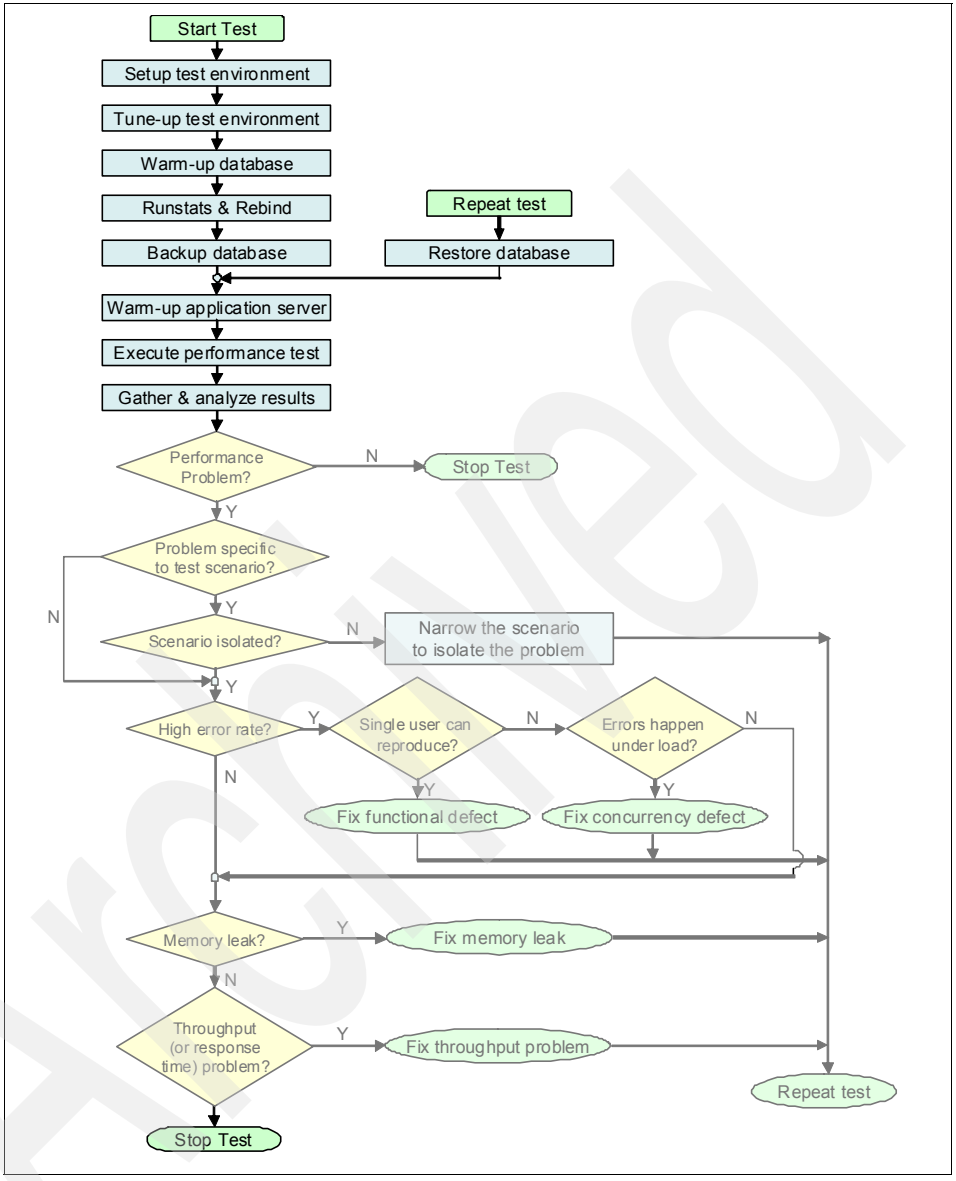


Figure 23-2 Highlighted test execution steps with shaded problem determination steps

Archived

23.3 Executing stress tests

Most of the performance testing is done at or above the peak workload expected. In the case of stress testing, the workload is mostly much larger than the peak workload expected.

During a stress test, many test attributes and control attributes, as shown in Figure 23-1 on page 508, can be changed. However, for the present discussion we focus on the following, for any given scenario:

- The number of virtual users
- The type of virtual users and their workload distribution, such as their browse-buy ration, which can be altered
- Thinktime - the time elapsed between two successive client interactions with the site

23.3.1 Testing for throughput

The preferred approach for stress testing a system is to stick to the realistic think time and gradually vary the number of virtual users so that the throughput exceeds the expected peak throughput (T_P), then reaches the throughput level corresponding to the maximum business capacity (T_P), and, eventually, reaches the breaking point (T_x) after sustaining the maximum system capacity (T_C).

For more details and a graphical representation of the above statements refer to 20.4, “Typical performance characteristics of a WebSphere Commerce site” on page 460.

This approach will give you realistic test results that are easy to interpret. The increase in throughput will be accompanied by an increase in CPU consumption, eventually maximizing the CPU when maximum throughput is achieved (T_M).

However, in case you are testing on a test environment that is not as powerful as your production environment and you find that you are limited by your test hardware such that you are not able to increase the number of virtual users to take the system to the breaking point (T_x), then an alternative would be to decrease the thinktime. Decreasing the think time in your test scenarios way past the expected think time is akin to increasing virtual users way past the expected value. You can even turn your think time down to zero to maximize the stress on your site for a given number of virtual users.

This approach also benefits if you need to maximize your CPU consumption and throughput very quickly, instead of having a long trial and error process. However, the trade-off for this approach is the lack of ready-to-use test results.

Archived

You would need to scale the results that you get from this test methodology so that you can interpret the corresponding behavior on your production environment for more realistic scenarios.

Such a scaling activity would require running a few experimental test cases and interpreting the results to come up with a scaling factor. Most of the time, especially for small and medium sized customers, such a scaling may turn out to be a linear factor. Any such scaling would need to be repeated with any major change in your system hardware, software, application, data, or even scenarios.

Once the system has reached maximum throughput (T_C), the response times of further client interactions may not be relied upon. In our testing we find that WebSphere Application Server scales to increase in workload extremely well. So, although WebSphere Application Server handles all the additional work given to an application under stress, the queuing-up of workload leads to longer response times. For example, as the number of virtual users keep increasing and there are no more CPU cycles to accommodate this additional workload, then each virtual user gets a smaller and smaller time share of the CPU, driving the response times higher.

These response times may be unacceptable for shoppers or users of the site.

So, although the above approach provides you with an easy mechanism to stress the system and to look for bottlenecks, it still leaves the question of observed response times unanswered.

23.3.2 Testing for concurrency

In addition to testing for the throughput targets, we also need to make sure that the site is able to handle the required number of concurrent users. The testing done to test for throughput doubles for testing the concurrency.

Similar to peak throughput expected, the business will have peak concurrency expected. There would also be business requirements of your site to maintain performance past peak concurrency expected, which we call the required business concurrency. Corresponding to the throughput breaking point, there would be a concurrency breaking point.

23.3.3 Analyzing stress test results

For each test scenario executed:

- For peak workload expected and minimum business capacity required, the error rate must be less than the acceptable error margin, as defined by your

business requirement. For example, the acceptable error margin may include page errors less than 0.1%, resulting in less than 1% failed scenarios.

- Concurrency, throughput targets should be met or exceeded for both peak workload expected as well as for minimum peak business capacity required.
- Response time targets should be met or beat for both peak workload expected as well as for minimum peak business capacity required.
- For excess capacity past minimum business capacity (as shown in Figure 20-4 on page 466), although response time is allowed to go past the maximum requirement, the acceptable error rate should still conform to your business requirements.

If any of these passing criterion are not met then it should be considered a defect. If error rate is the cause of the test case failure then the defective code needs investigation.

If overall low performance is a concern then we need to go through the iterative process of performance tuning and re-executing the test case, as shown in Figure 23-1 on page 508.

Last but not the least, if the concern is memory leak, core dump, or gradual throughput degradation, then refer to Chapter 24, “Analyzing test results and solving performance problems” on page 521.

23.4 Scalability testing

Whereas stress testing involves stressing the system way past the business capacity requirements, the scalability test primarily focuses on performance of the system within and around business capacity requirements. In the case of stress testing after the system has reached its maximum system capacity, response times starts degenerating very significantly, and as such there is not much to be learned from response times. In the scalability test case, response time is very critical, just like the throughput for a given set of test attributes, for example, concurrency, think time, and scenario.

The size of the catalog or the number of hosted stores in the system would also have a big impact on the performance of the system. Thus, more than one data parameter may also be varied from one hurdle to another. In the previous example we have only used users as a parameter for our stress test, but in reality there may be more than one.

Your database size is a major contributor to what throughput or response times your site may be able to provide.

Archived

In our previous discussion on stress testing we focused on increasing workload (virtual users and thinktime) for any given scenario. For example, we focussed on the test attributes only. In scalability testing we follow the same tactic, plus we test for different control attributes.

This is an arbitrary distinction since different data sets, hardware setups, and so on, should be stressed as well. You may decide to combine both your stress and scalability testing in one series of test cases, starting from low load to peak load, and then eventually to the breaking point load. However, if the thinktime in your stress test is not realistic, then you would have the same additional work of scaling the test results to your production environment size before any meaningful decision could be made from them, as discussed in 23.3.1, “Testing for throughput” on page 514.

The reason that we find this distinction useful is that, depending your site, for a given deployment you may need more or a different level of focus on either stress or scalability. By lumping the two together you may lose the focus and even end up doing more testing than you actually require.

To test for valid response times the system needs to operate within or at its maximum system capacity. The site can still be operating way past the expected peak workload as defined by your business. Some sample business capacity numbers are discussed in “Expected peak capacity versus maximum business capacity” on page 461.

23.5 Soak, endurance, or reliability testing

Reliability testing is done around expected peak capacity (P) and higher. Usually, the testing should be conducted close to required minimum business capacity (B) but not more than the maximum system capacity (C).

The key factors of reliability testing is the duration for which the test is run as well as the variable workload to mimic typical day-to-day activities on your site.

Reliability runs are usually run for days and can be half a day, one day, and so on, generally going up to seven days. A three-day reliability run is very common, since this is usually long enough to start showing symptoms of many performance problems, should they exist. A three-day test also gives you an opportunity to start it on a Friday, observe the test for a little while, then leave it running for the weekend, and then come back on Monday and observe the test in execution for another little while before it completes.

Figure 20-4 on page 466 gives an example of a three-day reliability test. The workload curve is not a straight line, but is modulated as shown in Figure 20-4 on

page 466. In addition to varying the workload from the shoppers, extra workload of the likes generated by DB clean, WCA, massload, Sales Center activity, MQ, Staging, and Authoring server will be added as well. This utilities can be executed manually or, preferably, automatically.

23.6 High Availability testing

In Chapter 20, “Introduction to performance testing” on page 453, we introduced the concept of High Availability testing in WebSphere Commerce performance testing. Generally, before starting the High Availability testing, having a good understanding of the utilities or solutions that we use to achieve High Availability in WebSphere Commerce site is very important. For example, if we want to use DB2 High Availability and Disaster Recovery (HADR) to achieve High Availability in the WebSphere Commerce database tier, several concepts should be kept in mind:

- The mechanism of HADR to achieve High Availability in a database
- Which outage scenarios HADR covers
- Whether there are any pre-implementation recommendations from HADR
- Whether there are any limitations in HADR which that impact the implementation

The general approach to apply High Availability testing on a WebSphere Commerce Web site can be classified as:

- Before enabling the High Availability utilities, execute a normal performance test case to establish a baseline for performance comparison purposes. Most times, the normal ratio for performance testing scenarios is not appropriate for High Availability testing, since we should simulate an appropriate and realistic scenario to test the behavior of High Availability utilities. In that case, appropriate customization is needed to implement the High Availability testing.
- Implement the High Availability utilities on the WebSphere Commerce site, and evaluate whether it is functional.
- With High Availability utilities enabled, drive normal performance execution to identify the performance impact to an additional High Availability configuration. We should collect the data for site throughput, error rate, resource (CPU, memory, disk I/O) utilization, and network status. With those dates on hand, compare the results one by one to identify and establish an overall picture of the performance impact of High Availability components enabled on the WebSphere Commerce site.

Archive

- Simulate the planned or unplanned outage by automatic script or manually, to identify the performance overhead brought from different outages (or we can say that some of them are disasters), and how can High Availability utilities help the Commerce site to achieve failover.
- With most of the outage scenarios' data collected, draft a performance comparison table to identify how much benefit the candidate High Availability solution gives to the WebSphere Commerce site. This table will help to make an appropriate/reasonable business decision about how to achieve High Availability on the Commerce site.

The key point here is that High Availability is much more a concept built on the customer's point of view, so that the comparison and judgement according to the result that we got should be assessed and evaluated from the customer's view, which sometimes is the most difficult step in driving the performance test.

Archived

Archived

24

Analyzing test results and solving performance problems

Collecting test results is as important as executing the test itself. Test execution can be useless if appropriate data is not gathered. What data and how much data should be gathered depends on the objective of the test case. Obviously, if you are in the process of troubleshooting a specific performance problem you may gather more data than the first time that you are executing a test case. The amount of data generated during a test case execution impacts the performance of the test case. As such, it is crucial to have the correct amount of logging/tracing and frequency of gathering data.

Note that in this chapter by the term *test environment* we mean the environment on which the test is run. It is possible that after successful testing this environment may actually be used or switched over as your production environment.

© Copyright IBM Corp. 2008. All rights reserved.

24.1 Test results to be collected and verified

For each test executed the following results would typically be recorded for analysis, as well as future reference:

1. Input attributes: For more information about these attributes refer to Figure 23-1 on page 508.
   - Number and type of virtual users
   - Think time
   - Scenario
   - Duration
2. Control attributes: For more information about these attributes refer to Figure 23-1 on page 508.
   - Store including file assets and data
   - Environment
     - Hardware and site topology
     - Hardware and software configuration: This can often be important since component levels may change during your site development, depending on the duration of your project, as well as the number of products integrating with WebSphere Commerce. It is important to know exactly what software stack created the successful result.
3. Output values:
   - Minimum/average/maximum response time for all page hits: Refer to “Response time” on page 9 for further details.
   - Minimum/average/maximum test scenario response time: Refer to “Response time” on page 9 for further details.
   - Transaction/page hit/scenario throughput: Refer to 1.2.4, “Throughput” on page 9 for further details.
   - Page hit/scenario failure ratio
   - Resource utilization (memory, CPU, I/O, and so on): Refer to Chapter 15, “Operating system monitoring tools” on page 323, for further details
   - Additional information as required by your business, such as orders/hour.
   - Logs such as WebSphere Application Server logs, JVM logs, database logs, and test client logs.

Archived

The purpose of recording this information is that:

- One may understand what is being tested.
- One may diagnose performance concerns without necessarily having to rerun the test case. This can also speed up getting support from subject matter experts who may not be directly involved in testing, including IBM Support.
- Someone can reproduce the exact same test and reproduce the same exact result in the future, for example, for comparison or for problem determination purposes.
- To have sufficient information about the results of test cases, which can then be used to predict the site behavior as the workload of the site changes. This information could also help better understand the site configuration changes that may be required to accommodate the workload changes.

Verify pass criterion

For all the test cases executed, you must analyze the results from various angles:

- Test results within business requirements
  - Acceptable failure rate

    Failures can happen due to various reasons, including functional defects and performance defects such as deadlocks. As discussed earlier, functional defects should ideally not happen during performance testing since if they do, they should not be acceptable. Performance defects, however, may sometimes be acceptable. For example, deadlocks in databases are inevitable. We always try to minimize and eliminate the possibility of having deadlocks in a database, but they are inevitable, depending on the workload at the site. As such, some margin of error, based on your business requirements, in a performance test is generally acceptably.

    The page hit failure rate should be less than your business requirements. Another business requirement that you need to track is the scenario failure rate. The page hit failure rate should not result in a scenario failure rate of more than this specification. The difference between the two is that whereas a page hit, such as catalog browsing, may result in a non-functional error, the user should be able to refresh the page and continue with shopping. For example, if there are more than ten page hits per scenario, 0.1% page hit errors and 1% scenario failure rate may be acceptable. Notice that in absolute numbers a scenario is still more resilient to failures than the sum page hit errors.

Archived

- Acceptable response time

  Although the minimum and maximum response times for a page hit and a scenario are important, the average response time is what would generally be tracked. The average response times should also be within your business specifications.

- Acceptable concurrency and throughput

  Ensuring that the throughput targets are met is also important even though it is the level of concurrency (number of virtual users) that seems to be driving the throughput. The level of concurrency is certainly more critical, but the throughput numbers provide you with revenue generation information for the scenario or site design that shoppers interact with. However, scenarios that may be long or non-intuitive may not always translate into orders.

- No memory leak or excessive fragmentation: For more information refer to “Solving memory problems in WebSphere applications” on page 527.
- No degradation of throughput or response time over time: For more information refer to “Solving throughput and response time problems” on page 549.

24.2 Common troubleshooting steps

WebSphere Commerce applications can be complex and as such troubleshooting performance problems can be one of the most important and complex tasks in the your site development cycle.

Here we discuss a general strategy for troubleshooting performance problems:

1. Set up the test environment.

   This includes your WebSphere Commerce site as well as any test tools required.

2. Warm up your site and back up your site before test execution.

   Before starting your test you must run a test to warm up your test environment. Warming up refers to running a very small load on your site for long enough so that various caches can be populated and the database can build an access plan.

   We recommend that you take a backup of your database after this warm-up so that you have a clean restart point should you need to rerun the test case later on.

After you take the backup and restart the WebSphere Commerce application you will need to rerun through your scenario at least once before executing your test case so that Dynamic Cache can be populated.

3. Narrow down the part of your scenario causing the performance concern.

   This is to isolate the problem so that you can focus on it without being distracted by any other interaction. Once you have narrowed down the interaction or the set of interactions that may be causing the performance concern, this would also make it easy for your to reproduce the problem quickly.

4. High error rate.

   All functional defects (which can be reproduced always even by a single user, for example, without load on site) should be resolved prior to starting performance testing since finding functional defects by doing performance testing is very expensive due to time and resource required for performance testing. However, it is possible that some functional defects were not caught prior to running a performance test and thus generate errors.

   If, however, the errors are caused due to concurrent users accessing the site, then it is possible that code-related or database-related deadlocks may be causing the problems. In such a case more detailed tracing may be required to isolate the problem. It is also possible that such problems may be caused due to system configuration such that it is not able handle high workload causing overflows or time outs.

In remaining part of this chapter we explore strategies for addressing memory, throughput, and response time concerns in greater detail.

Archive

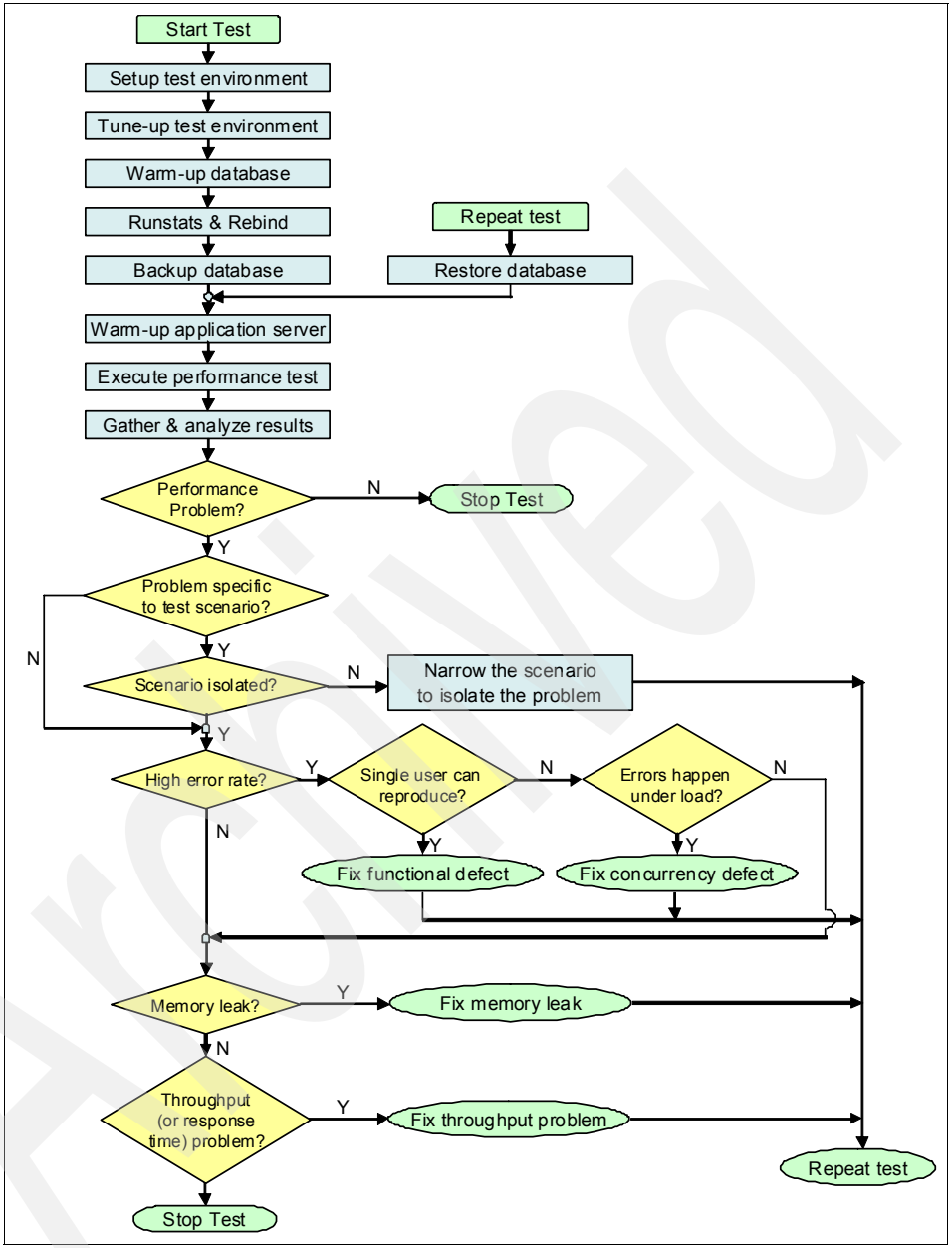


Figure 24-1 Test execution cycle with highlighted problem determination steps

Archived

24.3 Solving memory problems in WebSphere applications

In Web applications based on WebSphere Application Server, such as WebSphere Commerce, memory utilization can impact system performance significantly. One of the most common memory problems is memory leak, which causes severe performance degradation. In theory, memory leaks should not happen in Java because it has Garbage Collection (GC). However, GC only cleans up unused objects that are not referenced anymore. Therefore, if an object is not used, but is still referenced, GC does not remove it, which leads to memory leaks in JVM problems. Beside memory leaks, other memory problems that you might encounter are memory fragmentation, large objects, and tuning problems. In many cases, these memory problems can cause the application server to crash. Many users first notice that application server performance gradually declines, and eventually crashes with OutOfMemory exceptions.

Memory problems are hard to troubleshoot because they have multiple causes. This article provides methods of identifying the root causes of different memory problems and their corresponding solutions. It also introduces a recommended methodology to detect and solve memory leak problems in WebSphere Commerce.

Archive

Figure 24-2 shows the entire process for determining and solving memory problems. There are five kinds of solution listed in this diagram, which are further explained in the following sections:

- Tuning the max heap
- Tuning Xk/Xp
- Identifying by swprofiler
- Tuning the cache size
- Performing the heap dump

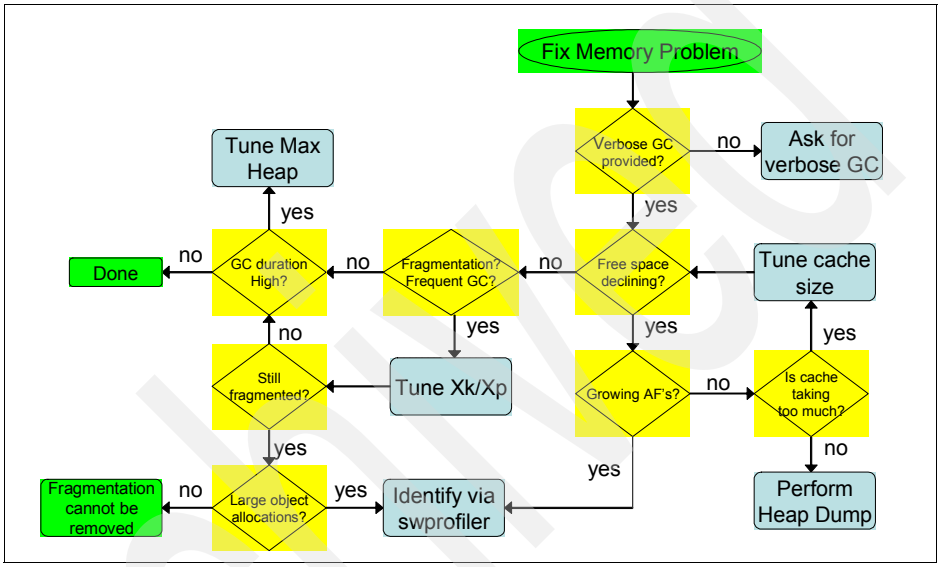


Figure 24-2 Process diagram for memory analysis methodology

24.3.1 Gather verbose Garbage Collection logs

To monitor the usage of JVM memory, get JVM verbose Garbage Collection logs from WebSphere Application Server (that is, native_stdout.log or native_stderr.log under the WebSphere Application Server Installation dir/profiles/default/logs/server1 directory). The default setting of WebSphere Application Server does not enable this, but you can enable it using the following WebSphere Application Server v6.0 example:

1. Open the WebSphere Application Server administration site by typing `http://hostname:port/ibm/console`. The port is the number of the HTTP administrative port, which is 9060 by default. Type an ID (any ID without a password) and log in to it.
2. Select **Servers** → **Application servers** → **server1** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.

3. Select **Verbose garbage collection**, as shown in Figure 24-3.
4. Click **Apply** and click **Save** at the top of this page.
5. Restart WebSphere Application Server.

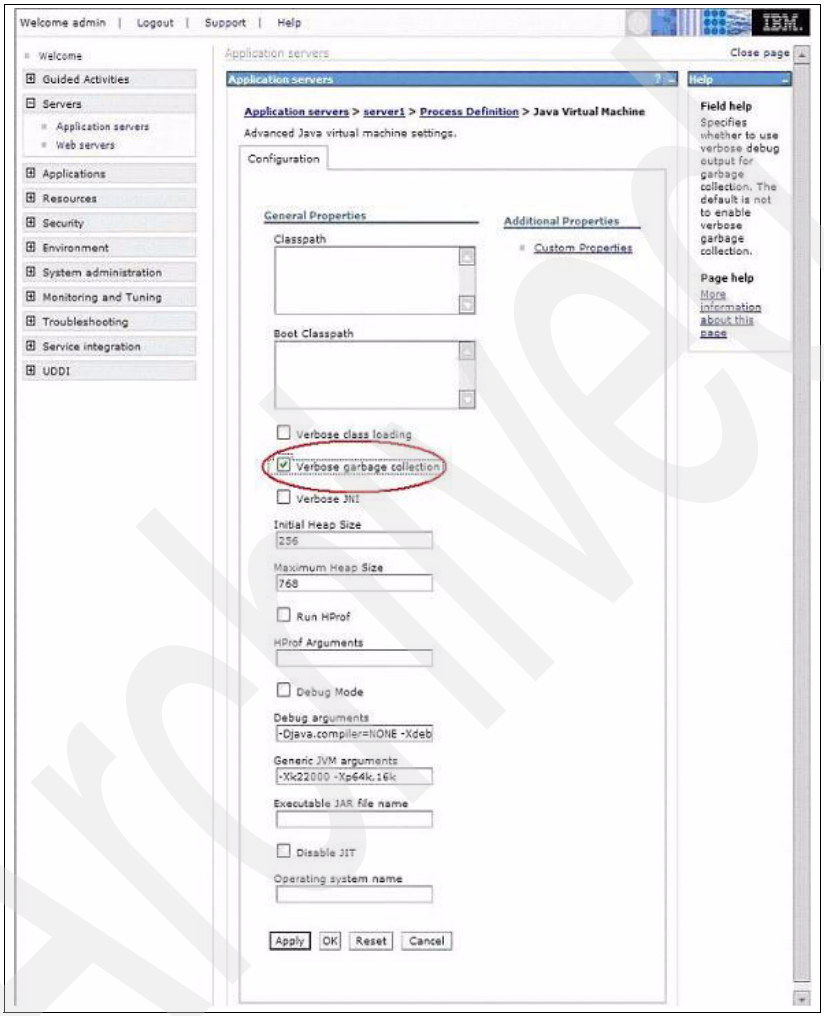


Figure 24-3 Enable verbose GC in WebSphere Application Server V6

After restarting WebSphere Application Server, you see the verbose GC output in native_stdout.log or native_stderr.log.

Archive

24.3.2 Analyzing verbose GC logs

To analyze memory problems in the application server, the first step is to gather GC information. You need a tool to analyze this information.

There are many tools for verbose GC log analysis, such as Tivoli Performance Viewer, Dump JVM (DMPJVM), and the WebSphere Resource Analyzer. These tools can abstract useful information, and illustrate the trend of JVM heap size usage over time.

After you analyze your native_stdout.log or native_stderr.log, you should generate charts with the following information:

- Occupancy (MB)
- Allocation rate (KBps)
- Total GC pause time (ms)
- Mark and sweep time (ms)
- Compact time (ms)
- GC cycle length and distribution (ms)
- Free space after GC (MB)
- Free space before AF (allocation failure) (MB)
- Size of request that caused AF (bytes)

Among these charts, some are helpful in monitoring the effects of GC and detecting many problems. You can use *GC Cycle length and distribution* to analyze GC frequency and distribution, *Free Space after GC* to analyze memory leak, and *Free Space before AF* and *Size of Request that caused AF* to analyze fragmentation or large objects. Other charts can also assist in the analysis.

Archived

Identifying memory leaks

Figure 24-4 and Figure 24-5 on page 532 show some examples of free space after GC. In a normal Free Space after GC graph, where the application is using the Java heap properly, the red line should be approximately on a horizontal line, as in Figure 24-4. In Figure 24-5 on page 532, the declining red line means that the free space available to allocate is decreasing. If you suspect a memory problem, continue running the test until an OutOfMemory exception occurs because some downward trends in free space will stabilize after a period of time. This helps you to get better support from WebSphere Application Server and the JDK if the problem is related to or the JDK.

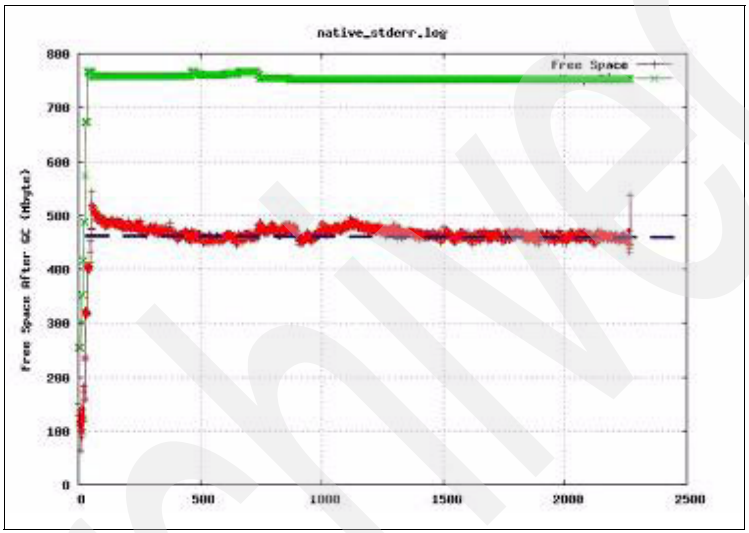


Figure 24-4 Example of normal Free Space after GC chart

Archived

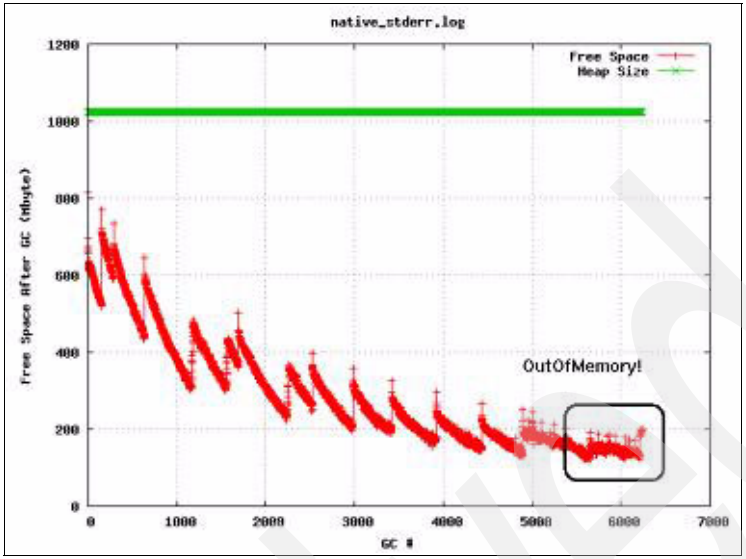


Figure 24-5 Example of Free Space after GC chart with problem

Identifying memory fragmentation

If there is a memory problem, but no reduction in free space after GC, check the charts for “Free Space Before AF” and “Size of Request that caused AF”. AF means that an object needs heap space, but there is not enough contiguous space available in the JVM heap for it. Generally, AF occurs when the JVM heap is used up. However, AF also occurs if all the free space is fragmented, so that there is no contiguous space for this object. This problem is greatly magnified if there are large object allocations within the application because it becomes unlikely for the heap to have large contiguous space for these large objects. “Free Space before AF” means the size of free space when AF occurs. This space should be a small value because the heap size is nearly used up when AF occurs. Therefore, the red line is always near the bottom of the chart. Figure 24-6 on page 533 shows normal usage without fragmentation problems.

Severe fragmentation causes frequent GC cycles, and thus performance degradation. Rising GC frequency is another indicator of fragmentation.

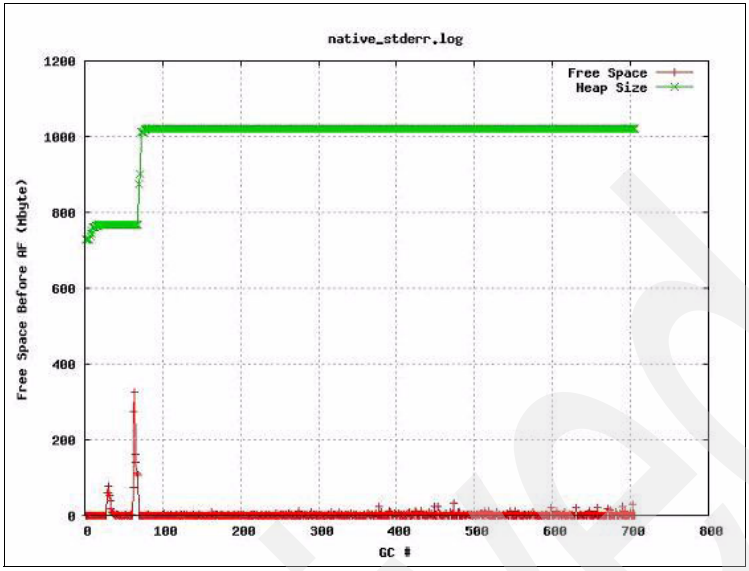


Figure 24-6 Normal "Free Space before AF" chart

24.3.3 Option 1: Tune max heap size to optimize GC frequency

If the free space after GC does not decline, check the GC cycle length and distribution, and the total GC pause time. If the time since the last AF in "GC cycle length and distribution" is not too small, but the complete time in "Total GC Pause Time" is high, it means that GC is not very frequent and GC duration is very high. The duration of each GC cycle should be monitored and not exceed 10 seconds, except for a compaction occurring within the cycle. In this situation, the heap is probably too large for the application and GC takes a long time to clean up objects in this large heap, so reduce the maximum heap size. In the other cases where GC frequency is too high, the heap is probably too small for the application and GC needs to run frequently, so increase the maximum heap size.

To tune the JVM max heap size:

1. Open the WebSphere Application Server administrative console, http://hostname:port/ibm/console, and log in.
2. Expand **Servers** → **Application servers** → **server1** → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.
3. Change the max heap size to a larger value.
4. Click **Apply** and click **Save** at the top of this page.
5. Restart WebSphere Application Server.

6. Try your test case again and see if the problem disappears.

Note: i5/OS® should have no maximum heap size according to the *System i™ Tuning Guide* because the allocation model is different from other platforms. After it is set to unlimited, it may take a couple of days to stabilize at 3 GB.

24.3.4 Tactic 2: Tune -Xk and -Xp to minimize fragmentation

If the free space after GC does not decline, but the time since the last AF in the "GC cycle length and distribution" is always small, there might be some large objects or heap fragmentations. You can try to tune the Xk/Xp parameters to remove most of the fragmentations.

kCluster, pCluster, and fragmentation

Java objects located in the Java heap are usually mobile. That is, the garbage collector can move them around if it decides to re-sequence the heap. Some objects, however, cannot be moved either permanently or temporarily. Such immovable objects are known as *pinned objects or dosed objects*. Pinned and dosed objects are the immovable objects on the Java heap. GC does not move these objects during compaction. These are the major cause of heap fragmentation.

All objects that are referenced from JNI to call external programs are pinned. Use of JDBC-2 drivers is a case in point. All objects on the heap that are referenced from the thread stacks are dosed.

In the Java SDK Release 1.3.1, Service Refresh 7 and later, the garbage collector allocates what is called a *kCluster* as the first region at the bottom of the heap. A kCluster is an area of storage that is used exclusively for class blocks. It is large enough to hold 1280 entries and each class block is 256 bytes long.

The GC then allocates a *pCluster* as the second object on the heap. A pCluster is an area of storage that is used to allocate any pinned objects. It is 16 KB long.

When the kCluster is full, the GC allocates class blocks in the *pCluster*. When the *pCluster* is full, the GC allocates a new pCluster of 2 KB. Because this new pCluster can be allocated anywhere in the heap and must be pinned, it can lead to fragmentation problems.

How fragmentation occurs

The pinned objects effectively deny the GC the ability to combine free space during heap compaction. This can result in a heap that contains a lot of free

space but in relatively small, discrete amounts, so that an allocation that appears to be well below the total free heap space fails. When the request fails, we need to run a full GC compaction to free up memory. During the compaction, processing in the JVM comes to a halt. The more frequently we run compactions, the larger the degradation of performance.

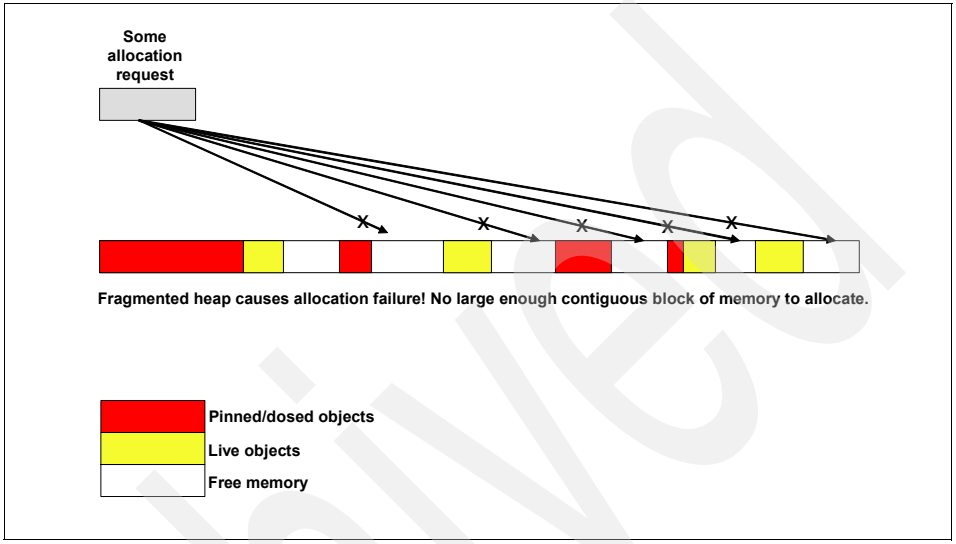


Figure 24-7 Heap fragmentation causes allocation failures (AF)

Archived

How to avoid fragmentation

Java SDK Release 1.3.1 at SR7, and later, provides command-line options to specify the size of the JVM kCluster and pCluster regions. Refer to Table 24-1 for a summary of the switches. For additional information see the IBM Java technology Web page Diagnosis Documentation, which has a section on garbage collection and performance tuning. The Web address is:

http://www.ibm.com/developerworks/java/jdk/diagnosis/142.html

Table 24-1 JVM kCluster and pCluster sizing switches

JVM region	JVM switch	
kCluster	**-Xk**	
pCluster overflowsize	**-Xp sz,ovfl**	The -Xp switch has two parameters: sz = the pCluster size parameter in KB ovfl = the overflow size parameter in KB

Set the initial sizes of the clusters large enough to help avoid fragmentation issues occurring on your Web site. It is not unusual in a large Java application, such as WebSphere Application Server, that the default kCluster space might not be sufficient to allocate all class blocks.

In Example 24-1, the pinned size valuec(4265) and classes size value (3955) are about the right size needed for the -Xk parameter. However, we recommend that you add 10% to the reported value (3955).

Example 24-1 Sample garbage collection output using -verbosegc switch

```
<GC(VFY-SUM): pinned=4265(classes=3955/freeclasses=0) dosed=10388
movable=1233792 free=5658>
```

Archived

In the preceding example, -Xk4200 is probably a reasonable setting. The difference between pinned (=4265) and classes (=3955) provides a guide for the initial size of pCluster. However, because each object might be a different size, it is difficult to predict the requirements for the pCluster and pCluster overflow options.

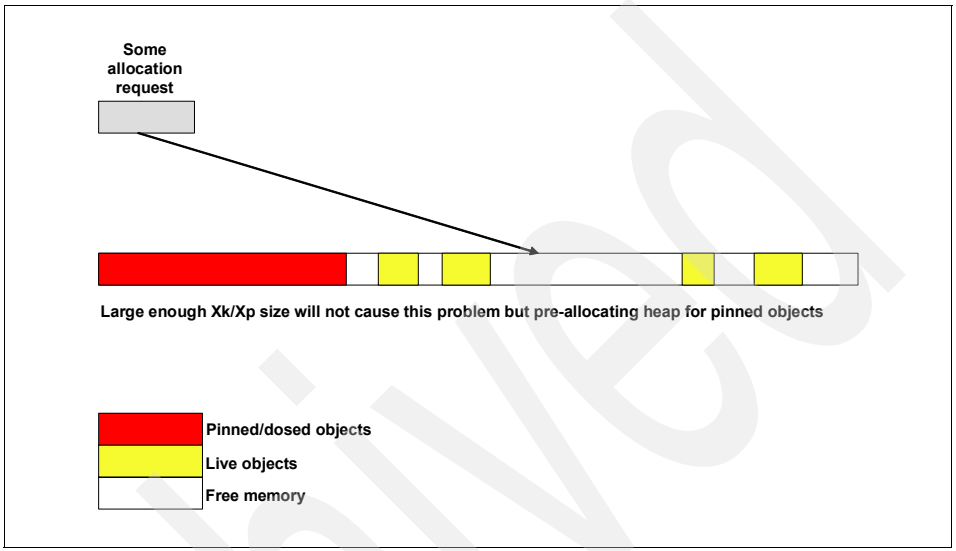


Figure 24-8 By tuning kCluster (Xk) and pCluster (Xp) size, fragmentation can be avoided

Configuring the kCluster

Set -Xk to handle objects up to the specified size using the -Xk option:

```
-Xk maxNumClass
```

Here, maxNumClass specifies the maximum number of classes that the kCluster can contain.

-Xk instructs the JVM to allocate space for maxNumClass class blocks in kCluster. The GC trace data obtained by setting -Xtgc2 can help provide a guide for the optimum value of the maxNumClasses parameter. You must keep -Xtgc2 enabled until memory fragmentation is satisfactory.

Configuring the pCluster

Specify the pCluster and pCluster overflow sizes using the -Xp command-line option:

```
-Xp sizeClusterKB[,sizeOverflowKB]
```

Archived

Here, sizeClusterKB specifies the size of the initial pCluster in KB and sizeOverflowKB optionally specifies the size of overflow (subsequent) pClusters in KB. The default values of sizeCluster and sizeOverflow are 16 KB and 2 KB, respectively. If your application suffers from heap fragmentation, turn on the GC trace (-Xtgc2) and specify the -Xk option. If the problem persists, experiment with higher initial pCluster settings and overflow pCluster sizes.

Setting Xk and Xp values in WebSphere Administration Console

Figure 24-9 is an example of setting Xk and Xp, which specifies -Xk22000 -Xp64k,16k in generic JVM arguments. If the problem persists, experiment with higher initial *pCluster* settings and overflow *pCluster* sizes. After this tuning, if there are fragmentations left, you can suspect large object problems.

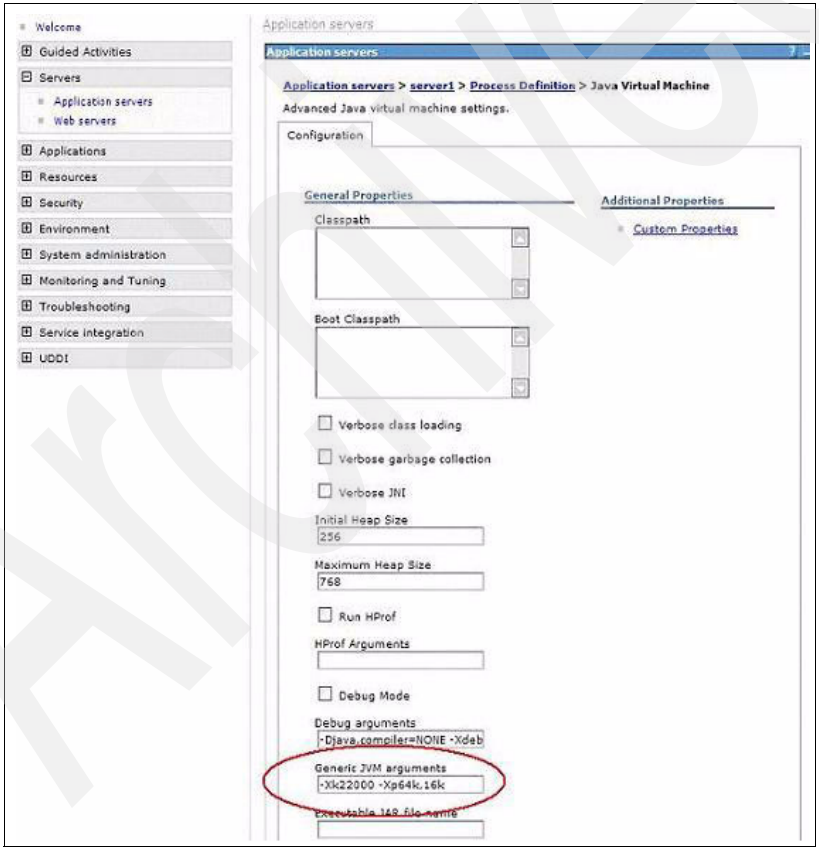


Figure 24-9 Tuning Xp/Xk parameters in WebSphere Administrative console

Archived

24.3.5 Tactic 4: identifying by swprofiler

There are two situations for which you can suspect that unusual large objects exist. One is if, after tuning Xk and Xp parameters, there are still fragmentation problems. In this situation, the free space before AF has a large value, and stays at a high level, as shown in Figure 24-10. The free space is even larger than 500 MB when AF occurs. In this case, suspect some unusual large objects.

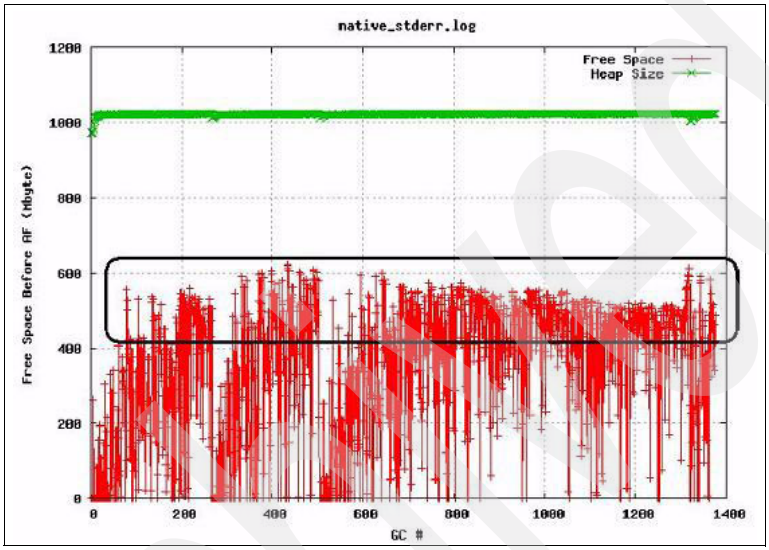


Figure 24-10 Large free space when AF occurs

Archived

In another situation, the free space after GC is declining. The AF request space is not only large, but also growing, as shown in Figure 24-11. In this case, you can also suspect problems associated with memory allocations for large objects.

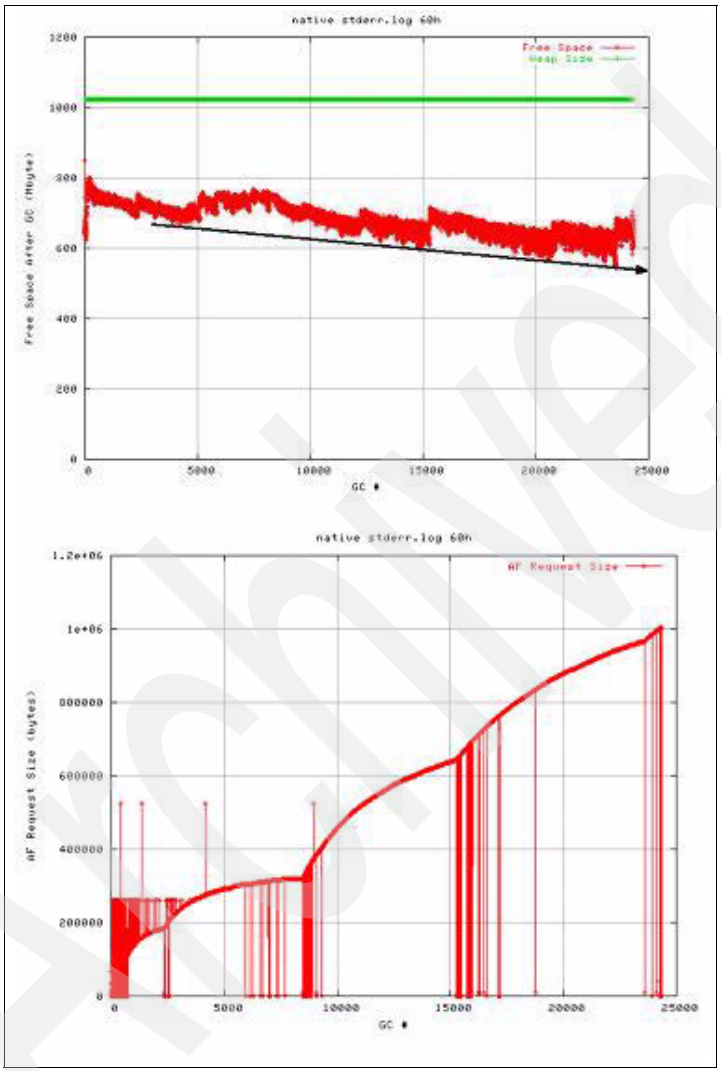


Figure 24-11 Free space after GC declining with AF request space growing

To identify the large object problem, you to use the swprofiler tool. This tool helps you print the stack information of an object by setting the allocation limit and depth. To use this tool:

1. Download the profiler.zip file and unzip it. Note that with the later version of the JDK, a built-in function has been created to replace the use of the

swprofile. For details, see Technote “How to identify the Java stack of a thread making an allocation request larger than a certain size.”

2. Get the proper lib file from the unzipped folder, and copy it to the bin path under the WebSphere Application Server installation folder. For example, copy this file to WAS_Home\java\jre\bin.
3. For an AIX, Linux, Linux on zSeries, or Windows platform, type `-Xrunswprof` in the Generic JVM arguments field in the WebSphere Application Server administrative console. If your platform is Solaris, this command is **`-Xrunallocprof`**.
4. Restart WebSphere Application Server.

You see information about the swprofiler in native_stdout.log or native_stderr.log after restarting WebSphere Application Server, such as in Example 24-2.

Example 24-2 Confirmation of profiler starting up in the log

```
Swprofiler loaded OK
Allocation limit: XXXX, Depth: YYY
```

You can try to configure this allocation limit and depth. After that, when the JVM needs to allocate an object that requires space larger than the allocation limit, the tool records this allocation in the stderr log. In WebSphere Application Server V6, to set the limit and depth values:

1. Open the WebSphere Application Server administrative console, http://hostname:port/ibm/console, and log in.
2. Expand **Servers** → **Application servers** → **server1** → **Java and Process Management** → **Process Definition** → **Custom Properties**.
3. Click **New** and add two pairs of properties and values (Example 24-3).

   Example 24-3 Sample values for allocation limit and allocation depth

   ```
   ALLOC_LIMIT 600000
   ALLOC_DEPTH 10
   ```

Archived

4. Save your changes and restart the server.

   After setting the allocation limit and depth (10 levels of thread stack in this example), you see the information about the allocation stack printed in the stderr log, as in Example 24-4.

Example 24-4 Sample allocation request

```
<AF[591]: completed in 106 ms>
Large object allocated: size 22616428
at testalloc in class com/test/OOMTest 21616424
at service in class javax/servlet/http/HttpServlet
```

The final step is to locate the suspect large object by finding `Large objects` in native_stderr.log. If your problem is with growing objects, as illustrated in Figure 24-12, find and fix objects with increasing size. After fixing the large object problems, confirm whether the new Free Space after GC, Free Space before AF, and AF Request Size graphs are normal. It is important to know that you cannot remove the fragmentation. You can only reduce or minimize it to a degree.

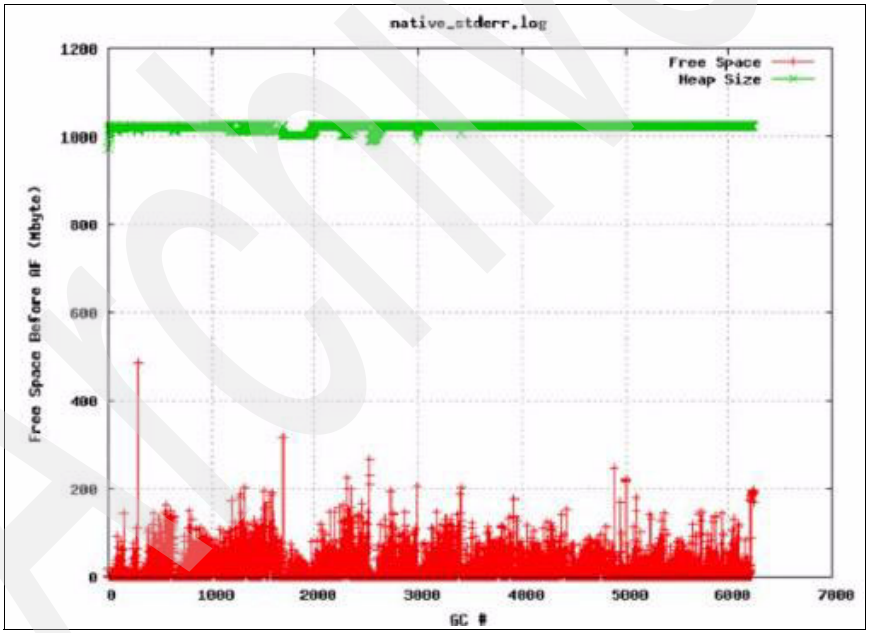


Figure 24-12 Free space before AF improves after removing suspected large objects

Archive

24.3.6 Tactic 4: tuning the cache size

If the free space after GC declines with no growing AF, then most likely cache tuning is causing the memory problems. Caching does not necessarily mean DynaCache. It includes other types of cache that are being used. One solution is reducing the in-memory cache size and letting the overflow entries use the disk cache if possible.

Sometimes an excessive number of cache objects look like a memory leak since the cache grows as the application receives an increasing load. Customers and testers need to find a stabilization point where the system does not generate OutOfMemory errors due to too many cached objects. Tune the cache size according to how much heap size is being used. In Figure 24-13, the example shows an OutOfMemory exception. After further examination, you see that the cached page is 60–100 KB in size and the number of cache entries is set to 5000. Therefore, half of this 1 GB heap is allocated to the cache.

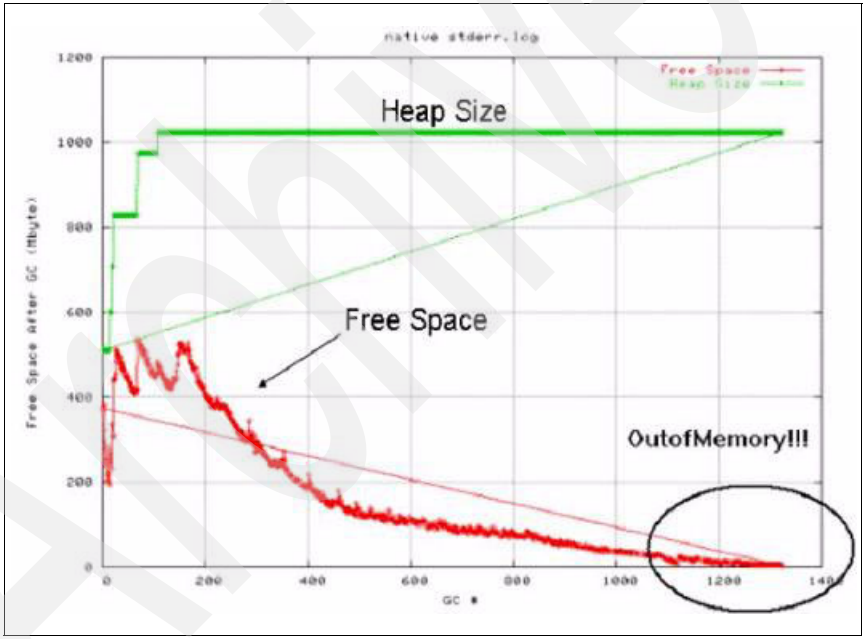


Figure 24-13 Free space reaching zero because of increasing cache size

Note: A quick way to rule out caching, such a Dynamic Caching, concerns that may work for you is to try turning-off Dynamic Caching altogether, to force disk caching, or to keep all cached objects in memory. The behavior of your application under such circumstances may reveal whether caching policies or disk offload may be contributing to your caching concerns.

Note, however, that this trial-and-error approach may change your application's error path and thus deviate from your error-causing scenarios.

In most cases, you can adjust the DynaCache size to eliminate such problems. To tune the DynaCache size:

1. Log in to the WebSphere Application Server administrative console.
2. Expand **Servers** → **Application servers** → **server1** → **Container Services** → **Dynamic Cache Service** → **Cache size**.

Archived

3. Set the value of the cache size that you want, as shown in Figure 24-14.
4. Save your changes and restart the application server.

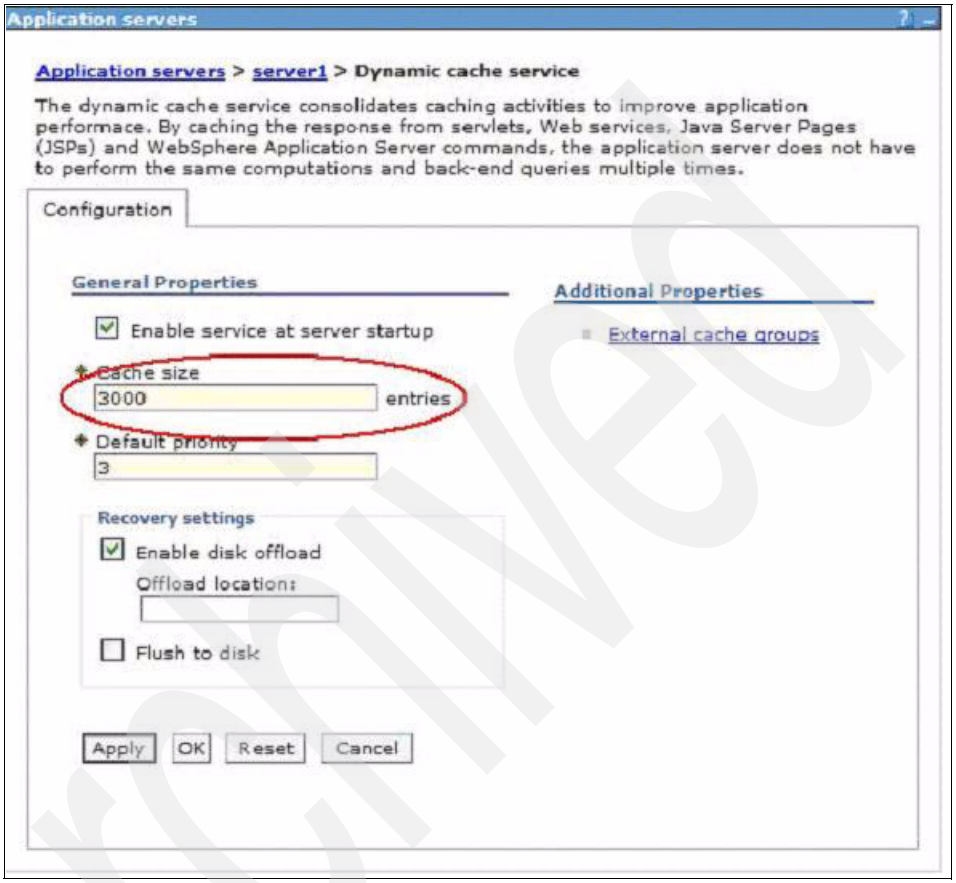


Figure 24-14 Tuning the DynaCache size

Archived

Figure 24-15 shows the memory usage of the same test case after tuning the number of cache entries to 3000, and enabling the disk offload to let the overflow entries use the disk cache. The application has stabilized at 100 MB free space in the heap after three times the duration of the first test where the OutOfMemory exception was encountered. From Figure 24-15, you can see at the beginning the cache is warming up, causing a drop in free space. However, towards the end of the free space, the cache is stabilizing, which means that it is now fully warmed.

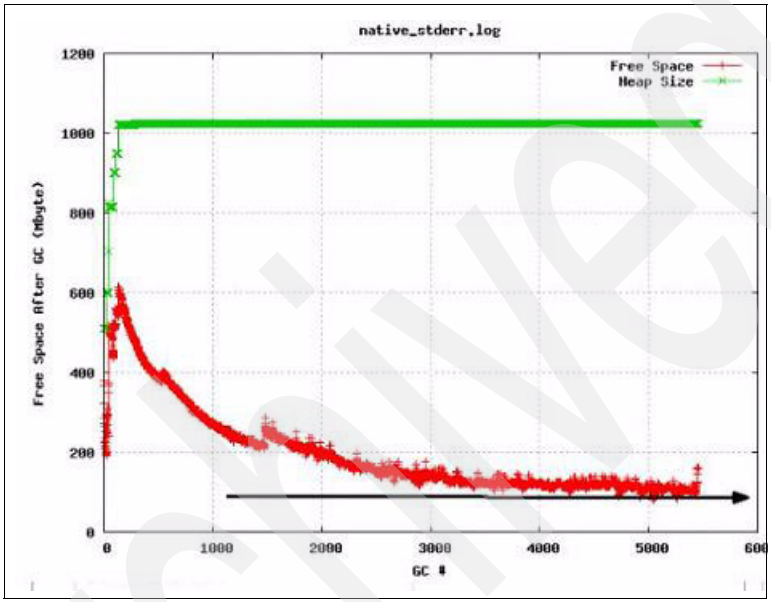


Figure 24-15 Free space after reducing cache size

24.3.7 Tactic 5: performing the heap dump

If all the charts are normal except “Free Space after GC declining,” suspect a memory leak. In this situation, we recommend performing a heap dump.

There are some tools that can help you perform this analysis. We used IBM HeapDump, a utility shipped with the IBM JDK. It lets you dump all the living objects in the Java heap into a text file called heapdump. This tool analyzes the memory usage of every Java object. This is a step to find which of them are consuming JVM space.

Archived

Here is an example of setting up the heapdump in WebSphere Application Server v6.0. To configure IBM_HeapDump, add the name and value pairs in the WebSphere Application Server administrative console, as shown in Figure 24-16.

Name	Value
IBM_HEAPDUMP	TRUE
IBM_HEAP_DUMP	TRUE
IBM_HEAPDUMPDIR	<your directory for heapdump output file>
IBM_HEAPDUMP_OUTOFMEMORY	TRUE
IBM_JAVADUMP_OUTOFMEMORY	TRUE
IBM_JAVA_HEAPDUMP_TEXT	TRUE

Figure 24-16 Name and value pairs

To do this:

1. In the administrative console, open **Servers** → **Application Servers** → **server_name** → **Java and Process Management** → **Process Definition** → **Environment Entries** → **New**. From there, you can set these name-value pairs. This setting is specific for WebSphere Application Server v6.0. If you do not set the IBM_HEAPDUMPDIR, the default output directory is the root directory of your application server.
2. Save your modification and restart the application server. Record the JVM PID of your application server process.
3. At the time when you want to collect a heapdump, you can signal it by running kill -3 JVM_PID. This generates files named heapdump.date.time.pid.txt and javacore.date.time.pid.txt under your IBM_HEAPDUMPDIR. This tool signals multiple heapdump points at the point after a memory leak occurs. This is important to analyze the root cause of the leak.

The commands for generating the heapdump file are different on different operating systems:

- On Solaris, use:

  ```
  kill -HUP JVM_PID
  ```

- On most UNIX platforms:

  ```
  use kill -3 JVM_PID
  ```

- On a Windows system, there is a sequence of commands that causes a heapdump, as shown in Example 24-5.

Example 24-5 Forcing heap dump on Windows system

```
WAS_HOME\profiles\instance_name\bin\wsadmin.bat
wsadmin>set jvm [$AdminControl completeObjectName  type=JVM,process=server1,*]
wsadmin>$AdminControl invoke $jvm dumpThreads
```

With the generated heapdump and javacore files, you can analyze what caused this memory leak problem. Various analysis tools exist to help with this investigation. One useful tool is the Memory Dump Diagnostic for Java tool. To acquire you must download the IBM Support Assistant from the following Web site and perform the following steps:

http://www-306.ibm.com/software/support/isa/

When the heap dump is available, run the Memory Dump Diagnostic for Java tool:

1. Start IBM Support Assistant.
2. In IBM Support Assistant, select the **Tools** tab.
3. On the left side, click **WebSphere 6.1**.
4. On the right side, click **Memory Dump Diagnostic for Java**.

The Memory Dump Diagnostic tool displays a list of leak candidates after analyzing the heapdump files. This list contains the suspicious object paths, as shown in Figure 24-17.

Figure 24-17 Example of Memory Dump Diagnostic analysis result

Archive

24.4 Solving throughput and response time problems

This section focuses on the analysis of throughput problems for WebSphere Commerce applications and provides guidelines that have been proven to be effective and efficient from our work experience. Here, we also describe a general methodology for diagnosing WebSphere throughput degradation problems. It also provides suggestions on how to solve them to improve performance. It contains three main sections:

- How to identify throughput problems in a performance test: This section introduces the main indicators of throughput degradation found in performance testing.
- How to analyze and solve throughput problems: This section introduces a general methodology on how to deal with throughput degradation, explains the detailed working process of analyzing throughput degradation, and provides possible solutions.
- Example of gradual throughput degradation (GTD) analysis and solution: This section takes you through a GTD example.

24.4.1 Identifying throughput problems in performance testing

In performance testing, testers may encounter throughput or response time problems that seriously impact the application's performance. Identifying the throughput degradation problems and analyzing them to come up with the corresponding solution is an important task for WebSphere Commerce application developers and testers. With these throughput degradation problems solved, the performance of the WebSphere application improves significantly.

- Lower throughput or higher response times

  This refers to throughput being lower than the required target or the response time being higher than the target. Code change, performance tuning, or hardware scalability may be able to increase the performance to your desired target.

- Throughput or response time degradation

  This behavior is similar to lower throughput or higher response time, as mentioned above, except that there is a degradation or *regression* compared to a certain baseline established in the past. The baseline could be on another release of some product in your software stack or on another code release. Again, code change, performance tuning, or hardware scalability may be able to increase the performance to your desired target. In this case you have another tactic for troubleshooting, and that is the comparison with the previous test execution.

Archived

- Gradual throughput (or response time) degradation (GTD)

  This is a systemic, and potentially complex, problem that you may encounter in performance testing. The main problem is that during the test interval (say, 3 hours), the throughput decreases gradually, or the response time gets longer and longer, and this trend does not stabalize at any acceptable level. You can easily find this trend in your test report. If the downward slope is not significant then you may have to either stress your system more or let the test run for a longer duration to observe a noticeable depreciation.

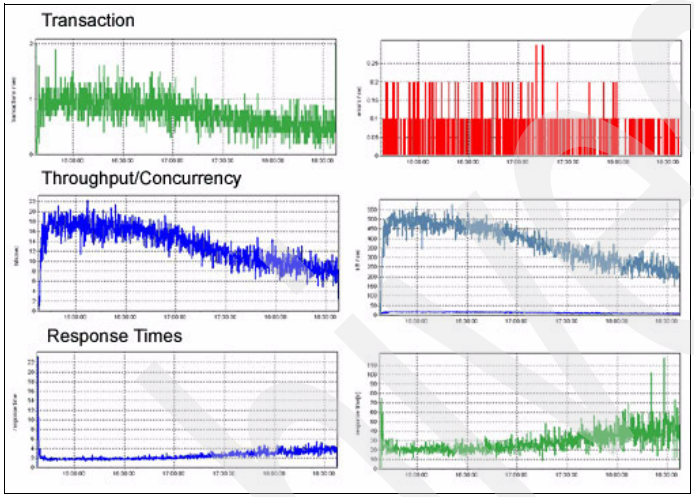


Figure 24-18 Throughput degradation problems in the stress test tool report

24.4.2 Analyzing and solving throughput problems

This section summarizes our throughput analysis methodology, and then gives a detailed working process of how to analyze and solve the throughput degradation problems.

Throughput analysis methodology

The main causes of throughput degradation can be anything from code issues, database issues, configuration issues, to test data or method issues. We can use test reports to identify throughput degradation easily, but finding its root cause usually requires thorough investigation.

Here we summarize a throughput analysis methodology:

- The methodology does not cover all database problems. Generally, database problems can refer to any bottleneck due to database, including SQL queries, database tuning parameters, indexing issues, and data distribution problems.

- Figure 24-19 assumes that a previous performance baseline has been established. To set a throughput baseline for a test case, and if the scenario has been tested in a previous release or version of the site, we often use its previous result as our baseline. If the scenario is new, we often use the final actual test result in the new release to set up our baseline for future comparison.
- In Figure 24-19, *divide and conquer* means running with smaller scenarios, such as home page only, logon only, and browse only, to isolate the problematic portion of the scenario, rather than investigating the whole end-to-end run.
- Cost/SQL is the execution cost per SQL query. Fetch time is not recorded in execution costs.
- Access plans can change based on accumulated data. You can use DB2 Explain utilities to find more clues.

Figure 24-19 is the recommended scheme for throughput analysis. For a detailed description of each step, refer to the section below, ““Throughput degradation analysis and solution” on page 552.

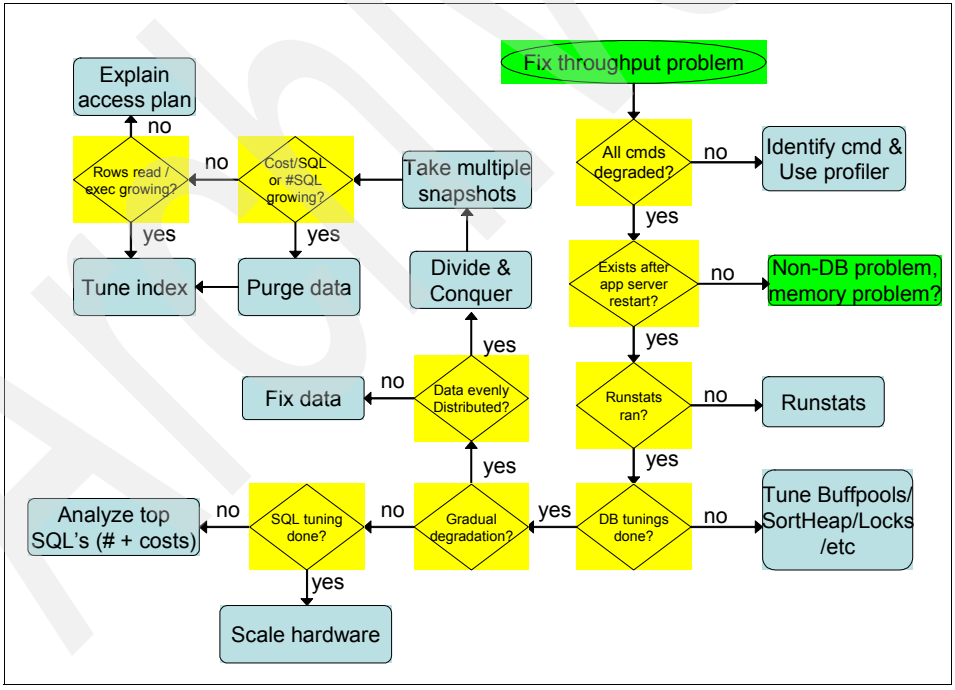


Figure 24-19 Throughput analysis methodology

Throughput degradation analysis and solution

When encountering a throughput problem in performance testing, follow these steps to analyze and solve the problem. First, we should identify whether the problem is a low throughput problem, a throughput degradation problem, or a GTD problem. The main identification method is to check whether there is a downward trend for throughput charts in the test reports, or an upward trend for the response time during the test. If so, it is a gradual throughput degradation problem, as shown in Figure 24-18 on page 550. Otherwise, it is a low throughput problem, which may or may not be solved by performance tuning alone, and you may need to scale your hardware.

You can start analyzing the problem using this detailed process:

1. Check to see whether all WebSphere application commands degrade when compared to the baseline result. You can do this by checking the average response time of all the commands in our test report. If only certain commands are slow, it usually means a design problem or code issue, and you can use the RAD profiler or an equivalent Java profiler to pinpoint the culprit in the code.
2. Check to see whether throughput degradation exists after restarting WebSphere Application Server. If the problem is resolved after a server restart, it probably relates to a non-database-specific problem. The may be a memory problem, such as memory leak, heap fragmentation, or large object allocation. In some cases, the problem may be caused by a WebSphere Application Server defect, in which case you need to involve its service team. However, if you cannot solve the problem by restarting the server, go to step 3 to continue the analysis.
3. Check to see whether the database has been optimized. For DB2, check whether runstats has been run on the DB server. If not, start runstats. Runstats is important to improve DB2 performance when the data volume is large or the system has been running for a long time. Runstats can also help to optimize the DB2 access plan, which makes DB2 more efficient. For Oracle, you can optimize database performance with the following command, where you need to provide your schema name:

   ```
   execute dbms_utility.analyze_schema ('schema_name', 'COMPUTE').
   ```

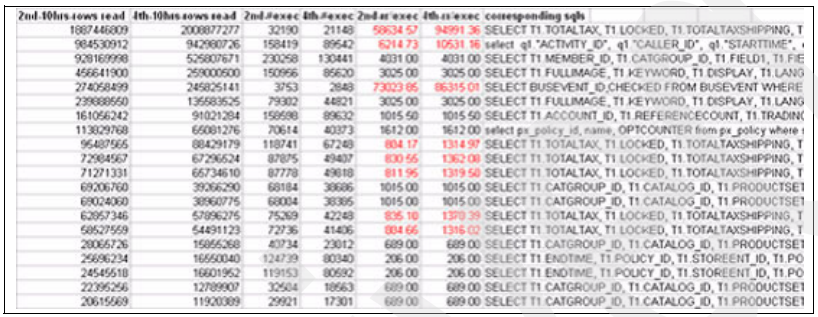
   This article mainly uses DB2 and runstats as our example.
4. Check to see whether database tuning has been done. If not, try to tune DB2 parameters. The available tuning objectives include bufferpools, sortheap, and locks. If the problem persists after DB tuning, there are two possible problems:
   - If the throughput is not gradual degradation, examine the DB2 snapshot file to analyze the status of the top SQL queries (the number of executions and costs of each query), and to find which query is causing the problem.

- If the throughput is a gradual degradation (the focus of this article), go to step 5 to continue the analysis.

5. Check to see whether the data is evenly distributed. If not, fix the data problem. Unevenly distributed data is created by improper warming up or unbalanced operation during the test. For example, the tester uses some fixed users to place an order in the warm-up, which creates thousands of orders related to these users in the database. On the other hand, if the tester only uses some fixed users to place orders in the formal testing, the corresponding data will accumulate. This makes DB2 queries use table scans instead of index scans and degrades database performance. The better method is to omit the warm-up users from the formal test and select the random users from the bigger user scale. For example, select 20 users randomly from 400 users to do both warm-up and formal testing.
6. For divide and conquer, try to use the smallest scenarios to reproduce the problem. This can isolate the scale of the possible causes of throughput degradation. To accomplish this, divide the test scenarios into different groups and test separately, find the groups that caused throughput degradation, then divide those groups again and again until you narrow down to the minimum scenario group that caused the problem.
7. Run the scenarios confirmed in step 6 for a long time and take multiple snapshots during the test. For example, take a 10-hour snapshot separately in the first and second day, and then compare these two snapshots.
8. Compare multiple snapshots to see whether the cost/SQL and execution number of SQL queries are growing. If yes, purge the data and tune the index, if needed. Through comparing these files, you can identify the top-growing cost queries. The cost can be one of the following metrics: execution time per query execution, user CPU time per query execution, and system CPU time per query execution. Note that other costs such as *fetch time* are not included in the execution time reported by the snapshot. Notice that you must search the same SQL query in multiple snapshots to find what is *growing*.
9. If the cost/SQL entries are all constant, compare the snapshots to see whether rows read/execution is growing for some SQL queries. If so, try to tune the corresponding index to improve the performance. If not, analyze the access plan (for example, using DB2 Explain utilities) to see which can be amended.
10. In steps 8 and 9, you can identify the top queries that have growing cost/SQL or growing rows read per execution. Usually, these two characteristics of identified queries are the main indicators for performance degradation. To solve these problems, drop the extraneous index, add a new index, or periodically clean out the large volume of obsolete data in some tables. If

there are no growing cost SQL queries in the snapshot, analyze the access plan to see which can be amended based on the accumulated data.

Figure 24-20 is an example that reflects the statistics of rows read/execution per SQL through comparing snapshots. We made this chart by comparing two snapshots in a four-day test. One is a 10-hour snapshot taken on day 2 and the other is a 10-hour snapshot taken on day 4. The numbers in red mean that corresponding queries have growing costs and need to be amended.



2nd.10hrs.rows read	4th.10hrs.rows read	2nd./exec	4th./exec	2nd.rr/exec	4th.rr/exec	corresponding sqls
1887446809	2008877277	32190	21148	58634.57	94991.36	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSHIPPING, T
984530912	942980726	158419	89542	6214.73	10531.16	select q1."ACTIVITY_ID", q1."CALLER_ID", q1."STARTTIME",
928169998	525807671	230258	130441	4031.00	4031.00	SELECT T1.MEMBER_ID, T1.CATGROUP_ID, T1.FIELD1, T1.FIE
456641900	259000500	150966	85620	3025.00	3025.00	SELECT T1.FULLIMAGE, T1.KEYWORD, T1.DISPLAY, T1.LANG
274058499	245825141	3753	2848	73023.65	86315.01	SELECT BUSEVENT_ID,CHECKED FROM BUSEVENT WHERE
239088550	135583525	79302	44821	3025.00	3025.00	SELECT T1.FULLIMAGE, T1.KEYWORD, T1.DISPLAY, T1.LANG
161056242	91021284	158598	89632	1015.50	1015.50	SELECT T1.ACCOUNT_ID, T1.REFERENCECOUNT, T1.TRADING
113829768	65081276	70614	40373	1612.00	1612.00	select px_policy_id, name, OPTCOUNTER from px_policy where
96487565	88429179	118741	67248	804.17	1314.97	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSHIPPING, T
72984567	67296524	87875	49407	830.55	1362.08	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSHIPPING, T
71271331	65734610	87778	49818	811.95	1319.50	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSHIPPING, T
69206760	39266290	68184	38686	1015.00	1015.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRODUCTSE1
69024060	38960775	68004	38385	1015.00	1015.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRODUCTSE1
62957346	57896275	75269	42248	835.10	1370.39	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSHIPPING, T
58527559	54491123	72736	41406	804.66	1316.02	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSHIPPING, T
28066726	15865268	40734	23012	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRODUCTSE1
25696234	16550040	124739	80340	206.00	206.00	SELECT T1.ENDTIME, T1.POLICY_ID, T1.STOREENT_ID, T1.PO
24545518	16601952	119153	80592	206.00	206.00	SELECT T1.ENDTIME, T1.POLICY_ID, T1.STOREENT_ID, T1.PO
22395256	12789907	32504	18563	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRODUCTSE1
20615569	11920369	29921	17301	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRODUCTSE1

Figure 24-20 Example of identifying growing cost SQL queries

Example of throughput degradation analysis

This section introduces an example of throughput degradation that we encountered while testing a sample WebSphere Commerce application. The test was run on an AIX platform in a 1-node test environment (for example, DB2, IBM HTTP Server, and WebSphere Application Server were installed on the same machine). This throughput degradation problem could be observed in a 3-hour stress comparison test. Based on the test result from the previous code release, we set the throughput baseline at 11,580 transactions per hour and estimated a 10% performance improvement in the new release. The test target of this case was set at 12,738 transactions per hour.

Finding throughput degradation

In the first run, we checked the report of our stress test tool. The throughput curve looked fairly straight and the degradation was not obvious at first. The rate of degradation was around 5% per hour. However, with such a degradation rate, in a few days the application cannot handle any load with reasonable response times. Figure 24-21 shows the throughput chart in the stress test report.

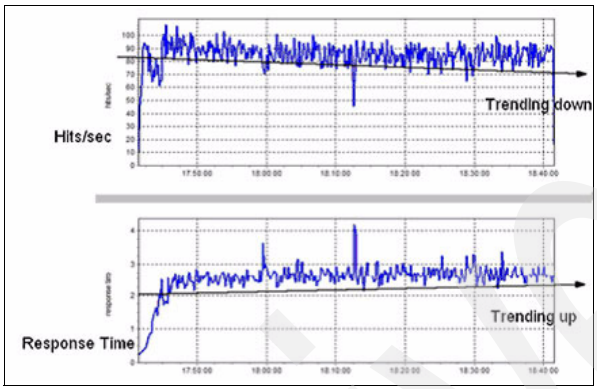


Figure 24-21 Find throughput degradation in stress test tool report

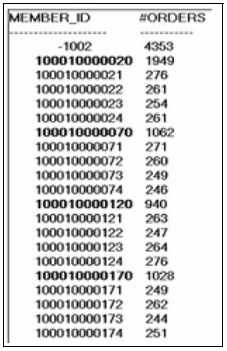
This was a gradual throughput degradation (GTD) problem. We followed steps 1 to 3 to analyze the problem and got the following results:

- All WebSphere Commerce commands were degraded, especially commands corresponding to orders.
- When analyzing native_stderr.log, we found that the GC cycles were fine, and there were no memory issues, such as memory leak, heap fragmentation, and large object allocation.
- After restarting the applications server, the throughput degradation still existed.
- Runstats and common DB2 tuning had been done.

In step 4, we doubled SORTHEAP to 2048 to avoid sort overflows because some queries needed to create temporary tables.

Archived

In step 5, when checking the database we found some data issues. For example, after running this query, `select member_id, count(*) from orders group by member_id having count(*) > 50`, we got the result shown in Figure 24-22.



MEMBER_ID	#ORDERS
-1002	4353
100010000020	1949
100010000021	276
100010000022	261
100010000023	254
100010000024	261
100010000070	1062
100010000071	271
100010000072	260
100010000073	249
100010000074	246
100010000120	940
100010000121	263
100010000122	247
100010000123	264
100010000124	276
100010000170	1028
100010000171	249
100010000172	262
100010000173	244
100010000174	251

Figure 24-22 Data issue found in the database

The data in Figure 24-23 on page 557 indicates that four users had more orders than other users. After reviewing our test steps, we found the causes of this problem were:

- In the warm-up, we often used four fixed users to run scenarios.
- There were 200 users in the database, but only 20 users were selected to run scenarios in the formal test.

The above two factors made the data distribution uneven in the database, so our solution was in two parts:

- Increase the number of users from 200 to 400.
- In the warm-up and formal test, randomly select each virtual user from 20 non-overlapped users. After these changes, the data issues did not occur.

In step 6, we narrowed the scenarios to order the shopping flow.

In step 7, we kept the test running for four days and took two 10-hour snapshots separately on day 2 and day 4.

In step 8, after comparing the two snapshots, we found no growing costs/execution for any query.

Archived

In step 9, after comparing the two snapshots, we identified that the top SQL queries had growing rows read per execution, shown in red in Figure 24-23.

2nd-#exec	4th-#exec	2nd-rr/exec	4th-rr/exec	
32190	21148	58634.57	94991.36	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSH
158419	89542	6214.73	10531.16	select q1."ACTIVITY_ID", q1."CALLER_ID", q1."STAI
230258	130441	4031.00	4031.00	SELECT T1.MEMBER_ID, T1.CATGROUP_ID, T1.FIEL
150956	85620	3025.00	3025.00	SELECT T1.FULLIMAGE, T1.KEYWORD, T1.DISPLAY
3753	2848	73023.85	86315.01	SELECT BUSEVENT_ID,CHECKED FROM BUSEVEN
79302	44821	3025.00	3025.00	SELECT T1.FULLIMAGE, T1.KEYWORD, T1.DISPLAY
158598	89632	1015.50	1015.50	SELECT T1.ACCOUNT_ID, T1.REFERENCECOUNT, T
70614	40373	1612.00	1612.00	select px_policy_id, name, OPTCOUNTER from px_pol
118741	67248	804.17	1314.97	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSH
87875	49407	830.55	1362.08	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSH
87778	49818	811.95	1319.50	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSH
68184	38686	1015.00	1015.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRC
68004	38385	1015.00	1015.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRC
75269	42248	835.10	1370.39	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSH
72736	41406	804.66	1316.02	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSH
40734	23012	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRC
124739	80340	206.00	206.00	SELECT T1.ENDTIME, T1.POLICY_ID, T1.STOREENT
119153	80592	206.00	206.00	SELECT T1.ENDTIME, T1.POLICY_ID, T1.STOREENT
32504	18563	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRC
29921	17301	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRC
29687	16788	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRC
29678	16877	689.00	689.00	SELECT T1.CATGROUP_ID, T1.CATALOG_ID, T1.PRC
87875	49407	51.00	51.00	SELECT T1.TOTALTAX, T1.LOCKED, T1.TOTALTAXSH

Figure 24-23 Top SQL with growing rows read per execution

Identifying these growing SQLs gave us clues to the solution of this throughput degradation problem. Throughput had increased based on these events:

- We dropped extraneous index MEMBER_ID+TYPE+STOREENT_ID on the ORDERS table, so queries will use the correct index MEMBER_ID+STATUS+STOREENT_ID.
- We created the CHECKED index for the BUSEVENT table.
- We periodically cleaned the CTXMGMT/BUSEVENT table in the test.

Figure 24-24 shows the same SQL execution status after we applied the modifications just mentioned. Most of the growing rows read per execution have been solved.

1st-rows read	2nd-rows read	1st-#exec	2nd-#exec	1st-rr/exec	2nd-rr/exec	
3377	2838	3378	2839	1.00	1.00	SELECT T1.TOTALTAX, T1.LOCI
252064	314758	20589	17482	12.24	18.00	select q1."ACTIVITY_ID", q1."C
5457	4738	5467	4747	1.00	1.00	SELECT BUSEVENT_ID,CHECK
11269	9510	16359	13983	0.69	0.68	SELECT T1.TOTALTAX, T1.LOCI
14102	16263	6787	5751	2.08	2.83	SELECT T1.TOTALTAX, T1.LOCI
74873	90367	12641	10717	5.92	8.43	SELECT T1.TOTALTAX, T1.LOCI
11827	13479	5436	4598	2.18	2.93	SELECT T1.TOTALTAX, T1.LOCI
8418	7242	11634	10028	0.72	0.72	SELECT T1.TOTALTAX, T1.LOCI
138662825	170820878	6787	5751	20430.65	29702.81	SELECT T1.TOTALTAX, T1.LOCI
20640	17554	20640	17554	1.00	1.00	SELECT T1.ACTIVITY_ID, T1.CA

Figure 24-24 Growing rows read per execution solved

After fixing the growing SQL costs, the throughput seemed stable for the first three hours, as shown in Figure 24-25.

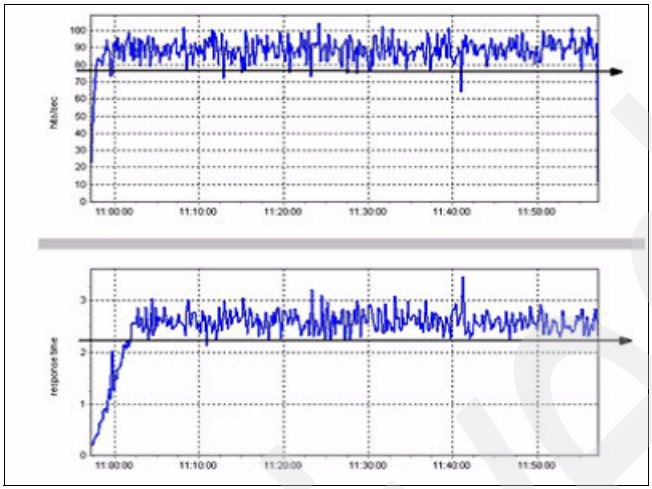


Figure 24-25 Throughput was stable in the first three hours

However, the degradation still existed in the long run, as shown in Figure 24-26.

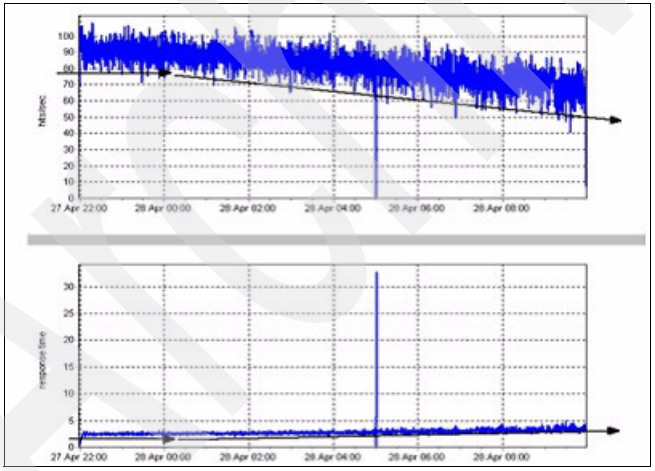


Figure 24-26 Throughput degraded in the long run

Then we analyzed the access plan and decided to do runstats and rebind the database during a long-running test. Figure 24-27 and Figure 24-28 on page 560 show how runstats helped to optimize the access plan. In this example, access plan overall costs do not include the actual fetch time.

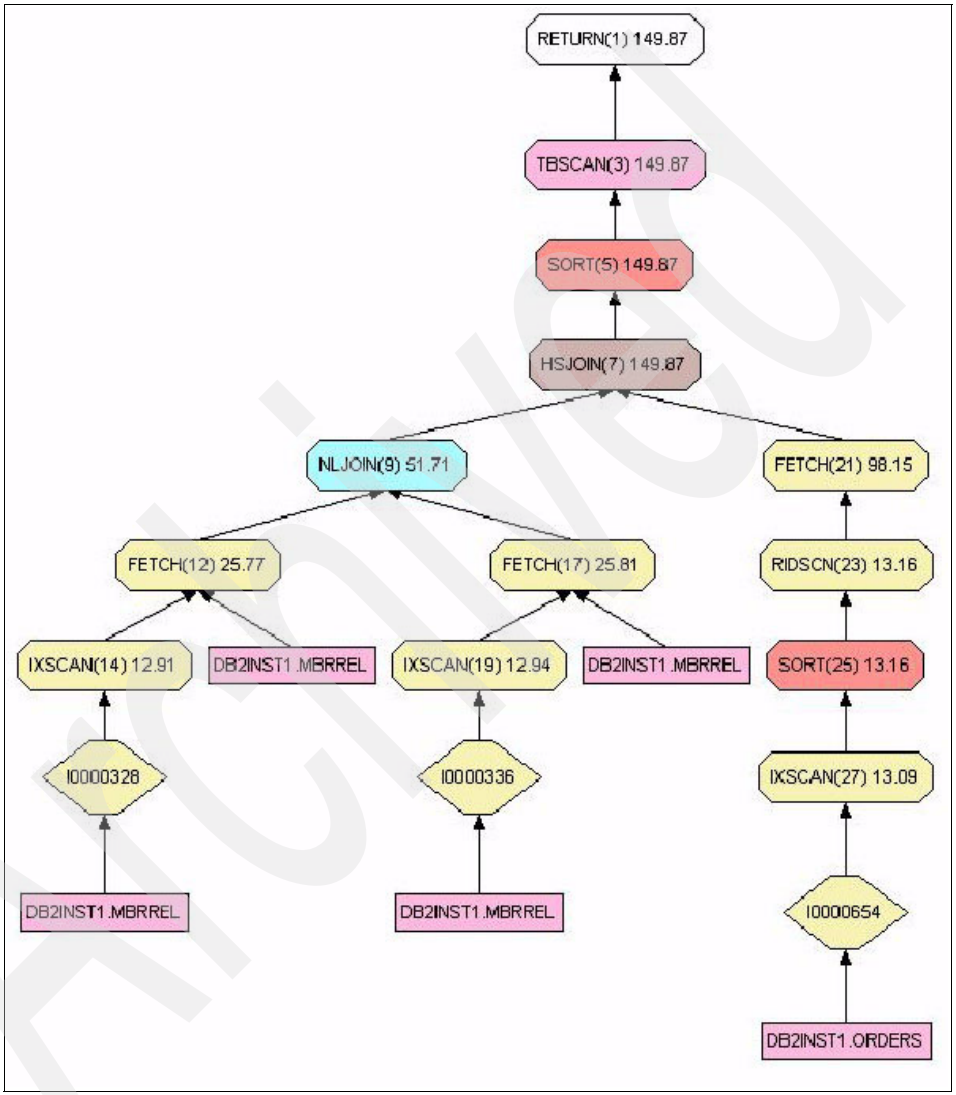


Figure 24-27 Access plan before runstats

Archive

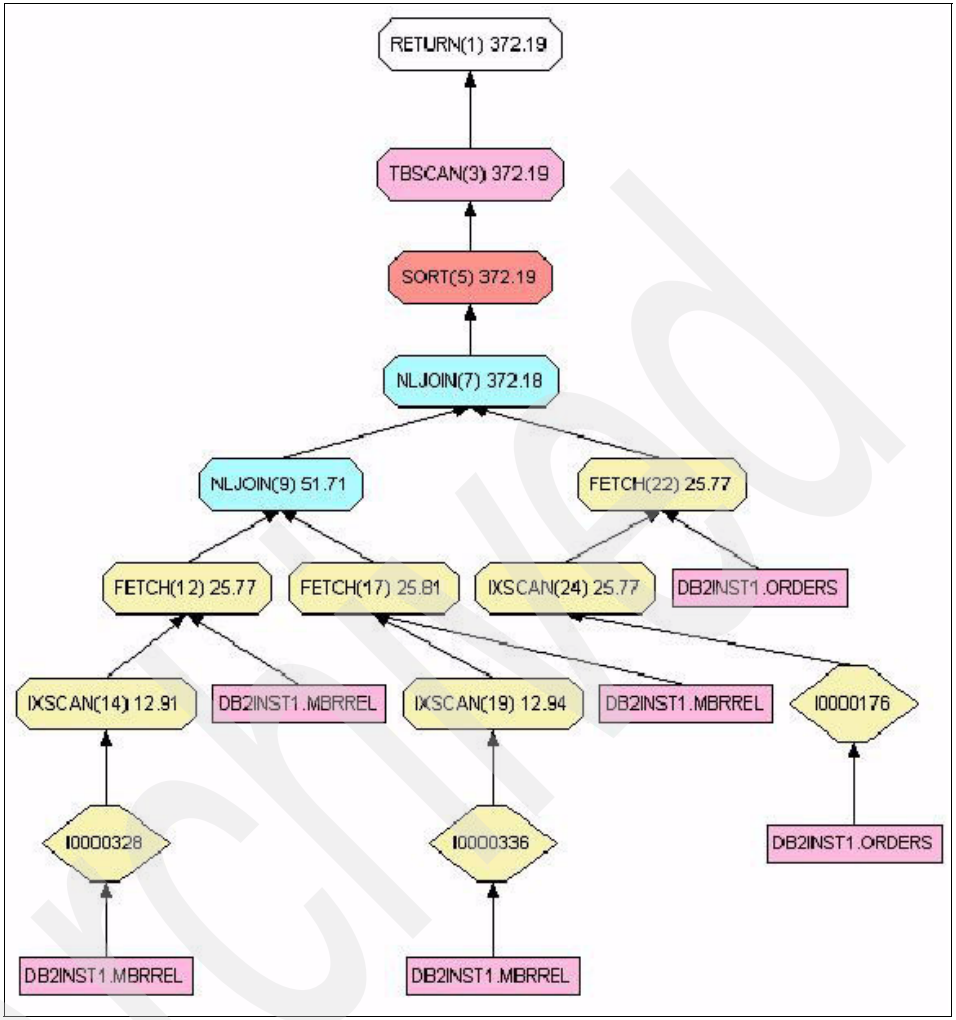


Figure 24-28 Access plan after runstats

After we did runstats and a rebind of the database in the middle of a long-running test, the overall throughput had been stabilized again and the throughput degradation problem was solved successfully. Therefore, running runstats and rebinding the database should be done regularly to ensure that database indexes are not out of date.

Archived

Part 7

Maintenance

In Part 7, we complete the HA and performance messages conveyed in this book by discussing the maintenance of those tier environments that support WebSphere Commerce. For example, the maintenance of the:

- Databases
- WebSphere Application Server tier
- Web servers
- Load Balancer

© Copyright IBM Corp. 2008. All rights reserved.

Archived

25

Database maintenance

This chapter discusses the topics related to database maintenance. IBM DB2 Universal Database has already delivered some useful utilities to help customers maintain their database, no matter whether for a test server or a production server. Besides that, WebSphere Commerce also deploys essential utilities to support High Availability and high performance in the database tier.

Archived

© Copyright IBM Corp. 2008. All rights reserved.

25.1 DB2 database maintenance in WebSphere Commerce

A WebSphere Commerce site will suffer significant performance degradation if the database is not being properly maintained. This section introduces guidelines for tasks that are required to maintain a WebSphere Commerce DB2 database.

25.1.1 DB2 database maintenance utilities

The general DB2 database maintenance flow is shown in Figure 25-1.

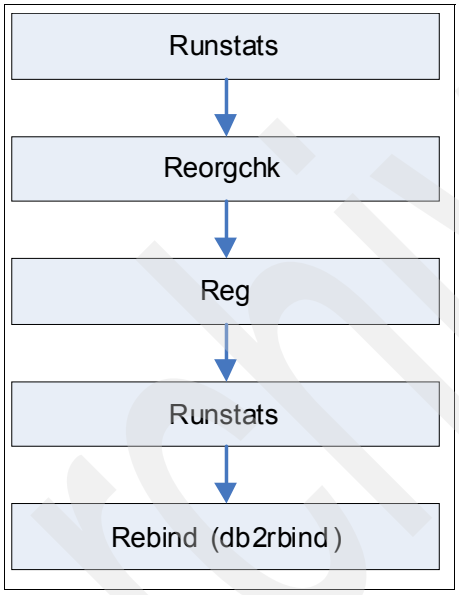


Figure 25-1 DB2 general maintenance flow

Runstats

The RUNSTATS command will update the statistics that are used by the optimizer when determining access paths to the data. If the statistics are not up to date, the system will suffer performance degradation.

Most WebSphere Commerce sites update statistics on a daily or weekly basis (generally over the weekend). You should also consider updating the statistics after schema changes or massive update or load (such as refreshing the catalog). Remember to run db2rbind after RUNSTATS so that the static packages can take advantage of the new statistics.

Archived

The following SQL statements can be used to identify when the statistics were last updated for all the tables and indexes on the database:

```
db2 "select tabschema, tabname, stats_time from syscat.tables order by
stats_time asc"
db2 "select indschema, indname, tabschema, tabname, colnames,
stats_time from syscat.indexes order by stats_time asc"
```

You might be able to get some output as in Example 25-1.

Example 25-1 Sample database statistics output

```
DB2INST1 CATENTSHIP 2007-07-11-19.27.18.034540
DB2INST1 CATENTTYPE 2007-07-11-19.27.18.120666
DB2INST1 CATGPCALCD 2007-07-11-19.27.18.185215
DB2INST1 CATGPENREL 2007-07-11-19.27.19.254761
DB2INST1 CATGROUP 2007-07-11-19.27.19.359756
DB2INST1 CATGRPATTR 2007-07-11-19.27.19.554302
DB2INST1 CATGRPDESC 2007-07-11-19.27.19.631565
```

Depending on the size of your database, you might need to use a finer- grained method for updating statistics, such as profiling or sampling. For more information, guidelines, and examples, see the topic on the RUNSTATS command in the DB2 Information Center:

http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/core/r0001980.htm

Or you can refer to the article titled “RUNSTATS in DB2 Universal Database, Version 8.2” on the developerWorks Web site:

http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0412pay/

In general, you should perform RUNSTATS on tables and indexes in the following situations:

- After data has been loaded into the database (for example, massload)
- After a table has been reorganized with the REORG utility
- After the table and its indexes have been extensively updated by data updates, deletions, and insertions (for example, stage propagation, dbclean)

Reorgchk and Reorg

The REORG command reorganizes a table by compacting information and reconstructing the rows to eliminate fragmented data. The REORGCHK utility uses different algorithms to find the tables and indexes that need to be reorganized. The REORGCHK command will output a table listing all the table

and index objects. An asterisk (*) on the REORG column will indicate that the calculated results exceed the set bounds of its corresponding formula, and that the table might need to be reorganized.

Unless you use the “CURRENT STATISTICS” specifier, the REORGCHK command will update the statistics for all the objects on the database. If you are running REORGCHK after RUNSTATS, you can use the “CURRENT STATISTICS” specifier to avoid updating the statics twice. You can also omit RUNSTATS and have REORGCHK update the statics, but this method provides less flexibility.

Once the tables or indexes to be reorganized have been identified using the REORGCHK command, the REORG command has to be explicitly invoked for each object. DB2 does not offer a way to automatically reorganize all the tables or indexes that were identified by REORGCHK.

Table reorganization is commonly performed in any one of the following ways:

- DBA to run REORG for each table identified by REORGCHK
- Explicitly executing REORG for the tables and indexes that are most likely to need a reorganization (for example, the USER table after using dbclean to delete guest shoppers)
- Implementing a script to select the tables or indexes that contain an asterisk (*) on the REORG column, and invoke the REORG command for each of them (not recommended)
- Enabling Automatic Reorganization on DB2 8.2 (See below.)

A standard (offline) REORG will lock for write the tables being reorganized. If you need to allow updates to the tables, online table reorganization can be used instead.

Follow the links in the DB2 Information Center for more information about the REORGCHK command and the REORG INDEXES/TABLE command.

Rebind (db2rbind)

Static packages need to be rebound after executing RUNSTATS to make use of updated statistics. Packages can be rebound one by one using the REBIND command or all at once by using the **db2rbind** (rebind all packages) command. The use of the **db2rbind** command is as follows:

```
db2rbind dbname -l db2rbind.log all
```

Note: The command will not rebind a package if it is in use.

WebSphere Commerce includes 16 out-of-the-box stored procedures that will be benefited by the rebind:

- adjustinventory
- allocateitem
- allocbora
- availableinv
- availinvstore
- availradate
- availreceipts
- backorderitem
- currentversion
- deletebackorder
- expectedinv
- getitems
- inventoryallocation
- raallocation
- reverseinventory
- shipitems

The following SQL can be used to identify the last bind date for all the packages in the database:

```
db2 "select pkgschema,pkgname,last_bind_time from syscat.packages order
by last_bind_time desc"
```

You may get the output shown in Example 25-2.

Example 25-2 Sample output for checking last bind date for packages

```
DB2INST1 P7001184 2007-07-11-19.29.24.498032
DB2INST1 P7001131 2007-07-11-19.29.23.996449
DB2INST1 P7001094 2007-07-11-19.29.23.742995
DB2INST1 P7001068 2007-07-11-19.29.23.685187
DB2INST1 P7001032 2007-07-11-19.29.23.318409
DB2INST1 P7001005 2007-07-11-19.29.23.158569
DB2INST1 P7000945 2007-07-11-19.29.22.678456
```

Note: Automatic Runstats and Reorg are available from DB2 8.2 (8.1.7). Since the automatic maintenance feature internally schedules a classic reorganization for the table, it locks the table being reorganized for writing. Therefore, if you are using automatic maintenance, ensure that the maintenance is only scheduled for when the traffic to the site is minimal.

Archived

25.1.2 WebSphere Commerce Database Cleanup utility

Keeping obsolete data affects runtime performance and makes the database difficult to manage. In order to keep the database consistent, policies should be defined and enforced to remove outdated information from the database.

WebSphere Commerce includes a Database Cleanup utility that allows you to delete objects from the database. You may want to do this if you have changed a lot of information in your database and have unused tables or rows.

When the Database Cleanup utility deletes an object, the records in the object's tables are deleted to preserve the referential integrity of the database. The Database Cleanup utility deletes records in child tables based on the delete rule of the referential integrity definition in the database schema. You can set the delete rule to on delete cascade, on delete set null, or on delete restrict. If you add new tables, ensure that the referential integrity and delete rule is properly defined. Otherwise, the Database Cleanup utility cannot work with your new tables.

The general approach to implement WebSphere Commerce Database Cleanup utility is as shown in Figure 25-2.

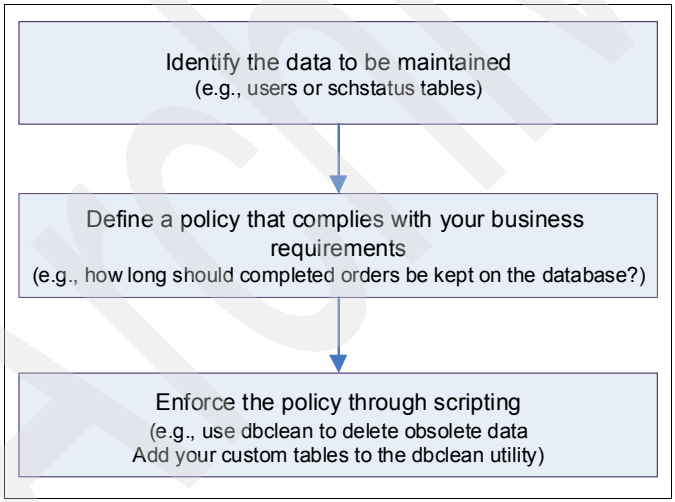


Figure 25-2 WC database cleanup utility

Archive

The following command can be used to record the cardinality (number of rows) of each table. Run this command on a monthly basis and save the output. Comparing the current table cardinality with that of previous months will allow you to identify those tables that require the most attention and maintenance.

```
db2 "EXPORT TO tablecard.csv OF DEL SELECT tabschema, tabname, definer,
card, fpages FROM syscat.tables WHERE type = 'T' AND (days (current
date) - days (stats_time)) < 7"
```

Tips for implementing DBClean:

1. If you are migrating from an existing version of WebSphere Commerce, you can run the Database Cleanup utility after your migration. Remember to evaluate the types of data on your system and how they affect database maintenance. Typically, user and order data can be quite large, resulting in large database tables. When you clean the database, this will be time consuming since it can fill up your database transaction log files or potentially lock database tables when your store is running.
2. You should only run the Database Cleanup utility on a staging server to clean the staglog object. The staging database is different from the production database. The staging database only has configuration data without the operation data. Deleting configuration data might cause a delete cascade on the operation data. When the Stage Propagate utility propagates the deletion to the production database, this might cause a cascade delete to the operation data (which you want to keep). To clean configuration data, run the Database Cleanup utility on the production database.
3. Depending on the amount of cleanup required for your database, you should consider running the DB2 REORGCHK utility prior to running the DBClean to improve performance during the cleanup.

Steps for implementing DBClean in WebSphere Commerce

The general steps to implement DBClean in WebSphere Commerce are:

1. Identify table that needs to be cleaned.

   The Database Cleanup utility refers to the CLEANCONF table to determine which tables and which rows to delete when a particular object and object type are specified. The following table describes preconfigured deletion scenarios from the CLEANCONF table. You can configure your own deletion objects by adding similar rows to the CLEANCONF table.

   About the detailed information for database Cleanup utility objects, you can refer to the description in Commerce Information Center from this link:

   http://publib.boulder.ibm.com/infocenter/wchelp/v5r6m1/index.jsp?topic=/com.ibm.commerce.admin.doc/refs/rduobjects.htm

2. Discuss the requirements for keeping data.

Syntax for DBClean configuration and execution

To add a new configuration to the Database Cleanup utility, use the following syntax as a reference. For example, object Obj1 consists of table *sample*, which contains the following columns: columnA, columnB, lastupdate, and columnC. To configure the Database Cleanup utility to delete all objects with columnA> 10, and where lastupdate is n days ago:

1. Open a DB2 command prompt.
2. Type the following to configure the Cleanup utility for this table, and add a new object named “Ojb1” into cleanconf table.

   ```
   db2 insert into cleanconf (objectname, type, statement, namearg,
   sequence, daysarg) values ('Obj1', 'obsolete', 'delete from sample
   where columnA > 10 and (days(CURRENT TIMESTAMP)- days(lastupdate)) >
   ?', 'no', 1, 'yes')
   ```

   Note: In the above command, the question mark (?) is replaced by the -days parameter from the following command line. The 'no' indicates that the name parameter is not used in the statement. The 'yes' indicates that the -days parameter is used in the statement. 'obsolete' describes the cleanup type for object Obj1. You can use other words, but you must use the same word in the -type argument when you invoke the Database Cleanup utility.

3. Execute the DBClean utility to clean the records that have been in existence for two days from the new table by typing the following:

   ```
   ./dbclean.sh -object o1 -db dbname -dbuser user -type obsolete -days
   2 -loglevel 1
   ```

Performance consideration for DBClean

DBClean can be tuned to gain performance improvement. The CLEANCONF table holds the actual SQL statement that the script references. Sample SQL stored in the CLEANCONF table to clean up the objects can be found at this linkl:

http://publib.boulder.ibm.com/infocenter/wchelp/v5r6m1/index.jsp?topic=/com.ibm.commerce.admin.doc/refs/rduobjects.htm

For other objects that can be cleaned up using DBClean, review the list at above link. A rule of thumb is to clean up, at the minimum, obsolete tables and objects. Some of the cleanup utility objects are:

- Order
- Address
- Usertraffic
- Member

- Promotion
- Catalog
- Cacheivl
- Scheduler
- Staging

For more information, see this link about Database cleanup utility objects supported by Commerce:

http://publib.boulder.ibm.com/infocenter/wchelp/v5r6m1/index.jsp?topic=/com.ibm.commerce.admin.doc/refs/rduobjects.htm

For example, if a user wants to delete guest shoppers who have no orders associated and who has been in the system for more than 10 days, a new object called guest_shopper can be created by inserting the following SQL into the CLEANCONF table and calling DBCLEAN on the object (Example 25-3).

Example 25-3 Sample SQL to delete guest shoppers with DBClean

```
Deletes guest_shopper ids older than 10days **
delete from member
where member_id in (select users_id from users where registertype='G'
and (days(CURRENT TIMESTAMP) - days(lastsession)) >= 10
and (users_id not in (select member_id from orders)) and (users_id >
0))
```

Archived

25.1.3 Commerce DB2 database maintenance solution

The recommended Commerce DB2 database maintenance solution is the combination of DB2 general database maintenance utilities and the Commerce database cleanup utility, as shown in Figure 25-3.

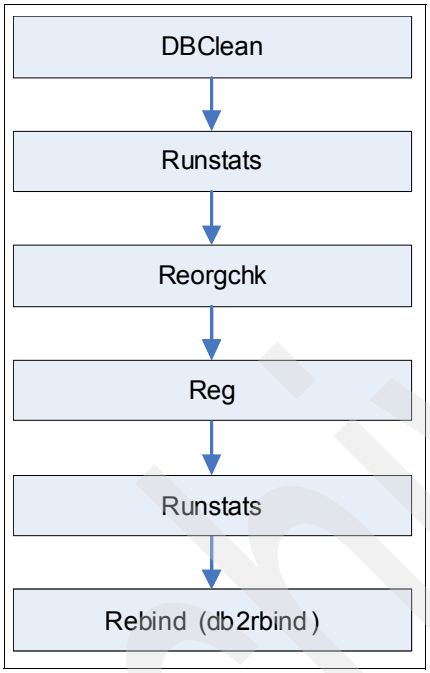


Figure 25-3 Commerce DB2 database maintenance solution

Archived

26

Maintain and update WebSphere Application Server tier

While in production, continue to monitor the important parameters that we discussed in Chapter 17, “Monitor and tune WebSphere Application Server for WebSphere Commerce” on page 375.

In this chapter, we describe how to maintain and update the WebSphere Application Server machines in a WebSphere Commerce High Availability configuration. We describe actions that need to be taken in order to perform hardware upgrades, OS upgrades, product upgrades, fix installations, deployment requiring an EAR update, and log maintenance.

We group the types of upgrades into types that do not require a server outage or restart, and types that do require an outage. As our application server configuration only consists of active nodes, the upgrade types that require a server outage are further distinguished according to whether upgraded and not yet upgraded servers can be active at the same time (compatible upgrade) or not (incompatible upgrade).

© Copyright IBM Corp. 2008. All rights reserved.

26.1 Maintenance not requiring planned outages

There are some maintenance tasks that do not require any application servers to be stopped:

- Application server log maintenance
- Deployment of cachespec.xml
- Deployment of new custom code including EJBs and JSPs
- Deployment of WebSphere Commerce instance configuration file

26.1.1 WebSphere Application Server log maintenance

The following WebSphere Application Server logs should be maintained on a regular basis.

FFDC

The first failure data capture (FFDC) feature preserves the information that is generated from a processing failure and returns control to the affected engines. The captured data is saved in a log file for analyzing the problem. FFDC is intended primarily for use by IBM Service. FFDC instantly collects events and errors that occur during the WebSphere Application Server runtime. The information is captured as it occurs and is written to a log file that can be analyzed by an IBM Service representative. The data is uniquely identified for the servant region that produced the exception.

If not properly maintained, this log can grow very big, causing you to run out of disk space and halting application server processing.

Hence, we recommend that you rotate this log regularly.

You can also configure the property files to automatically purge FFDC logs. The FFDC configuration properties files are located in the properties directory under the Application Server product installation. You must set the ExceptionFileMaximumAge property to the same value in all three files: ffdcRun.properties, ffdcStart.properties, and ffdcStop.properties. You can set the ExceptionFileMaximumAge property to configure the amount of days between purging the FFDC log files. The value of the ExceptionFileMaximumAge property must be a positive number.

Perform the following steps to configure the number of days between the FFDC log file purges. The value is in days.

1. Open the ffdcRun.properties file. The file is located in the app_server_root/properties directory.

Archived

2. Change the value for the ExceptionFileMaximumAge property to the number of days between the FFDC log file purges. The value of the ExceptionFileMaximumAge property must be a positive number. The default is seven days. For example, ExceptionFileMaximumAge = 3 sets the default time to three days. The FFDC log file is purged after three days.
3. Save the ffdcRun.properties file and exit.
4. Repeat the previous steps to modify the ffdcStart.properties and ffdcStop.properties files.

The FFDC file management function now removes the FFDC log files that have reached the maximum age and generates a message in the SystemOut.log file.

Garbage collection log (native_stderr.log)

The GC log should be recycled regularly.

SystemOut.log and SystemErr.log

In the SystemOut.log, turn off any custom code tracing.

Change the log file size to a minimum of 5 to 10 MB.

Change the number of historical files to more than 5.

Trace.log

We recommend that this file be turned off on production.

The recommendation is to enable the trace service, but do not configure any trace specification strings. If trace is required, turn it on dynamically during runtime for a short period of time and turn it off as soon as enough data is gathered.

Threaddump and heapdump

Threaddump and heapdump are automatically generated when an out-of-memory condition occurs. These files could potentially get very large (some heapdumps could reach more than 500 MB in size).

You should consider redirecting these files to a separate hard drive that has sufficient disk space and archive or delete these files once you are done using them for problem determination.

Archived

26.1.2 Deployment of cachespec.xml

cachespec.xml can be dynamically reloaded at this location:

```
WC_eardir/Stores.war/WEB-INF
```

If it is loaded properly, you will see a message in the SystemOut.log similar to this:

```
[5/10/07 15:43:42:438 EDT] 0000000a ConfigManager I   DYNA0047I:
Successfully loaded cache configuration file
C:\WebSphere\AppServer60\profiles\demo\installedApps\WC_demo_cell\WC_de
mo.ear\Stores.war\WEB-INF/cachespec.xml.
```

This is very handy when you want to quickly test any changes made in the cachespec.xml without having to restart the application server.

However, this is only the runtime copy of the file on the individual application server node. The master copy of the file on deployment manager has not yet been updated. Once the file is finalized, you must do a regular deployment of the application update in order to synchronize with the master configuration of this file on deployment manager. Otherwise, while the master configuration of this file is of an older version, an application update is deployed using the normal process, your temporary runtime copy of the cachespec.xml will be lost and overwritten with the older version.

26.1.3 Rollout update

WebSphere Commerce Version 6 takes advantage of a new feature in WebSphere Application Server V6 called rollout update, whereby in a clustered environment, an application update request is sent to one node at a time, and only when the first node completes its update and comes back online is the change repeated on a subsequent node. Hence, this process ensures that at any given time during the application update, at least one node remains online and actively processing user requests. Because of this new feature, the following changes no longer require an outage: any partial application updates such as new custom code deployment, store publish, and file updates inside Commerce EAR.

Depending on the network speed, the size of the WebSphere Commerce EAR, and application server processing power, the rollout update process could take some time. Anywhere from fewer than 5 minutes to more than 30 minutes is possible.

Deployment of new custom code including EJBs and JSPs

A sample tutorial on how to deploy JSP using a rollout update is found here:

http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/topic/com.ibm.commerce.samples.doc/tutorial/tdedeployjsp.htm

Deployment of Commerce instance configuration file

Once the WebSphere Commerce instance configuration file (located in WC_installdir /instances/instance_name/xml/instance_name.xml) is changed, either manually or through WebSphere Commerce configuration manager, you must also ensure that you run the ANT target UpdateEAR target to update the runtime configuration with the new configuration file. This process ensures that all nodes running the WebSphere Commerce instance place an updated copy of the configuration file in the following location:

```
WAS_profiledir/installedApps/WC_instance_name_cell/WC_instance_name.ear
/xml/config/wc-server.xml
```

Run ANT script

The UpdateEAR target uses a partial application (.zip) or instance configuration file (xml) to update the deployed WebSphere Commerce Enterprise application.

- Prerequisites

  The administrative server must be running to run the ConfigureCommerceServer target and any subtargets.

  The createInstance.properties file must exist for this Ant target to work. For information about generating the properties file, see ANT targets.

- Required parameters: instance_name

  The name of the WebSphere Commerce instance with which you are working (for example, demo). For example:

  ```
  WC_installdir/bin/config_ant.sh -DinstanceName=instance_name
  UpdateEAR
  ```

26.2 Planned outages

In addition to hardware and OS upgrades, this section describes a few common tasks that may require planned outages of the WebSphere Application Server tier.

Archive

26.2.1 WebSphere Application Server fix pack/APAR upgrade

When applying WebSphere Application Server fixes, it is required that you stop all Java processes associated with Java SDK shipped with WebSphere Application Server, the nodeagent process, the deployment manager process, and all server processes that belong to serviceable products, such as the IBM HTTP Server.

Follow the README for installation details.

26.2.2 WebSphere Commerce fix pack/APAR upgrade

When installing WebSphere Commerce fixes, follow instructions in the fix pack installation guide or APAR readme.

You only need to apply the fixes on ONE WebSphere Commerce node—the one where you created the WebSphere Commerce instance.

When installing a WebSphere Commerce APAR that requires no update to the database, you do not need to stop any servers.

However, when installing the WebSphere Commerce fix pack, a database update is also required. You need to stop the application servers to ensure data integrity.

Archived

27

Maintain and update Web servers

In this chapter we describe how to maintain and update the Web server machines in WebSphere Commerce High Availability configurations. We describe actions that need to be taken in order to perform hardware upgrades, OS upgrades, product upgrades, fix installations, deployment of new static content, and log maintenance.

We group the types of upgrades into types that do not require a server outage or restart, and types that do require an outage. As our Web server configuration only consists of active nodes, the upgrade types that require a server outage are further distinguished according to whether upgraded and not yet upgraded servers can be active at the same time (compatible upgrade) or not (incompatible upgrade).

Each upgrade type requires different preparation and execution processes. Also, some upgrades require a certain order of updating the application server tier and the Web server tier.

© Copyright IBM Corp. 2008. All rights reserved.

27.1 Maintenance not requiring planned outages

There are some maintenance tasks that do not require any Web server to be stopped:

- Web server log maintenance
- Deployment of new static content (if certain prerequisites are met)

27.1.1 Maintain IBM HTTP Server logs

Special care needs to be taken of the log files on a Web server. For IBM HTTP Server, there are two types of log files:

- IBM HTTP Server logs (access log and error log)
- IBM HTTP Server Plug-in log

IBM HTTP Server logs (Apache logs)

If you have turned logging on (see 18.1.2, “Access log” on page 394), you need to make sure that there is enough disk space for the access and error logs. As the amount of information in the access log can grow very large, you should also periodically rotate the log files by moving or deleting the existing logs. There are two ways of doing this:

- The so-called graceful restart of the server.
- Using log piping to the rotatelogs utility.

Graceful restart

By using a graceful restart, you can instruct IBM HTTP Server to open new log files without losing any existing or pending connections from clients. However, in order to accomplish this, the server must continue to write to the old log files while it finishes serving old requests. Therefore, you need to wait for some time after the restart before doing any processing on the log files. A typical scenario that simply rotates the logs and compresses the old logs to save space is shown in Example 27-1.

Example 27-1 Scripted log rotation and compression

```
cd WC_Install_Dir/instances/Instance_Name/httplogs
mv access_log access_log.old
mv error_log error_log.old
IHS_Install_Dir/bin/apachectl -k graceful -f
   WC_Install_Dir/instances/Instance_Name/httpconf/httpd.conf
sleep 600
gzip access_log.old error_log.old
```

Archived

You may use the cron utility to regularly execute scripts on UNIX systems.

Log piping

The executable *IHS_Install_Dir*/bin/rotatelogs can be used to rotate the access and error logs. To use it, open the httpd.conf file (*WC_Install_Dir*/instances/*Instance_Name*/httpconf/httpd.conf) and modify the CustomLog and ErrorLog directives, as shown in Example 27-2:

Example 27-2 Using log piping

```
CustomLog "| IHS_Install_Dir/bin/rotatelogs
WC_Install_Dir/instances/Instance_Name/httplogs/access_log.%Y%m%d
86400" common
ErrorLog "| IHS_Install_Dir/bin/rotatelogs
WC_Install_Dir/instances/Instance_Name/httplogs/error_log.%Y%m%d
86400"
```

This example would create a new access log every day and append the current date to the name, for example, access_log.20070724.

You still need to regularly execute operating system scripts that archive or remove the rotated logs, but you do not need to restart the Web server.

Important: To use the syntax shown in Example 27-2 on page 581 for log piping, IBM HTTP Server Version 6.0.2.1 or later is required.

IBM HTTP Server Plug-in

In "Update the Web server configuration" on page 176, we have configured the plug-in log to be written to the *WC_Install_Dir*/instances/*Instance_Name*/httplogs/http_plugin.log file.

Like the access and error logs, the plug-in log can be renamed or moved while IBM HTTP Server Plug-in is still writing to the file. IBM HTTP Server Plug-in will continue on to writing to the moved file. You may therefore use the graceful restart option of IBM HTTP Server to rotate the plug-in log, too (see Example 27-1 on page 580).

27.1.2 Deploy new static content

Deploying new versions of your WebSphere Commerce application mostly includes updating the static content on the Web servers. Remember that with WebSphere Commerce, the Web servers are used only to serve static content and to forward requests for dynamic pages to the application servers.

Deploying new static content does not require a server outage or restart if the following prerequisites are met:

- Old and new versions of the static content must work with both old or new versions of the application. If both old and new static content files work with the old version of the application, the Web servers should be updated first. If both old and new static content files work with the new version of the application, the application servers should be updated first.

  If the new static content is incompatible with the old application and the old static content is incompatible with the new application, or if the two versions of the application must not be used in the same user session, but site downtime is not desired, then Web servers and application servers should be updated in separate groups, using Web server to application server affinity, as described in 27.2.3, “Incompatible upgrades” on page 588.

- No change of content expiration times is required (see Chapter 18, “Monitor and tune Web servers” on page 391).

  If a change is required here, but otherwise the first prerequisite is met, you may use the graceful restart option to restart the server and apply the new expiration times after changing the httpd.conf file.

27.2 Maintenance involving planned outages

Upgrades to the Web server tier may not always be possible without stopping one or more Web servers for some time. After explaining a technique called quisling for gracefully removing a Web server from the Load Balancer cluster (see 27.2.1, “Quiescing a Web server” on page 583), we look at different types of upgrades requiring server outage:

- Compatible upgrades, where an updated Web server can be restarted while non-updated Web servers are still online (See 27.2.2, “Compatible upgrades” on page 587.)
- Incompatible upgrades, where updated Web servers must not be restarted while any non-updated Web servers are still online (See 27.2.3, “Incompatible upgrades” on page 588.)

In both cases, not all the servers need to be stopped at the same time when following our instructions. The WebSphere Commerce site can be kept online.

While in many cases our High Availability configuration allows us to keep the WebSphere Commerce site online, there might still be upgrades (typically at the application and database tiers) that require taking the entire site down. We describe how to configure the Web servers to display a maintenance page before

making these kinds of updates (see 27.2.4, “Maintenance Web page for site downtimes” on page 592).

Important: When performing the upgrades described in the following sections, one or more of your Web servers are down for a certain amount of time. To ensure that the remaining Web servers can still handle the incoming requests without performance degradation, utilization of each Web server should not be more than 50% (CPU, memory, open files, sockets, and so on) at peak times when all Web servers are online. If this is not the case, consider performing upgrades at off-peak times or temporarily adding more Web servers.

27.2.1 Quiescing a Web server

Quiescing a Web server is a technique for gracefully removing a Web server from the Load Balancer cluster when server affinity (stickyness) is configured for the Web server on any port (see 19.3, “Server affinity” on page 437).

Rather than just allowing existing connections to complete without being severed, this method allows existing connections to complete and forwards subsequent new connections to the quiesced server from those clients with existing connections that are designated as sticky, as long as the quiesced server receives the new request before stickytime expires.

Remember, with WebSphere Commerce, we do not need to use server affinity, as the user’s HTTP session is maintained by the application server tier while it is identified by cookies that are stored in the client browser. However, server affinity may increase performance (again, see 19.3, “Server affinity” on page 437).

Archive

To quiesce a server when using IBM WebSphere Edge Components Load Balancer, log on to your Load Balancer node, run **`dsserver`** (if necessary) and **`lbadmin`**, and in the GUI, connect to your host (as described in steps 1 on page 193 to 4 on page 195). Then follow these simple steps:

1. In the tree view, right-click **Manager**, as shown in Figure 27-1.

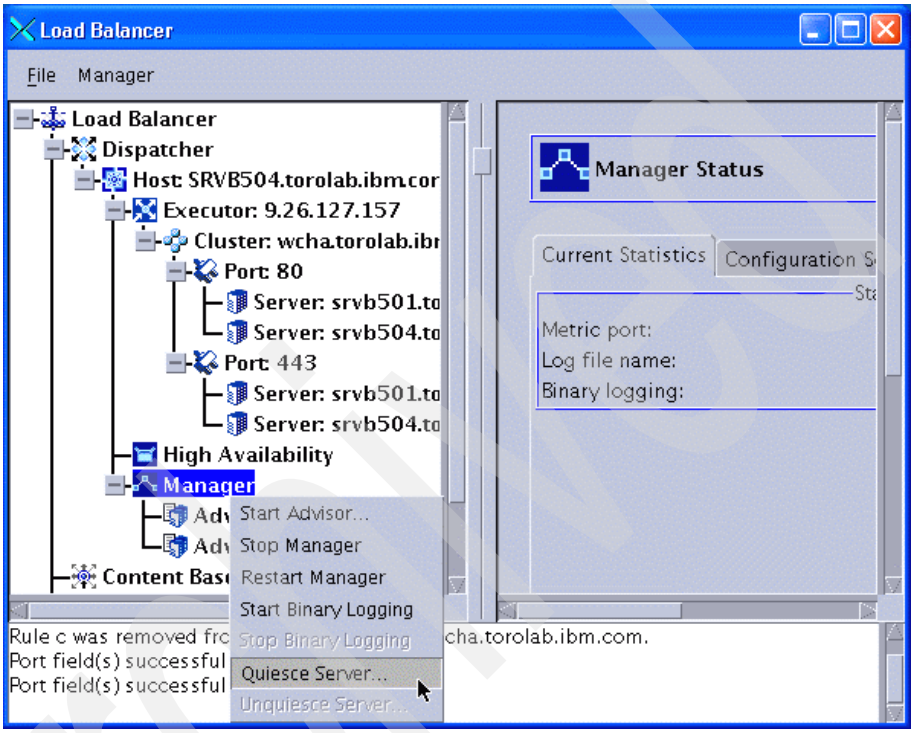


Figure 27-1 Quiescing a server

Archive

2. Click **Quiesce Server**. This brings up a pop-up window, as shown in Figure 27-2.

Figure 27-2 Selecting the server to quiesce

3. Select the server that you want to quiesce in the Server box. Depending on the sticky time and the current user activity, quiescing a server with sticky time may take quite long. If you do not want to wait, you may check the **Quiesce now?** check box. The server will then be removed from the cluster immediately, allowing existing connections to complete, but ignoring any sticky times configured for the server's ports.

Log on to the active Load Balancer as root and execute the following command:

```
dscontrol manager quiesce server Server_IP_or_hostname [now]
```

Example 27-3 shows how we quiesce our Web server node 1.

Example 27-3 Quiescing Web server node 1

```
dscontrol manager quiesce server srvb501.torolab.ibm.com
```

Again, using the *now* option would ignore sticky times.

Important: Also quiesce the Web servers on the standby Load Balancer.

Archived

To reactivate a server:

1. In the tree view, right-click **Manager** and in the Manager context menu select **Unquiesce Server**, as shown in Figure 27-3.

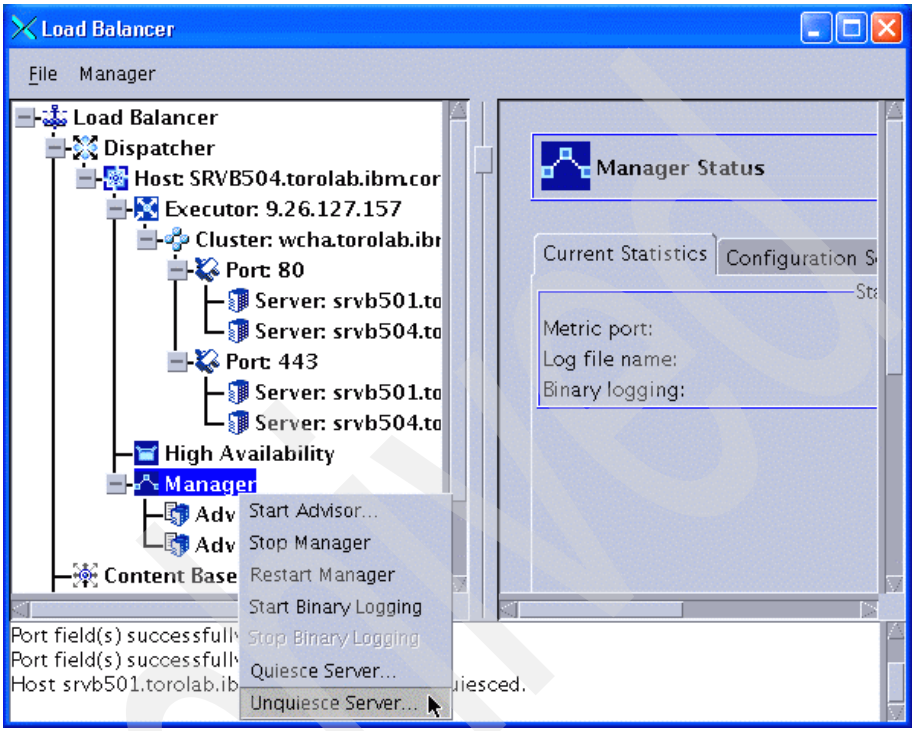


Figure 27-3 Unquiescing a server

A pop-up window for selecting the server to unquiesce is displayed (Figure 27-4).

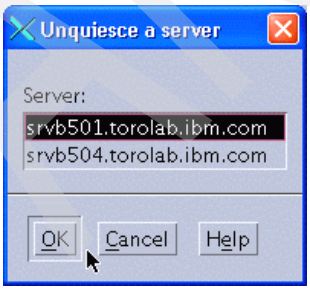


Figure 27-4 Selecting the server to unquiesce

2. Select the server to unquiesce and click **OK**.

Unquiescing a server may also be scripted on the command line as follows:

```
dscontrol manager unquiesce server Server_IP_or_hostname
```

27.2.2 Compatible upgrades

We refer to upgrades as compatible if Web servers to which the updates have already been applied can be taken online while Web servers that have not yet been updated are still online. Without server affinity in the Load Balancer, this means that requests may be routed to updated and non-updated Web servers within the same user session.

Compatible upgrades that require a Web server to be offline while the update is being applied typically include:

- Hardware and capacity upgrades
- OS upgrades
- Software product upgrades
- Refresh packs, fix packs
- Fixes, APARs, iFixes, eFixes, and so on

If only compatible upgrades had to be performed, the Web server tier could be utilized over 50% at peak times, as there would only need to be enough capacity to compensate for one server being down at a time. However, this is usually not sufficient, as we discuss further in 27.2.3, “Incompatible upgrades” on page 588”.

Instructions for all compatible upgrade types

For each Web server, repeat the following steps:

1. Quiesce the server (see 27.2.1, “Quiescing a Web server” on page 583).
2. Perform the update (see specific instructions below).
3. Test your Web server by sending requests directly to the Web server rather than to the Load Balancer.
4. Unquiesce the server (see 27.2.1, “Quiescing a Web server” on page 583).

Instructions to specific types of compatible upgrades

This section lists details for step 2 above, for each type of compatible upgrade.

Hardware and capacity upgrades

This type of upgrade most likely requires a server shutdown. After restarting, the Web server may simply be restarted. Some parameters possibly need to be adapted to the new hardware, in order to utilize increased capacities.

Archived

OS upgrade

If a new OS is installed, the Web server software might need to be reinstalled and reconfigured. See 8.5, “Install IBM HTTP Server” on page 127, and 11.1, “Add additional Web servers” on page 186, for how to configure it by copying the configuration from another Web server.

Software product upgrade

If a different major version of IBM HTTP Server (or even a different Web server product) is installed, this might require reconfiguration different from the configuration described in this book for IBM HTTP Server. Do not even think about it.

Refresh packs and fix packs

Refresh packs and fix packs normally do not require any reconfiguration. Refer to 8.5.2, “Install fixes” on page 133, for instructions on how to apply refresh packs and fix packs.

Fixes, APARs, iFix, eFix, and so on

Fixes normally do not require any reconfiguration. Fixes are installed similarly to fix packs. Also refer to the installation instructions that are included with the fixes.

27.2.3 Incompatible upgrades

There are certain types of upgrades at the Web and application server tier that make it impossible to update the Web servers (or the application servers) one by one.

If a new application version with major differences from the current application is to be deployed, old static content might not be compatible to the new JSPs, and vice versa. Further, it might not be possible to switch between updated and not yet updated Web servers (by Load Balancer), or between updated and not yet updated application servers (by the IBM HTTP Server Plug-in on the Web server) within one user session. (Switching might lead to application flow and GUI inconsistencies, depending on the nature of the upgrades.)

If this is the case, it might still be okay to use the old version of the application (for example, old static content, old application) and the new version (for example, new static content, new application) in parallel in different user sessions. For example, one user could see the old application while another user already sees the new application.

Archived

If this is okay, an upgrade may be performed as follows:

1. Configure Load Balancer server affinity and Web server to application server affinity such that a client always connects to the same Web server (within a certain sticky time) and the application servers are divided into two separate groups, as shown in Figure 27-5.

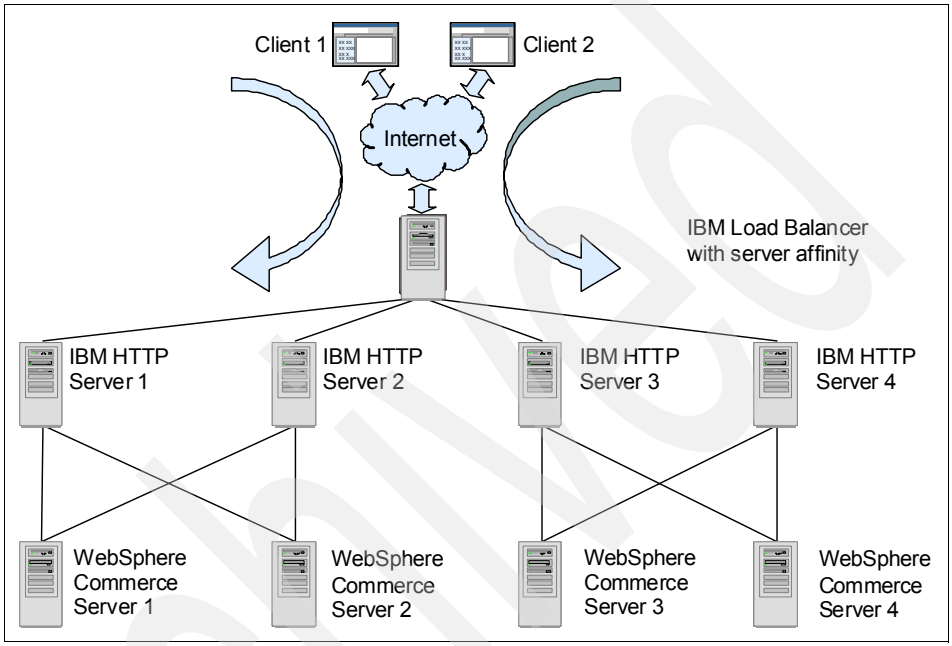


Figure 27-5 Separating application servers

Refer to 19.3, “Server affinity” on page 437, for details on how to set up server affinity in Load Balancer. For performance reasons, server affinity should be enabled here all the time.

Binding Web servers to certain application servers can be done by editing the plug-in configuration:

a. First disable automatic plug-in propagation for your Web servers in the Network Deployment Manager administrative console for each Web server by navigating to **Servers** → **Web servers** → ***WebServer_Name*** → **Plug-in properties** and deactivating **Automatically propagate plugin propagation** (see Figure 10-3 on page 177). When you are done with all Web servers, save your changes to the master configuration.

b. On each Web server, one -by-one, use the process for compatible upgrades described above (see 27.2.2, “Compatible upgrades” on page 587). Back up and then edit the plug-in configuration file (*WC_Install_Dir*/instances/*Instance_Name*/httpconf/plugin-cfg.xml) as the

non-root_user. Find the definition for your application server cluster and comment out (using <!-- and -->) the server elements belonging to the other group. Example 27-4 shows our plug-in configuration with one of the two servers in the cluster commented out.

Example 27-4 Plug-in configuration with modified cluster definition

```
...
<ServerCluster Name="Cluster_Name" ...>
   <Server CloneID="11tf0t8ul" Name="Node1_Name_Cluster_Member1" ...>
      <Transport Protocol="http" .../>
      <Transport Protocol="https" ...> ... </Transport>
   </Server>
<!-- <Server CloneID="11trs2mtc" Name="Node2_Name_Cluster_Member2" ...>
      <Transport Protocol="http" .../>
      <Transport Protocol="https" ...> ... </Transport>
   </Server> -->
   <PrimaryServers>
      <Server Name="WC_demo_node_WC_demo_01_goro"/>
<!--  <Server Name="goro2Node01_WC_demo_01_goro2"/> -->
   </PrimaryServers>
</ServerCluster>
...
```

c. Save the file and restart your Web server.

HTTP sessions and dynamic cache objects should also only be replicated within the two groups of application servers, and not between members of different groups, in case sticky time is exceeded at the Load Balancer, but the HTTP session is still valid. If sticky time is expired and a client is diverted from a not-yet-updated Web server to an updated Web server, a new session should be started (or a notice should be displayed informing the user about an expired session).

Refer to “HTTP session management” and “Configure distributed session management” in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, for details on setting up session replication and to “Cache replication” in *WebSphere Application Server V6 Scalability and Performance Handbook,* SG24-6392, for details on setting up cache replication. Essentially, you need to do this:

i. Set up an additional replication domain (one was created when setting up the cluster). (Go to the administrative console and select **Environment** → **Replication domains**.)

ii. On each application server, configure one of the two replication domains for session and cache replication, such that there are two

equally sized groups of servers that match the groups configured for Web server to application server affinity above. (Go to the administrative console and select **Servers** → **Application servers** → ***AppServer_Name*** → **Web Container Settings** → **Session management** → **Distributed environment settings** → **Memory-to-memory replication** → **Replication domain** and **Servers** → **Application servers** → ***AppServer_Name*** → **Container Services** → **Dynamic cache service** → **Consistency settings** → **Full group replication domain**.) Save your changes and synchronize with all nodes.

iii. One by one, restart your application servers.

2. Now quiesce the first group of Web servers, one by one, as described in 27.2.1, “Quiescing a Web server” on page 583.
3. Upgrade the first group of Web servers and application servers.

   Typically, the application tier upgrade is performed as EAR deployment or update on the Network Deployment Manager. To prevent the upgrade from being distributed to the nodes of the second group of application servers, there are two options:

   – Do not check **Synchronize changes with nodes** when saving the changes to the master configuration, then manually synchronize the nodes of the first group in the **System administration** → **Nodes** view. (Select the nodes and click **Synchronize**.)

   – Stop the nodeagents in the servers of the second group before performing the upgrade. As *non-root_user*, execute the following command:

     ```
     WAS_Install_Dir/profiles/Profile_Name/bin/stopNode.sh
     ```

4. Restart the first group of application servers and Web servers.
5. Unquiesce the first group of Web servers (see 27.2.1, “Quiescing a Web server” on page 583). New users will now be able to see the new application.
6. Quiesce the second group of Web servers. All users will now see the new application.
7. Upgrade the second group of Web servers and application servers.

   The EAR deployment or update has already been performed on the Network Deployment Manager. (See step 3.) Now the nodes of the second group of application servers need to be synchronized. Depending on which option you chose in step 3, do one of the following:

   – Go to **System administration** → **Nodes** and synchronize all nodes (select the nodes and click **Synchronize**) in the second group, so the EAR update is copied to each of the servers.

– Restart the nodeagents on the servers in the second group by executing the following command as *non-root_user*:

```
WAS_Install_Dir/profiles/Profile_Name/bin/startNode.sh
```

The changes should be automatically synchronized after starting the nodeagents.

8. Restart the second group of application servers and Web servers.
9. Unquiesce the second group of Web servers.
10. Unconfigure Web server to application server affinity by undoing the changes made in step 1 on page 589:
    a. One by one, undo the changes to plugin-cfg.xml on each Web server and restart the Web server.
    b. Using the Network Deployment Manager administrative console, change session and cache replication back to just one replication domain. Save your changes and synchronize with all nodes.
    c. One bye one, restart your application servers.

27.2.4 Maintenance Web page for site downtimes

In the previous sections we have described techniques to apply updates without taking the site offline. Although at limited capacity, the site has still been able to server customers.

In some cases, however, the site needs to be taken offline completely to deploy application updates or to perform database maintenance. With WebSphere Commerce, this may, for example, be the case in the following situations:

- Incompatible application (EAR) updates need to applied at the application server tier. For example, new and old application versions cannot be used within one user session, and temporary server affinity from Web servers to application servers (see 27.2.3, “Incompatible upgrades” on page 588) cannot be configured (or is not desired).
- Necessary database schema or content updates cannot be performed while the site is being accessed and the database is in use.

If the site needs to be taken down, a temporary page should be returned to all clients upon all requests, informing the users of the maintenance in progress.

To configure this such that all user sessions can be finished first (for example, a user should be able to finish browsing and purchasing items), the Web servers should be quiesced as described in 27.2.1, “Quiescing a Web server” on page 583, and reconfigured one by one.

For this technique to work properly, server affinity must be configured for the Load Balancer to ensure that a user does not switch between Web servers still serving the WebSphere Commerce site and Web servers already reconfigured to show the maintenance page (see 19.3, “Server affinity” on page 437).

To configure a Web server to show a maintenance page:

1. Quiesce the server.
2. Copy your maintenance page (for example, maintenance.html) into your instance’s document root directory. This is the directory configured as DocumentRoot in your httpd.conf file (for example, *WC_Install_Dir*/instances/*Instance_Name*/web).
3. Modify httpd.conf:
   a. Find these two lines and comment them out (prepend with a #):

      ```
      LoadModule was_ap20_module
      Plugin_Install_Dir/bin/mod_was_ap20_http.so
      WebSpherePluginConfig
      "WC_Install_Dir/instances/Instance_Name/httpconf/plugin-cfg.xml"
      ```

   b. Configure your maintenance.html file as an error page by adding (or modifying) the ErrorDocument directive:

      ```
      ErrorDocument 404 /maintenance.html
      ```

      This way, all URIs that would normally be handled by the plug-in will result in 404 errors, as they do not point to any existing files on the Web server. Using the ErrorDocument directive, the maintenance page will be displayed whenever a 404 error occurs.

   c. Save the httpd.conf file.
4. Restart your Web server (see “Restart the Web server” on page 184).

After making all necessary changes at the database, application server, and Web server tiers, undo the changes described above on all Web servers, then restart the servers. Quiescing is not necessary this time, as only static content has been served and no user session has been established.

Tip: It is easier to provide a copy of the httpd.conf file that contains the changes described in step 3 above. For future upgrades requiring a site outage, instead of steps 3 and 4 above, just stop your Web server and restart it, passing the file name of the modified copy of httpd.conf as the -f parameter to **`apachectl`**.

Archived

Archived

28

Maintain and update Load Balancer

In this chapter, we describe how to maintain and update the Load Balancer machines in WebSphere Commerce High Availability configurations. We describe actions that need to be taken in order to perform hardware upgrades, OS upgrades, product upgrades, fix installations, and log maintenance for IBM WebSphere Edge Components Load Balancer.

Application updates (new custom code and static content) typically do not affect the Load Balancer machines in our configuration, so we can look at the Load Balancer machines independently from other nodes.

Like we did for the Web servers (see Chapter 27, “Maintain and update Web servers” on page 579), we group the types of upgrades into types that do not require a server outage or restart, and types that do require an outage. Unlike Web servers, we cannot use multiple Load Balancers in active/active configurations (as the site is only accessible through one IP address). Rather, we have an active/passive configuration with one standby node monitoring the active node (see 7.2, “Introduction to Load Balancer High Availability” on page 85, and 11.3, “Configure Load Balancer High Availability” on page 226).

© Copyright IBM Corp. 2008. All rights reserved.

28.1 Maintenance not requiring planned outages

With IBM WebSphere Edge Components Load Balancer, log file maintenance is the only type of maintenance not requiring you to stop the Dispatcher server.

Maintain Load Balancer logs

Load Balancer posts entries to a server log, a manager log, a metric monitor log (logging communications with Metric Server agents if these are used), and a log for each advisor that you use. Additionally, for the Dispatcher component only, entries can be made to a subagent (SNMP) log.

You can set the logging level to define the expansiveness of the messages written to the log. At level 0, errors are logged and Load Balancer also logs headers and records of events that happen only once (for example, a message about an advisor starting to be written to the manager log). Level 1 includes ongoing information, and so on, with level 5 including every message produced to aid in debugging a problem if necessary. The default for the manager, advisor, server, and subagent logs is 1.

You can also set the maximum size of a log. When you set a maximum size for the log file, the file will wrap. When the file reaches the specified size, the subsequent entries will be written at the top of the file, overwriting the previous log entries. You cannot set the log size to a value that is smaller than the current one. Log entries are timestamped so you can tell the order in which they were written.

The higher you set the log level, the more carefully you should choose the log size. At level 0, it is probably safe to leave the log size to the default of 1 MB. However, when logging at level 3 and above, you should limit the size without making it too small to be useful.

You can set the logging level and log file size using the Load Balancer administrative GUI (**`lbadmin`**) or using the **`dscontrol`** command. Refer to 19.2, “Tuning Load Balancer parameters” on page 425, and to “Using Load Balancer logs” in the *Load Balancer Administration Guide,* GC31-6858, for detailed information.

28.2 Maintenance involving planned outages

Upgrades to the Load Balancer hardware, operating system, or software, typically require stopping the network dispatcher. When using Load Balancer High Availability (see 11.3, “Configure Load Balancer High Availability” on

page 226), we can stop the active or the standby Load Balancer from performing the desired maintenance or upgrades without affecting WebSphere Commerce site availability. However, for the time of updating the standby Load Balancer, the active Load Balancer is a single point of failure. Therefore, we recommend using a temporary (standby) machine, if you want true High Availability all along and to reduce the timeframes for the existence of single points of failure.

Like for Web servers, we also look at different types of upgrades requiring server outage:

- Compatible upgrades, where an updated Load Balancer can exist in a High Availability configuration with a not-yet-updated Load Balancer (See 28.2.1, “Compatible upgrades” on page 597.)
- Incompatible upgrades, where an updated Load Balancer cannot exist in a High Availability configuration with a not-yet-updated Load Balancer (See 28.2.2, “Incompatible upgrades” on page 602.)

28.2.1 Compatible upgrades

We refer to upgrades as compatible if an updated Load Balancer can exist in a High Availability configuration with a not-yet-updated Load Balancer.

Compatible upgrades that require Load Balancer to be stopped while the update is being applied typically include:

- Hardware and capacity upgrades
- OS upgrades
- Software product upgrades
- Refresh packs and fix packs
- Fixes, APARs, iFixes, eFixes, and so on

Instructions for all compatible upgrade types

The instructions in this section apply to all the types of upgrades mentioned above. You may follow a procedure using a temporary machine, reducing single point of failure time, if your upgrades take some time. Alternatively, you can follow a simpler procedure.

Upgrade using a temporary machine

To upgrade your primary Load Balancer and Standby Load Balancer using a temporary machine:

1. Install a new Load Balancer machine and apply all needed maintenance and upgrades. See 8.6, “Install Load Balancer” on page 140. Configure the machine for load balancing your Web server cluster as described in 11.2, “Configure Load Balancer” on page 193. If you have done performance tuning

as described in Chapter 19, “Monitor and tune Load Balancer” on page 417, also apply your tuning to the new machine. We refer to this machine as *temporary Load Balancer*.

2. For all three machines, open the administration GUI and connect to the host, as described in steps 1 on page 193 to 4 on page 195.
3. Make sure that you save your Load Balancer configurations on all three machines (as described in step 15 on page 208).
4. Make sure that your primary Load Balancer is in active state. Click **High Availability** in the tree view and select **Current statistics** in the right pane to view the state (see Figure 11-49 on page 231). If your recovery strategy is Auto and your primary Load Balancer is functional, it should be in active state. If your recovery strategy is Manual and your primary Load Balancer is in standby state, perform the following steps (see also step 4 on page 229):
   a. In the tree view, right c-click **High Availability** and select **Take over** from the context menu, as shown in Figure 28-1.

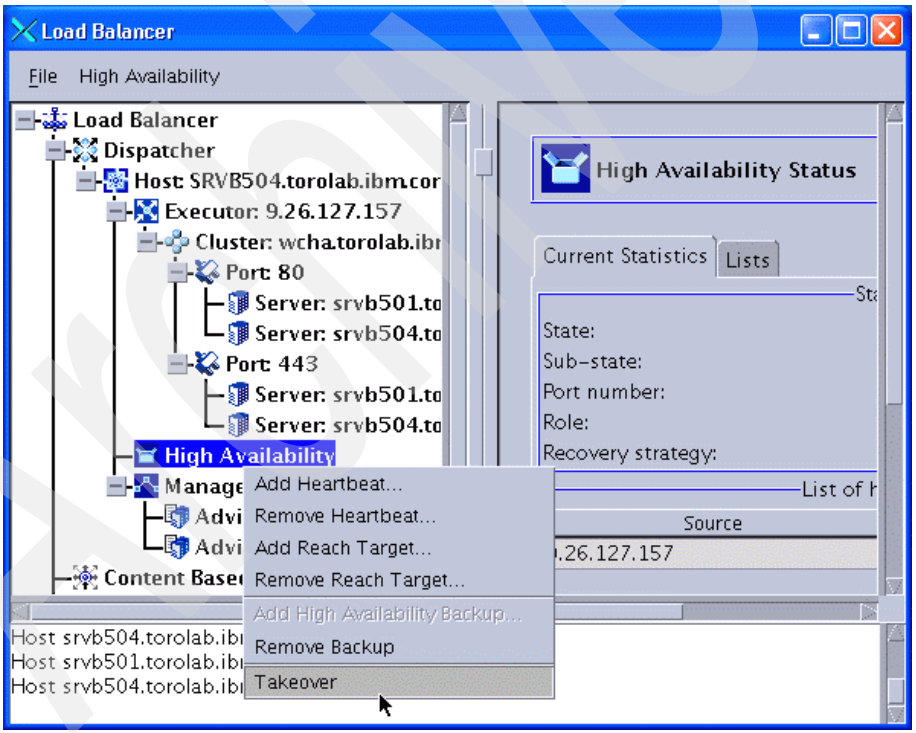


Figure 28-1 Taking over load balancing from the active machine

A pop-up window is displayed, as shown in Figure 28-2.

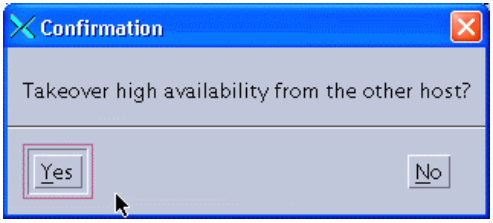


Figure 28-2 Take over confirmation dialog

b. Click **Yes** to take over load balancing from your Standby Load Balancer.

5. Remove the High Availability configuration from your Standby Load Balancer by right-clicking **High Availability** in the tree view (of the Standby Load Balancer GUI now) and selecting **Remove Backup** from the context menu. Then stop the executor, either by right-clicking **Executor** and selecting **Stop Executor**, or by using the command `dscontrol executor stop` (see also 11.3.5, “Test Load Balancer High Availability” on page 244).

   Note: At this point, until you have completed the temporary High Availability configuration as described in the next two steps, your primary Load Balancer is a single point of failure.

6. Remove the High Availability configuration from your primary Load Balancer by right-clicking **High Availability** and selecting **Remove Backup**.
7. Reconfigure High Availability using your primary Load Balancer as the primary machine and the temporary Load Balancer as the standby/backup machine. Follow the instructions in 11.3, “Configure Load Balancer High Availability” on page 226, but use manual as the recovery strategy.
8. Upgrade your Standby Load Balancer. See “Instructions for specific types of compatible upgrades” on page 601.
9. Now force a takeover of the temporary Load Balancer from your primary Load Balancer, so that your temporary machine becomes active. See step 4 on page 598 (which explains how to do this for the primary Load Balancer in case it is in standby state).
10. Remove the High Availability configuration and stop your primary Load Balancer. See step 5 (which explains how to do this for the Standby Load Balancer).

Archived

Note: At this point, until you have completed the next temporary High Availability configuration as described in the next two steps, your temporary Load Balancer is a single point of failure.

11. Remove the High Availability configuration from your temporary Load Balancer. (Step 6 on page 599 explains this for the primary Load Balancer.)
12. Reconfigure High Availability using your temporary Load Balancer (which is still load balancing traffic at this point) as the primary machine and the (updated) Standby Load Balancer as the standby/backup machine. Follow the instructions in 11.3, “Configure Load Balancer High Availability” on page 226, but use manual as the recovery strategy.
13. Upgrade your primary Load Balancer. See “Instructions for specific types of compatible upgrades” on page 601.
14. Now force a takeover of the Standby Load Balancer from your temporary Load Balancer, so that your Standby Load Balancer becomes active. See step 4 on page 598 (which explains this for the primary Load Balancer in case it is in standby state).
15. Remove the High Availability configuration from your temporary Load Balancer and stop your temporary Load Balancer. See step 5 on page 599 (which explains this for the Standby Load Balancer).

Note: At this point, until you have completed the final High Availability configuration as described in the next two steps, your Standby Load Balancer is a single point of failure.

16. Remove the High Availability configuration from your Standby Load Balancer. (Step 6 on page 599 explains this for the primary Load Balancer.)
17. Reconfigure High Availability using your (updated) primary Load Balancer as the primary machine and the Standby Load Balancer (which is still load balancing traffic at this point) as the standby/backup machine. Follow the instructions in 11.3, “Configure Load Balancer High Availability” on page 226. You may use automatic or manual recovery. If you choose automatic, the primary Load Balancer will take over as soon as the High Availability configuration is complete. If you choose manual, your Standby Load Balancer will continue to route traffic.

Archived

Upgrade without using a temporary machine

If you think that using a temporary machine is too complicated and do not mind your active Load Balancer being a single point of failure for some time, you may follow this following simple procedure that does not need a temporary machine:

1. Make sure that you save your Load Balancer configurations on both the primary Load Balancer and the Standby Load Balancer (as described in step 15 on page 208).
2. Stop the Load Balancer that is in standby state. To see which machine is in standby state, click **High Availability** in the left pane of the GUI and **Current Statistics** in the right pane to see the state (Figure 11-49 on page 231). To stop the Load Balancer, either right-click **Executor** and select **Stop Executor**, or use the command **`dscontrol executor stop`** (see also 11.3.5, “Test Load Balancer High Availability” on page 244).
3. Upgrade the machine that you just stopped. See “Instructions for specific types of compatible upgrades” on page 601.
4. Restart the Load Balancer by reloading the configuration that you saved before. In the GUI, right-click **Host: *Hostname*** in the tree view and select **Load New Configuration**. A pop-up window is displayed for choosing the configuration file. Choose the one that you just saved.
5. If your High Availability recovery strategy is manual, you may now force a takeover of the machine that you just upgraded from the active machine. See step 4 on page 598 (which explains it for the primary Load Balancer in case it is in standby state).
6. Stop the Load Balancer on the machine that still needs to be upgraded (see step 2 to see how to stop the Load Balancer). If your recovery strategy is auto, the other machine will now take over load balancing.
7. Upgrade the machine that you just stopped. See “Instructions for specific types of compatible upgrades” on page 601.
8. Restart the Load Balancer by reloading the configuration that you saved before, as explained for the other machine in step 4.
9. If your recovery strategy is manual, you may now choose to take over from the previously upgraded machine or let that machine continue with load balancing.

Instructions for specific types of compatible upgrades

This section lists details for the steps above about how to upgrade a machine.

Hardware and capacity upgrades

This type of upgrade most likely requires a server shutdown. After restarting, the Load Balancer server may simply be restarted by using the command **`dsserver`**

and reloading a configuration file. Some parameters possibly need to be adapted to the new hardware in order to utilize increased capacities.

OS upgrade

If a new OS is installed, Load Balancer might need to be reinstalled and reconfigured. See 8.6, “Install Load Balancer” on page 140. Configure the machine for load balancing your Web server cluster as described in 11.2, “Configure Load Balancer” on page 193. If you have done performance tuning as described in Chapter 19, “Monitor and tune Load Balancer” on page 417, also reapply your tuning to the new installation.

Software product upgrade

If a different major version of Load Balancer (or even a different Web server product) is installed, this might require reconfiguration different from the configuration described in this book for Load Balancer V6.0.

Refresh packs and fix packs

Refresh packs and fix packs normally do not require any reconfiguration. Refer to 8.6.2, “Install Load Balancer refresh pack” on page 143, for instructions on how to apply refresh packs and fix packs.

Fixes, APARs, iFix, eFix, and so on

Fixes normally do not require any reconfiguration. Fixes are installed similarly to fix packs. Also refer to the installation instructions that are included with the fixes.

28.2.2 Incompatible upgrades

We refer to upgrades as incompatible if an updated Load Balancer cannot exist in a High Availability configuration with a not -yet-updated Load Balancer. Possible reasons for incompatibility include:

- Installation of a new major release with an incompatible High Availability mechanism
- Migration to a new solution (for example, different software or a hardware-based solution)
- Migration to a different High Availability solution (for example, TSA)
- Migration to a new operating system.

In the case of an incompatible upgrade, we recommend installing the new high available load balancing configuration in parallel to the current installation, on new, dedicated hardware.

Archived

If you can use dedicated hardware, you need to make sure that no IP traffic is routed to the new installation during the setup phase, for example, by using different IP addresses or a separate network. When the installation is done, take down your current Load Balancer hardware and switch over to using the correct IP addresses on the new hardware.

If you want to reuse your existing hardware, you need to stop load balancing before installing the new solution. If you have a Web server that can handle the traffic for your site during the time frame for setting up the Load Balancer, you can temporarily assign the Web server cluster IP address to that Web server. Otherwise, we recommend taking the site down by having one Web server display a maintenance page (see 27.2.4, “Maintenance Web page for site downtimes” on page 592) and temporarily assigning the cluster IP address to that Web server.

Archive

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 608. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- *Mastering DynaCache in WebSphere Commerce*, SG24-7393

Other publications

These publications are also relevant as further information sources:

- *Load Balancer Administration Guide,* GC31-6858

Online resources

These Web sites are also relevant as further information sources:

- Edge caching

  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/concepts/cdc_esi.htm

- Tom Alcott: Everything you always wanted to know about WebSphere Application Server but were afraid to ask—Part 3

  http://www-128.ibm.com/developerworks/websphere/techjournal/0606_col_alcott/0606_col_alcott.html

© Copyright IBM Corp. 2008. All rights reserved.

- Implementing a Highly Available Infrastructure for WebSphere Application Server Network Deployment, Version 5.0, without Clustering

  http://www.ibm.com/developerworks/websphere/library/techarticles/0304_alcott/alcott.html

- High Availability Cluster Multi-Processing (HACMP)

  http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.hacmp.doc/hacmpbooks.html

- Solving memory problems in WebSphere applications

  http://www.ibm.com/developerworks/websphere/library/techarticles/0706_sun/0706_sun.html

- Solving performance degradation problems in WebSphere applications

  http://www.ibm.com/developerworks/websphere/library/techarticles/0706_lou/0706_lou.html

- RUNSTATS command

  http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/core/r0001980.htm

- RUNSTATS in DB2 UDB Version 8.2

  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0412pay/

- Reliable Scalable Cluster Technology (RSCT)

  http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsctbooks.html

- IBM Tivoli System Automation for Multiplatforms

  http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.1.html

- DB2 Information Center

  http://publib.boulder.ibm.com/infocenter/db2help/index.jsp

- Replication solutions for common scenarios

  http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod_overview/iiyrcintrsbdd.html

- Session management

  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp?topic=/com.ibm.commerce.admin.doc/concepts/csesmsession_mgmt.htm

Archived

- Deprecated and removed features

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rmig_deprecationlist.html

- Creating profiles using the graphical user interface

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/tpro_instances.html

- Caching Web 2.0 store pages

  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp?topic=/com.ibm.commerce.web20storesolution.refapp.doc/tasks/tsm_web20_extend.html

- iostat command

  http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.cmds/doc/aixcmds3/iostat.htm

- dtdgen utility

  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/index.jsp?topic=/com.ibm.commerce.data.doc/refs/rml_dtdgen.htm

- IBM Support Assistant Version 4.0

  http://www-306.ibm.com/software/support/isa/

- JVM Diagnostic Guide:

  http://www-128.ibm.com/developerworks/java/jdk/diagnosis/

- WebSphere Application Server MustGathers for debugging JVM Hang/Crash/OOM:

  http://www-1.ibm.com/support/docview.wss?uid=swg21145599

- Diagnostic Tool for Java garbage collector

  http://www.alphaworks.ibm.com/tech/gcdiag

- Heap Analyzer

  http://www.alphaworks.ibm.com/tech/heapanalyzer

- HeapRoots

  http://www.alphaworks.ibm.com/tech/heaproots

- Tutorial: Deploying precompiled JSP files to your WebSphere Commerce Server

  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/topic/com.ibm.commerce.samples.doc/tutorial/tdedeployjsp.htm

Archived

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Index

© Copyright IBM Corp. 2008. All rights reserved.

D

Archived

E

Archive

F

G

H

Archived

I

J

Archived

K

L

M

Archived

N

O

P

Archive

Q

R

S

Archived

T

Archived

U

V

W

Archived

X

Z

Archived

WebSphere Commerce High Availability and Performance Solutions

IBM CONFIDENTIAL

Archived

Archived

WebSphere Commerce High Availability and Performance Solutions

High Availability solutions for unplanned and planned outages

Installing and configuring a highly available system

Monitoring and performance tuning

Building a high performance and high availability Commerce site is not a trivial task—from having the correct capacity hardware to handling the workload to properly test the code change before deploying in a production site. This IBM Redbooks publication covers several major areas that need to be considered when using WebSphere® Commerce and provides solutions on how to address them. Some of the topics discussed are:

- How to build a Commerce site to deal with various kinds of unplanned outage. This includes utilizing IBM WebSphere Application Server Network Deployment 6.0 and IBM® DB2® High Availability Disaster Recovery (HADR) in a Commerce environment.
- How to build a Commerce site to deal with planned outages such as software fixes and operation updates. This include use of the WebSphere Application Server Rollout Update feature and the use of the Commerce Staging Server and Content Management.
- How to proactively monitor the Commerce site and prevent potential problems from occurring. We discuss various tools, such as WebSphere Application Server build-in tools and Tivoli®'s Performance Viewer.
- How to utilize Dynacache to future enhance your Commerce site's performance. This includes additional Commerce command caching introduced in the Commerce fix pack and e-spot caching.
- The methodology of doing performance and scalability testing on a Commerce site.

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

SG24-7512-00 ISBN 0738431338