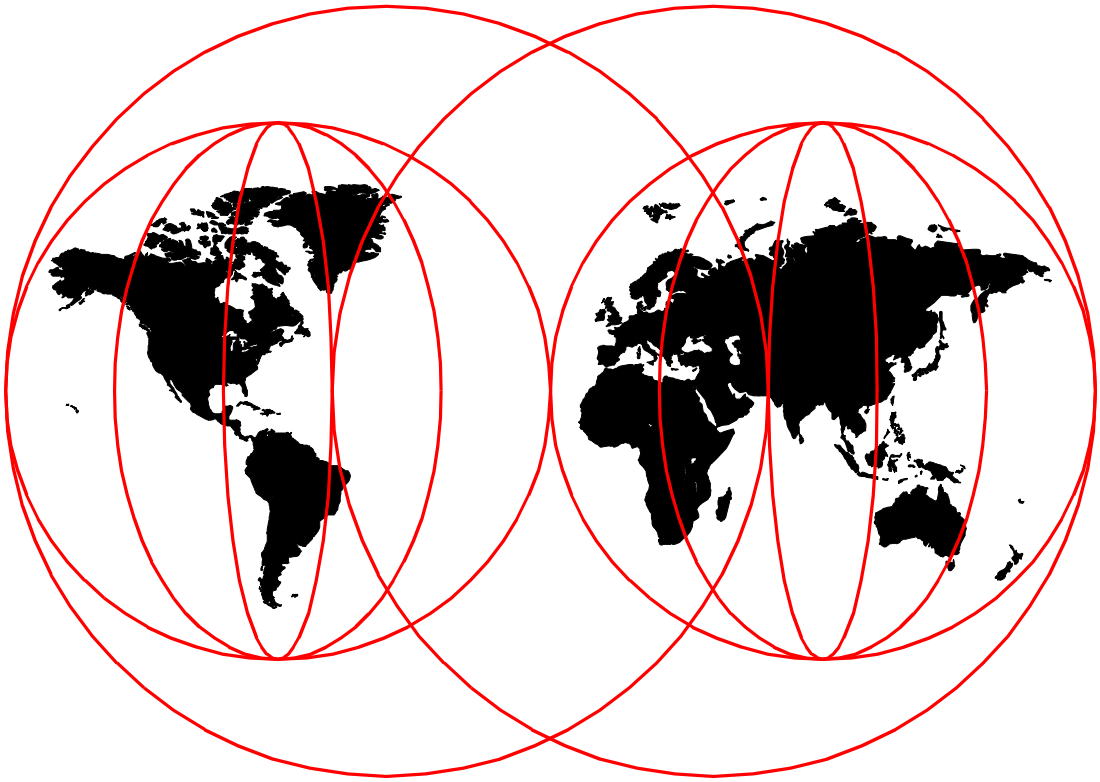
WebSphere V3 Performance Tuning Guide

Ken Ueno, Tom Alcott, Jeff Carlson, Andrew Dunshea, Hajo Kitzhöfer, Yuko Hayakawa, Frank Mogus, Colin D. Wordsworth



International Technical Support Organization

www.redbooks.ibm.com

SG24-5657-00

International Technical Support Organization

SG24-5657-00

WebSphere V3
Performance Tuning Guide

March 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special notices” on page 185.

First Edition (March 2000)

This edition applies to:

- IBM WebSphere Application Server Version 3.02
- IBM HTTP Server Version 1.3.6.2
- IBM Java Development Kit Version 1.1.6.9

for use with the AIX V4.3.2 operating system.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Preface

This redbook will help you to design and configure WebSphere Application Server V3.02 for AIX, Solaris, and Windows NT for better performance. The main tuning objectives for WebSphere are to improve performance, response time, and resource utilization.

This redbook gives some general recommendations and describes specific tuning methodologies. It provides hints and tips on the various factors and variables that can enhance the performance of WebSphere including AIX networks and IBM HTTP Server performance tuning. We introduce performance test tools and monitoring tools. WebSphere tuning methodology examples are included. This redbook also contains an overview of WebSphere Site Analyzer. We also provide a performance tuning guide for AFS, which is a part of WebSphere Performance Pack.

Some knowledge of WebSphere Application Server V3.02, DB2 UDB, and AIX are assumed. Note that in this redbook we will not discuss the basic configurations of WebSphere, DB2 UDB, or AIX.

The team that wrote this redbook

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

Ken Ueno is an advisory International Technical Support Representative at the International Technical Support Organization, Raleigh Center. He manages residencies and produces redbooks. Before joining the ITSO, he worked in Internet Systems, IBM Japan Systems Engineering Co., Ltd in Japan as an I/T Specialist. He can be reached at kenueno@us.ibm.com.

Tom Alcott is an advisory I/T specialist in the United States. He has been a member of the World Wide WebSphere Technical Sales Support team since its inception. Before he started working with WebSphere, he was a systems engineer for IBM's Transarc Lab supporting TXSeries. His background includes over 18 years of application design and development on both mainframe-based and distributed systems. He has written and presented extensively on a number of WebSphere runtime and security issues.

Jeff Carlson is a Staff Software Engineer on the AFS/WSPP team at the IBM Transarc Lab in the United States. He has over 10 years of experience in the support of various mainframe, PC/LAN, and UNIX workstation products. Jeff graduated Magna Cum Laude from Robert Morris College with a Bachelor's degree in Business Administration. At the IBM Transarc Lab, he has written extensively on AFS performance tuning.

Andrew Dunshea is a Performance Analyst from IBM New Zealand. He has 10 years of experience in application development. His areas of expertise include object-oriented software development and analysis, systems programming, and performance analysis.

Dr. Hajo Kitzhöfer is an Advisory International Technical Support Organization (ITSO) Specialist for RS/6000 SP at the Poughkeepsie Center. His areas of

expertise include RS/6000 SP, SMP, and Benchmarks. He now specializes in SP System Management, SP Performance Tuning, and SP hardware.

Yuko Hayakawa is an I/T Specialist in Japan. she has four years of experience in the e-commerce field. She has worked at IBM for 4 years. Yuko has written extensively on database tuning and session management.

Frank Mogus is a Systems Consultant in Canada. He has several years of UNIX experience and has worked with The Braegen Group for four years.

Colin D. Wordsworth is a software engineer in Australia. He has 16 years of experience in applications development. He has been with IBM for the last five years, and is currently involved in setting up the e-business centre for excellence in Western Australia.

Thanks to the following people from the International Technical Support Organization, Raleigh Center:

Carla Sadtler
Gail Christensen
Shawn Walsh
John Ganci
Barry Nusbaum
Margaret Ticknor
Mike Haley

Thanks to the following IBM employees:

Ruth Willenborg, Manager, WebSphere Performance, Raleigh
Chris Forte, WebSphere Performance, Raleigh
Ron Bostick, WebSphere Performance, Raleigh
Charlie Bradley, WebSphere Performance, Raleigh
Jerry Cuomo, Manager, WebSphere Performance and Security, Raleigh
Graeme N. Dixon, STSM, WebSphere Development, IBM Transarc Lab
Jason R McGee, WebSphere Development, Raleigh
Songquan Liu, WebSphere Development, IBM Transarc Lab
Tianyu Jiang, WebSphere Development, IBM Transarc Lab
Cindy Tipper, WebSphere Development, IBM Transarc Lab
Chris Newbold, WebSphere Development, IBM Transarc Lab
Chriss Stephens, WebSphere Development, IBM Transarc Lab
Gabe Montero, WebSphere Development, Raleigh
Allan Dickson, WebSphere Development, Raleigh
Linh Nguyen, Project Manager, IBM HTTP Server Development, Raleigh
David Allen, IBM HTTP Server Development, Raleigh
Tom Hartrick, Manager, Site Analyzer Development, Raleigh
Gopi Attaluri, DB2 Performance Team, Toronto
Steve Schormann, DB2 Performance Team, Toronto
Richard Nesbitt, Special Events Development, Raleigh
Steve King, System Administration, Raleigh
Marco Pistoia, T.J. Watson Research
Tetsuya Shirai, ITSO San Jose Center
Joanne Luedtke, Manager, ITSO Austin Center

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks review” on page 197 to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an internet note to redbook@us.ibm.com

Chapter 1. Overview of WebSphere Application Server V3

This chapter gives an overview of the WebSphere Application Server V3, which comes in three editions:

- WebSphere Application Server, Standard Edition
- WebSphere Application Server, Advanced Edition
- WebSphere Application Server, Enterprise Edition

The Enterprise Edition builds on the Advanced Edition. It combines the TXSeries and Component Broker with the Advanced Edition. The Enterprise Edition is *not* covered in this redbook.

1.1 WebSphere Application Server V3

Our discussion will cover two editions of WebSphere: Standard and Advanced. Both products are based on and support key open-industry standards such as HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML), Extensible Markup Language (XML), Secure Sockets Layer (SSL), Java, JavaBeans, Common Object Request Broker Architecture (CORBA), Lightweight Directory Access Protocol (LDAP), and most importantly the following Enterprise Java APIs:

- Enterprise JavaBeans (EJB) technology is a reusable Java component for connectivity and transactions (EJB support is provided only in the Advanced Edition).
- Java Server Pages (JSP) represent inline Java code scripted within Web pages.
- Java Servlets are used in building and deploying server-side Java applications.
- Java Interface Definition Language (JIDL), is used to connect to CORBA objects and applications.
- Java DataBase Connectivity (JDBC) is for connections to relational databases. WebSphere supports JDBC within its Connection Manager and within EJBs, for distributed database interactions and transactions.
- Java Messaging Service (JMS) is to be supported via MQSeries for asynchronous messaging and queuing and for providing an interface.
- Java Transaction Service (JTS) and Java Transaction API (JTA) are low-level APIs for interacting with transaction processing systems and relational databases, respectively. WebSphere uses these within EJBs for supporting distributed transactions.
- Java Naming and Directory Interface (JNDI) is for communicating with directories and naming systems and is used in WebSphere Application Server to look up existing EJBs and interact with directories.
- Java Remote Method Invocation over Internet Inter-ORB Protocol (RMI/IIOP) is for communicating with other ORBs (Object Request Brokers) and CORBA-compliant applications.

Version 3 of WebSphere introduces a new runtime architecture for the WebSphere Application Server. This runtime architecture is divided into the following components:

- Administration Server (“adminserver”) - The adminserver is the Systems Management runtime component of WebSphere. The adminserver is responsible for runtime management, security, transaction coordination, and workload management. In most cases (exceptions will be outlined later) the adminserver runs on all nodes in a WebSphere Administrative domain and controls the interaction between each node and application server process in the domain.
- Administrative Console (“adminclient”) - The adminclient is the graphical user interface used for administration of a WebSphere Administrative domain. The adminclient can run on one of the nodes that the adminserver is running on, or it can be a remote node that attaches to a running adminserver.
- System Configuration Repository - WebSphere stores all runtime configuration information in a persistent repository. In Standard Edition this repository can be either UDB V5.2, UDB V6.1, Oracle 8.05, or InstantDB (which ships with the Standard Edition). Advanced Edition supports UDB and Oracle as noted for Standard Edition. This repository can either exist on the same physical server as WebSphere or can be configured on a remote server.
- Application Server - WebSphere V3 introduces the notion of an application server process that is separate from the runtime server (adminserver). In either WebSphere Standard or Advanced Edition, you can define multiple application servers, each of which has its own JVM. These application servers in turn can have either a servlet engine, EJB container, or both defined to them depending on application requirements.

The components of WebSphere V3 are depicted below.

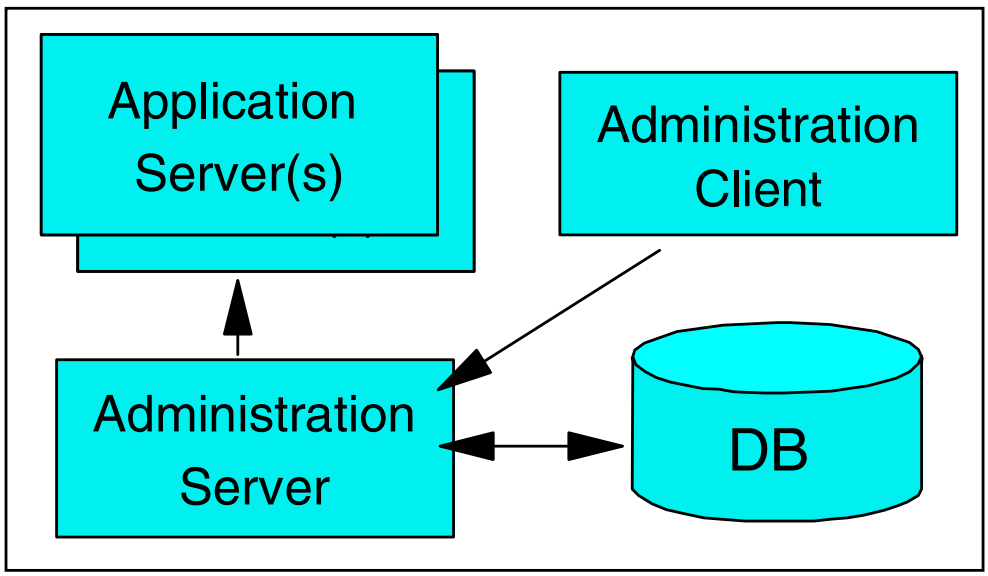


Figure 1. WebSphere components

1.2 Standard Edition

WebSphere Standard Edition is a single system, extremely easy-to-use Web application server. Programmers developing applications think that they are implementing extended HTML content. The content/page styles include:

- Static HTML (HTML, .gif, .wav, etc.)
- HTML with imbedded client-side scripts, for example JavaScript
- Java Server Pages (JSP)

WebSphere Standard Edition's objective is to be a simple, easy-to-use but complete solution for building an active Web site and basic Web applications that integrate with databases.

WebSphere Standard Edition does not provide the Work Load Management (WLM) functionality that is available in WebSphere Advanced Edition, but does allow for multiple JVMs on a single physical server. WebSphere Standard Edition is also limited to a single node/machine unlike WebSphere Advanced Edition. These JVMs can be mapped to multiple virtual hosts on a single HTTP server to provide support for hosting multiple Web sites on a single application server. This is useful for Internet service providers (ISPs) or for a company wishing to host the Internet and Intranet Web site on a single server, since WebSphere Standard Edition has single system application server scalability, achieved through the use of a product such as Network Dispatcher, which is a part of WebSphere Performance Pack. An example configuration is shown in Figure 2.

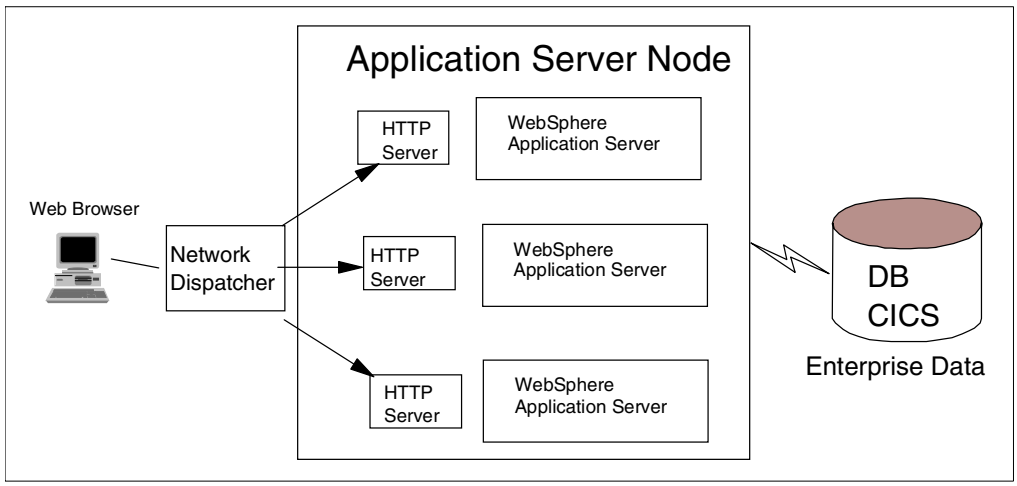


Figure 2. WebSphere Application Server with Network Dispatcher

1.3 Advanced Edition

WebSphere Advanced Edition extends the WebSphere Standard Edition's functions across multiple machines to provide complete support for developing new high-performance, scalable and available, transactional Web-driven applications. WebSphere Advanced Edition focuses on new applications (JSPs and EJBs) that access relational databases for persistent state data.

WebSphere Advanced Edition also supports distributed system management for networks and clusters of WebSphere Advanced Edition systems, and supports a shared name and security between networks and clusters of WebSphere Advanced Edition systems.

In an object-oriented distributed computing environment, clients must have a mechanism to locate and identify the objects as if the clients and objects were all on the same machine. A naming service provides this mechanism. In the EJB server environment, the Java Naming and Directory Interface (JNDI) is used to provide a common front-end to the naming service.

JNDI provides naming and directory functionality to Java applications, but the API is independent of any specific implementation of a naming and directory service. This independence ensures that different naming and directory services can be used by accessing them through the JNDI API. Therefore, Java applications can use many existing naming and directory services, for example, the Lightweight Directory Access Protocol (LDAP) or the Domain Name System (DNS). WebSphere V3 supports JNDI via CosNaming only.

In the Advanced Edition environment, the main component of the security service is an EJB server that runs as a component of the adminserver that contains the security beans. When system administrators administer the security service, they manipulate the security beans.

After an EJB client is authenticated, it can attempt to invoke methods on the enterprise beans that it manipulates. A method is successfully invoked if the principal associated with the method invocation has the required permissions to invoke the method. These permissions can be set by application (an administrator-defined set of Web and object resources) and by method group (an administrator-defined set of Java interface-method pairs). An application can contain multiple method groups.

In general, the principal under which a method is invoked is associated with that invocation across multiple Web servers and EJB servers (this association is known as delegation). Delegating the method invocations in this way ensures that the user of an EJB client needs to authenticate only once. HTTP cookies are used to propagate a user's authentication information across multiple Web servers. These cookies have a lifetime equal to the life of the browser session.

A transaction is a set of operations that transforms data from one consistent state to another. The EJB server manages transactions for EJB applications by using the mechanism defined in the Java Transaction API (JTA).

For most purposes, enterprise bean developers can delegate the tasks involved in managing a transaction to the EJB server. The developer performs this delegation by setting the deployment descriptor attributes for transactions. The enterprise bean code itself does not need to contain transactional logic.

The Work Load Management (WLM) functionality in WebSphere Advanced Edition introduces the notion of modelling of application server processes. Clones, which are instances of a model, can be created either on a single machine or across multiple machines in a cluster. In either case the WebSphere Advanced Edition WLM provides workload distribution and failover. A sample configuration is shown in Figure 3 on page 5.

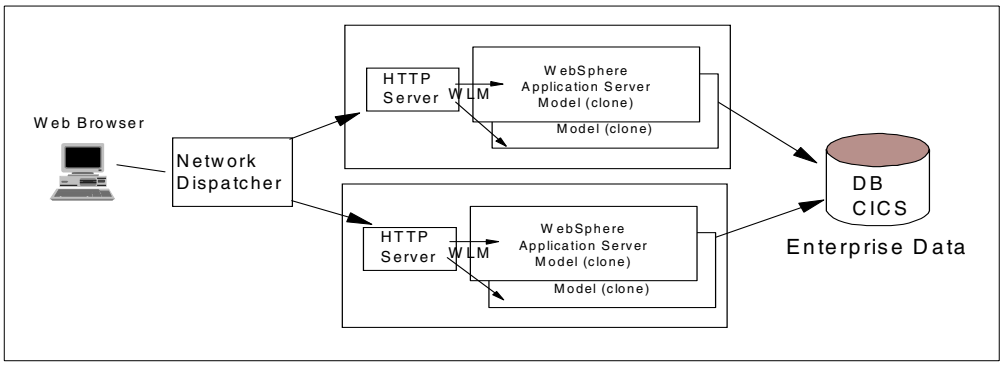


Figure 3. WebSphere Application Server WLM with Network Dispatcher

While the WLM in WebSphere Advanced Edition can work in conjunction with the Network Dispatcher (ND) component of WebSphere Performance Pack as depicted above, WLM can also be used independently of ND. The WebSphere product family provides support for scalability in two dimensions:

- The number of concurrent connected users (horizontal scaling)
- The number of concurrent requests (vertical scaling)

Horizontal scaling is used to add support for additional concurrent users. For example, if a machine is serving 100 transactions per second to 100 users with the desired response time, and you want to be able to serve more concurrently accessed users, then horizontal scaling techniques of adding additional hardware with the HTTP server and WebSphere Application Server can be used. This allows you to serve nearly twice the client load, at the same response time.

Vertical scaling is used to increase the concurrent requests. For example, if a machine is serving 100 transactions per second to 100 users, and you want to be able to serve high transaction rates (that is 200 tps), then vertical scaling techniques of adding additional Application Servers to the machine can be used. This assumes that the machine is not already running at capacity (if it is, then you would also have to add more processors).

WebSphere Performance Pack helps with the number of connected users. The WLM in WebSphere Advanced Edition uses multithreading, and connection pool management helps to increase the number of concurrent requests.

Since the focus of this redbook is the WebSphere Application Server, we will not be testing system topologies using WebSphere Performance Pack. However, it should be mentioned that lab-based performance testing has demonstrated near linear scalability (greater than 96%) across nodes when using WebSphere Performance Pack. WebSphere Performance Pack also has been used in an impressive list of production sites, including the Atlanta and Nagano Olympic games.

Chapter 2. Performance tuning approach

This redbook is a performance tuning guide for Web application servers. We outline various performance tuning points along with suggestions and recommendations for initial starting values. Web application server performance tuning is an ongoing learning experience especially with WebSphere Application Server V3. Since the largest factor in Web application server performance is the specific application, specifics of performance tuning for your application might vary from those presented here. Accordingly, we will not present the results of performance benchmark tests that we performed for this publication.

There are three main areas that affect Web application server performance:

- The hardware capacity and settings
- The operating system settings
- The Web application server settings

In this redbook, we do not discuss hardware capacity or settings. Though it might sound trite, the simplest and best way to improve performance is to increase memory and processor speed as long as you are not I/O or network bound. If you have the latter problem, then reconfiguring clients and servers into the same server will help (assuming free cycles on the app server machine).

Your network topology and network adapter card settings can also affect overall performance. You might want to consider that outgoing response packets from Web application server are much larger than incoming request packets from browsers to Web application server.

We discuss the operating system settings in Chapter 3, “AIX TCP/IP tuning” on page 15.

The Web application server settings are the main topics of this redbook. In this book, we focus on WebSphere Application Server V3 for AIX. The WebSphere Application Server V3 architecture provides several decision points that affect the overall performance of your Web applications. Each decision point, in turn, offers trade-offs that you might view favorably or negatively depending on your circumstances. There is no one-size-fits-all solution.

There are four fundamental decision points that have their own tuning options:

1. The Web server (HTTP server)
2. The WebSphere Application Server Engine
3. The Java Virtual Machine (JVM)
4. The database server

We discuss all of them in this redbook. Chapter 4, “Web Server: IBM HTTP Server 1.3.6” on page 21 discusses the Web server settings. Chapter 5, “WebSphere Engine” on page 29 and Chapter 6, “Security” on page 49 discuss the WebSphere Application Server Engine and JVM. The database server settings are covered in Chapter 7, “Database tuning” on page 57 and Chapter 8, “Session management” on page 75.

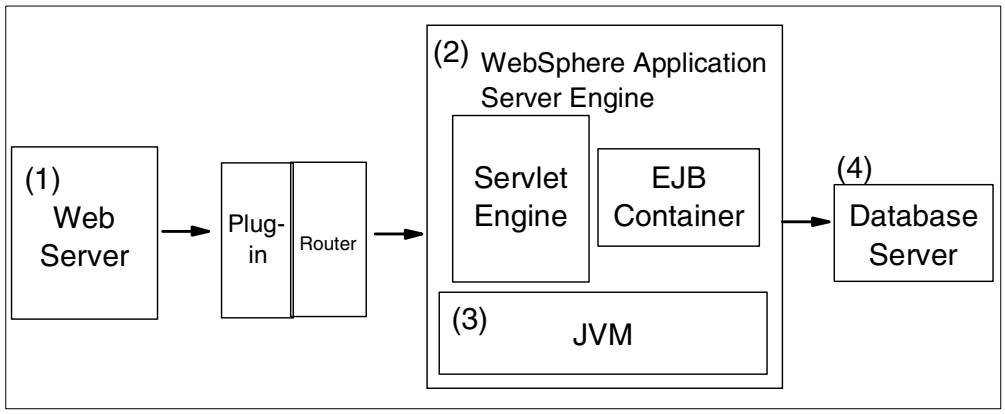


Figure 4. Fundamental tuning points of WebSphere Application Server settings

2.1 WebSphere Application Server V3 topologies

The performance testing done for this redbook was based on four distinct topologies.

2.1.1 Topology 1: single machine

In order to establish a baseline for our performance testing we configured a single machine with the topology depicted in Figure 5. In our single machine topology (topology 1) the database server with the business application data and the configuration repository reside on the same machine as the WebSphere Application Server (WAS).

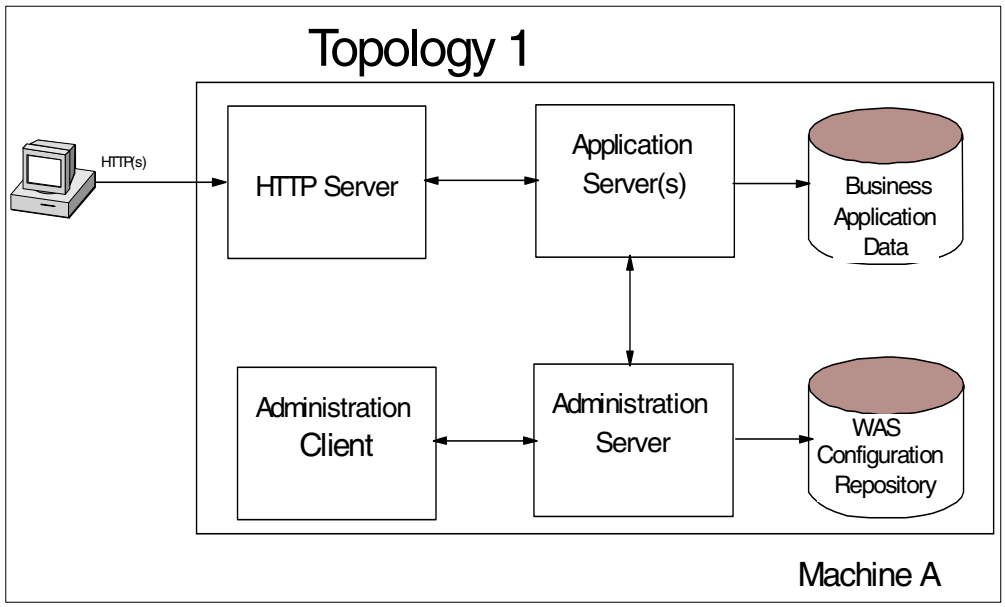


Figure 5. Topology 1

While this configuration may be suitable for a development environment, it should be avoided for production environments where performance is a concern.

Installation of a database on a UNIX system involves changes to the UNIX OS kernel. Depending on the database and operating system the changes are either made explicitly by the systems administrator or by the installation program for the database. For example, kernel changes are made automatically on AIX when installing DB2, while installation of DB2 on Solaris or HP-UX require explicit modification of the kernel. Oracle also requires modification of the kernel parameters. The result is that a server running both a database server and WebSphere Application Server is optimized for database performance and not the application server.

The other issue and perhaps more significant factor with running both processes on the same physical server is that under load you have both WebSphere and the database competing for system resources.

When we moved the database processes to a second remote server (shown later in topology 2), we observed a significant performance improvement over running the application server processes and database server processes on the same node as WAS.

Note for Windows NT

Though Windows NT uses a thread-based architecture as opposed to the process-based architecture of UNIX, performance on Windows NT suffers to a similar degree when hosting both the database and application server on the same machine.

Performance Tip

Run the WebSphere Application Server and the database server on separate servers.

2.1.2 Topology 2: separating the database server from WAS

While it is possible to host both WAS and its configuration/administrative repository on the same physical server as shown in topology 1 in Figure 5 on page 8, in practice most organizations choose to maintain a single database server that hosts all databases, most likely residing on a separate machine from WAS. This means putting the WAS configuration/administrative repository on the same database server as the business application data, facilitating a more streamlined system of backups, system maintenance and performance tuning.

Topology 2, shown in Figure 6 on page 10, would be typical for a small Web site where all the application server and HTTP server processes are hosted on a single machine. In this topology, we used the same database server for the WAS configuration repository, business application data, and session persistence manager.

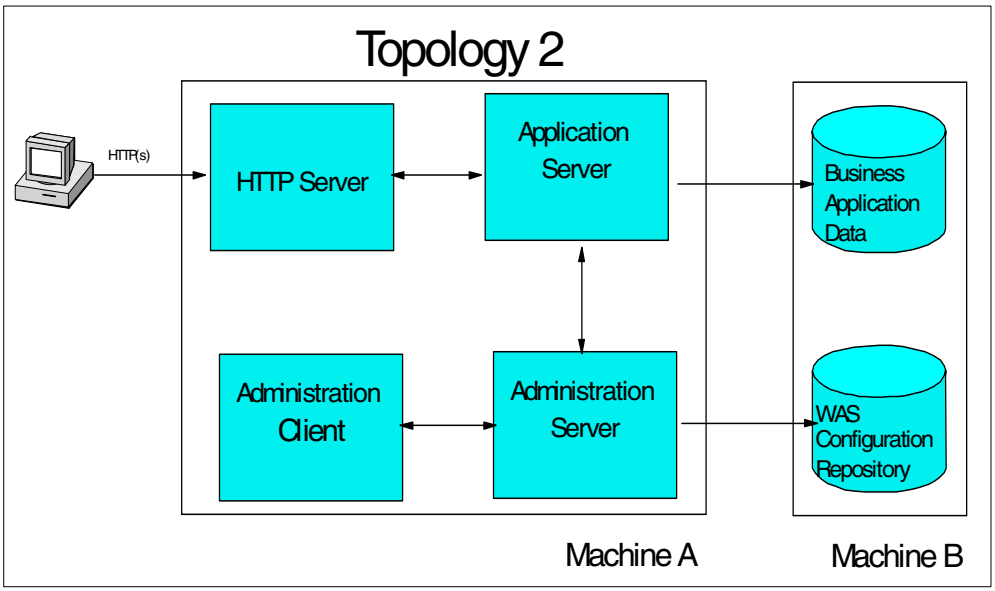


Figure 6. Topology 2

2.1.3 Topology 3: multiple application servers

Topology 3, shown in Figure 7, expands on topology 2 by adding a clone of the application server process on the same server that the other application server and the HTTP server processes are running on. This topology demonstrates the application process scalability and WLM capability for a single server.

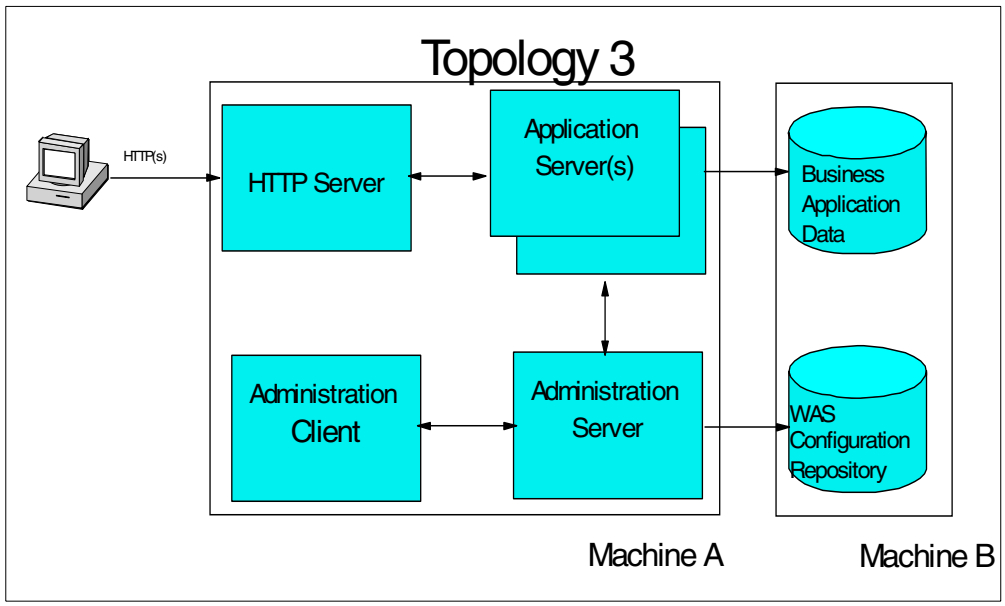


Figure 7. Topology 3

2.1.4 Topology 4: advanced topology

In topology 4, shown in Figure 8 on page 11, we have separated the HTTP server from the application server using the servlet redirector that comes with WebSphere Application Server. This topology is typical of many customer

environments where the HTTP server is physically separated from the application server and runs on one side of a firewall, either outside the firewall or between two firewalls inside a Demilitarized Zone (DMZ). For our testing we did not actually configure a firewall.

In this topology, the HTTP server communicates locally with the servlet redirector using Open Servlet Engine (OSE). The servlet redirector in turn, forwards or “redirects” the servlet requests to the application server running on another physical server via Internet Inter-ORB Protocol (IIOP).

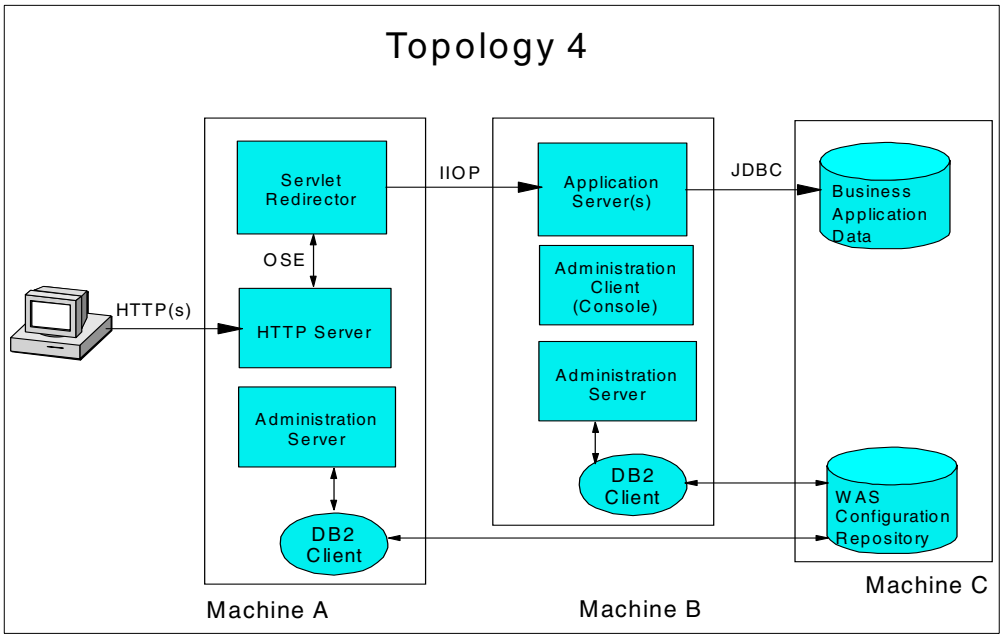


Figure 8. Topology 4

2.1.5 Our test application

The application used for our performance tests is known as the Trade application. This sample application program is an internal tool. It was designed to test the WebSphere Application Server for aspects of scalability, performance and competitiveness. The Trade application is a collection of Java classes, Java servlets, Java Server Pages and Enterprise JavaBeans. The application is designed to be portable between the WebSphere 3.x Standard, Advanced and Enterprise Editions.

The Trade application simulates an online stock trading Web site. The application allows a user to perform the following actions using a Web browser:

- Registration
- Site login
- Inquiry of the current price for a stock ticker symbol
- Purchase of shares
- Sale of shares
- Portfolio review
- Logout from site

2.1.5.1 Trade application components

The Trade application contains six major objects:

- Trade
- Account
- Registry
- Profile
- Holding
- Quote

These objects and their primary data fields and methods are depicted in Figure 9.

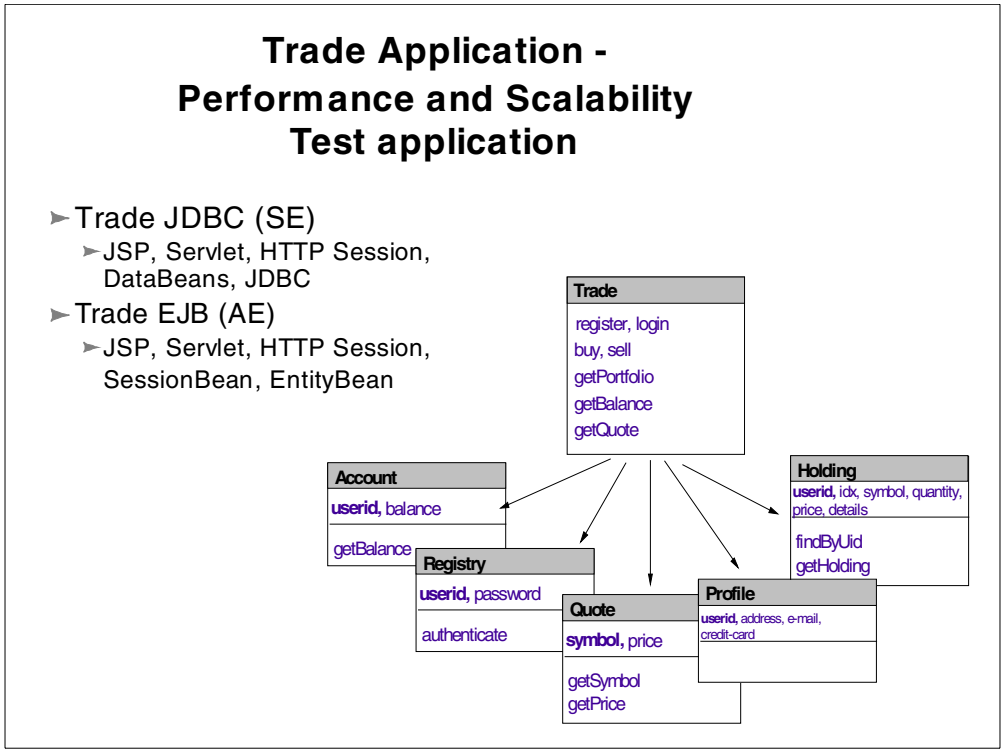


Figure 9. Trade application components

The Trade object is a stateless object that encapsulates the primary operations of the Trade application. When the Trade application is configured to use EJBs, the Trade object takes the form of a Stateless Session EJB. Otherwise the Trade object takes the form of a standard Java class.

The five remaining objects: Account, Registry, Quote, Profile and Holding are data objects. When the Trade application is configured to use EJBs, these objects are Container Managed Entity EJBs. Otherwise, these objects are encapsulated in a Java data bean. The data bean is a standard Java class that uses JDBC prepared SQL statements to manipulate data within a database table.

The topology for the Trade application is shown in Figure 10 on page 13.

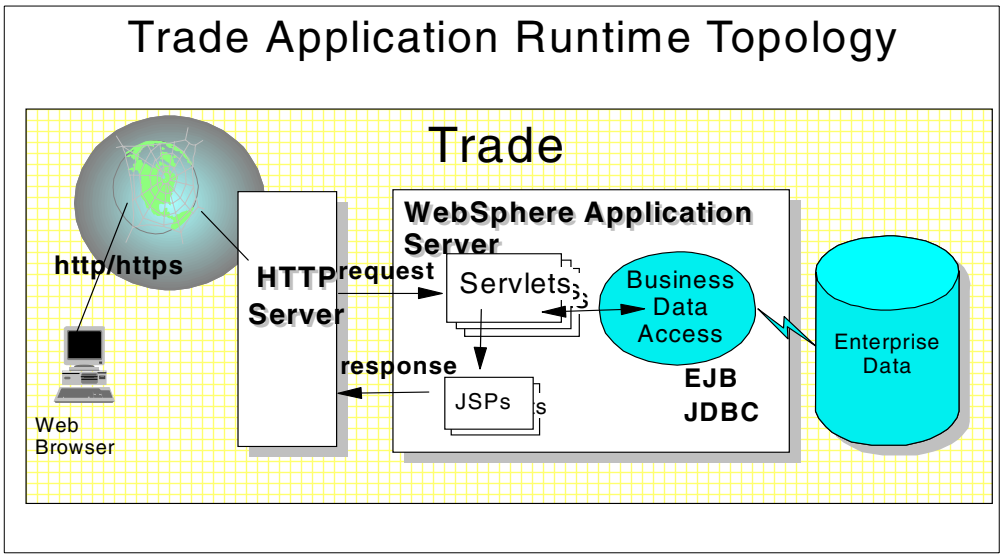


Figure 10. Trade application runtime

The Trade servlet is the controller for the application with calls from servlet instances to the DataBeans and JDBC as required. As the controller for the application, the Trade servlet coordinates:

- Registering or logging in
- Buy scenarios (which includes getting quotes and buying stocks)
- Sell scenarios (which includes browsing your portfolio and sales of holdings)
- Logging out

2.2 Network and hardware configuration

All tests, unless otherwise stated, were run on a nine-node SP frame. Eight of the SP nodes were 2x332 MHz CPUs with 512 MB of RAM, while the ninth node was a 4x332 MHz CPU with 1 GB of RAM. Each node has a 9 GB disk assigned to it.

This allowed us to simulate multiple physical servers while minimizing the network latency. While in the real world there would be some network latency, the purpose of this publication is to determine the optimal settings for the WebSphere Application Server runtime, and this is best done by eliminating external influences such as network latency. The hardware and corresponding software configuration for our performance testing is shown in Figure 11 on page 14.

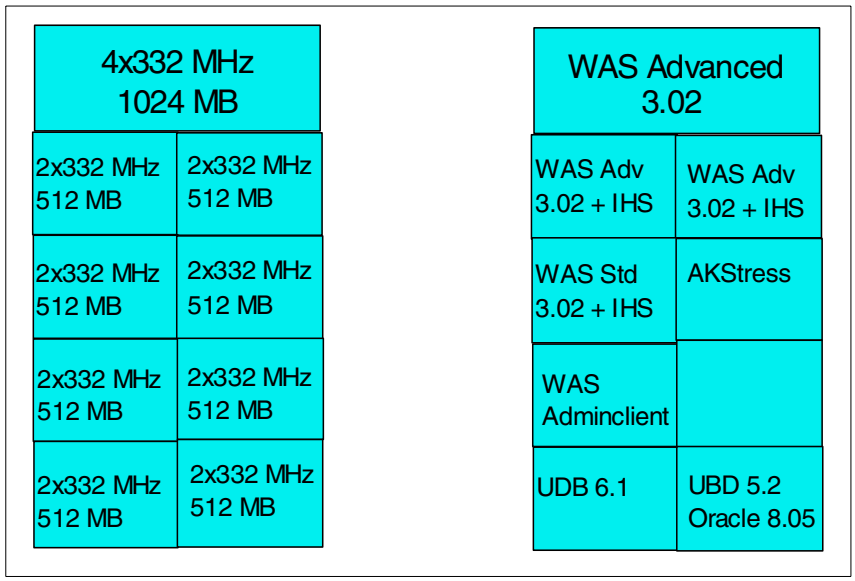


Figure 11. Testing hardware and software configuration

The software used was:

- AIX 4.3.2 + PTF
- WebSphere Application Server Advanced Edition V3.02 for AIX
- UDB V5.2 + FP11
- UDB V6.1 + FP1c
- Oracle 8.05
- IBM HTTP Server V1.3.6
- JDK 1.1.6.9 (PTF9)

Chapter 3. AIX TCP/IP tuning

Before you start WebSphere tuning, we recommend reviewing and modifying the system settings of your Web site, especially the TCP/IP network settings because when Web browsers access WebSphere, they have to establish TCP/IP network connections. In this chapter, we show you several TCP/IP network tunable parameters that should be modified. We provide an AIX TCP/IP overview as an appendix. For more in-depth knowledge of AIX performance tuning, we recommend the *AIX Performance Tuning Guide*, SR28-5930 and *RS/6000 SP System Performance Tuning*, SG24-5340.

3.1 Network tuning with no command

The tunable parameters should be set to customized values during the AIX installation process.

Use the `no` command to display the current settings and change it as shown below.

```
no -o <parameter>
no -o <parameter>=<newvalue>
```

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

Figure 12. Usage of no command

In the output of the `no -a` command taken from AIX 4.3.2 in Figure 13 on page 16, the appropriate tunables are highlighted.

The following are specific details for setting the network tunables for AIX systems:

`thewall`

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

`sb_max`

Provides an absolute upper bound on the size of TCP and UDP socket buffers per socket. Limits udp_sendspace, udp_recvspace, tcp_sendspace and tcp_recvspace. The units are in bytes. For tuning, increase size, preferably in multiples of 4096. This value should be at least twice the size of the largest value for tcp_sendspace, tcp_recvspace, or udp_recvspace. This ensures that if the buffer utilization is better than 50% efficient, then the entire size of the

tcp and udp byte limits can be utilized. Changes take effect immediately for new connections and remain in effect until the next reboot.

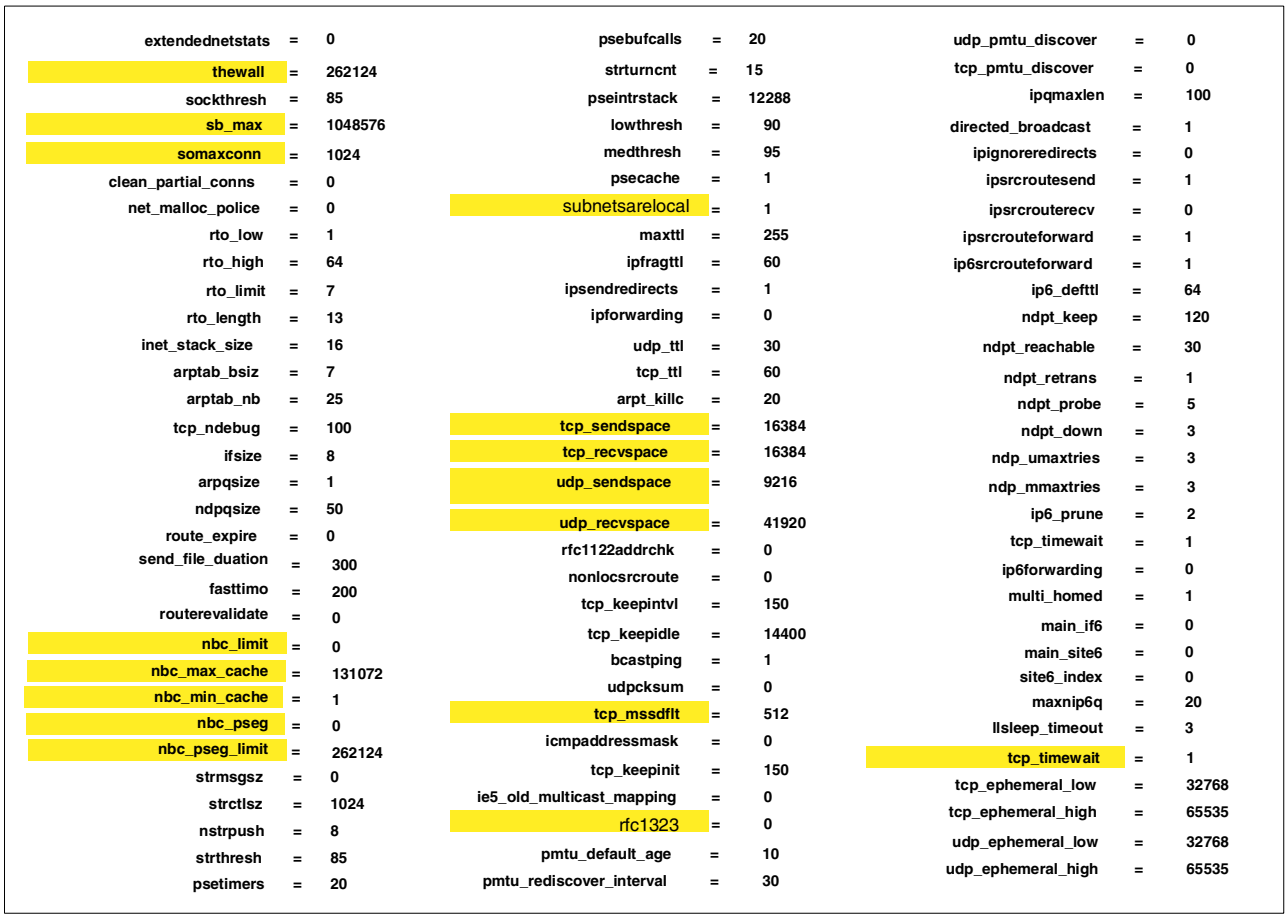


Figure 13. Displaying Network Options

somaxconn

Specifies the maximum listen backlog. The default is 1024 bytes. somaxconn is a runtime attribute.

tcp_sendspace

Provides the default value for the size of the TCP socket send buffer, in bytes. This is never higher than the major network adapter transmit queue limit. To calculate this limit, use:

(major adapter queue size) * (major network adapter MTU)

For tuning increase size, preferably to a multiple of 4096. Changes take effect immediately for new connections and remain in effect until the next reboot.

tcp_recvspace

Provides the default value for the size of the TCP socket receive buffer. The value is in bytes. This is never higher than the major network adapter transmit queue limit.

(major adapter queue size) * (major network adapter MTU)

For tuning, increase size, preferably to a multiple of 4096. Changes take effect immediately for new connections and remain in effect until the next reboot.

`tcp_sendspace and tcp_recvspace`

These tunables establish the size of the TCP window on a per datagram socket connection. The effective size used is the least common denominator between the sending side tcp_sendspace and the receiving side tcp_recvspace. The size depends on the network you are sending over.

`udp_sendspace`

Provides the default value for the size of the UDP socket send buffer, in bytes. Set to 65536, because anything beyond 65536 is essentially ineffective. Since UDP transmits a packet as soon as it gets any data, and since IP has an upper limit of 65536 bytes per packet, anything beyond 65536 runs the small risk of getting thrown away by IP. For tuning, increase size, preferably to a multiple of 4096. Should always be less than udp_recvspace, but never greater than 65536. Changes take effect immediately for new connections and remain in effect until the next reboot.

`udp_recvspace`

Provides the default value for the size of the UDP socket receive buffer. A suggestion for the starting value of udp_recvspace is 10 times the value of udp_sendspace, because UDP may not be able to pass a packet to the application before another one arrives. For tuning, increase size, preferably to a multiple of 4096. Should always be greater than udp_sendspace and sized to handle as many simultaneous UDP packets as can be expected per UDP socket. Changes take effect immediately for new connections and remain in effect until the next reboot.

`rfc1323`

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

`tcp_timewait`

The tcp_timewait option is used to configure how long connections are kept in the timewait state. It is given in 15 second intervals, and the default is 1.

`nbc_max_cache`

Specifies the maximum size of the cache object allowed in the Network Buffer Cache (NBC). This attribute is in bytes; the default is 131072 (128 KB). A data object bigger than this size is not put in the NBC.

`nbc_pseg_limit`

Specifies the maximum amount of cached data allowed in private segments in the Network Buffer Cache. This value is expressed in KB. The default value is half of the total real memory size on the running system. Since data cached in private segments is pinned by the Network Buffer Cache, nbc_pseg_limit

controls the amount of pinned memory used for the Network Buffer Cache in addition to the network buffers in global segments. When the amount of cached data reaches this limit, cache data in private segments may be flushed for new cache data so that the total pinned memory size doesn't exceed the limit. When nbc_pseg_limit is set to 0, all caches in private segments are flushed.

`nbc_pseg`

Specifies the maximum number of private segments that can be created for the Network Buffer Cache. The default value is 0. When this option is set at non-0, a data object between the size specified in nbc_max_cache and the segment size (256 MB) is cached in a private segment. A data object bigger than the segment size is not cached at all. When the maximum number of private segments exist, cache data in private segments may be flushed for new cache data so that the number of private segments do not exceed the limit. When nbc_pseg is set to 0, all caches in private segments are flushed.

`nbc_limit`

Specifies the total maximum amount of memory that can be used for the Network Buffer Cache. This attribute is in KB. The default value is derived from thewall. When the cache grows to this limit, the least-used cache objects are flushed out of cache to make room for the new ones.

`tcp_mssdflt`

Default maximum segment size used to communicate with remote networks. This tunable is used to set the maximum packet size for communication with remote networks; however, only one value can be set even if there are several adapters with different MTU sizes. It is the same as the MTU for communication across a local network except for one small difference: the tcp_mssdflt size is for the size of the data only, in a packet. You need to reduce the tcp_mssdflt for the size of any headers so that you send full packets instead of a full packet and a fragment. The way to calculate this is as follows:

MTU of interface - TCP header size - IP header size - rfc1323 header size

which is:

MTU - 20 - 20 - 12 , or MTU - 52

Limiting data to MTU - 52 bytes ensures that, where possible, only full packets will be sent.

If set higher than the MTU of the adapter, IP or an intermediate router may fragment packets. Changes take effect immediately for new connections and remain in effect until the next reboot.

`subnetsarelocal`

Specifies that all subnets that match the subnet mask are to be considered local for purposes of using MTU instead of the maximum segment size (MSS). This is a configuration decision with performance consequences. If the subnets do have the same MTU, and subnetsarelocal is 0, TCP sessions may use an unnecessarily small MSS. Changes take effect immediately for new connections and remain in effect until the next reboot.

`MTU`

Limits the size of packets that are transmitted on the network, in bytes. Default value is adapter dependent. The range of values is 512 bytes to 65536 bytes.

To obtain the current setting, use the `lsattr` command. For example,

```
lsattr -E -l interface tr0
```

To change the value, use the `chdev` command. For instance,

```
chdev -l interface -a mtu=<newvalue>
```

Because all the systems on the LAN must have the same MTU, they must change simultaneously. Change is effective across boots.

3.2 Tuning parameter summary

Table 1 shows you a summary of tunable parameters for the `no` command. In Table 2 on page 20, we show you the parameter settings for Web site tuning. IBM has tested Web server performance with SPECweb96. We've summarized the information, which you can get from the following Web site:

`http://www.spec.org.`

Table 1. The tunable parameters of the no command

Parameter	Default Value	Range
thewall	1/2 of RAM or 1048576 (1 GB)	
sb_max	65536	
subnetsarelocal	1 (yes)	0 or 1
somaxconn	1024	
tcp_sendspace	16384	0 to 64 KB (if rfc1323=0) 0 to 4 GB (if rfc1323=1)
tcp_recvspace	16384	0 to 64 KB (if rfc1323=0) 0 to 4 GB (if rfc1323=1)
udp_sendspace	9216	0 to 65536
udp_recvspace	41600	
rfc1323	0	0 or 1
tcp_timewait	1	
nbc_max_cache	131072	
nbc_seg_limit	0	
nbc_pseg	0	

Parameter	Default Value	Range
nbc_limit	786432	
tcp_mssdflt	512	512 to unlimited

Table 2. The results of SPECweb96 and tunable parameters

	S7A IHS 1.3.4	7026 - H70 IHS 1.3.4	S80 IHS 1.3.6	43P-260 IHS 1.3.6
SPECweb96	20200	11774	40161	4597
Processor	262 MHz RS64-2	PowerPC RS64-II 340MHz	450 MHz RS64-III	200 MHz POWER3
# of Processors	12	8	12	2
Memory	8 GB	4 GB	16 GB	4 GB
OS	AIX 4.3.2 + APAR IX86737	AIX 4.3.2 + APAR IX86737	AIX 4.3.3	AIX 4.3.3
MTU	ATM MTU=9180	ATM MTU=9180 & Jumbo Frame Gigabit MTU=9000	Gigabit Jumbo Frame MTU=9000	Jumbo Frame Gigabit MTU=9000
no				
sb_max	262144	262144	262144	262144
somaxconn	8192	8192	16384	16384
nbc_max_cache	100000	100000	60000	100000
nbc_seg_limit	10000	10000		
tcp_sendspace	28000	28000	28000	28000
tcp_recvspace	28000	28000	28000	28000
tcp_timewait	5	5	5	5
nbc_pseg			80000	20000
nbc_limit			393216	

Chapter 4. Web Server: IBM HTTP Server 1.3.6

As of January 2000, IBM HTTP Server for AIX on an RS/6000 S80 holds the world record for the performance benchmark result by SPECweb96. For detailed information, see `http://www.spec.org/`

This chapter focuses on Web Server performance tuning. We concentrate on the Web server produced by IBM, the IBM HTTP Server V1.3.6 (IHS), which is based on the popular open standards-based Apache Web server.

While the settings will be different for each Web server and application the methodology used to performance tune IHS still holds true for your Web server application combination.

Most of these settings will need to be tested with the individual applications that will be run on the server. This tuning guide provides information which you can use as a starting point for your performance tuning tests. You probably save your time with this publication.

In the following sections we cover some of the performance tuning directives found in the httpd.conf file. You can use a text editor to modify the parameters in the httpd.conf file in the directory /usr/HTTPServer/conf on AIX.

There are several directives that we can specify in the httpd.conf file. We have categorized them in five different areas:

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

You may also refer to `/usr/HTTPServer/htdocs/<LANG>/manual/misc/perf-tuning.html` by Dean Gaudet. This is a very good reference for the Web site that mainly serves HTML contents. In other words, some information in these performance notes is not appropriate for WebSphere.

4.1 Process handling

This category is primarily related to the httpd processes.

MaxClients - Limit on the total number of simultaneous HTTP requests that IHS can serve. Since IHS uses one child server process for each HTTP request, this is the limit of the number of child server processes that are able to run simultaneously. The default value is 150 and maximum value is 2048.

The value of this directive can significantly impact your application performance, particularly if it is too high. The optimum value depends on your application. In general:

- Use a lower MaxClients value for a simple, CPU intensive application.

- Use a higher MaxClients value for a more complex, database intensive application with longer wait times.

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

A good approach to tuning this parameter is to start with a setting of 50 and capture the performance under normal load. Repeat your test with settings of 40 and 60. Use this data to refine your tuning. Use small increments and decrements rather than large ones.

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

StartServers - The number of child server processes that are created when IHS is started. Default value is 5.

MaxSpareServers - Specifies the upper number of idle httpd child processes which are not handling any requests.

MinSpareServers - Specifies the lower number of idle httpd child processes which are not handling any requests.

If there are fewer than MinSpareServers, then the parent process creates new child processes at a maximum rate of 1 per second. If there are more than MaxSpareServers, then the parent process kills off the excess child processes. Default value of MaxSpareServers is 10 and MinSpareServers is 5.

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

MaxRequestsPerChild - restricts the number of requests handled by each child httpd process. Once this value is reached, the child process terminates. One of the intentions of this parameter is to limit the lifetime of an httpd client process in order to prevent it from using too much memory resource in case of memory leaks. The number specified can be fairly high if stable operation is expected. Default value is 10000.

ListenBacklog - The maximum length of the queue of pending connections from the clients. Generally no tuning is needed or desired, however on some systems it is desirable to increase this when under a TCP SYN flood attack. See the backlog parameter to the listen(2) system call. Default value is 511.

4.2 Connection

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

gets an HTTP/1.1 request, IHS can re-use the connection until it receives the connection close request.

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

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

MaxKeepAliveRequests - The maximum number of requests to allow during a persistent connection. Set to 0 to allow an unlimited number.

Note

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

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

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

Default value is 300.

4.3 Resource usage

These directives restrict the amount of system resource usage by the httpd child process.

Note

The directives explained in this section are not included in the default configuration file since they are normally not used, because they can be specified at the operation system level, if required. These directives should only be used when the values need to be set lower than what the operating system permits.

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

RLimitMEM - Sets the number of bytes per process. This directive also takes one or two parameters. The first parameter sets the soft resource limit for all processes and can be set to a number or “max”. The “max” indicates to the server

that the limit should be set to the maximum resource limit allowed by the operating system.

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

```
#
RLimitCPU 5 max

RLimitMEM max

RLimitNPROC max max
```

Figure 14. RLimit directives in the httpd.conf

SendBufferSize - Specifies the TCP buffer size to a specific number of bytes.

4.4 Name resolution

This category includes the directives that affect the httpd processes specifically in the client-parsing phase in the runtime environment.

HostnameLookups - Enables or disables DNS lookups to be performed such that host names (rather than IP addresses) can be logged. To increase performance in any case, you should set the HostnameLookups directive to off. The default is off.

Note

When using allow from domain or deny from domain directives there is a double, reverse DNS lookup. For best performance avoid using these directives.

4.5 Fast Response Cache Accelerator

Fast Response Cache Accelerator (FRCA) provides a kernel level cache to store server static HTML documents and images. There are several limitations when using FRCA. FRCA can cache only static contents. In other words, dynamic contents which are generated by servlets, JSPs and EJBs are not cached by FRCA. FRCA does not support protected pages, POST method, or any pages over SSL connections. But many Web sites still use static pages. Therefore, FRCA can provide performance improvements for these sites.

The following are the basic steps for setting up FRCA for AIX.

FRCA is new function of IBM HTTP Server 1.3.6 for AIX. IHS 1.3.3 for AIX does not support it. For AIX 4.3.2, you have to apply APAR IX86737. If you are using AIX 4.3.3, no APAR is required. For FRCA support you install the http_server.frca fileset. Then, configure AIX for FRCA. During the AIX FRCA configuration, the Network Buffer Cache options will be requested, which can be obtained by the `no` command.

Network Buffer Cache options:

- nbc_limit: sets the maximum size of the network buffer cache in KB. This value should not be set higher than 1/2 the value of `thewall` which you can get with `no -a` command.
- nb_max_cache: sets the maximum size of a cache object that will be allowed in the Network Buffer Cache.
- nbc_min_cache: sets the minimum size of a cache object that will be allowed in the Network Buffer Cache.

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

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

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

Note

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

There are several directives that are related to FRCA:

- AfpaBindLogger [-1, 0, 1, ..., n]: allows you to bind the FRCA logging thread to a specific CPU on a multiple processor machine.
- AfpaCache on | off: allows you to turn FRCA on or off for a particular scope such as a directory.
- AfpaEnable: enables the FRCA to listen on the TCP port specified by the Port directive or the default port 80.
- AfpaLogFile file_path_and _name log_format: sets the FRCA log file name, location and format
  - Log formats:
    - CLF: Command Log Format
    - ECLF: Extended Common Log Format
    - V-CLF: Common Log Format with virtual host information
    - V-ECLF: Extended Common Log Format with virtual host information
    - BINARY: Binary Log Format with virtual host information

- AfpaSendServerHeader true | false: specifies whether or not FRCA will send the HTTP Server header in the response.
- AfpaLogging on | off: turns FRCA logging on or off.
- AfpaMaxCache [size]: specifies the maximum file size in bytes which can be added to the FRCA cache.
- AfpaMiCache [size]: specifies the minimum file size in bytes which can be added to the FRCA cache.
- AfpaRevalidationTimeout [seconds]: sets the time interval for files cached to be revalidated.

In our test environment, we specified the AFPA configuration directives in the /usr/HTTPServer/conf/httpd.conf file. See below in Figure 15.

```
#
LoadModule ibm_afpa_module

AddModule mod_ibm_afpa.c

AfpaEnable
AfpaCache on
AfpaLogFile /usr/HTTPServer/logs/afpa-log V-ECLF
AfpaMinCache 0
AfpaMaxCache 100000
AfpaLogging on
AfpaBindLogger -1
AfpaSendServerHeader true
```

Figure 15. AFPA configuration in the httpd.conf file

Then, start IHS with the `/usr/HTTPServer/bin/apachectl start` command.

There are three ways to monitor FRCA:

- Fast Response Cache Accelerator Log: the FRCA log can be used to observe files being served out of the cache.
- Using netstat to monitor the Network Buffer Cache: using the command `netstat -c` you can observe the current status of the Network Buffer Cache.
- Using `frcactrl` to monitor the FRCA kernel: using the command `frcactrl stats` you can observe statistics from the FRCA kernel such as the total number of requests handled and the total number of successful cache hits.

Now, you can set up FRCA successfully.

We tested FRCA with two scenarios. One is for HTML static documents and the other is for a dynamic document that is created by a servlet.

For the HTML static document test case, we accessed the welcome page, which includes several gif files. We saw significant performance improvements with FRCA. In our environment, using FRCA allowed twice as many HTTP requests than without FRCA. In addition the CPU utilization of the Web server machine was lower than the non-FRCA case. With the `frcactrl stats` command, we noticed that our HTTP requests hit cached data.

The other test case for dynamic content showed that using FRCA did not improve performance. FRCA is designed for static content only as we described before. From this test result, we can tell that using FRCA does not affect for dynamic contents. In other words, FRCA does not cause any performance decreases for dynamic content. Of course, there is no cache hit with FRCA for dynamic content.

4.6 APAR for Web server performance

There are several APARs that are related to Web server performance. The following AIX APARs are related to all AIX Web servers, so they should always be installed:

- IX88664:

  Abstract: server performance is suboptimal

- xIX86737:

  Abstract: Web server performance improvements

- xIY02324:

  Abstract: wrong http response on if-unmodified-since header

- xIY00957:

  Abstract: get engine can crash the system

The `instfix -ik` command tells you if you've applied the APAR. See Figure 16.

```
# instfix -ik IX88664
    All filesets for IX88664 were found.
# instfix -ik IX86737
    Not all filesets for IX86737 were found.
# instfix -ik IY00957
    All filesets for IY00957 were found.
```

Figure 16. instfix -ik command

The above output tells us that we have not installed all filesets for IX86737.

Now, we show you how we can find the filesets which we need to install for IX86737.

We also used the `instfix -icvk` command to determine the status of IX86737 on our system. See Figure 17 on page 28.

```
# instfix -icvk IX86737
#Keyword:Fileset:ReqLevel:InstLevel:Status:Abstract
IX86737:bos.64bit:4.3.2.6:0.0.0.0:!:web server performance improvements
IX86737:bos.adt.include:4.3.2.6:0.0.0.0:!:web server performance improvements
IX86737:bos.adt.prof:4.3.2.6:0.0.0.0:!:web server performance improvements
IX86737:bos.adt.syscalls:4.3.2.2:0.0.0.0:!:web server performance improvements
IX86737:bos.atm.atmle:4.3.2.6:0.0.0.0:!:web server performance improvements
IX86737:bos.mp:4.3.2.7:0.0.0.0:!:web server performance improvements
IX86737:bos.net.ipsec.rte:4.3.2.3:4.3.2.0:-:web server performance improvements
IX86737:bos.net.nfs.client:4.3.2.6:4.3.2.0:-:web server performance
improvements
IX86737:bos.net.tcp.client:4.3.2.6:4.3.2.10:+:web server performance
improvements
IX86737:bos.net.tcp.server:4.3.2.6:4.3.2.9:+:web server performance
improvements
IX86737:bos.rte.libc:4.3.2.6:4.3.2.11:+:web server performance improvements
IX86737:bos.rte.tty:4.3.2.6:4.3.2.8:+:web server performance improvements
IX86737:bos.sysmgt.serv_aid:4.3.2.4:4.3.2.6:+:web server performance
improvements
IX86737:bos.up:4.3.2.7:4.3.2.11:+:web server performance improvements
IX86737:devices.common.IBM.atm.rte:4.3.2.3:0.0.0.0:!:web server performance
improvements
IX86737:devices.common.IBM.ethernet.rte:4.3.2.2:4.3.2.3:+:web server
performance improvements
IX86737:devices.pci.14100401.rte:4.3.2.3:0.0.0.0:!:web server performance
improvements
IX86737:devices.pci.14107c00.com:4.3.2.5:0.0.0.0:!:web server performance
improvements
```

Figure 17. instfix -icvk command

There are 18 filesets for IX86737. The Status field tells us if we've installed the fileset or not. If it says "+", it means that we've already applied the correct level of fileset. If it shows "-", it indicates that we need to apply the correct level of fileset which the ReqLevel field shows you.

For example, we've installed bos.net.nfs.client.4.3.2.0, but IX86737 also requires that we apply 4.3.2.6 level of bos.net.nfs.client.

Chapter 5. WebSphere Engine

Version 3.0 of the WebSphere Application Server introduced a new runtime and systems management infrastructure. The new Systems Management infrastructure introduced a new “look and feel” that is significantly different from the administration tools in prior versions of WebSphere. Primary among these changes is the storing of the Application Server configuration in 34 entity EJBs in a UDB or Oracle database (InstantDB is also available for Standard Edition). This means that changes to the system configuration need to be made through the Administrative Console (“adminclient”) pictured below, not by changing settings in properties files as was the case in previous versions. While there are still a number of properties files in the WebSphere directory structure, these are intended primarily for debugging and should not be used for making the configuration changes described below.

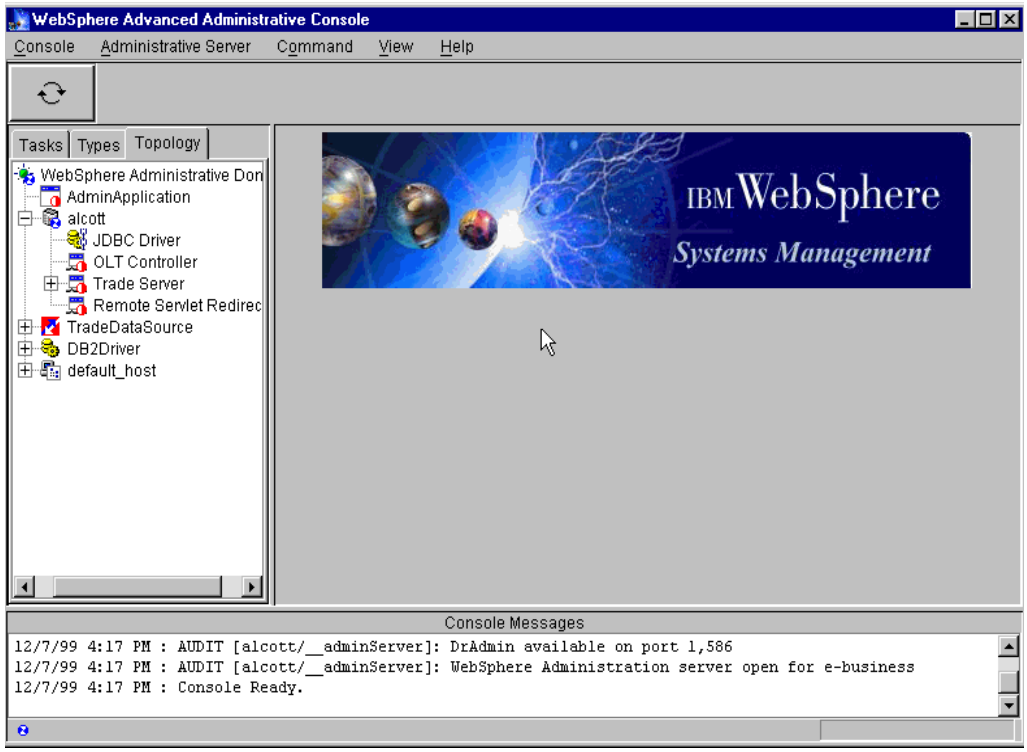


Figure 18. Administrative Console

For an application server in WebSphere V3 there are a number of settings that have an impact on performance. In addition to the settings for the application server itself there are also a number of settings for the various components of an application server: Servlet Engine, Web Application, and the EJB Container that can affect performance.

The WebSphere Engine has several parameters that will influence the overall performance of your Web site. The following parameters are set using the WebSphere Application Server Administrative Console:

- JVM
- Transport Queue
- Servlets Auto Reload

- EJB Container
- ORB

The parameters listed above will be discussed in detail in the following sections.

5.1 JVM

WebSphere requires a JVM to run. Though WebSphere ships with a default JVM, on some platforms alternate JVMs can be used. In general, upgrading to the latest 1.1.n level will provide better performance as JVM technology is continually improving.

5.1.1 Selecting a JVM

On AIX, two JDKs are currently supported: JDK 1.1.6 PTF 9 and JDK 1.1.8 PTF 4. All tests for this redbook were run using JDK 1.1.6 PTF 9.

JDK for Windows NT

As of late December 1999, the only supported JVM for Windows NT is the IBM Developer Kit and Runtime Environment for Windows, Java Technology Edition, Version 1.1.7p that ships with WebSphere V3. It has incorporated via "backporting" some of the features from JDK 1.2 such as JDBC as well as the Java ORB from JDK 1.3.

JDK for Solaris

On Solaris, a Solaris-tuned JVM (currently Solaris JDK 1.1.7_08) available from SunSoft will typically outperform the base Solaris reference implementation from JavaSoft. WebSphere V3 ships the JavaSoft reference implementation. WebSphere for Solaris ships with the JavaSoft JDK from JavaSoft. You can also go to the Sun Web site and download the SunSoft JDK for Solaris. The SunSoft JDK typically performs better than the JavaSoft JDK.

5.1.2 Tuning the JVM

The JVM offers several tuning parameters that will impact the performance of WebSphere (which is primarily a Java application), as well as your own applications. In WebSphere V3, JVM parameters are set via the command line arguments of the Application Server on the Administrative Console. The Command Line Arguments are located on the General tab for each application server (Trade Server in Figure 19 on page 31). Highlight the application server for your node in the WebSphere Administrative Domain to get to this tab.

Click **Topology** tab **-> <node> -> <application server>** as shown in Figure 19 on page 31.

There are five topics for JVM performance tuning:

- Heap Size
- JIT
- Garbage Collection

- Java Stack Size
- Native Thread Stack Size

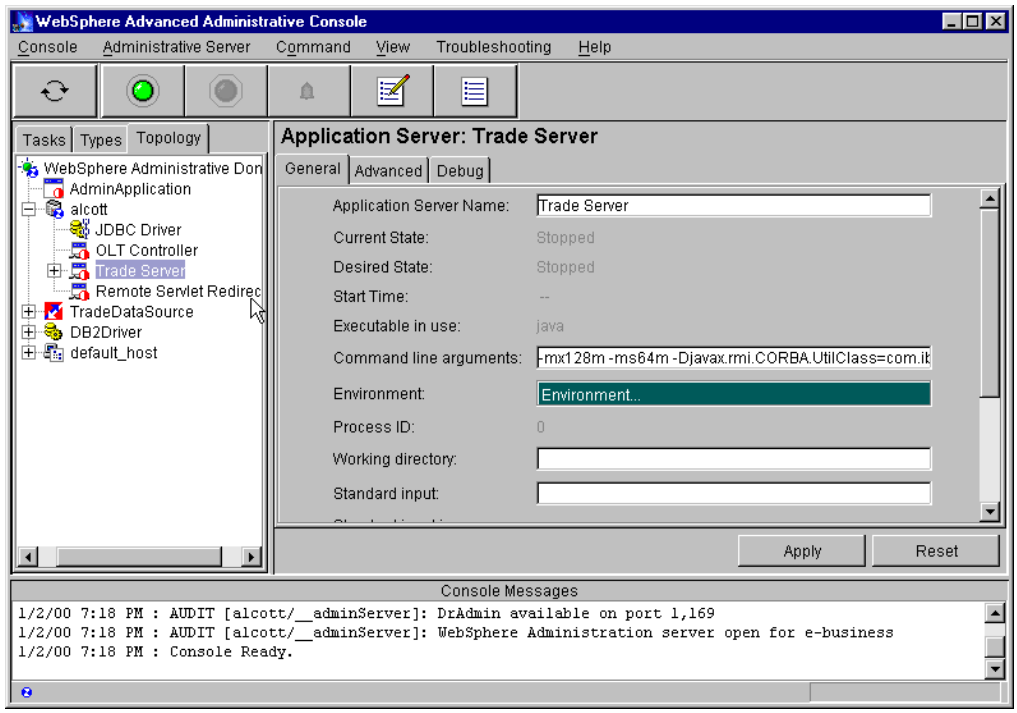


Figure 19. JVM Parameters settings

5.1.3 JVM heap size

Java mx and ms are used to set:

- The maximum heap for the JVM
- The starting (minimum) heap for the JVM

When running performance tests, the best performance will typically be realized when ms and mx heap values are equal. Setting mx and ms heap values equal during performance runs provides highly repeatable results by eliminating any heap growth. This may not necessarily be appropriate in a production environment where other system resources may need to utilize the physical memory that would be allocated by setting these two values equal.

As a starting point on AIX (also Windows NT and Solaris with the JavaSoft JDK) consider setting the maximum heap size to 1/4 the total physical memory on the server and setting the minimum to 1/2 of the maximum.

For example, for a server with 512 MB of memory:

-ms64m -mx128m (mx = 1/4 physical memory, ms = 1/2 mx)

Solaris with the SunSoft JDK

Sun recommends that ms be set to somewhere between 1/10 and 1/4 of the mx setting. They do *not* recommend ever setting ms and mx to be the same. This has to do with their garbage collection algorithms and the need for Sun to track the memory as it is allocated. An example JVM setting is as follows:

-ms32m -mx128m (mx = 1/4 physical memory, ms = 1/4 mx)

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

The other issue to consider when increasing the heap size is that while throughput will be improved, pause times will increase. Large heaps can take several seconds to fill up, but garbage collection occurs less frequently. In general, it is probably good practice *not* to set the maximum heap size larger than 256 MB without extensive testing and monitoring (with `-verbosegc`) of your application under load.

The sizing guidelines mentioned above should prove adequate for most applications. You can refine your heap size settings even further by running tests utilizing the `-verbosegc` command line argument. This additional output is logged by stderr for the application server.

Some sample output from `-verbosegc` is shown below. This first entry occurred during server startup without a load applied. The first thing to consider will be increasing the minimum heap size (-ms).

```
<AF[1]: managing allocation failure. need 1040 bytes, action=1
(80688/16777208)>
<GC: not GC'ing classes: 0 verifier running>
<GC(1): freed 258785 objects, 14314168 bytes in 323 ms, 85% free
(14394256/16777208)>
  <GC(1): mark: 102 ms, sweep: 105 ms, compact: 116 ms>
  <GC(1): moved 37493 objects, 2124448 bytes in 116 ms>
<FIN: async finalizer thread waking>
```

Of course you'll need to further examine the output in order to make an estimate of how much to increase the minimum heap size. By looking for at the last occasion where the JVM heap size was increased you'll have a good starting point. In the example below we can see on the last line that the heap size was increased to just over 29 MB by the JVM. As a result we chose to increase the minimum heap size to 32 MB.

```
<AF[22]: managing allocation failure. need 1552 bytes, action=2
(18071624/27222008)>
<FIN: async finalizer thread waking>
<AF[22]: synchronously running 399 finalizers>
<GC: not GC'ing classes: 0 verifier running>
<GC(29): freed 6368 objects, 488752 bytes in 499 ms, 67% free
(18255704/27222008)>
  <GC(29): mark: 450 ms, sweep: 49 ms, compact: 0 ms>
<AF[22]: managing allocation failure. need 1552 bytes, action=3
(18255704/27222008)>
<AF[22]: zeroed 6 of 6 soft refs>
<GC: not GC'ing classes: 0 verifier running>
<GC(30): freed 102 objects, 9568 bytes in 316 ms, 67% free
(18265272/27222008)>
  <GC(30): mark: 272 ms, sweep: 44 ms, compact: 0 ms>
<AF[22]: managing allocation failure. need 1552 bytes, action=3
(18265272/27222008)>
<AF[22]: managing allocation failure. need 1552 bytes, action=4
(18265272/27222008)>
<AF[22]: Heap Expansion due to GC ratio: 0.171187.>
<AF: expanded heap by 2637824 to 29859832 bytes, 70% free>
```

In this example it turned out that increasing the minimum heap size to 32 MB was still not sufficient to prevent garbage collection from occurring during startup, but it did prevent the JVM from growing the heap during testing. We then increased the minimum heap size to 48 MB. This setting did prevent garbage collection from occurring during system startup and also resulted in our best throughput.

In our testing we found the best performance came with settings of -ms192m and -mx192m. Recall that this was on a two-way AIX server with 512 MB of RAM.

5.1.4 JIT

By default on JDK 1.1.6 for AIX and JDK 1.1.7 for Windows NT the Just In Time (JIT) compiler is turned on. This results in significantly better performance and we validated this in our tests. It's unlikely that you would ever want to turn the JIT compiler off for performance reasons, but in some cases it might be necessary to do so for debugging purposes.

As with the other JVM arguments, this is specified on the Application Server command line arguments. The specific arguments vary with each JDK. For AIX and Windows NT the arguments to turn JIT off are:

```
-Djava.compiler=off (AIX)

-nojit (Windows NT)
```

Again, these arguments turn JIT off, so you would normally not specify these except for debugging purposes.

Note

With JDK 1.1.8 for AIX you will need to turn JIT off, in which case it would be appropriate to specify this argument.

5.1.5 Garbage collection

The WebSphere performance lab in Raleigh has seen cases on Solaris using EJBs where turning off asynchronous garbage collection ("`-noasyncgc`") actually improved performance significantly. You can also turn off class garbage collection to enable more class reuse. The parameter is set on the server command line:

`-noasyncgc` (using `-noasyncgc` on Solaris may occasionally improve performance)

`-noclassgc` ("no" will enable more class reuse; default is for classgc to be on)

Specifying `-noasyncgc` had no significant impact on performance nor did specifying `-noclassgc`.

5.1.6 Java stack and native thread stack size

Java stack size and the native thread stack size are set with the `-oss` and `-ss` command line arguments. Specifying a size of 819200 for both resulted in a slight performance improvement (less than 1%).

Setting these values to 409600 did result in performance degradation of approximately 16% from our baseline (819200). Conversely we encountered problems in starting the application server when they were set to too large a value. When we set these to 1638400 we received a `java.lang.OutOfMemoryError: cannot create anymore threads` message. If you wish to specify a value on the command line of your application server, recommended starting points would be:

`-oss=819200`: Java Stack Size

`-ss=819200`: Native Thread Stack Size

Command Line Arguments - Making Changes

When making changes to some of the settings described above such as JVM heap size (-ms and -mx) take care when typing. If you make a mistake you'll likely encounter the error depicted below: `Server start failed` (in Figure 20 the name of the server is "Default Server"). In addition to the messages in red in the console messages section, you'll notice the yellow icons with "?" in them. You'll need to highlight the server you were trying to start (as depicted below), and stop the server (right mouse button --> stop or depress the red button at the top of the console). In some cases it might be necessary to affect a "force stop" of the server (highlight the server --> right mouse button - force stop) which marks the processes as stopped in the repository. Once you've stopped the server review, and correct your command line arguments before starting the server again (don't forget to click the **Apply** button after you make your changes or corrections).

An additional word of caution: for some arguments the order matters. If you change the -oss, -ss, classgc, or Djava.compiler arguments, these need to come immediately after the -ms, -mx arguments and immediately preceding any other command line arguments.

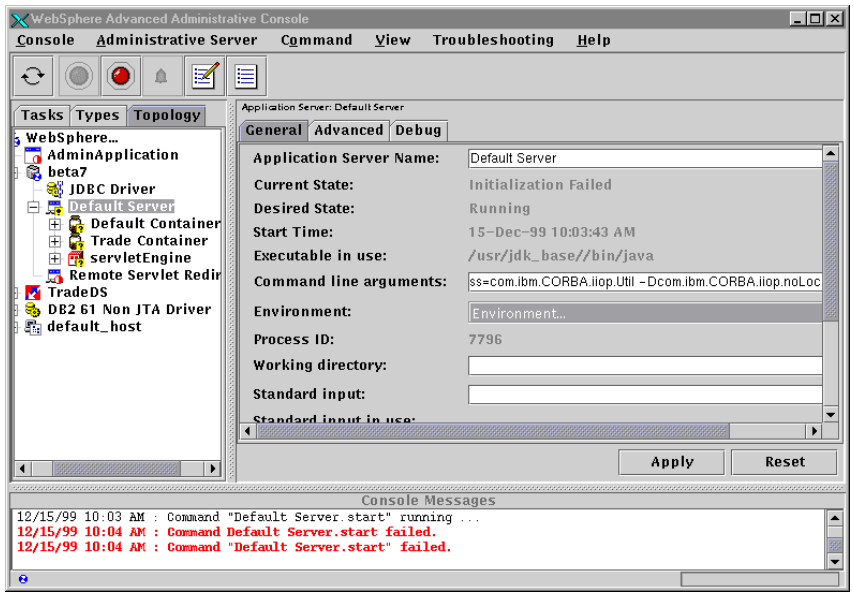


Figure 20. Command line arguments

5.2 Transport queue

Remember that each application server in your WebSphere Application Sever product is comprised of an enterprise bean container and servlet engine. To route servlet requests from the Web server to the servlet engines, the product establishes a transport queue between the Web server plug-in and each servlet engine.

You can adjust the transport queue to improve performance. There are three parameters that you need to consider:

1. Queue type
2. Transport type
3. Maximum connections

as shown in Figure 21.

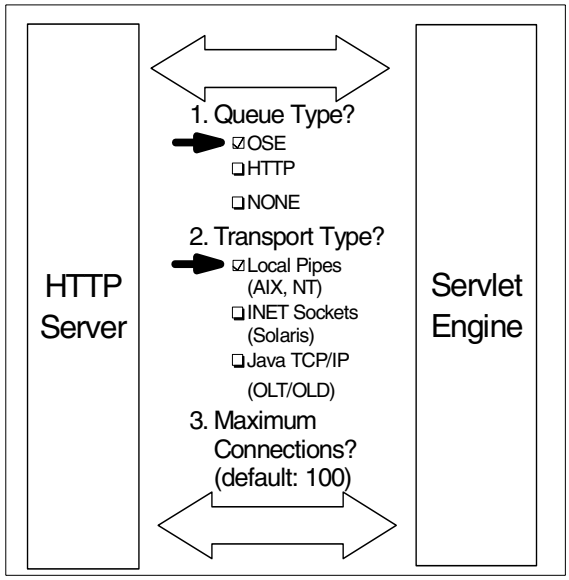


Figure 21. Transport queue tuning

The queue type is of central importance. If distributing servlet requests from the Web server to servlets on remote machines known as “redirecting servlets”, you should use an IIOP-based queue. We will discuss this setting in 5.2.4, “Queue type for servlet redirector” on page 38. If not constrained to use IIOP, you should use Open Servlet Engine (OSE). The second tuning point is the transport type. And the last one is the number of maximum connections.

5.2.1 Queue type: OSE

The OSE will provide the best performance for the transport queue between the Web server plug-in and WebSphere servlet engine. The queue type is configured from the Advanced tab for the Servlet Engine (**Topology tab -> WebSphere Administrative Domain -> node -> Application Server -> Servlet Engine -> Advanced**) as shown in Figure 22 on page 37.

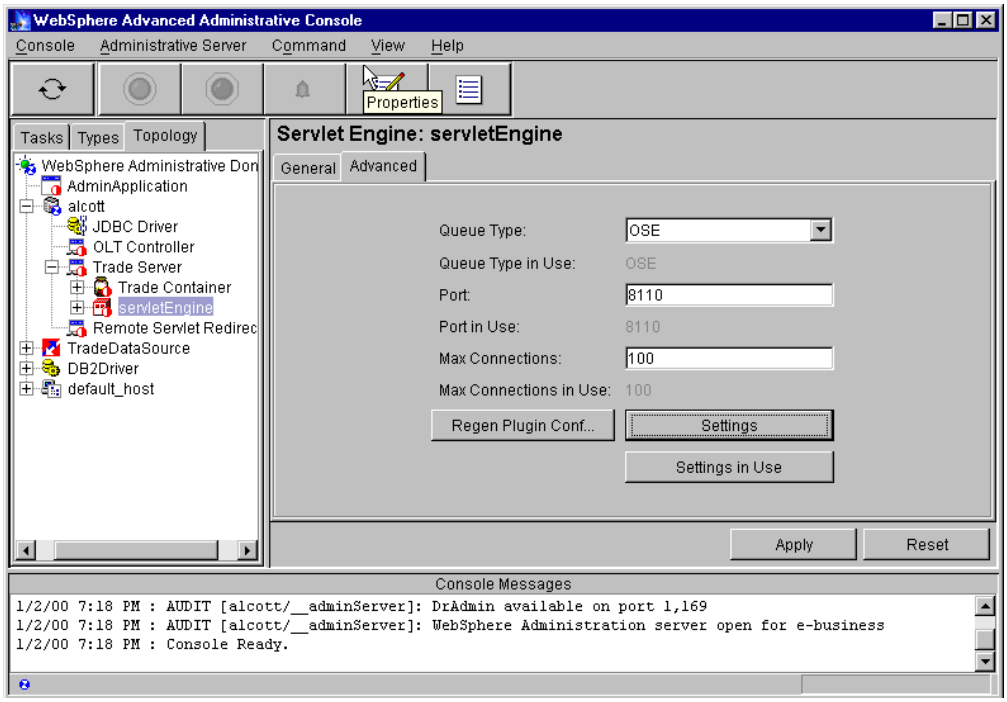


Figure 22. Servlet Engine: Advanced

5.2.2 Transport type

The Servlet Engine Transport Type is edited by selecting **Settings** from the Advanced tab for the servlet engine after choosing OSE. You have three choices as shown in Figure 23.

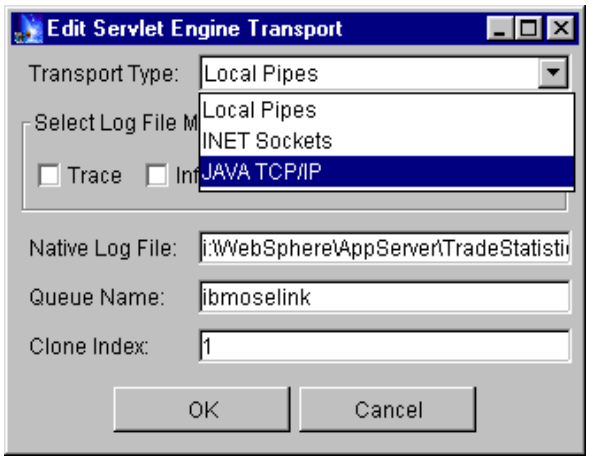


Figure 23. Transport Type

Local Pipes is the default on Windows NT and AIX and typically performs fastest on those platforms.

INET Sockets is the default on Solaris and typically performs better under load than Local Pipes.

JAVA TCP/IP is only provided for debugging in special cases. For the most part you should never specify Java TCP/IP since OLT/OLD support for this transport is lacking in the runtime.

For those readers familiar with WebSphere V2.0 the Servlet Engine Transport Type replaces ose.outofproc.transport.type.

5.2.3 Maximum connections

The Max Connections parameter on the Servlet Engine Advanced tab, as shown in Figure 25 on page 40, specifies the number of connections to use for the communication channel between the Web server and the WebSphere engine. Each connection represents a request for a servlet.

Typically, optimum performance and stability will be achieved by setting the MaxConnections slightly less than or equal to the number of threads or processes that are running within the Web server.

For example, if the IHS AIX MaxClients parameter is set at 50, the MaxConnections parameter should be set to 50 or slightly lower.

More is not necessarily better for this. In our testing with MaxClients set to 50 on IHS, we found that performance did not vary significantly as we varied Max Connections from 30 to 55. Once we increased Max Connections to 60 however we noticed a significant performance decrease.

In testing for your application and hardware infrastructure you'll want to make small changes (5 or less) in this parameter until you determine what's optimal for your environment.

For those familiar with WebSphere V2, the Max Connections property replaces the ose.outofproc.link.cache.size property.

5.2.4 Queue type for servlet redirector

When configuring WebSphere in a cluster that includes use of a servlet redirector it is possible to change the Queue Type for the Servlet Engine from “OSE” to “NONE”. You can continue to use the OSE (default) or HTTP queue type but it's counterproductive. When using the servlet redirector there is no longer a local queue or transport (INET Sockets or Local Pipes). All incoming servlet requests flow over RMI/IIOP from the servlet redirector to the application server(s) where the servlet engine(s) are running.

By specifying "NONE" you eliminate the cost of starting and running a process that is never used. This prevents the application server from unnecessarily loading the transport layer and all of its associated threads, which saves resources on the server machine(s). The result is that you have more resources available to perform "real work" (responding to servlet requests) and performance is better.

Recall that in the configuration where the application server is located on a separate server from the HTTP server and the servlet redirector is used, all communication between machines is via IIOP. Hence there is no need for a queue on the machine that is running the application server. This is illustrated in the diagram in Figure 24 on page 39, which shows one possible means of configuring the servlet redirector.

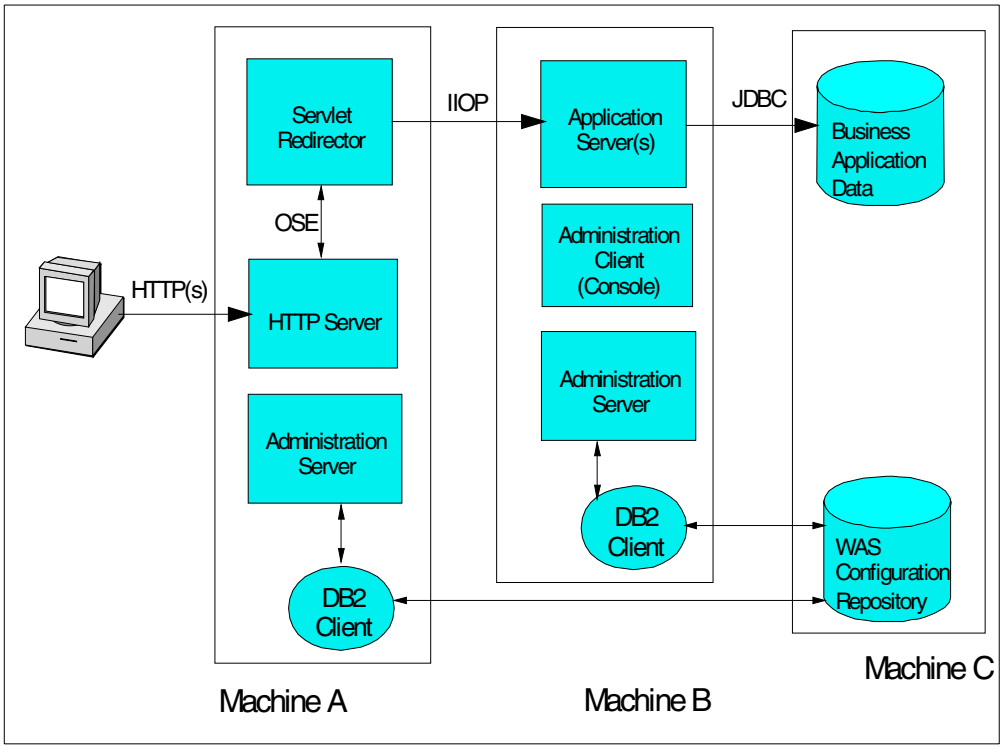


Figure 24. Servlet redirector and queue type

To change the queue type, navigate to the Servlet Engine for the application server via the Topology tab. Highlight the Servlet Engine, then click the **Advanced** tab and change the queue type from OSE to NONE, and then click the **Apply** button as depicted in Figure 25 on page 40.

In our testing we saw performance improvements of up to 14% by changing the queue type from OSE to NONE for the configuration depicted in Figure 24 on page 39.

Note

You cannot specify "none" as the queue type in conjunction with INET Sockets on Solaris; you must specify OSE as the queue type on Solaris.

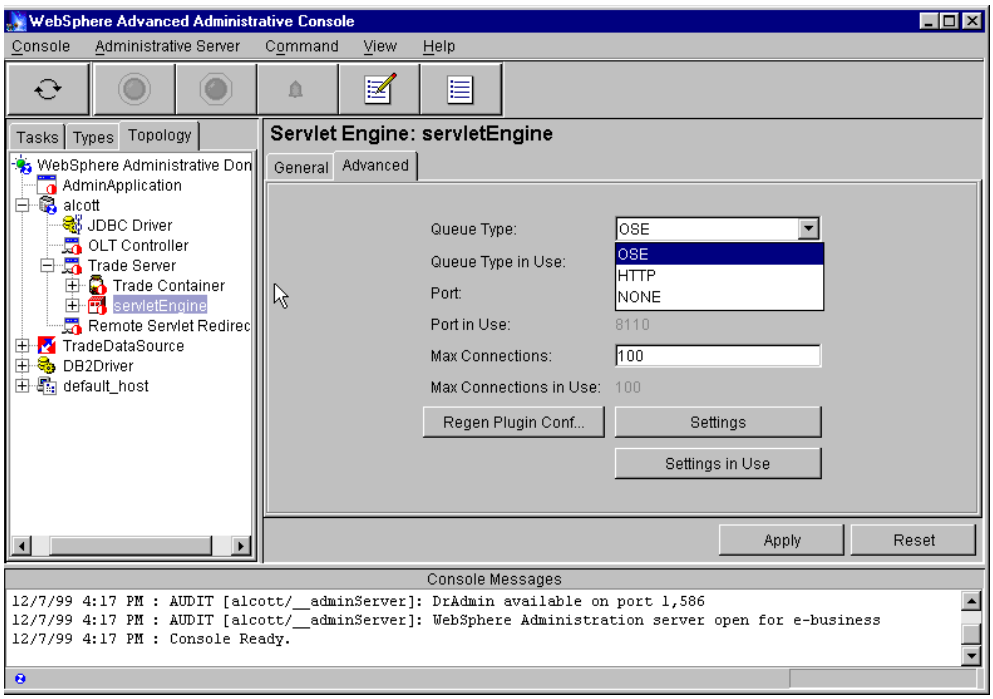


Figure 25. Queue type

5.3 Servlets Auto Reload

With WebSphere V3, you can set parameters specific for each Web application that is deployed. WebSphere has an auto reload capability that specifies whether to automatically reload servlets in the Web application when their class files change. This capability may simplify testing and management of your Web site by enabling you to quickly modify your site without restarting the server; however, this dynamic ability to reload servlets and associated polling will have a negative impact on performance. Once you are in production mode, you should turn off Auto Reload.

This property is accessed by navigating to the web application: **Topology -> WebSphere Administrative Domain -> Node -> Application Server -> servlet Engine -> web application -> Advanced** tab (the values for Node, Application Server, Servlet Engine and web application in the picture are: alcott,Trade Server, serveltEngine and trade_app) and then scrolling down to the bottom of the tab to Auto Reload. Using the pull-down, change the value from True (the default) to False and select Apply and Changing (alcott in the picture) (trade_app in the picture). See Figure 26 on page 41.

There is also a Reload Interval property, as shown in Figure 26 on page 41, that can be used to control the number of seconds between class reloading. An alternative to changing Auto Reload to False would be to change the value for Reload Interval to a very high interval for reload.

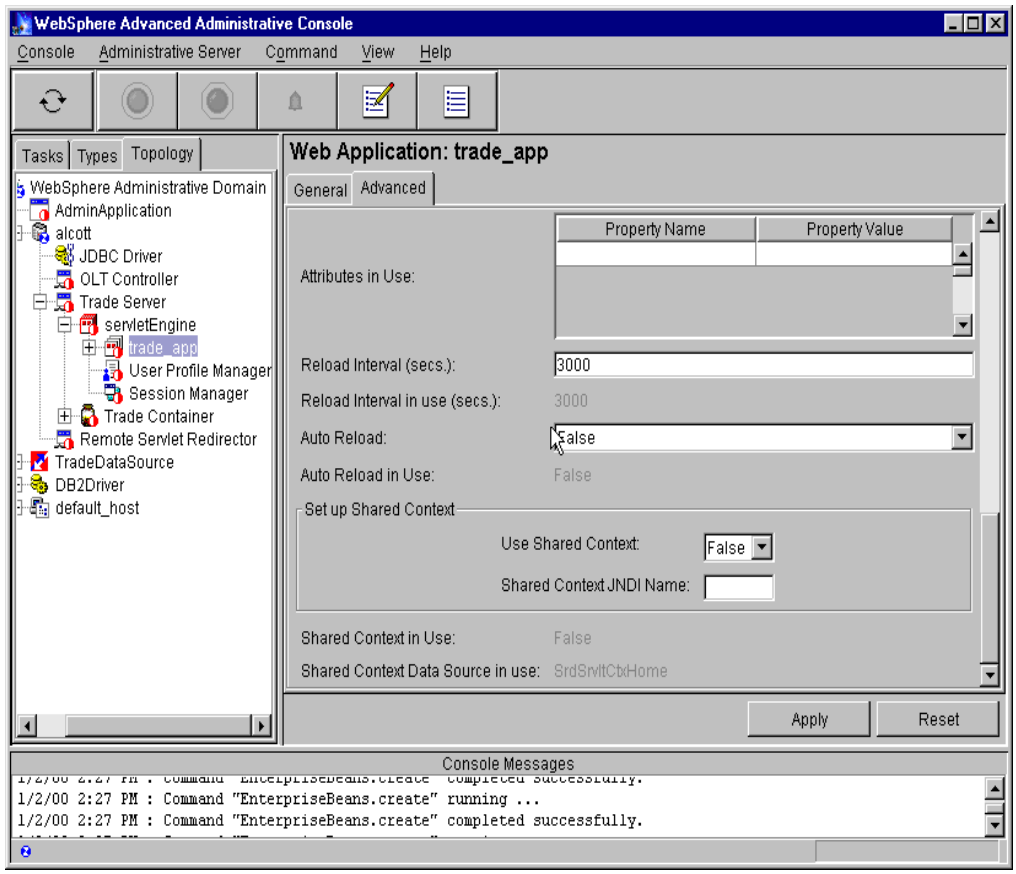


Figure 26. Web Application: Auto Reload

5.4 EJB Container

With the EJB Container in WebSphere V3, you can specify several settings for the container cache that can impact performance. These settings are accessible from the Advanced tab on the EJB container (**Topology tab -> node -> Application Server -> EJB Container**). This is depicted for the Default Container in Figure 27 on page 42. The defaults are shown in the same figure.

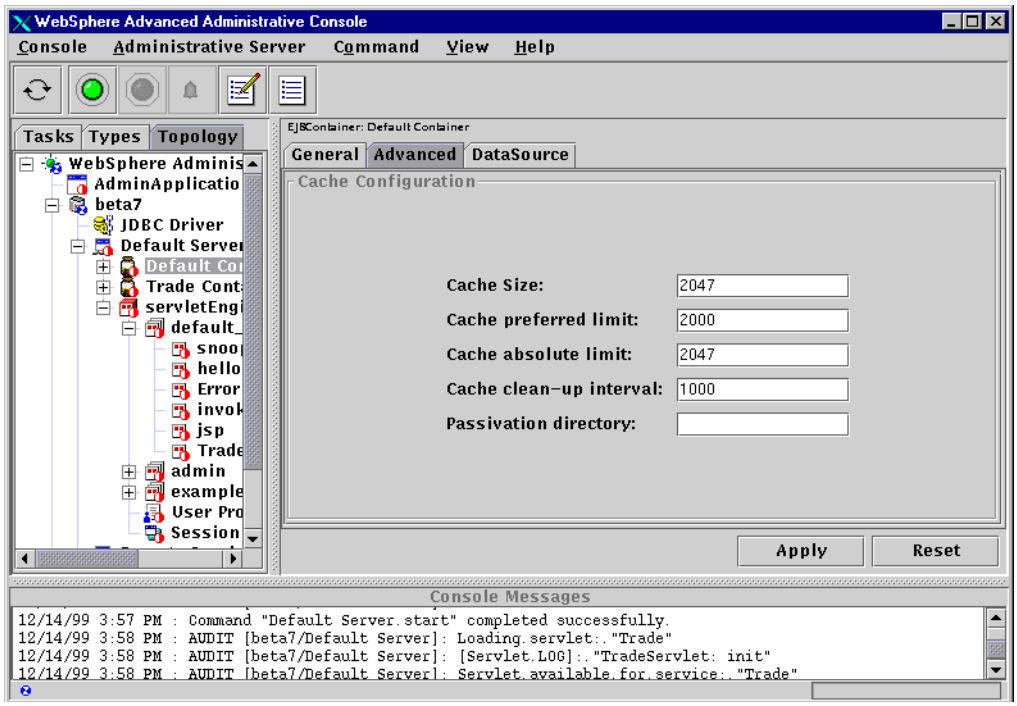


Figure 27. EJB Container: Advanced tab

5.4.1 Cache size

This parameter actually controls the number of buckets in the cache's hash table not the size of the container cache.

Though exposed for modification it is strongly recommended that this not be changed, unless directed to by product support for debugging purposes.

5.4.2 Cache preferred limit

This is a "soft" limit that the cache manager will use as a trigger to start throwing unused entries out of cache. The idea behind this is that the cache manager tries to maintain some unallocated entries that may be quickly allocated as needed. Determining which elements may be freed from the cache is not without cost and there is a background thread that runs which attempts to free enough elements to ensure that some unallocated entries are maintained. If this thread runs while the application server is idle, then when the application server needs to allocate new cache entries it does not pay the performance cost of removing elements from the cache.

Sizing the cache involves estimating the working set size for the concurrent load you expect the application server to be subjected to. In this case, we can define the working set to be the total number of stateful session beans that are concurrently active (involved in a transaction) plus the total number of entity beans that are active. If you have 100 transactions each accessing 20 entity beans then you'll have 2000 entity beans active.

In our testing we never generated a load that required more active beans than were available with the default cache (recall from 2.1.5, "Our test application" on page 11 that the Trade application consists of a stateless session bean and 5 entity beans). To do so would have required more than 400 concurrent clients.

We did, however, decrease the size of the cache several times under load before we noticed a change in performance. In fact performance did not change significantly until we decreased the Cache preferred limit to 20, and Cache absolute limit to 27.

Our recommendation would be to estimate the number of active beans in the manner described earlier (the number of beans * the number of concurrent clients that are accessing beans) to use this as a starting point for your testing. You'll then need to monitor the total number of active beans (you can see the number in the Active Beans column) under load using the WebSphere Resource Analyzer depicted in Figure 28 and adjust the number of beans upward as necessary. Note that increasing the number of beans in cache can increase the JVM in use, so you may need to modify the JVM maximum heap size as a result.

5.4.3 Cache absolute limit

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

5.4.4 Cache cleanup interval

This is the interval in milliseconds for the background thread discussed above that attempts to ensure that there are always free elements in the cache (the difference between the cache absolute limit and the cache preferred limit). In general this parameter should be increased as the cache size increases but in our testing we never generated a load sufficient to require increasing the cache and thus were not able to determine a specific recommendation for changing this.

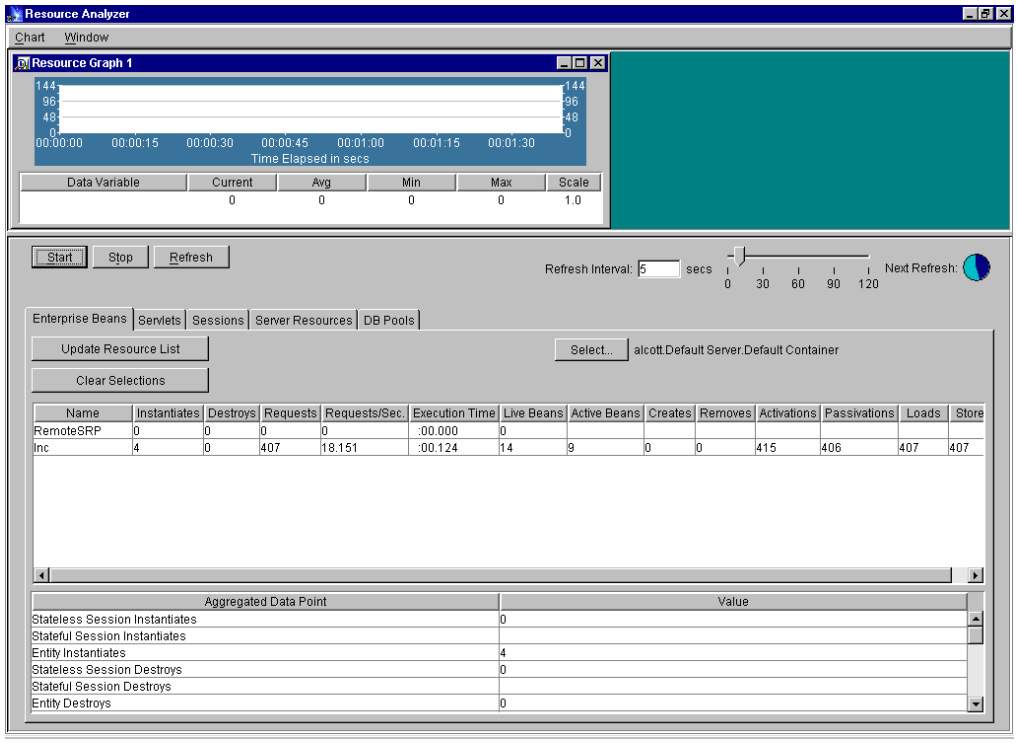


Figure 28. Monitor the total number of active beans

5.4.5 Option A and option C caching performance considerations

WebSphere V3.02 allows you to choose between Option A (“exclusive”) caching and Option C (“shared”) caching as defined by the EJB specification. The default for this property is “shared” and is specified by from the **Database access** property on each enterprise bean. This property is accessed by navigating to **Topology tab -> node -> Application Server -> EJB Container -> EnterpriseBean** as shown in Figure 29.

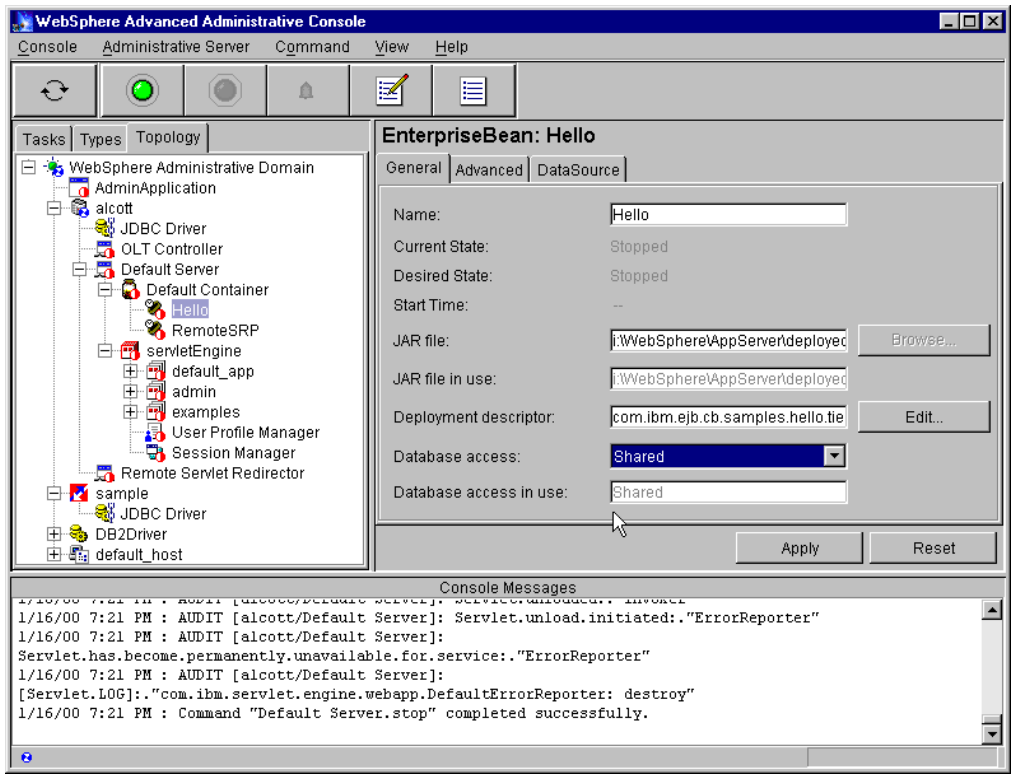


Figure 29. Option A and Option C caching

With option C caching (“shared”), the cache size will typically have very little effect on performance since beans are cached for the duration of the transaction they are involved in. To understand this, consider two sequential transactions that touch bean A. The first transaction loads the bean into the cache, does its work, commits, and removes the bean. The second transaction then has to reload the bean and repeat. As long as there is room in the cache, the performance of this scenario will be the same regardless of whether there are 100 or 10,000 other beans in the cache. The primary reason to change the cache size when using option C caching is to ensure that it can hold the entire working set required by highly concurrent loads.

With option A caching (“exclusive”) there is a potential performance benefit from a larger cache. Namely, that loading the persistent state of the bean from the database will be amortized over multiple transactions. If the cache can hold more beans and the workload has reasonable locality then the chances for a cache hit go up with cache size.

Note

There is a significant restriction for using option A caching: the application server must be the only updater of the data in the persistent store. This means you cannot use option A caching with WLM'ed servers or with a database that is shared among multiple applications that are trying to modify the data.

There is no guarantee that a larger cache will increase application performance. If the application has poor locality then the overhead of the larger cache (increased garbage collection in the JVM, increased process size, cost of cache management) may outweigh any benefits.

5.4.6 Number of containers

You'll typically realize your best performance by deploying all your EJBs in a single container per application server. This minimizes the cost (in JVM heap size) associated with creating a container inside the application server and allows the container to optimize the cache. The net effect is that when you create multiple containers, you've increased the process overhead in the JVM, without any gain in performance.

5.5 ORB

There are several settings controlling internal ORB processing that can be used to improve application performance when EJBs are used.

EJB programming is a remote programming style. Like other remote programming styles such as DCE and CORBA, when you invoke a remote object a local representation (proxy) is created. This local proxy is the local representation of the remote object. All client interaction is through this proxy, the client does not "talk" directly with the remote object.

Normally, Remote Method Invocation (RMI) is Pass By Value (a copy of the calling functions arguments used by the object being called), regardless of in- or out-process.

When both the client and the remote object are in the same JVM, for in-process calls (that is one EJB to another), local copies can be disabled allowing the object being called to get a reference to the calling arguments. In other words, we can get a reference to the calling arguments instead of using “pass by value" and creating a copy/proxy.

This eliminates the process cost of creating the "proxy object". This seems to improve performance by better than 10%.

In order to disable local copies, you need to add the following two arguments to the Command line arguments for your application server as shown in Figure 30:

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

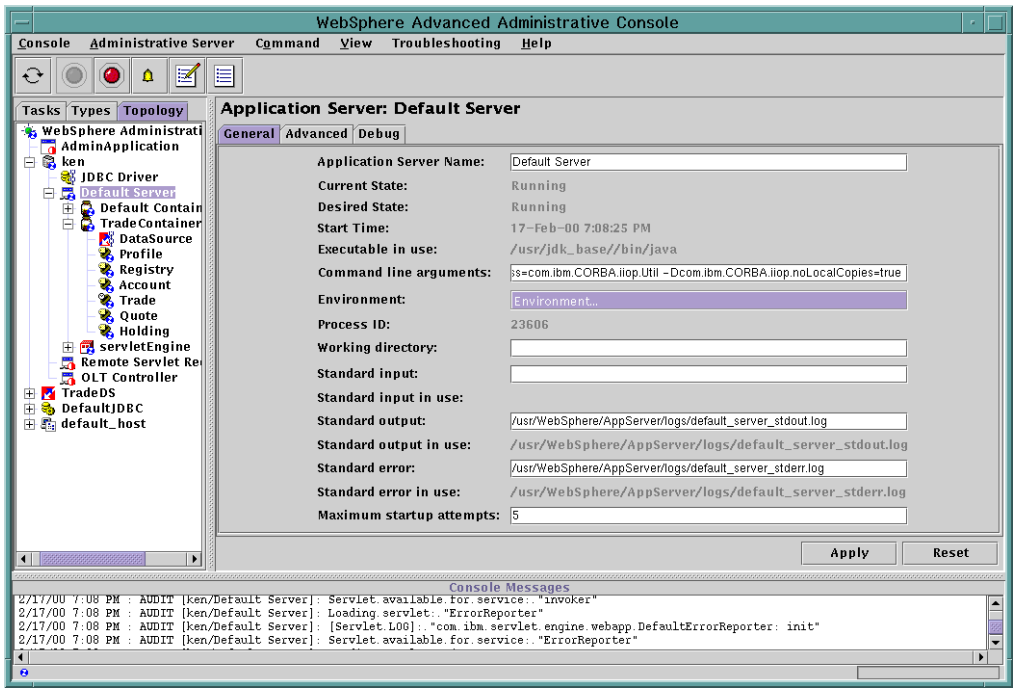


Figure 30. ORB Tuning

Use caution when using "noLocalCopies"

While the use of "noLocalCopies" can improve performance, there can also be unintended side effects. A possible unintended side effect is illustrated with the following example:

```
void execute() {
MyObject myObject = new MyObject();
myObject.setString("hello");
remoteObject.foo(myObject);
System.out.println(myObject.toString());
}

RemoteObject...
void foo(MyObject obj) {
obj.setString("hello world");
}
```

Without "noLocalCopies", the method call to `foo` passes a copy of `MyObject`. When the `foo` method modifies the value of the String field (via `setString`), it has no bearing on the original object. This results in the output of "hello".

With "noLocalCopies", the method call to `foo` passes an object reference of `MyObject`. When the `foo` method modifies the value of the String field (via `setString`), it has an effect on the original object. This results in the output of "hello world".

Tuning for the adminserver

We made the following modifications to the com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs line in the /<was_dir>/bin/admin.config file. We added three entries in addition to -mx128m, which is the default:

```
-mx128m -ms64m -Djavax.rmi.CORBA.UtilClass=com.ibm.CORBA.iiop.Util
-Dcom.ibm.CORBA.iiop.noLocalCopies=true
```

As we discussed before, the first argument increases the minimum heap size for the JVM that the WebSphere adminserver process uses. And also we've discussed what the second one does as well. Since WebSphere is built on an RMI/IIOP and EJB architecture (just like applications), the adminserver starts faster (~25%) and seems to perform better, as does the adminclient. This seems most pronounced on UNIX. There doesn't seem to be as much improvement on Windows NT.

Chapter 6. Security

In V3 of WebSphere Application Server the security infrastructure is designed to support a single integrated policy to govern the security of Web pages, servlets, and enterprise beans. The infrastructure provides for management of the security policies and services provided by both WebSphere Standard and Advanced Editions in a distributed manner utilizing the WebSphere systems console (adminclient).

Among the improvements in security implementation in V3 of WebSphere from V2 are the integration of HTML, JSP file, servlet, and enterprise bean security. V3 of WebSphere also provides for an HTTP Single Sign-On solution, as well as support for modes of delegation between WebSphere servers and secure Java clients. Finally V3 improves on Lightweight Directory Access Protocol (LDAP) support for use as a security registry when compared to V2.

6.1 WebSphere security overview

The WebSphere Application Server runtime relies on Remote Method Invocation/ Internet Inter-ORB Protocol (RMI/IIOP) for all interprocess communication. When WebSphere security is enabled all interprocess communication is encrypted with Secure Sockets Layer (SSL), thus all communications use RMI/IIOP/SSL. As is the case when encrypting HTTP with SSL for HTTPS, there are performance impacts. Once the security is enabled additional processes are invoked for the application.

As an example consider the process flow for the Inc Bean sample that ships with WebSphere depicted in Figure 31 on page 49.

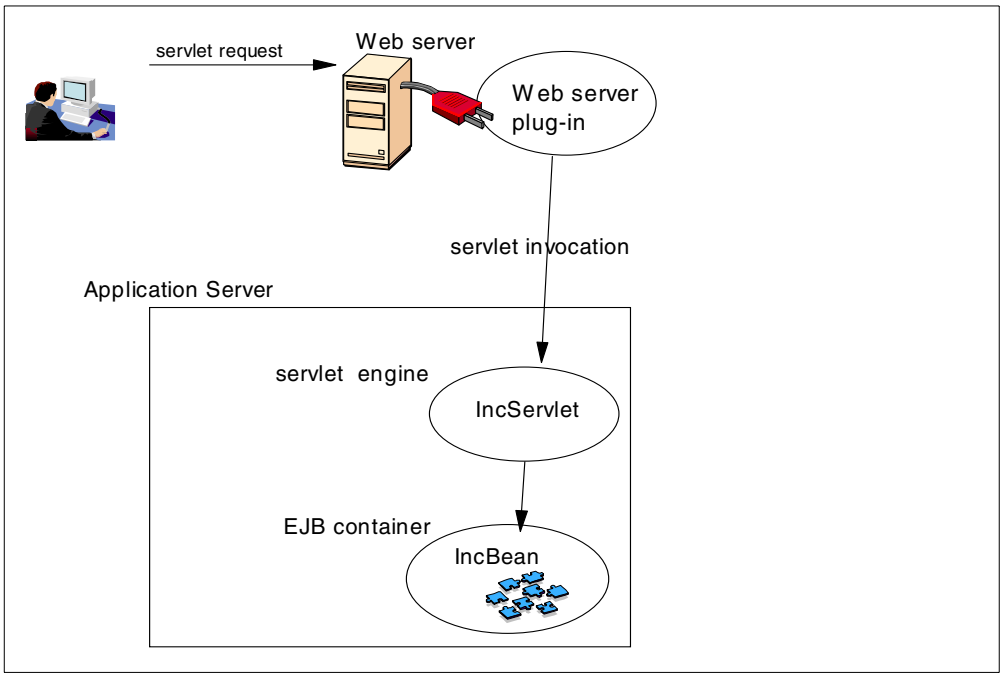


Figure 31. The process flow for the Inc Bean sample

The process flow is:

1. A request comes into the Web server.
2. The WebSphere plug-in to the Web server intercepts the request and determines that the URI is to be serviced by WebSphere.
3. The IncServlet is invoked.
4. The IncBean EJB is invoked.
5. The result is returned to the browser.

Now consider what occurs when security is enabled. This is depicted in Figure 32 on page 50.

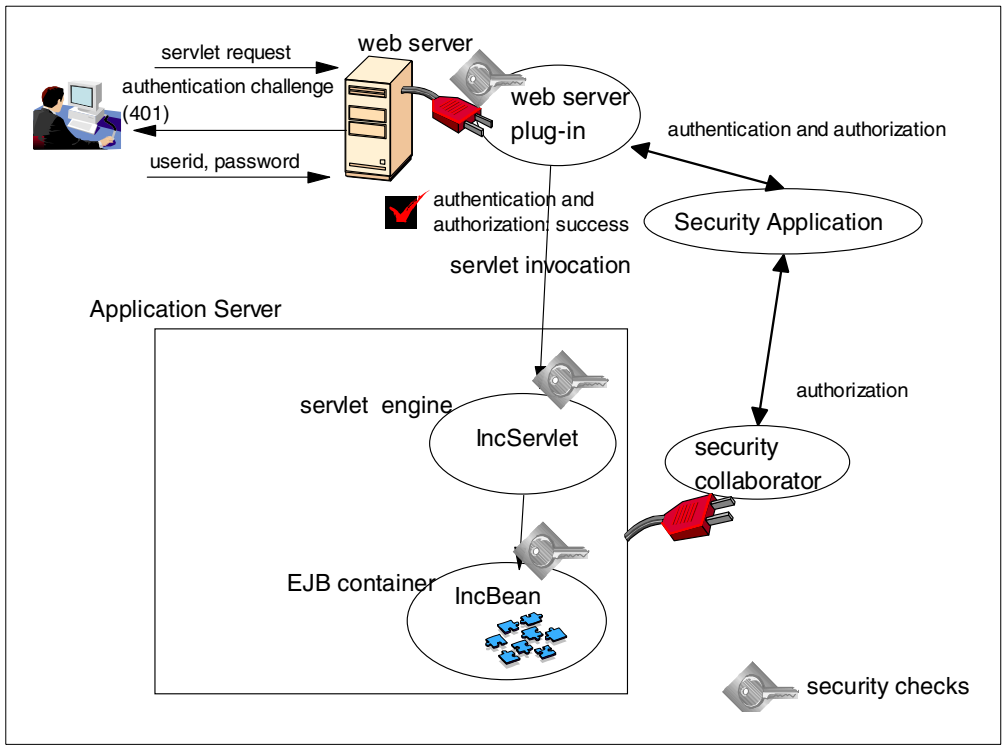


Figure 32. The process flow for the Inc Bean sample with WebSphere security

With security enabled the process flow becomes:

1. A request comes into the Web server.
2. The WebSphere plug-in to the Web server intercepts the request and determines that the URI is to be serviced by WebSphere and is protected.
3. A 401 challenge is issued back to the Web browser requiring the user to enter an ID and password.
4. The plug-in to the Web server contacts the security server with the user ID and password and the security server authenticates the user with this information.
5. Once authenticated, the plug-in to the Web server then determines if the user has the appropriate authorization (permissions) to access the protected resource (the IncServlet). A security context is then created and the request is passed on to the servlet engine to service the request.

6. Before allowing the request to be processed, the security context is checked to ensure that the user is authorized to invoke the method on the servlet.
7. When the IncServlet invokes a method on the IncBean, once again the security context is checked to ensure that the caller has the appropriate permissions to invoke the increment() method in the IncBean.
8. The method is then invoked and the result is returned to the browser from the servlet engine.

As can be seen, the number of steps have nearly doubled when security is enabled, plus the overhead of SSL is added to the additional steps. When security is enabled, performance is degraded by about 50%.

Now that we've explained why there's an impact, the obvious question is what can be done to minimize the performance impact.

6.2 Configuring security

Security in WebSphere V3 is a global setting enabled from the Tasks tab of the WebSphere Administration Console **Tasks -> Security ->Specify Global Settings** as shown in Figure 33.

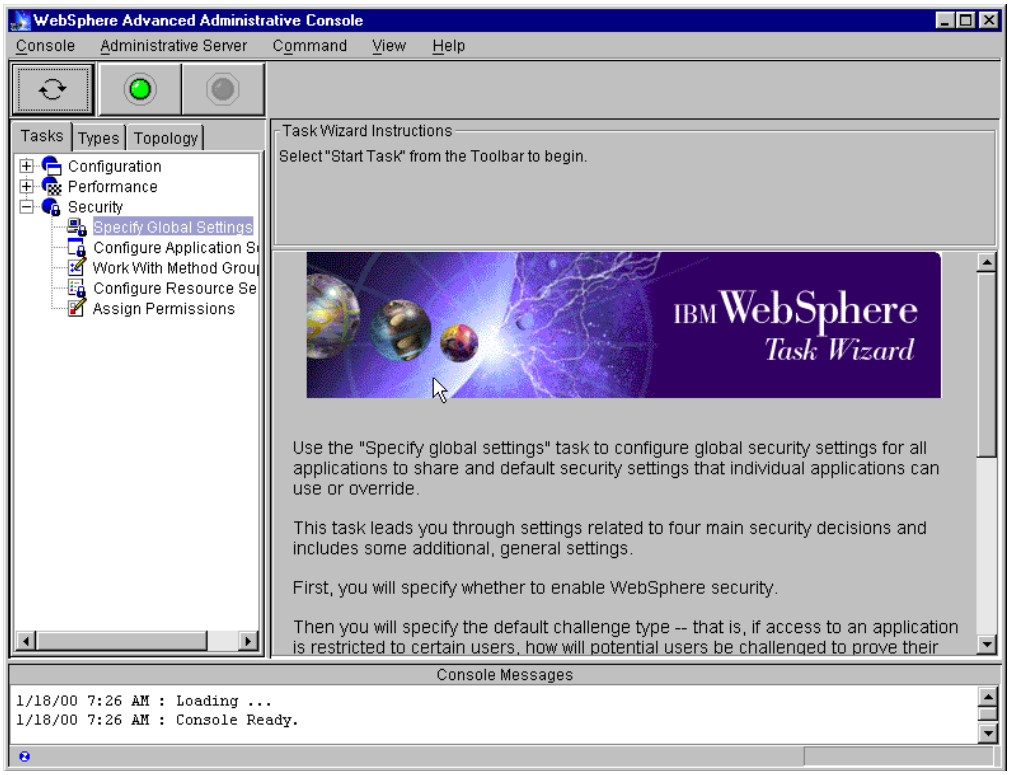


Figure 33. Security global settings

6.2.1 Enabling security

Selecting **Enable Security** as depicted in Figure 34 on page 52 will result in SSL encryption of all interprocess communication in WebSphere. Also, all subsequent invocations of the WebSphere Administrative Console will result in a challenge/authentication dialog prompt once the adminserver has been restarted.

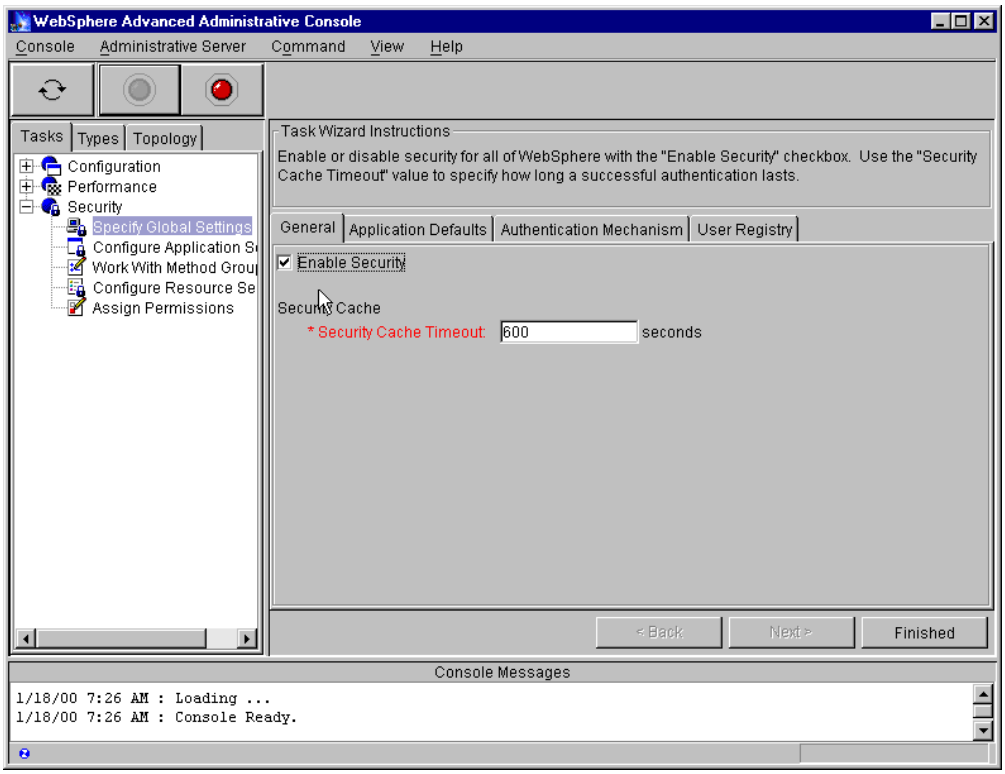


Figure 34. Enable Security

Note

Once security is enabled you'll still need to create an enterprise application to specify security for each object and its methods.

Customer will need to create an enterprise app to specify security for each object and its methods.

6.2.2 Security cache timeout

On the same tab on which you enable WebSphere security as depicted in Figure 34, there is setting for the Security Cache Timeout that can influence performance. This timeout specifies how often the security-related caches need to be refreshed. The information on beans, permissions, and credentials are all cached. Every time the cache is refreshed the values become invalid. Subsequent requests for those values potentially result in a database lookup or even an LDAP-bind or native authentication.

In a simple 20-minute test case, we raised the cache timeout from the 600-second default to 6000 seconds so that it did not time out during the run, and achieved a 40% performance improvement. You can experiment here to find the best trade-off (performance versus how long you want the security cache to remain valid) for your application based on your site usage and your business rules.

6.2.3 SSL V3 timeout

There is one property for the SAS (Secure Association Server) that is performance related. This is the *com.ibm.CORBA.SSLV3SessionTimeout* property value, which is set by default to 9600 secs. This is the time interval after which SSL sessions are renegotiated. This is a high setting, and probably won't realize any significant impact from modification. The SAS parameters are modified by editing the "sas.server.props" and "sas.client.props" files found in the <wasroot>/properties directory as depicted in Figure 35 and Figure 36 on page 54.

Also note that SAS establishes an SSL connection only if it goes out of the ORB (to another ORB), so if all the beans are CO-located within an ORB, then SAS's SSL performance should not be hindered.

```
#SAS Properties - Editable
#Wed Jun 16 17:27:51 EDT 1999
com.ibm.CORBA.loginUserid=demouser
com.ibm.CORBA.loginPassword=demopasswd
com.ibm.CORBA.securityEnabled=false
com.ibm.CORBA.authenticationTarget=LOCALOS
com.ibm.CORBA.delegateCredentials=methodDefined
com.ibm.CORBA.principalName=my.domain.name/demouser
#SAS Properties - DO NOT EDIT
#Wed Jun 16 17:27:51 EDT 1999
com.ibm.CORBA.SSLKeyRing=com.ibm.websphere.DummyKeyring
com.ibm.CORBA.SSLKeyRingPassword=WebAS
com.ibm.CORBA.SSLClientKeyRing=com.ibm.websphere.DummyKeyring
com.ibm.CORBA.SSLClientKeyRingPassword=WebAS
com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
com.ibm.CORBA.SSLTypeIClientAssociationEnabled=true
com.ibm.CORBA.SSLV3SessionTimeout=9600
com.ibm.CORBA.standardClaimQOPModels=integrity
com.ibm.CORBA.standardPerformQOPModels=confidentiality
com.ibm.CORBA.loginTimeout=30
com.ibm.CORBA.loginSource=properties
com.ibm.CORBA.LTPAServerAssociationEnabled=true
com.ibm.CORBA.LTPAClientAssociationEnabled=false
com.ibm.CORBA.keytabFileName=c:/websphere/etc/keytab5
com.ibm.CORBA.securityDebug=no
com.ibm.CORBA.securityTraceLevel=none
com.ibm.CORBA.bootstrapRepositoryLocation=c:/websphere/etc/secbootrep
com.ibm.CORBA.disableSecurityDuringBootstrap=false
LOCALOS.server.id=demouser
LOCALOS.server.pwd=demopasswd
```

Figure 35. sas.server.props

```
com.ibm.CORBA.securityEnabled=true
com.ibm.CORBA.loginSource=prompt
com.ibm.CORBA.loginTimeout=30
com.ibm.CORBA.loginUserid=
com.ibm.CORBA.loginPassword=
com.ibm.CORBA.keytabFileName=/usr/WebSphere/AppServer//etc/keytab5
com.ibm.CORBA.SSLTypeIClientAssociationEnabled=true
com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
com.ibm.CORBA.LTPAClientAssociationEnabled=false
com.ibm.CORBA.LTPAServerAssociationEnabled=true
com.ibm.CORBA.authenticationTarget=LocalOS
com.ibm.CORBA.SSLKeyRing=com.ibm.websphere.DummyKeyring
com.ibm.CORBA.SSLKeyRingPassword=WebAS
com.ibm.CORBA.SSLServerKeyRing=com.ibm.websphere.DummyKeyring
com.ibm.CORBA.SSLServerKeyRingPassword=WebAS
com.ibm.CORBA.SSLV3SessionTimeout=9600
com.ibm.CORBA.standardPerformQOPModels=confidentiality
com.ibm.CORBA.standardClaimQOPModels=integrity
com.ibm.CORBA.bootstrapRepositoryLocation=none
```

Figure 36. sas.client.props

6.3 The invoker servlet

The invoker servlet is a special servlet that is created as part of the sample default configuration. While useful for a development environment, this servlet exacts a very slight performance penalty. More important is the security risk that this servlet represents.

For those of you not familiar with “invoker”, this is a servlet that has a service method only. This service method searches the classpath for other servlets and then invokes (or executes) them. This allows you to place servlet class files in the default directory and execute them without defining them. The easiest way to see this is to use the servlet snoop. You’ll notice that snoop is defined in the default_app and can be called in a browser as `http://localhost/servlet/snoop`. When this happens you’ve called the servlet directly, and invoker is not required. However if you choose to call snoop by its class name, `http://localhost/servlet/SnoopServlet`, then invoker is used to service this request. If you look closely at the Request Information section on the page returned to the browser you’ll see different URIs and paths returned for each of these even though they are the same servlet. You can also go to the WebSphere Resource Analyzer and monitor the number of executions of each servlet as you press Enter on your browser.

While the performance cost of using the invoker servlet to execute your servlets is negligible, of significant importance is the fact that this exposes your Web site to unauthorized execution of servlets. In WebSphere V3 the permissions are defined by URL, so if you alias a servlet /servlet/something, then the permissions aren’t applied to /servlet/com.someone.something. This can we worked around by securing both URLs for each servlet. The other possible issue is that a provider of a support class library could sneak a servlet into their JAR file and use it as a back door into Web sites.

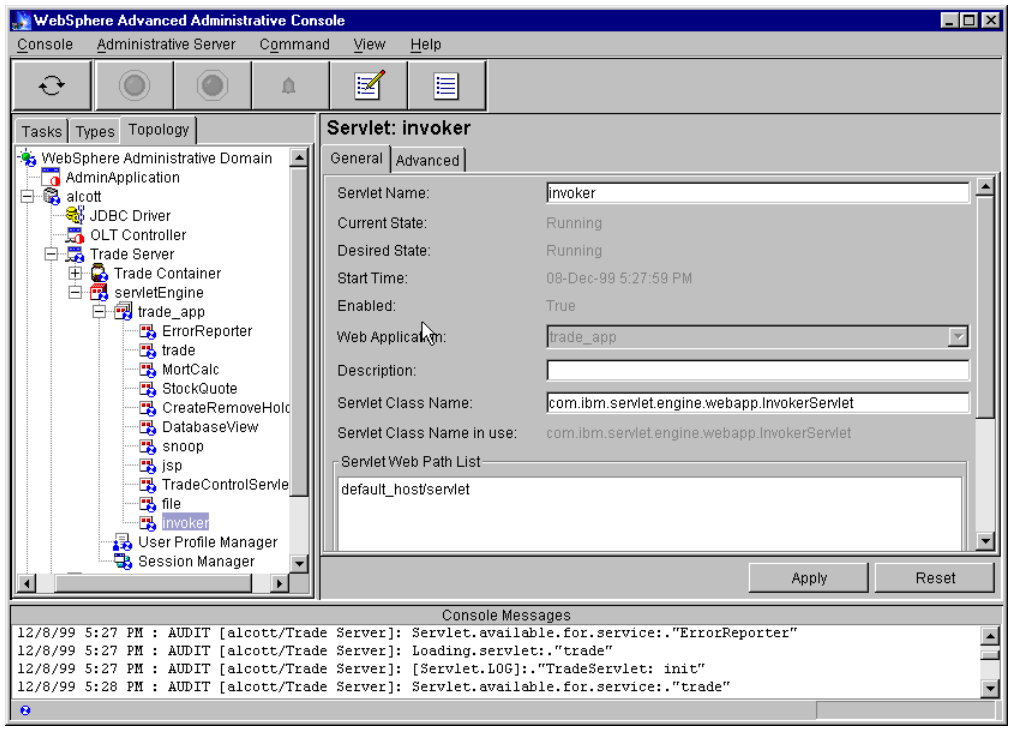


Figure 37. The invoker servlet

You remove the invoker servlet from the Web application to turn it off.

To enable the invoker servlet you either specify "Serve Servlets By Classname" when creating a Web Application (Tasks tab) or create a servlet in an existing application with the servlet class name: com.ibm.servlet.engine.webapp.InvokerServlet.

Chapter 7. Database tuning

WebSphere V3 uses a relational database, called the administrative repository, to store configuration information. The WebSphere product includes relational database software in the shipment; WebSphere Standard Edition ships with InstantDB and WebSphere Advanced Edition ships with UDB V5.2. Both editions support the administrative repository on UDB V5.2, UDB V6.1, and Oracle.

In addition to the administrative repository, many applications use a database as their back-end system. In this case, database tuning can have the largest impact on your application performance.

WebSphere also can use a database for persistent sessions, which we will discuss in Chapter 8, “Session management” on page 75.

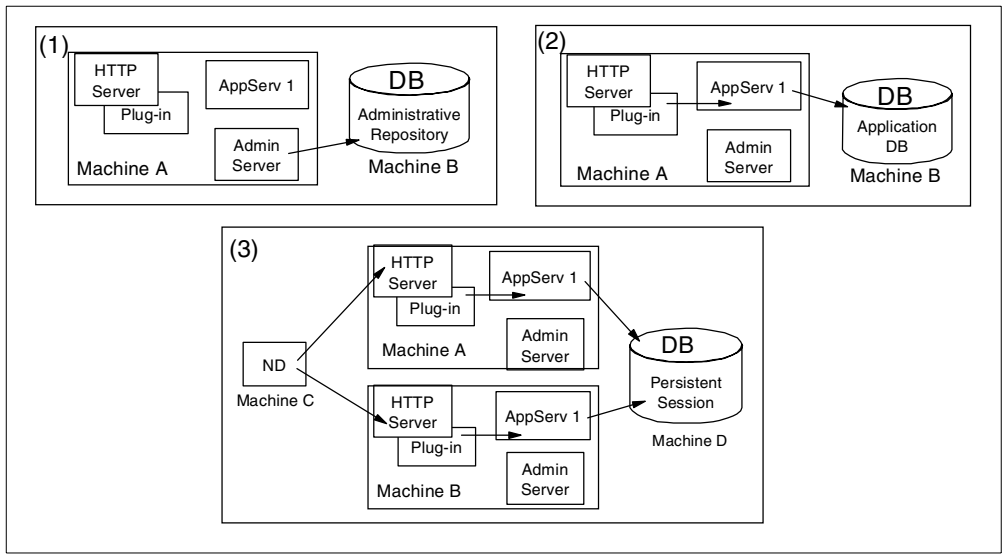


Figure 38. WebSphere and databases

Basically, there are two areas where tuning can affect database performance. The first area is in the WebSphere parameters relating to the database. The second is the database configuration. We will not go into detail on database configuration but we will take a look at some of the most effective tuning parameters.

In this chapter, we will discuss:

- Administrative repository implementation
- DataSource object settings
- Prepared statement cache sizing
- Database configuration

7.1 The WebSphere administrative repository

WebSphere Application Server uses a database to store configuration data. Parameters like servlet name, application server name, session timeout value, and others, are stored in this administrative repository. Generally, WebSphere

Admin Server keeps up to four database connections open to the administrative repository. And when you start an AdminClient the total number of connections can be up to seven.

The administrative repository needs to be created before bringing up WebSphere for the first time. For example, if you are building the database with UDB the following statements are used:

```
db2 create database was
db2 update database configuration for was using applheapsz 256
```

The database name “was” is the default repository name. You will specify this name later in the installation wizard when you start WebSphere for the first time.

It is recommended that you increase the applheapsz to 256. The applheapsz is discussed later in 7.4.2, “Applheapsz” on page 70.

The tables in the administrative repository will be created when you start an Admin Server (on AIX, `startupServer.sh` is executed) for the first time. Starting the Admin Server the first time will take a few minutes due to the tables being built.

WebSphere determines that the tables need to be built by looking at a setting in the /usr/WebSphere/AppServer/bin/admin.config file.

```
com.ibm.ejs.sm.adminServer.nameServiceJar=/usr/WebSphere/AppServer/lib/ns.jar
install.initial.config=false
com.ibm.ejs.sm.adminServer.initializer=com.ibm.ejs.security.Initializer,com.ibm
.servlet.engine.ejs.ServletEngineAdminInitializer,com.ibm.servlet.config.Initia
lSetupInitializer
```

Figure 39. admin.config (selection)

If install.initial.config is set to true, the tables will be created. If the initialization of tables has already been done, this setting will be false (see Figure 39).

If, at some time, you want to create and use a new administrative repository the steps are easy. Simply create the new database, change the install.initial.config to true (so the tables will be created) and change the URL of the database in the admin.config file to point to the new database (see Figure 40). This is especially useful in test environments where you can try different configurations by switching the administrative repository.

```
com.ibm.ejs.sm.adminServer.traceFile=/usr/WebSphere/AppServer/logs/tracefile
com.ibm.ejs.sm.adminServer.dbUrl=jdbc:db2:was52rem
server.root=/usr/WebSphere/AppServer
com.ibm.ejs.sm.util.process.Nanny.traceFile=/usr/WebSphere/AppServer/logs/nanny
.trace
com.ibm.ws.jdk.path=/usr/jdk_base/
com.ibm.ejs.sm.adminServer.dbPassword=db2inst1
```

Figure 40. admin.config (selection)

In this example, the administrative repository name is was52rem. The user ID and password for the database are also stored in the admin.config file with the following settings:

```
com.ibm.ejs.sm.adminServer.dbUser
com.ibm.ejs.sm.adminServer.dbPassword
```

Note that you can save WebSphere's configuration by backing up the database.

The relationship between WebSphere Application Server and its administrative repository is determined by the com.ibm.ejs.sm.adminServer.dbUrl setting in the admin.config file. If the database is on a remote machine, this is transparent to WebSphere. It only looks for the database name, user ID and password specified in the database catalog. Just make sure the database connection is configured correctly and working before you start the Admin Server.

Detailed information on setting up the administrative repository is found in *WebSphere Application Server Advanced Edition: Getting Started* (shipped with the product).

7.1.1 Serious event reporting

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

7.1.1.1 Serious event pool interval

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

You can change the polling interval and the types of messages reported by selecting the node in the Administration Client then selecting **Console-> Trace-> Serious Events**. See Figure 41.

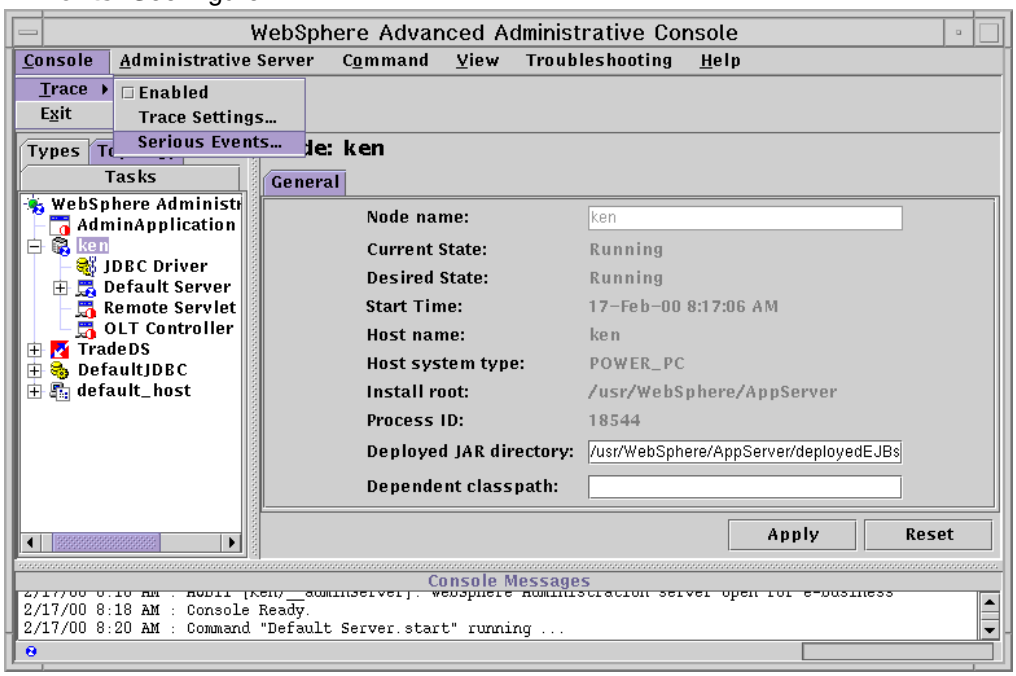


Figure 41. Serious Events

You get the new Serious Events window and under the **Preferences** tab shown in Figure 42, you can then adjust the **Serious Event Pool Interval** from the default 10 seconds.

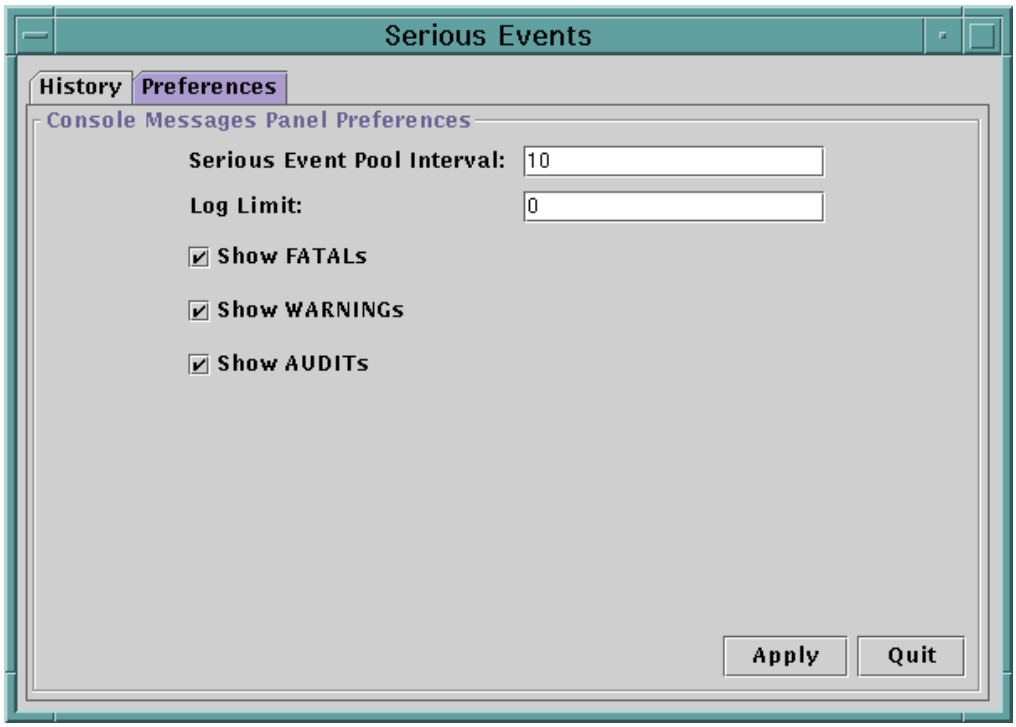


Figure 42. Serious Events Preferences

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

7.1.1.2 Log limit

A more important setting for performance is the Log Limit for the Serious Events table in the WebSphere configuration repository.

The default setting for the Log Limit is 0 (zero) as depicted in Figure 42 on page 60. This setting is misleading since this actually maps to no limit being set for the entries in the Serious Events table. Over time this will degrade database performance as the table grows and could result in a database error if sufficient space on the file is not allocated.

Setting this to a value of between 1000 and 5000 should provide adequate logging while not adversely impacting system performance.

7.2 DataSource object settings

In WebSphere Version 3 administration, databases are defined as DataSource objects. A DataSource object represents an application database which will be used for user applications. The WebSphere administrative repository is not considered a user application and therefore is not a DataSource object.

DataSource objects are used by servlets through the connection pool, by the Session Manager for persistent sessions, and by EJB containers.

7.2.1 Connection pooling

One of the most time-consuming procedures of a database application is establishing a connection to the database. In WebSphere Version 2 the Connection Manager API was used to reduce this overhead. WebSphere Version 3 provides a new API called connection pooling. Connection pooling is defined as a part of the JDBC 2.0 Standard Extension API.

Connection pooling establishes a pool of connections that user servlets can use. Once a connection is established, it is reused repeatedly so subsequent user requests incur only a fraction of the cost of a connect/disconnect. Connection pooling also lets you control the number of concurrent connections to a database server. This is useful if you have to manage the number of database users to comply with a database server license agreement.

Connection pooling is the recommended method to use when developing new servlets. The old Connection Manager API is supported by WebSphere Version 3, but only as a migration path. The differences between the Connection Manager API and connection pooling are discussed in the “What Is -> Connection Pooling” topic of the online documentation provided with WebSphere.

Servlets use a connection pool as follows:

- When a user makes a request over the Web to a servlet, the servlet uses an existing connection from the pool.
- When the request is satisfied, the servlet returns the connection to the connection pool for use by other servlets.

- The servlet uses a DataSource object to get a connection, and the connection pooling services are automatically provided through the DataSource object.

Note that the servlet does not access the connection pool directly. The servlet requests a connection through a DataSource object and the connection pooling is provided automatically.

You can modify the configuration of a connection pool using the adminclient. Select **Topology -> Datasource -> Advanced**. The defaults are shown in the figure below.

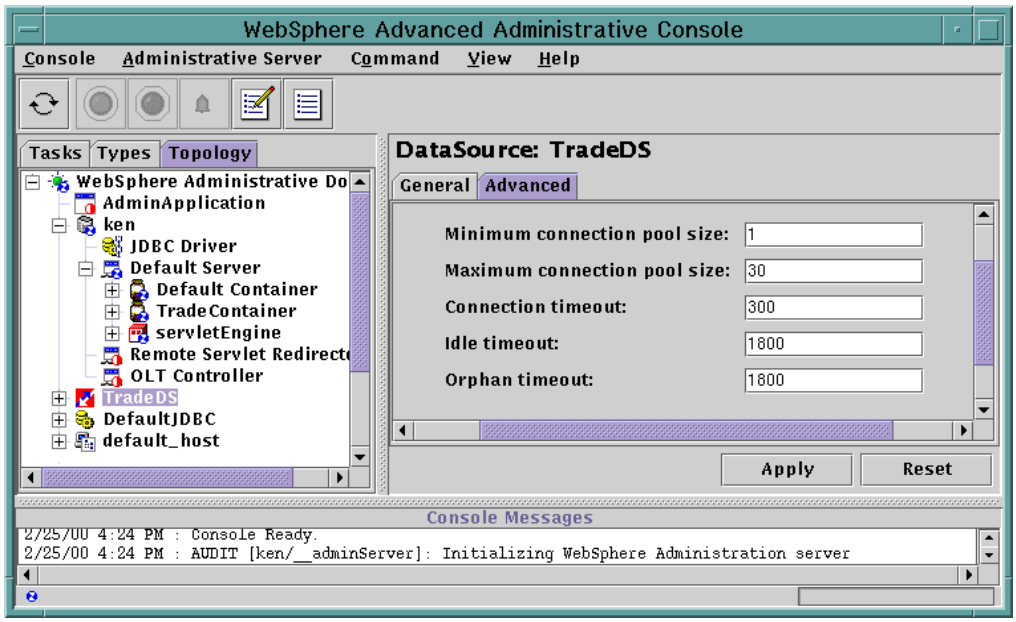


Figure 43. Datasource - Advanced

Minimum connection pool size

This is the number of connections which will be prepared at server startup.

Maximum connection pool size

This is the maximum number of connections. It includes in-use, idle, and orphaned connections. If a client request reaches minimum connection pool size, WebSphere creates a new connection. Then, if a request reaches Maximum connection pool size and all connections are in-use, the next request will have to wait for a connection to be released by other servlets.

Connection timeout

If a connection is not made available during this specified period, a com.ibm.ejs.dbm.jdbcext.ConnectionTimeoutException (a subclass of SQLException) will be returned to your application. In a production environment the default of 300 seconds is probably too long for acceptable performance. You'll want to test various settings for this parameter under load for your environment and response time criteria. A suggested start point for your testing would be in the range of 1-10 (seconds), again depending on your requirements.

Idle timeout

This parameter determines how long an idle connection can remain in the pool before being removed to free resources.

Orphan timeout

This parameter determines how long a connection can be owned by a servlet before being considered orphaned. WebSphere will consider a connection as orphaned, when the servlet uses the connection multiple times over an extended period and the connection becomes unresponsive. This might be caused by an unexpected problem.

7.2.1.1 Sample case 1

To see the effects of the connection pooling parameters we performed a primitive test of management connections. We used the ConnPoolTest.class, included in the WebSphere Samples Gallery, to access a database using the connection pooling API. The servlet accesses the SAMPLE database and executes a simple select statement.

The test configuration looked like topology 2, shown in Figure 6 on page 10, with the WebSphere Application Server residing on a different machine than the database server.

In our test environment, setting the maximum connection pool size to 3-5 gave the best performance. We set the max connections to higher than 5 and it resulted in high CPU activity with the `vmstat` command. Of course, the WebSphere node gets heavier stress than the database node in this simple SQL statement. We used the same spec machines for each node.

7.2.1.2 Sample case 2

Next, we used the sample Trade application with connection pooling to see the effect of changing the maximum and minimum connection pool size. We used five cases, setting the maximum connection pool size to 1, 3, 5, 7 and 10 respectively.

In our test environment, we saw the best performance when the maximum connection pool size was 5. Setting this parameter to 1 significantly reduced the performance (by as much as 50% of the performance with the setting at 5). Setting the maximum connection pool size to 5 or more, ensured there was less CPU idle time.

7.2.1.3 Rules of thumb

- If your application uses more than one database, you have to consider the total number of connections. If you create a clone environment or if you implement the session persistence feature, connections to the session database are also necessary. The total number of connections will be as follows:

  Maximum total number of database connections =

  Application DataSource Maximum connection pool size * Number of clone Servers +

  Session DataSource Maximum connection pool size * Number of clone Servers+

  WebSphere administrative repository connections+

(Connections from any other applications or temporary connections from command line)

- Each connection uses about 700 KB - 1800 KB of memory on each node. Increasing the maximum connection pool size seems to have a positive effect on performance until you reach the point where CPU processing becomes a bottleneck. Increasing the CPU power of the WebSphere node could allow you to increase the maximum connection pool size.
- Specifying too small a number of connections will cause the application to get a connection timeout.

DataSource Object for Persistent Sessions

It is important that a DataSource object for persistent sessions have its own connections aside from the connections to an application DataSource. By default, the maximum connection pool size of a DataSource is 30.

7.3 Prepared statements

Each time your application submits a query to a database, the database manager creates a query plan. The query is then executed with this plan. If your application is coded to use the PreparedStatement class instead of the Statement class an object is created that precompiles and stores the SQL (Structured Query Language) associated with the query plan. This approach is more efficient than running the same statement multiple times with a Statement object, which compiles the statement each time it runs.

In the example below the database builds the query plan only once when the first statement is executed.

```
PreparedStatement pstmt = con.prepareStatement(select exchange_rate from
rates where currency =  ?");
   //Initialize first parameter with currency
   pstmt.setString(1, Flan);
   ResultSet results = ps.executeQuery();
Subsequent invocations of the query are performed by substitution of the
appropriate variables
   pstmt.setString(1, Lira);
```

Figure 44. Example of the prepared statements

7.3.1 Prepared statement cache

WebSphere V3.x provides an additional performance improvement to that already provided by using the PreparedStatement class, with a Prepared Statement cache. By caching prepared statements, the initial overhead of the database manager creating an SQL query plan (as described above) for a PreparedStatement class is avoided. When possible, WebSphere will provide the database manager with an existing execution plan for the prepared statement.

The size of the prepared statement cache is specified by adding a command line argument to your application server, select **Topology-> Default Server-> General**. As shown in Figure 45, in the Command line arguments field specify:

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

This cache is shared among all DataSources. In our testing on AIX we found the following formula most appropriate for determining the appropriate setting:

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

For the application that we tested there were only two prepared statements. So for our tests with 100 concurrent clients, we set the prepared statement cache to 200 and noticed a significant performance improvement from tests with cache settings of 50, 100, 150, and did not notice any significant improvement in tests where the setting was higher than this.

For example, for 20 concurrent user threads, you would set the maximum connection pool size to 20. Assuming each client is doing just an entity read and there are two different PrepStmts (load and save), the prepared statement cache size should be at least 2*20 = 40.

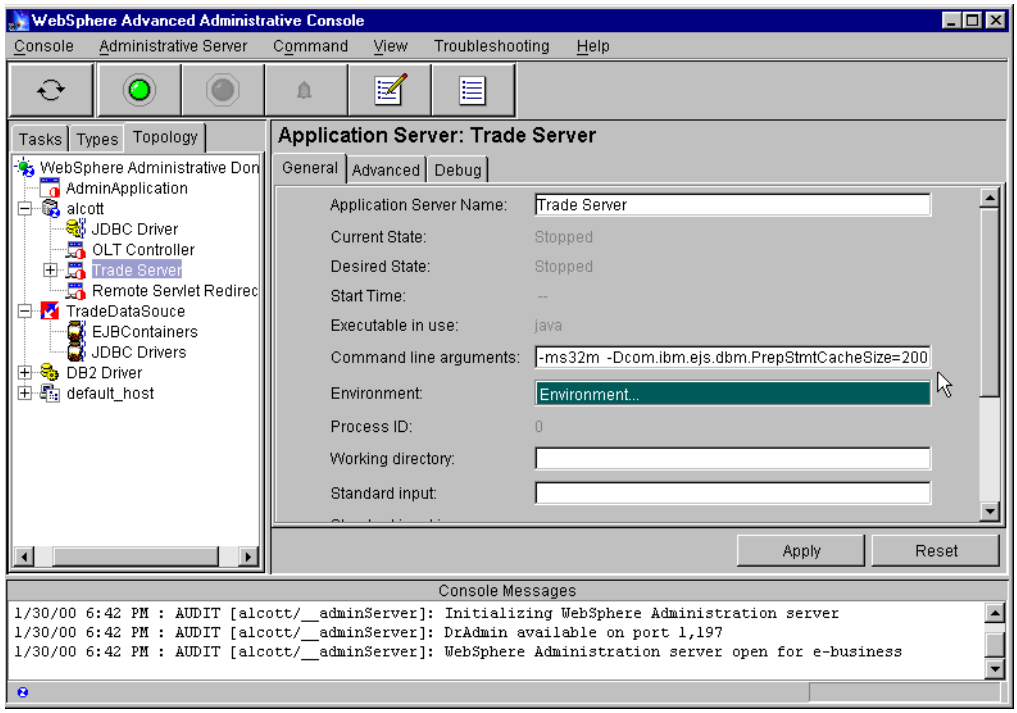


Figure 45. Prepared statement cache

For Windows NT

The WebSphere Performance team has found that the following formula works best on Windows NT:

PrepStmtCache > #PrepStmt * min (concurrent user threads, connection Pool Size)

7.3.2 Prepared statement key cache

This parameter controls the size of the intermediate cache used to map between SQL statements and entries in the prepared statement cache. Unlike the

prepared statement cache which is shared among *all* Data Sources, this parameter specifies the number of entries in cache for *each* Data Source. A reasonable value would be the expected number of unique prepared statements which you expect to have in the application. As in Figure 46, we can specify in the Command line arguments field:

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

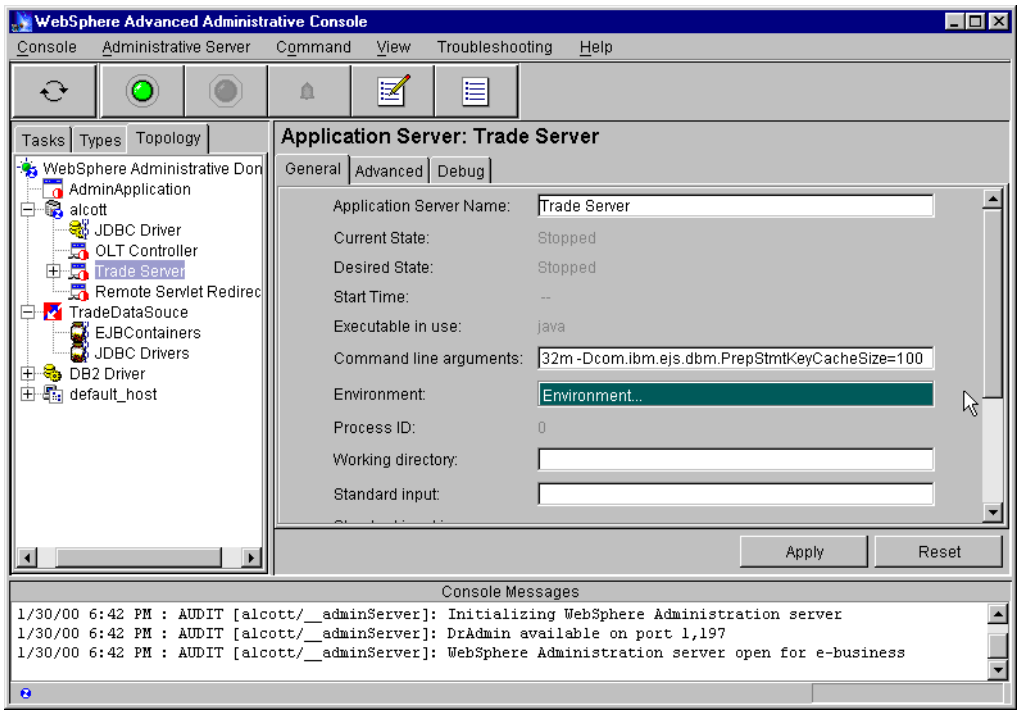


Figure 46. Prepared statement key cache

7.4 UDB configuration

In this topic we will discuss only general parameters to consider. All parameters which we describe can apply to your application database, the WebSphere administrative repository and the persistent session database. Table 3 shows the parameters we used in testing the performance of our test application, Trade servlet.

Table 3. Database configuration parameters

Parameter	Default Value
BUFFPAGE	1000 (4 KB page)
APPLHEAPSZ	128 (4 KB)
PCKCACHESZ	8 * maxappls, if it is smaller than 32, 32 will be the default value
MAXAPPLS	40
DFT_DEGREE	1
LOCKLIST	100 (4 KB)
MAXLOCKS	10 (%)

Parameter	Default Value
LOCKTIMEOUT	-1 (sec)
MAXAGENTS	200

7.4.1 Buffpage

The buffer pool is allocated when the first application connects to the database or when the database is explicitly activated. As an application requests data out of the database, pages containing that data are transferred to one of the buffer pools. Pages are not written back until the page is changed and one of the following occurs:

- All applications disconnect from the database.
- The database is explicitly deactivated.
- The database quiesces.
- Its space is required for another page that needs to be read into the buffer pool.
- A page cleaner is available and is activated by the database manager.

If the buffer pools are large enough to keep the required data in memory, less disk activity will occur. If the buffer pools are not large enough, the overall performance of the database can be adversely affected.

Note

To determine whether the buffpage parameter is active for a buffer pool, you can use the following command:

```
SELECT * from SYSCAT.BUFFERPOOLS
```

Each buffer pool that has an NPAGES value of -1 uses buffpage.

To change the value NPAGES of an existing buffer pool to -1, you can use the `db2 alter` command as follows:

```
db2 alter bufferpool ibmdefaultbp size -1
```

7.4.1.1 Update buffpage size

The default value for buffpage, 1000 (4 MB), is often not large enough. You can update a buffpage parameter with the following command:

```
db2 update db cfg for <database_name> using BUFFPAGE 2000
```

In our test case, we changed the buffsize value from 1000 (4 MB) to 2000 (8 MB). This change gave us a 2.11% performance increase in our sample trade application. Keep in mind that the buffpage parameter will hold a specific amount of memory, so you need to balance increased buffer pool sizes with the need for enough memory for the rest of the processes active on the machine.

7.4.1.2 Monitor buffpage size

Before updating the buffpage size, we recommend that you check the current usage of buffer pages. We used a DB2 snapshot of the data collected by the DB2 system monitor to do this.

DB2 has a system monitor that collects information about performance. The type of information collected can be determined by setting monitor switches. By

default, buffer related information is not collected so you will need to set up the monitoring switches. There are two ways to do this:

1. Set the `DFT_MON_BUFPOOL` parameter “on” with the following command:

   ```
   db2 update dbm cfg using DFT_MON_BUFPOOL ON
   ```

2. Use the `update monitor switches` command:

   ```
   db2 update monitor switches using BUFFERPOOL on
   ```

The first option, the `update dbm cfg` command, will be effective until you re-issue the command and specify Off. It doesn't affect applications which were connected to the database before the command was executed.

The second option, the `update monitor switches` command, affects only the application that executes this command. This value will be reset when the application terminates.

To confirm the current setting of monitor switches, use the following command:

```
db2 get monitor switches
```

To reset the statistics, use the following command:

```
db2 reset monitor all
```

To get a snapshot of the information, use the following command:

```
db2 connect to <database_name>
db2 get snapshot for database on <database_name>
```

The following command collects only buffer pool related information:

```
db2 get snapshot for bufferpools on <database_name>
```

Figure 47 shows the output of the `snapshot` command. As you can see there is a line that tells the number of buffer pool data physical reads. The higher this number is, the more impact you will see on performance because physical reads and writes are time consuming.

```
 Database Snapshot

Database name                               = TRADES
Database path                               =
/home/db2inst1/db2inst1/NODE0000/SQL00003/
Input database alias                        = TRADES
Database status                             = Active
Catalog node number                         = 0
Catalog network node name                   =
Operating system running at database server= AIX
Location of the database                    = Local
First database connect timestamp            = 12-03-1999 12:08:47.403708
Last reset timestamp                        =
Last backup timestamp                       = 12-02-1999 13:49:50.499393
Snapshot timestamp                          = 12-03-1999 12:15:58.821643

// omitting..

High water mark for database heap           = 340573

Buffer pool data logical reads              = 64580
Buffer pool data physical reads             = 116
Asynchronous pool data page reads           = 63
Buffer pool data writes                     = 72
Asynchronous pool data page writes          = 71
Buffer pool index logical reads             = 95101
Buffer pool index physical reads            = 104
Asynchronous pool index page reads          = 0
Buffer pool index writes                    = 43
Asynchronous pool index page writes         = 40
Total buffer pool read time (ms)            = 586
Total buffer pool write time (ms)           = 1041
Total elapsed asynchronous read time        = 41
Total elapsed asynchronous write time       = 949
```

Figure 47. Output of snapshot #1 (before update buffpage size)

Figure 48 shows a snapshot taken after the parameter was changed from 1000 to 2000. The number of reads has dropped to 0. This means that all read requests hit the buffer pool cache in the memory.

```
 Database Snapshot

Database name                               = TRADES
Database path                               =
/home/db2inst1/db2inst1/NODE0000/SQL00003/
Input database alias                        = TRADES
Database status                             = Active
Catalog node number                         = 0
Catalog network node name                   =
Operating system running at database server= AIX
Location of the database                    = Local
First database connect timestamp            = 12-03-1999 13:13:26.866103
Last reset timestamp                        = 12-03-1999 13:28:08.993190
Last backup timestamp                       = 12-02-1999 13:49:50.499393
Snapshot timestamp                          = 12-03-1999 13:32:00.568447

High water mark for connections             = 10

//

Buffer pool data logical reads              = 11318
Buffer pool data physical reads             = 0
Asynchronous pool data page reads           = 0
Buffer pool data writes                     = 1
Asynchronous pool data page writes          = 0
Buffer pool index logical reads             = 18692
Buffer pool index physical reads            = 0
Asynchronous pool index page reads          = 0
Buffer pool index writes                    = 5
Asynchronous pool index page writes         = 0
Total buffer pool read time (ms)            = 0
Total buffer pool write time (ms)           = 150
Total elapsed asynchronous read time        = 0
Total elapsed asynchronous write time       = 0
```

Figure 48. Output of snapshot #2 (after update buffpage size)

7.4.2 Applheapsz

Application heap contains memory blocks used by UDB to process application requests. To update the applheapsz parameter:

```
db2 update db cfg for <database_name> using applheapsz <block_number>
```

When we set this value to 512 (versus the default of 128) for our test database application, we saw a performance improvement of 5.44%.

Note

If you used the <was_dir>/bin/createdb2.sh shell script when you installed WebSphere, it updated the applheapsz parameter to 256 for the WebSphere administrative repository.

7.4.3 Pckcachesz

The package cache size (pckcachesz) is allocated from the application heap size (applheapsz). Access plans are stored in the package cache. You will need to consider resizing the applheapsz parameter if you increase the package cache size. When an application uses a specific section often the package cache will be efficiently used. You can use the following command to update the size of pckcachesz:

```
db2 update db cfg for <database_name> using pckcachesz <number>
```

7.4.4 Maxappls

This parameter specifies the maximum number of concurrent applications that can be connected (both local and remote) to a database. You can update this value with the following command:

```
db2 update db cfg for <database_name> using maxappls <number>
```

To estimate the total number of connections to an application database use the following formula:

Application DataSource Maximum connection pool size * Number of Clone Servers + (Connections from any other applications or temporary connection from command line)

You also need to consider this value for the session database. If you increase DataSource Maximum connection pool size without increasing this value, you will get SQL Error SQL1040N and your application will fail.

See also 7.4.6, “Locklist” on page 72 and 7.4.9, “Maxagents” on page 73.

7.4.5 Dft_degree

The dft_degree parameter specifies whether to enable SMP parallel procedures. The default value is 1, meaning not to use SMP parallel. SMP parallel is suitable for applications which use only a select statement and utilities which collect information from the database. If your application is an OLTP type, enabling this SMP parallel procedure might cause a decrease in performance.

In our test case using Trade (an OLTP application), performance decreased by approximately 36% when we set dft_degree to -1 (use SMP parallel) with the following command:

```
db2 update db cfg for <database_name> using dft_degree -1
```

Note

If you don't want to use SMP parallel at all, you should specify no for the INTRA_PARALLEL parameter in the Database Manager configuration.

```
db2 update dbm cfg using INTRA_PARALLEL no
```

7.4.6 Locklist

Locklist indicates the amount of storage that is allocated to the lock list. Locking is the mechanism that the database manager uses to control concurrent access to data in the database by multiple applications. Both rows and tables can be locked. There is one lock list per database and it contains the locks held by all applications concurrently connected to the database. Lock escalation is the process of replacing row locks with table locks, reducing the number of locks in the list. You can update this value with the following command:

```
db2 update db cfg for <database_name> using locklist <number>
```

We tested with two locklist values, 100 and 200. When we set this value to 200, we saw performance improvements with database lock snapshot. The value of lock waits went down 19%, lock list memory in use went down 20%, Time database waited on locks (ms) went down 38%. And also we noticed that the HTTP performance (request per second) went up 5%.

7.4.7 Maxlocks

The maxlocks parameter defines a percentage of the lock list held by an application. When the number of locks held by any one application reaches this percentage of the total lock list size, lock escalation will occur for the locks held by that application. Lock escalation also occurs if the lock list runs out of space. Therefore, when you increase maxappls, you should keep it lower than maxlocks or increase locklist to prevent lock escalation.

Monitoring Lock Escalation

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

Set the DFT_MON_LOCK parameter to ON with the following command:

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

Use the update monitor switches command:

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

To reset snapshot data, use the following command:

```
db2 reset monitor all
```

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

```
db2 connect to <database_name>
db2 get snapshot for locks on <database_name>
```

7.4.8 Locktimeout

If your application response time appears to be too long, review the locktimeout setting. The default setting is -1, which means no connection timeout. If you do not want the transaction to keep trying to get a connection, set the locktimeout value to a positive number measured in seconds. The application programmer will need to determine what locktimeout value is appropriate for the user community.

For example, user A connects to the database and is updating table T1. When the locktimeout value is set to the default (-1), user B will not be able to update table T1. There is no recommended number to use for the locktimeout value. The application programmer will need to experiment with different locktimeout values until the application response time is appropriate for the application users.

You can change the locktimeout value with the following command:

```
db2 update db cfg for <database_name> using locktimeout <number>
```

7.4.9 Maxagents

The maxagents parameter indicates the maximum number of database manager agents. The value of maxagents should be at least the sum of the values for maxappls in each database allowed to be accessed concurrently.

For Websphere applications the maximum number of connections can be calculated with the following formula:

Maximum total number of database connections =

Application DataSource Maximum connection pool size * Number of Clone Servers +

Session DataSource Maximum connection pool size * Number of Clone Servers +

WebSphere administrative repository connections +

(Connections from any other applications or temporary connections from a command line)

You can change the value of maxagents with the following command:

```
db2 update dbm cfg using maxagents <number>
```

Chapter 8. Session management

When establishing a Web application site, it is desirable to be able to track client status. In many cases, a client's desired procedure will consist of several Web pages and several methods of a servlet. A series of requests from the same client to the same browser is called a session. If your application requires sessions to be maintained across requests, WebSphere supports the Servlet API HttpSession Interfaces (javax.servlet.http.HttpSession). With this interface, an application can track client status and keep client information temporarily. WebSphere provides a Session Manager to handle these session objects.

8.1 Session information

WebSphere provides two ways to store session information on the application server. One way is to store the session in memory. The second way is to store the session in a database. Sessions stored in a database are called persistent sessions.

8.2 Keeping session information in memory

By default, the Session Manager uses the in-memory mode.

From the **Topology-> WebSphereAdminDomain-> hostname-> DefaultServer-> servletEngine-> Session Manager-> Enabled** tab as shown in Figure 49 on page 75, you can see that Enable Persistent Sessions is set to the default setting of No.

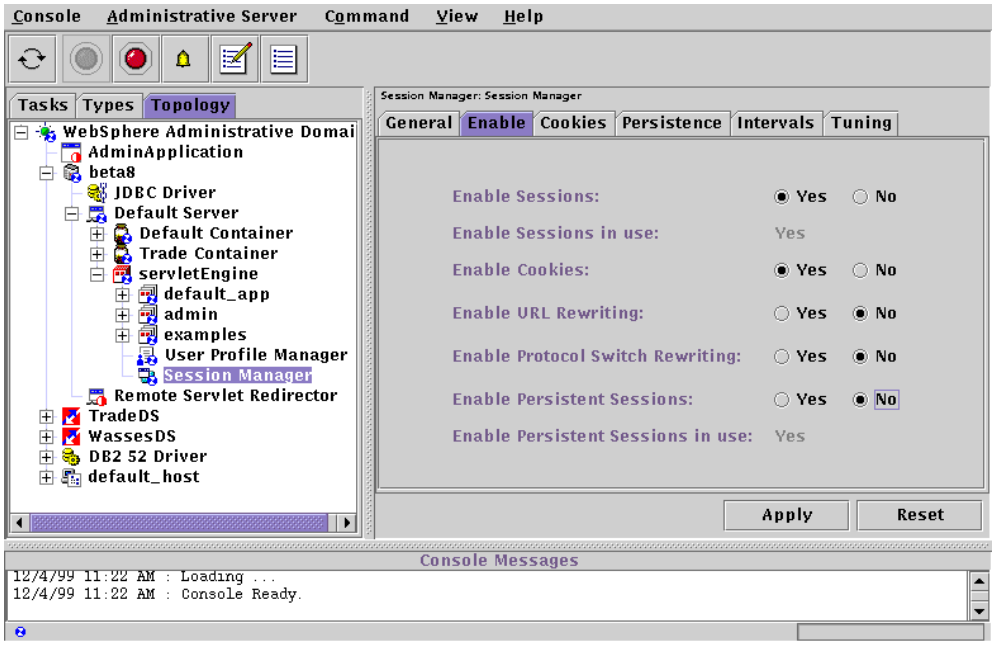


Figure 49. Session Manager - Enable

8.3 Persistent sessions

Using persistent sessions is of significant value to a Web application. By storing sessions in a database the session state is maintained even if you reboot the application server, or if an unexpected system error occurs. It also allows access from more than one instance of an application server, which allows for application server clustering. If you are considering cloning applications on a single node, or using more than one application server on separate machines, or using a combination of the two, sessions must be persisted to a database.

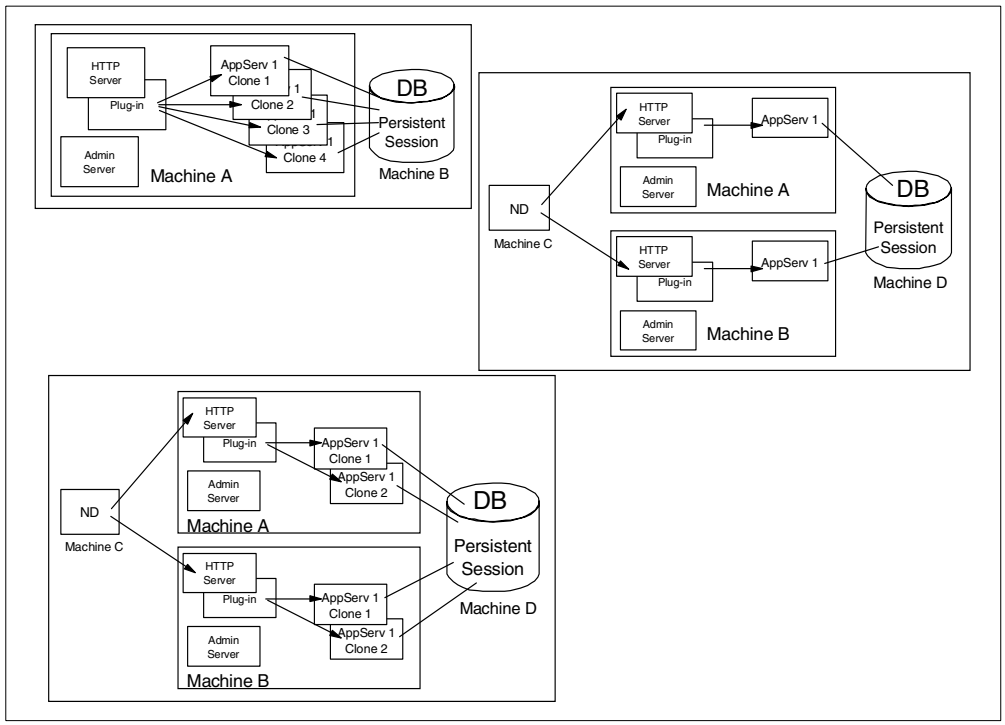


Figure 50. Persistent sessions

In a scenario with one application server, a persistent session is generally slower than a session in memory and may cause degradation in performance.

8.3.1 Database/Datasource configuration

To use persistent sessions, you have to create a database on your database server and then define it as a Datasource object in WebSphere. As we discussed in the previous chapter, you should modify your database settings. And also we recommend that you modify the number of connections in the Datasource setting for session persistence.

8.3.2 Session Manager configuration

To modify the Session Manager setting to allow persistent sessions:

1. On the **Topology-> WebSphere Administrative Domain-> hostname-> Default Server->** servletEngine**-> Session Manager-> Enable** tab, set Enable Persistent Sessions to Yes and click **Apply**.
2. On the **Session Manager -> Persistence** tab as shown in Figure 51, click the Datasource **Change** button and select the Datasource you defined before.

Keep the Persistence Type set to directodb, meaning that WebSphere will store the session information to a database using the JDBC connection. Persistence Type ejb will be supported in a future version of WebSphere. For the time being, do not select this option.

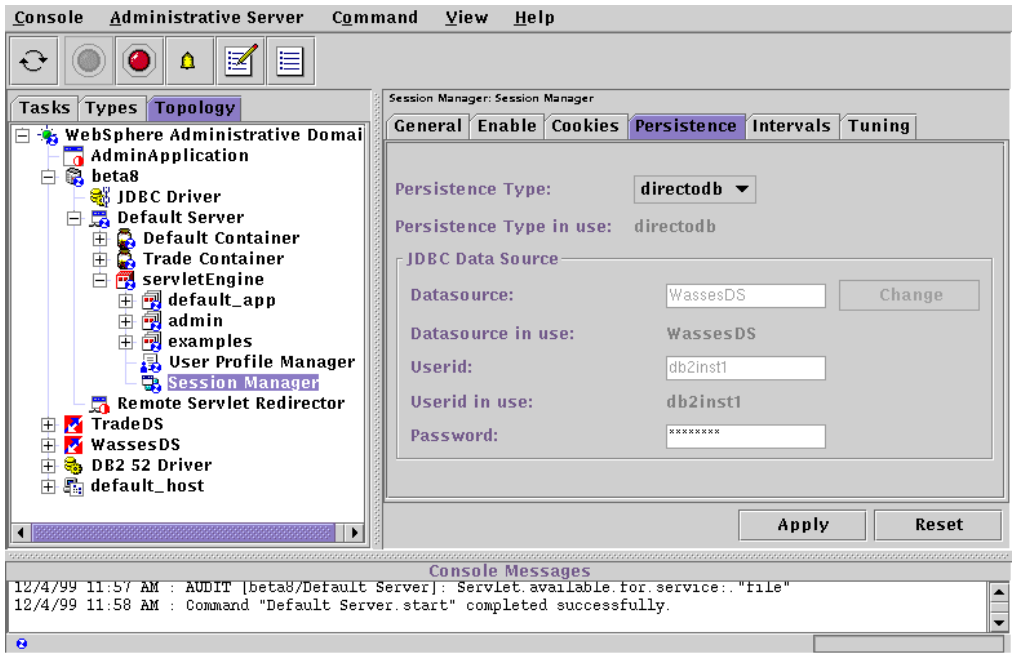


Figure 51. Session Manager - Persistence

8.4 Tuning the Session Manager

There are two tabs related to performance tuning on the Session Manager panel, the Intervals tab and the Tuning tab. We will discuss the tuning parameters in the following sections.

8.4.1 The Invalidate Time setting

Under the Intervals tab, you can specify the period a session is allowed to go unused until it is no longer considered valid in the Invalidate Time field. Integers representing seconds are allowed. A “-1” means the session will not be invalidated.

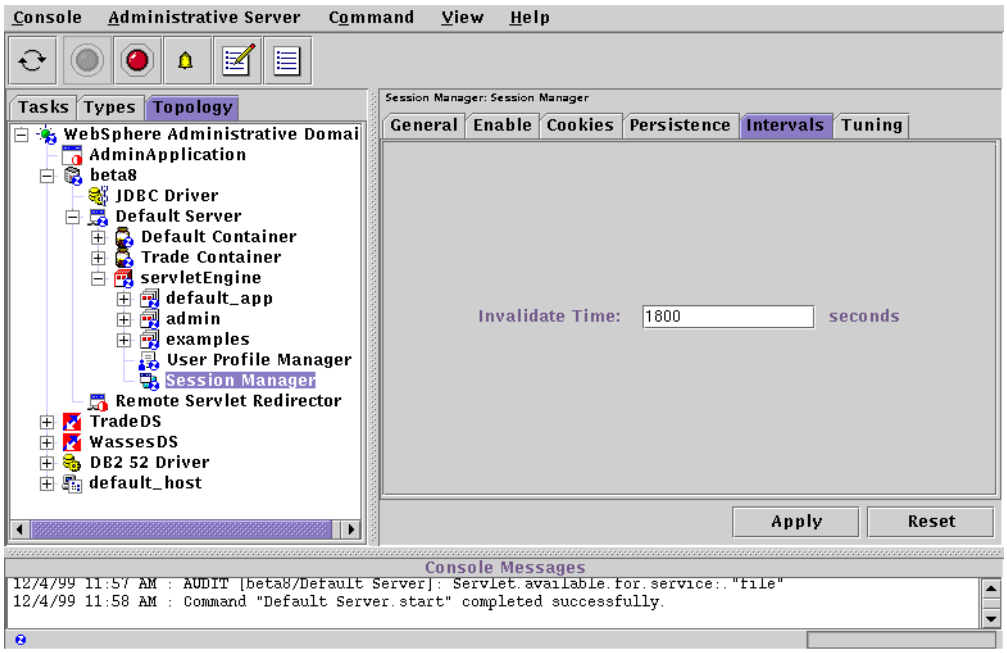


Figure 52. Session Manager - Intervals

If you are using in-memory mode sessions, a session that passes the Invalidate Time setting will disappear from memory.

If the session is persistent and the Invalidate Time has been reached, the rows that contain the session information will be deleted from the Sessions table in the database.

8.4.2 Monitor and estimate Invalidate Time

The correct Invalidate Time value for a particular application varies with machine capacity. To estimate it, you can use `vmstat` to monitor memory usage and the Resource Analyzer to confirm session numbers and usage. The default value (1800 seconds) may be desirable to lower it, so you might start testing around 600 seconds. On the Sessions tab with Resource Analyzer as shown in Figure 53 on page 79, you can see the number of sessions that have been created and invalidated within the monitoring refresh interval.

If you are using in-memory mode, the Invalidate Time should be set to an appropriate value that doesn't stress the actual memory size of the machine. If the value is too high and too many sessions are kept in memory, paging will go up and performance will suffer.

It depends on your application, but if possible, sessions should be closed with the invalidate method. If your application is not the type which can release sessions or if the clients tend to leave the site before the end of the procedure, the number of sessions can become large. Since the Invalidate Time parameter affects the amount of memory used for sessions, the setting of this parameter is extremely important and can affect application server response time. We will also discuss other options for managing the number of in memory sessions, such as Allow Overflow, and Base Memory Size later.

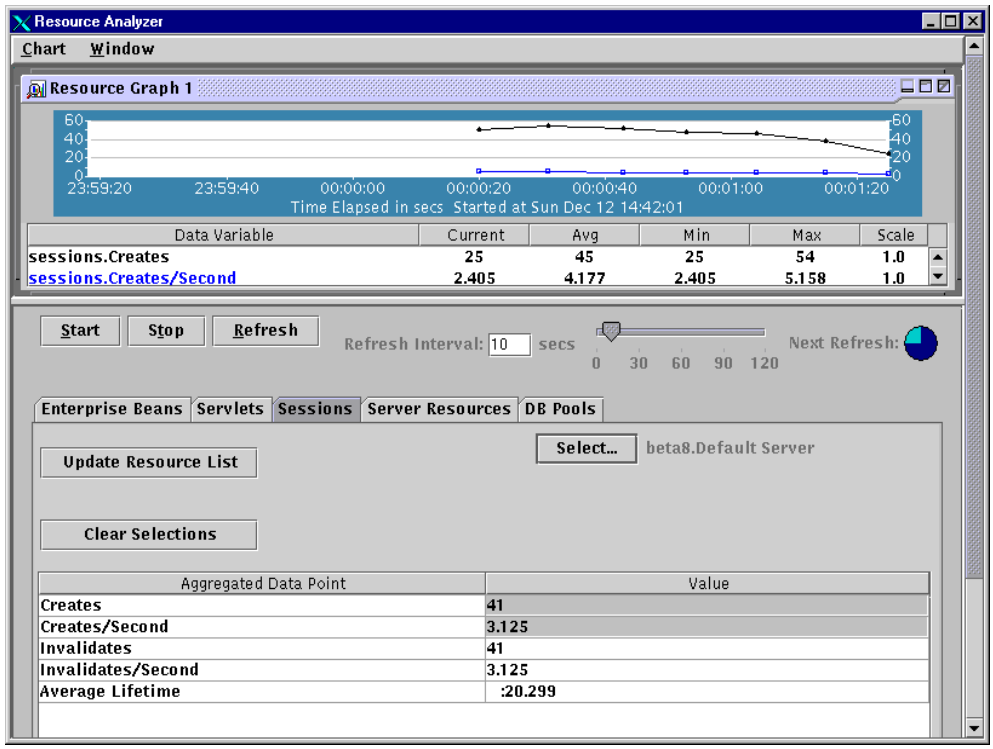


Figure 53. Resource Analyzer - Sessions

8.4.3 Tuning parameters on the Tuning tab

With WebSphere 3.0 Session Management, several new features have been added that allow advanced customers to tune the performance and operating characteristics of the Session Manager. You can also specify these features with the Tuning tab as shown in Figure 54 on page 80.

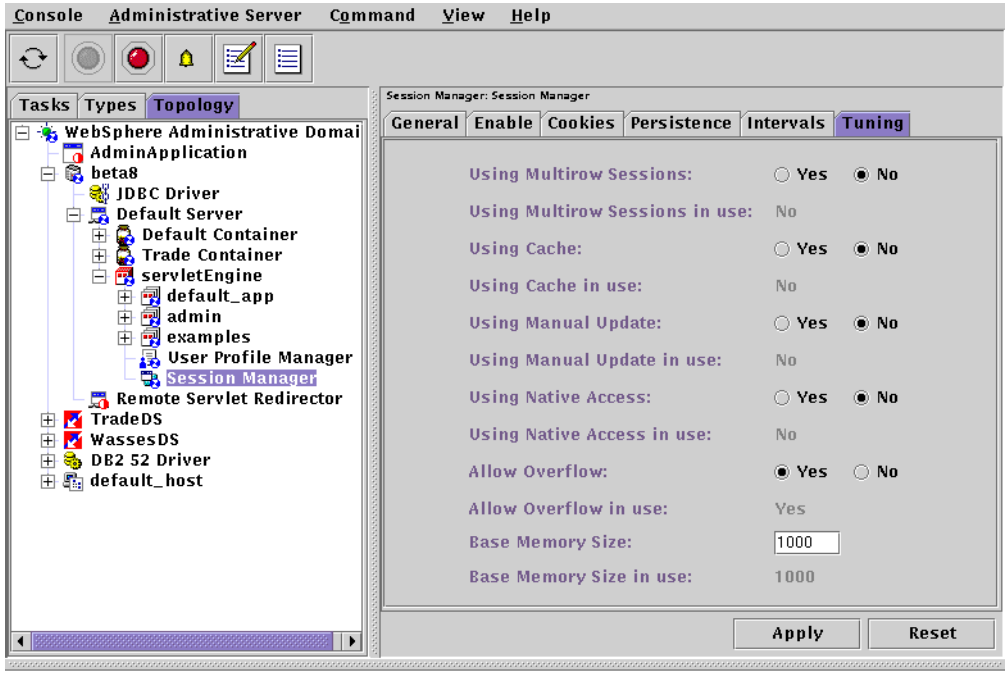


Figure 54. Session Manager - Tuning

8.4.4 Multirow sessions

By default (or select **No**), a single session maps to a single row in the database table used to hold sessions. With this setup, there are hard limits on the amount of user-defined, application-specific data that WebSphere Application Server can access. Specifically, with UDB, the maximum space available is 32,700 bytes (using the long varchar for bit data UDB type vs. BLOBs as a compromise between space and performance). With Oracle, the maximum amount is 2 MB (where the only choices are "raw" for a maximum of 2 KB, or the "long raw" data type of 2 MB).

WebSphere Application Server V3.0 Session Manager, however, allows for the use of a multirow schema option where each piece of application-specific data is stored in a separate row of the database. With this mode, the total amount that can be placed in a session is now bound only by the capacities of the database. We will relax the restriction and add a BLOB column for UDB. The only practical limit remaining is the size of the data object itself, where the above limits (32 KB for UDB or 2 MB for Oracle) should be more than enough for any single Java object.

This parameter is applicable only when using persistent sessions.

Another reason to use this setting is that you can store the session information separately by property ID. With this parameter, the session table is used as shown in Table 4 on page 81.

Table 4. Model of session table with setting using multirow

ID	PROPID	MEDIUM
<Cookie id A>	<Cookie id A>	-
<Cookie id A>	<Property name 1>	<Value of property 1>
<Cookie id A>	<Property name 2>	<Value of property 2>
<Cookie id B>	<Cookie id B>	-
<Cookie id B>	<Property name 1>	<Value of property 1>
<Cookie id B>	<Property name 2>	<Value of property 2>

The property name which you specified in the Session.putValue() method will be stored in the PROPID column.

When the application needs to refer to only one property, the Session Manager doesn't have to seek the whole object. The Session Manager selects the appropriate object more efficiently by using the Propid key.

The multirow schema potentially has performance benefits in certain usage scenarios, such as when larger amounts of data are stored in the session but only small amounts are specifically accessed during a given servlet's processing of an HTTP request. In such a scenario, avoiding unneeded Java object serialization is beneficial to performance.

To determine if this parameter is good for your application, create two session databases: one using the default setting (which uses a single row for each session), the other using multirow. Create two Datasource objects (one for each session database) and switch between the two using **Session Manager->Persistence -> Datasource.** When you choose the session database that specifies Using Multirow Sessions as Yes, the Session Manager will create a row for each session object. And then, you can compare the performance results of both settings.

8.4.5 Using cache

WebSphere will maintain a list of the most recently used sessions in memory and avoid using the database to read in or access the session when it determines that the cache entry is still the most recently updated copy. The default is No.

In our test cases, a 21% performance improvement was seen when this setting was Yes.

This value applies only when persistent sessions are enabled and the Persistence Type is "direct to database."

See 8.4.9, “Base memory size” on page 83 about controlling the size of the cache.

To efficiently use the cache, WebSphere Application Server Version 3.0 needs an "affinity" mechanism to help ensure cache hits. As an example, if Fred is visiting a Web site serviced by a WebSphere cluster consisting of 3 machines (A, B, and C), and his client hits machine A on a given request, then there is a performance

benefit to the WebSphere clustering mechanism by ensuring his client will use machine A on subsequent requests.

For WebSphere 3.0, the affinity features are provided by our sister product, the WebSphere Performance Pack (which includes among other things the Network Dispatcher and the Web Traffic Express Proxy Server facilities). In particular, two affinity features exist. The first and older feature is the "sticky port", where Network Dispatcher (ND) uses, on top of its load balancing functionality, the client's IP address to maintain the affinity. The second and newer feature is content-based or cookie-based affinity, where ND and the Web Traffic Express (WTE) Proxy Server work in conjunction to load balance, using cookies to identify clients and maintain affinity.

8.4.6 Using manual update

WebSphere Application Server Version 3.0 by default always updates the database with any changes made to the session during the servlet's processing of an HTTP request (that is the execution of the service() method). These updates minimally include the last access time of the session, and typically also include changes made by the servlet (that is updating or removing application data).

When manual update is turned on (select Yes), WebSphere will no longer automatically update the database at the end of a servlet's service() method. The last update times are cached and updated asynchronously prior to checks for session invalidation. For any permanent changes to the session as part of the servlet processing, the servlet must manually call the sync() method provided as part of the WebSphere extension to HttpSession, the com.ibm.websphere.servlet.session.IBMSession interface when running in manual update mode.

This value applies only when persistent sessions are enabled and the Persistence Type is "direct to database."

Depending on your application, performance can be improved by allowing the servlet to determine when a write to the database should be made. This is because the number of times an HTTP request's processing leads to changing a session (typically its application data) may actually be less than the number of times the session is accessed or read in. If one is able to combine manual updating (for minimizing database writes) along with caching (to minimize database reads), performance can greatly improve.

8.4.7 Using native access

Specifies whether to perform session persistence database updates using optimized JNI-based SQL access, written in the C programming language. The default is No. This parameter will be supported in a future version of WebSphere. For the time being, it is not used.

8.4.8 Allow overflow

By default, the number of sessions maintained in memory is specified by Base Memory Size. If you do not wish to place a limit on the number of sessions maintained in memory and allow overflow, set this value to Yes as default. Allowing for an unlimited number of sessions, however, can potentially exhaust

system memory and even allow for system sabotage (where somebody could write a malicious program that continually hits your site and creates sessions, but ignores any cookies or encoded URLs and never utilizes the same session from one HTTP request to the next).

When overflow is disallowed, the Session Manager will still return a session with the HttpServletRequest's getSession(true) method if the memory limit has currently been reached, but it will be an invalid session which is not saved. With the WebSphere extension to HttpSession, com.ibm.websphere.servlet.session.IBMSession, there is an isOverflow() method which will return "true" if the session is invalid. Your application could then check this and react accordingly.

8.4.9 Base memory size

The base memory size setting specifies the number of sessions in memory. The default is 1000. This value holds when you are using in-memory sessions, persistent sessions with caching, or persistent sessions with manual updates. (The manual update cache keeps the last n time stamps representing "last access" times, with n being the base memory size value.)

This number has several meanings:

- For the in-memory mode sessions:

  This value specifies the number of sessions in the base session table. Use the Allow Overflow property to specify whether to limit sessions to this number for the entire Session Manager, or allow additional sessions to be stored in secondary tables.

- For the persistent sessions:

  It also specifies the size of the cache, as well as the number of last access time updates that are saved in manual update mode. In either case, once this number is surpassed, these functions are bypassed (that is any sessions after this number are simply not cached, and any session updates past this number are automatically sent back to the database).

In other words, if the cache property is enabled, the base memory size specifies how many session updates will be cached before the Session Manager reverts to reading session updates from the database automatically.

This parameter can be used to tune performance when memory is used at a high rate. With the Resource Analyzer, check the usage of memory for sessions and tune this parameter. How you customize this setting will depend on your hardware system, the usage characteristics of your site, and your willingness to increase the stack sizes of the Java processes for your application servers to accommodate a larger value. The size of the session objects and number of sessions (base memory size) will need to be determined by the application

programmer. These values should be recorded and given to the systems administrator to make these changes.

Table 5. .Base Memory Size

Enable Persistence	Using Cache	Manual Update	Allow Over Flow	Base Memory Size
No (in-memory)	Yes / No	N / A	No	The number of sessions in the base session hash table.
No (in-memory)	Yes / No	N / A	Yes	The total number of sessions. Secondary hash table allowed to be created.
Yes (in-database)	No			Not regarded.
Yes (in-database)	Yes			The number of session updates will be cached in memory before the Session Manager saves changes into the database.
Yes (in-database)		No		Not regarded.
Yes (in-database)		Yes		The manual update cache keeps data related to invalidated time, and corresponding number of sessions at base memory size.

Chapter 9. Performance test tools

In order to test the throughput and scalability of a particular application or hardware architecture you will need to simulate a load. The WebSphere Application Server does not include any tools for the purpose of load generation or client side performance monitoring. There are a number of tools available for this purpose. Some are available for free and some are at cost. Any of the tools mentioned below can be used alone or in conjunction with the WebSphere Resource Analyzer for performance testing. Some of the tools, in no particular order one discussed in the following sections.

9.1 WebStone

WebStone is an open source benchmark tool that is freely available from Mindcraft. WebStone was originally developed by Silicon Graphics to measure the performance of Web server software and hardware products. Mindcraft Inc. acquired the rights to WebStone from Silicon Graphics and is currently providing support for WebStone 2.5.

WebStone simulates the activity of multiple Web clients thus creating a load on a Web server. The load can be generated from either one client computer or by multiple client computers. According to Mindcraft it is possible to run in excess of 100 simulated Web clients on a single computer.

As freeware though WebStone does not offer many of the features that are available from other products at cost. Among the features **not** supported as of this writing are:

SSL, POST, HTTP 1.1, cookies, dynamic workloads with database access, authentication, HTTP 1.0 keep-alive support, and multiple headers/URLs.

Further information and downloads of the source code and executables are available from Mindcraft at `http://www.mindcraft.com`.

9.2 AKtools

AKtools are a set of internal IBM applications which allow a user to test Web application performance and was the tool used in writing this redbook.

The two current applications are AKstress and AKrecord. AKstress is a high performance, simple threaded HTTP engine which is capable of simulating hundreds or even thousands of HTTP clients, using a highly configurable set of directives from a plain text configuration file. AKrecord is a simple eaves dropping proxy which will record a user's session against a Web server for later playback in AKstress.

AKtools provides a variety of functions:

- Fully configurable HTTP headers
- SSL support
- Support for HTTP/1.1
- Built-in cookie caching

- Result verification
- Full logging
- Overall and request-level statistics
- Simple protocol for sending statistics to third-party tools
- Proxy request capability
- AKstress process statistics

```
 Uptime:     0 hours 1 minutes 6 seconds
   Number of Threads:     5
   Pages Completed:     1000
   Pages To Be Completed:     1000
   Pages per second:      15.15
   Requests completed:     1000
   Requests per second:      15.15
   Failed Connections:    0
   Incorrect response codes:    0
   Content verification failed:     0
   Request write failures:     0
   Number of early closes:     0
   Number of early server closes:     0
   Number of request write failures:     0
   SSL handshake failures:     0

Request statistics for request /servlet/trade (Rec_Request_dHe8Ea)
   Successes:   1000
   Return Code Failures:   0
   Early Server Closes:   0
   Content Verification Failures:   0
   Min time (milliseconds):   0
   Max time (milliseconds):   1953
   Mean time (milliseconds):   323

akstress - execution complete
```

Figure 55. Sample output of AKstress

9.3 Apache Bench

The IBM HTTP Server (IHS) which is included with WebSphere Application Server does include the Apache Bench (AB) tool on UNIX platforms (Apache Bench is Perl script-based). This tool allows for HTTP client load simulation. You can specify the URL, number of total requests and the number of concurrent requests with this tool. A sample of the output from AB is provided below.

Further information and the source code for Apache Bench are available from the Apache Software Foundation at `http://www.apache.org`.

```
# ab -n 100 -c 10 http://ken/servlet/snoop
This is ApacheBench, Version 1.3
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Copyright (c) 1998-1999 The Apache Group, http://www.apache.org/

Server Software:        IBM_HTTP_Server/1.3.6.2
Server Hostname:        ken
Server Port:            80

Document Path:          /servlet/snoop
Document Length:        2740 bytes

Concurrency Level:      10
Time taken for tests:   1.222 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      315773 bytes
HTML transferred:       298660 bytes
Requests per second:    81.83
Transfer rate:          258.41 kb/s received

Connnection Times (ms)
              min   avg   max
Connect:        0     0    15
Processing:    11   113  1088
Total:         11   113  1103
```

Figure 56. Sample output of Apache Bench

9.4 Rational Suite Performance Studio

Performance Studio offers support for a variety of clients, both Web and non-Web based. Among the clients supported are HTML, DHTML, Document Object Model, Visual Basic, Visual C++, Java, ActiveX and PowerBuilder. Performance Studio records user inputs for playback as scripts that are used in performance testing.

More information on Rational Suite Performance Studio, including an evaluation copy is available from `http://www.rational.com`.

9.5 JMeter

JMeter is another freely available tool from the Apache Software Foundation. JMeter is a Java desktop application designed to test URL behavior and measure performance. Apache JMeter may be used to test server performance both on static and dynamic resources (files or CGI, servlets, Perl scripts). Simple to use, JMeter is limited to 20 concurrent requests per JMeter client, but can be used for initial performance testing.

JMeter is available from the Apache Software Foundation at:
`http://www.apache.org`.

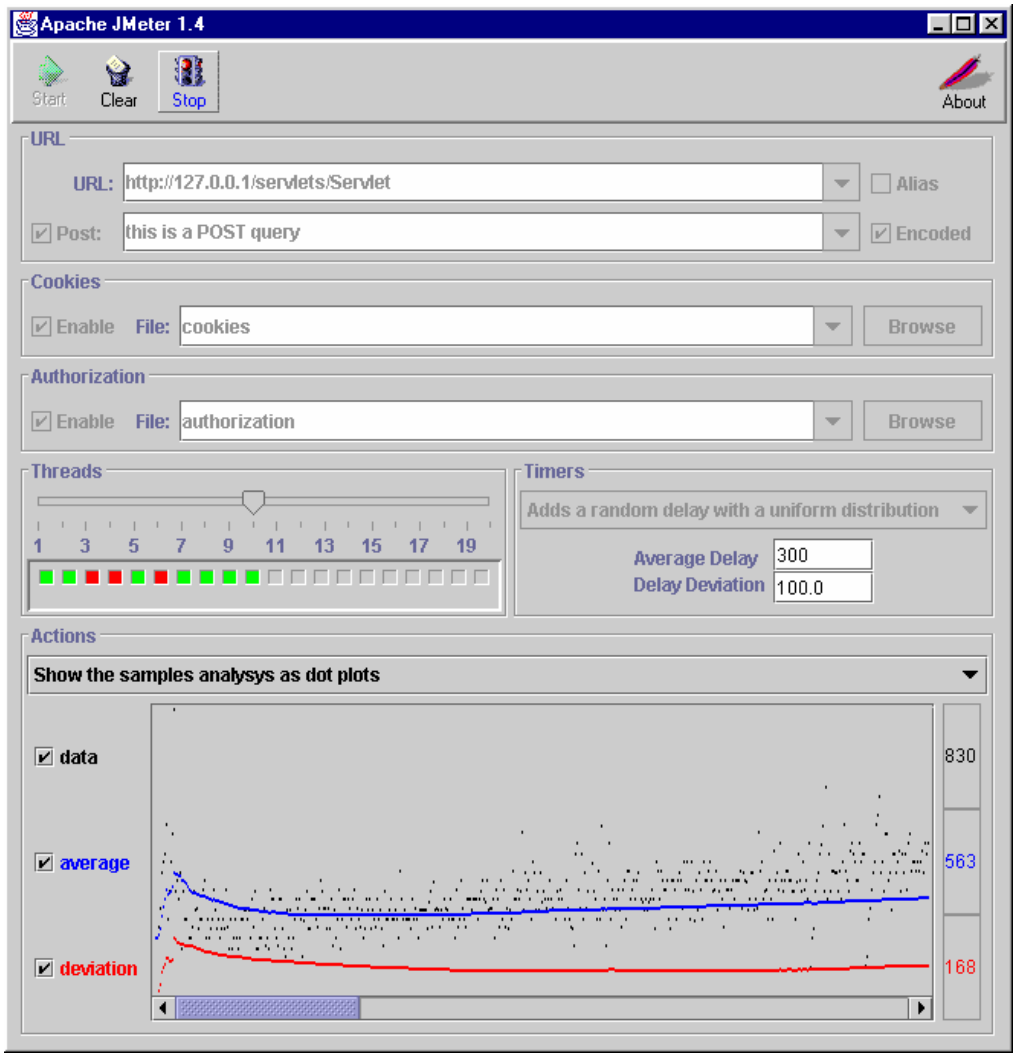


Figure 57. Sample output of JMeter

9.6 WebLoad

WebLoad from Radview provides support for the HTTP 1.0 and 1.1 protocols including cookies, proxies, SSL, keep-alive, and client certificates. Support is also provided for a variety of authentication mechanisms such as basic authentication, proxy, form, NT challenge response, client certificate, and user agent.

Additional information on WebLoad is available from Radview at `http://www.radview.com`.

9.7 LoadRunner

LoadRunner from Mercury Interactive Corporation supports a variety of different clients, including Web, ERP, database (DB), Java or remote terminal emulator (RTE).

Additional information on LoadRunner is available at `http://www.merc-int.com`.

Chapter 10. Monitoring tools

In Version 3 of WebSphere Application Server the Resource Analyzer provides a tool that can be used in conjunction with operating system tools such as `vmstat` to monitor a number of performance measures. For more in depth knowledge on AIX performance tuning, we recommend you to read the *AIX Performance Tuning Guide*, SR28-5930 and *RS/6000 SP System Performance Tuning*, SG24-5340. The Resource Analyzer will be described below in more detail. Another WebSphere Product: Site Analyzer, also comes with WebSphere Application Server and can be used to analyze your HTTP site usage patterns. Site Analyzer will be discussed in Chapter 11, “WebSphere Application Server Site Analyzer” on page 133.

10.1 WebSphere Application Server Resource Analyzer

WebSphere Application Server V3 expands on the summary server execution analysis that is available in V2 of the application server and provides a number of summary and discrete monitoring functions for a variety of resources.

If you intend to monitor performance using the Resource Analyzer in WebSphere you will need to explicitly define your servlets and JSPs as part of your Web application definition. Thus only the items shown below under the trade_app Web application can be monitored by WebSphere.

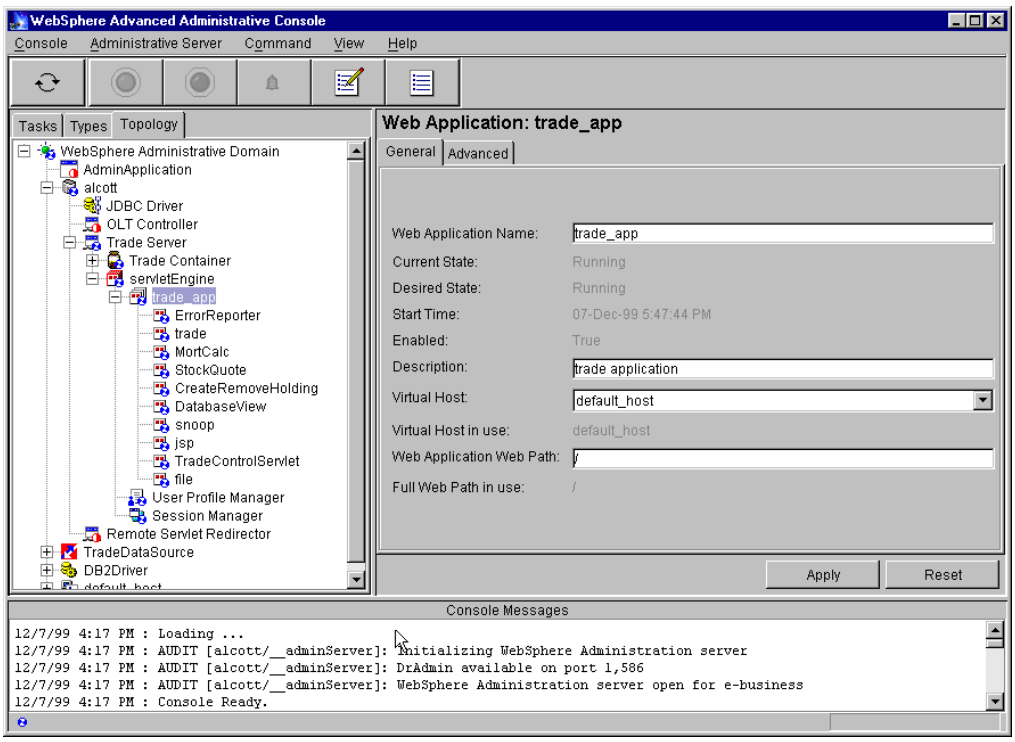


Figure 58. Web Application: trade_app

The WebSphere Resource Analyzer can be accessed from the Tasks tab of the WebSphere Administrative Console: **Task Tab -> Performance -> Resource Analyzer** and can be started once the Resource Analyzer is highlighted either by

clicking the green start button on the top of the pane or by double-clicking the Resource Analyzer as shown in Figure 59.

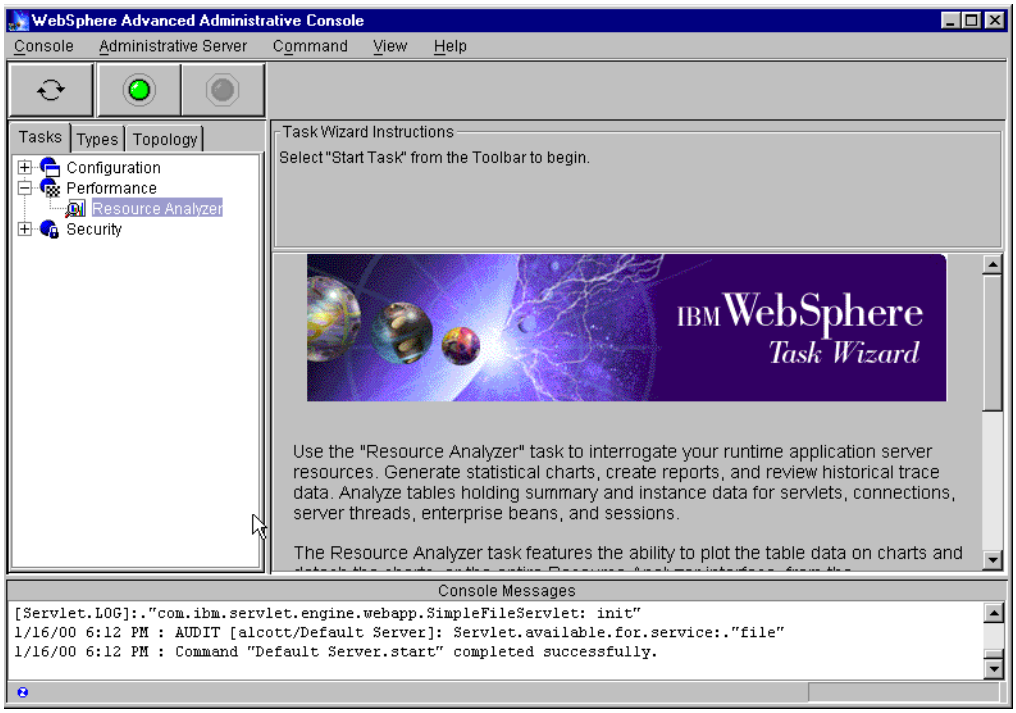


Figure 59. Resource Analyzer

Once started you can choose to either view the window inside the Administration Console or you can detach the task window so that it operates independently. This allows one to perform routine system administration tasks from the Topology tab and monitor performance at the same time. This is shown in Figure 60 on page 91.

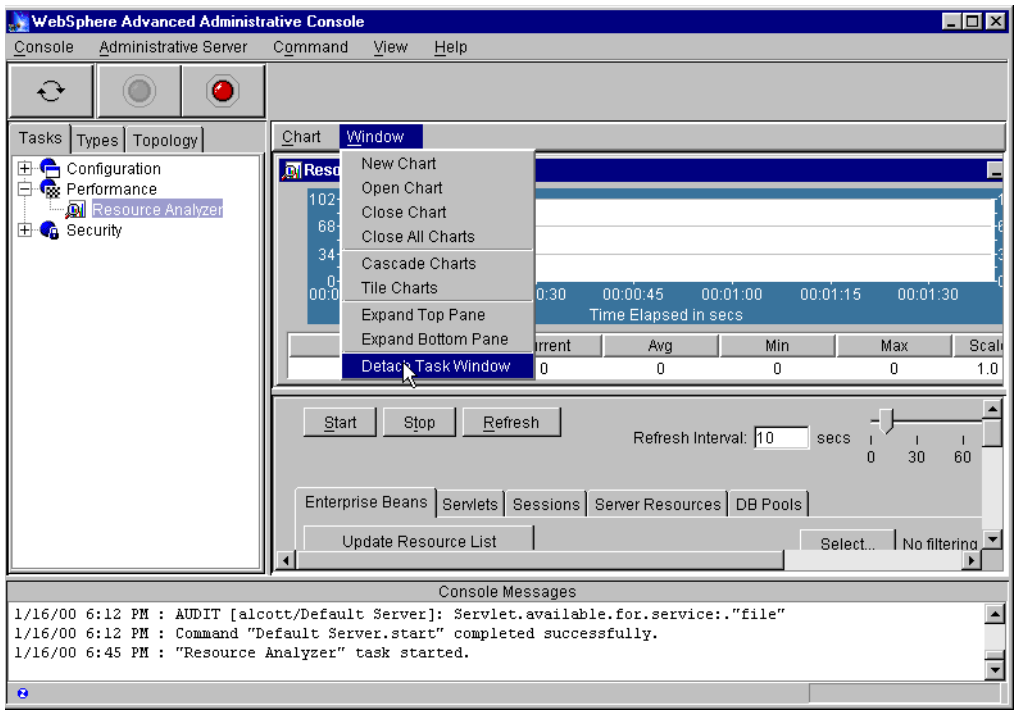


Figure 60. Detach Task Window

10.1.1 Enterprise beans

The first tab on the resource analyzer allows you to monitor execution of your EJBs. Upon clicking the **Select** button in the middle of the page, you'll be prompted with the dialog box shown below that allows you to specify monitoring at:

1. The server level
2. The container level
3. The individual EJB level

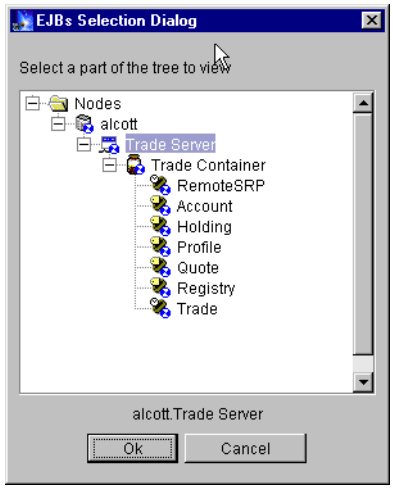


Figure 61. EJBs Selection Dialog

Once you specify the monitoring level, the information below will be displayed for the appropriate enterprise beans. Data specific to each enterprise bean will be displayed as well as aggregate summary data for all enterprise beans as noted below.

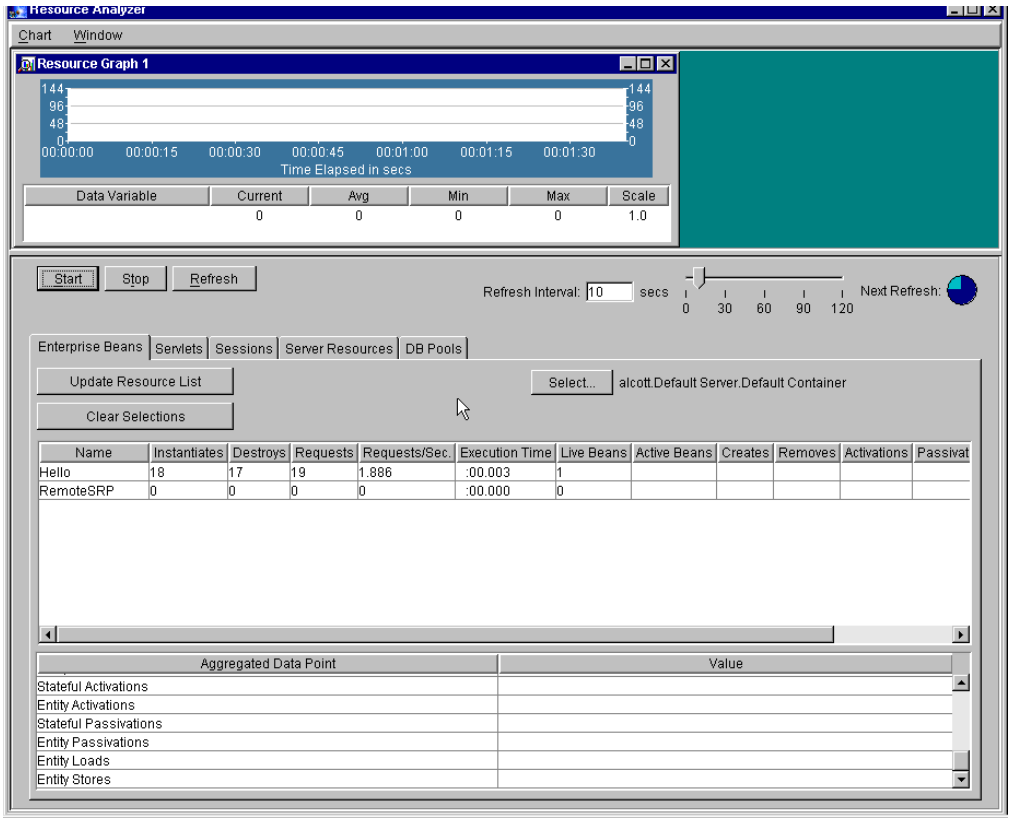


Figure 62. Resource Analyzer: Enterprise Beans

Name

Indicates the enterprise bean being monitored. The table might list two or more enterprise beans with the same name. To distinguish the beans from one another, drag your cursor over a bean to see the container with which the bean is associated. Each bean will be associated with a unique container.

Instantiates

Indicates the number of times the create() method was invoked against the enterprise bean, as measured during the most recently completed polling interval.

Destroys

Indicates the number of times an enterprise bean destroyed one of its enterprise bean instances, as measured during the most recently completed polling interval. The enterprise bean instance might have been destroyed by the remove() method, or by other means.

Requests

Indicates the total number of requests handled by instances of the enterprise bean, as measured during the most recently completed polling interval.

Because an active method can handle one or more requests concurrently, this value will be greater than or equal to the Requests/Sec. value.

Requests/Sec.

Indicates the average number for all instances of all the listed enterprise beans, as measured during the most recently completed polling interval.

Execution Time

Indicates the average number of seconds the enterprise bean has required to process requests from start to finish. The average is taken from the time the administrative server was started to the end of the most recently completed polling interval.

Live Beans

Indicates the number of enterprise bean instances at the server (pooled and active) at this point in time. This value applies to stateless and stateful session beans and entity beans.

Active Beans

Indicates the number of live beans that are active at this point in time. This value applies to stateful session beans and entity beans.

Creates

Indicates the number of times the create() method was invoked against the enterprise bean, as measured during the most recently completed polling interval. This value applies to stateful session beans and entity beans.

Removes

Indicates the number of times an enterprise bean instance was removed, as measured during the most recently completed polling interval. The enterprise bean instance might have been removed by the remove() method, or by other means invoked by the client. This value applies to stateful session beans and entity beans.

Activations

Indicates the number of times the enterprise bean's container activated (retrieved from secondary storage) the enterprise bean, as measured during the most recently completed polling interval. Note, because stateless session beans are never activated, this data point is not applicable to them.

Passivations

Indicates the number of times the enterprise bean's container passivated (transferred to secondary storage) the bean, as measured during the most recently completed polling interval. Note, because stateless session beans are never passivated, they are not included in the total.

Loads

Indicates the number of times a BMP or CMP entity bean loaded information from the database into itself during the polling interval.

Stores

Indicates the number of times an entity bean persisted its state in the database during the polling interval.

10.1.1.1 Aggregated Data Point

Stateless Session Instantiates

Indicates the total number of times the create() method was invoked against the listed stateless session beans, as measured during the most recently completed polling interval.

Stateful Session Instantiates

Indicates the total number of times the create() method was invoked against the listed stateful session beans, as measured during the most recently completed polling interval.

Entity Instantiates

Indicates the number of entity bean objects created by the server during the polling interval.

Stateless Session Destroys

Indicates the total number of times the listed enterprise beans destroyed a stateless session bean instance, as measured during the most recently completed polling interval.

Stateful Session Destroys

Indicates the total number of times the listed enterprise beans destroyed a stateful session bean instance, as measured during the most recently completed polling interval.

Entity Destroys

Indicates the total number of times the listed enterprise beans destroyed an entity bean instance, as measured during the most recently completed polling interval.

Requests

Indicates the total number of requests handled by the enterprise beans, as measured during the most recently completed polling interval.

Requests/Sec.

Indicates the average number of requests per second handled by the enterprise beans, as measured during the most recently completed polling interval.

Live Stateless Session Beans

Indicates the number of stateless session beans at the server (pooled and active).

Live Stateful Session Beans

Indicates the number of stateful session beans at the server (pooled and active).

Live Entity Beans

Indicates the number of entity beans at the server (pooled and active).

Active Stateful Session Beans

Indicates the total number of live stateful session beans that are active. This value will be less than or equal to the number of live stateful session beans.

Active Entity Beans

Indicates the total number of live entity beans that are active.

Stateful Session Creates

Indicates the total number of times the create() method was invoked against the listed stateful session beans, as measured during the most recently completed polling interval.

Entity Creates

Indicates the total number of times the create() method was invoked against the listed entity beans, as measured during the most recently completed polling interval.

Stateful Session Removes

Indicates the number of times stateful session bean instances were removed during the most recently completed polling interval. The enterprise bean instances might have been removed by the remove() method, or by other means invoked by the client.

Entity Removes

Indicates the number of times entity bean instances were removed during the most recently completed polling interval. The enterprise bean instance might have been removed by the remove() method, or by other means invoked by the client.

Stateful Activations

Indicates the number of times the stateful session bean containers activated (retrieved from secondary storage) the enterprise beans, as measured during the most recently completed polling interval.

Entity Activations

Indicates the number of times the enterprise bean containers activated (retrieved from secondary storage) the enterprise beans, as measured during the most recently completed polling interval.

Stateful Passivations

Indicates the number of times the enterprise bean containers passivated (transferred to secondary storage) the enterprise beans, as measured during the most recently completed polling interval.

Entity Passivations

Indicates the number of times the enterprise bean containers passivated (transferred to secondary storage) the enterprise beans, as measured during the most recently completed polling interval.

Entity Loads

Indicates the number of times entity beans loaded information from the database into themselves.

Entity Stores

Indicates the number of times entity beans stored information in the database.

10.1.1.2 How enterprise bean statistics apply to the bean types

Table 6 indicates which enterprise bean statistics apply to each type of enterprise bean.

Table 6. Enterprise bean statistics

Statistic	Stateless session beans	Stateful session beans	Entity beans
Instantiates	YES	YES	YES
Destroys	YES	YES	YES
Requests	YES	YES	YES
Requests per second	YES	YES	YES
Execution time	YES	YES	YES
Live beans (pooled and active)	YES	YES	YES
Active beans		YES	YES
Creates		YES	YES
Removes		YES	YES
Activations		YES	YES
Passivations		YES	YES
Loads			YES
Stores			YES

10.1.2 Servlets

The second tab on the Resource Analyzer allows you to specify which servlet resources you wish to monitor. As with Enterprise Beans you can specify a filtering level, by clicking the **Select** button. You will be prompted with a Servlet Selection Dialog box that allows you to select servlet monitoring at:

1. Server level
2. Servlet Engine level
3. Web Application level
4. By individual servlet

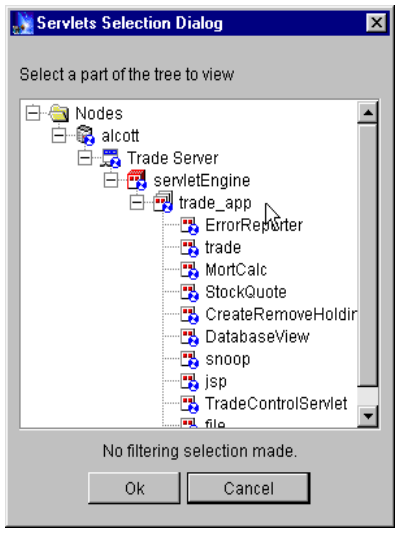


Figure 63. Servlets Selection Dialog

Again there are a number of statistics for each servlet resource as well as aggregate data for all servlet resources as described in Figure 64 on page 97.

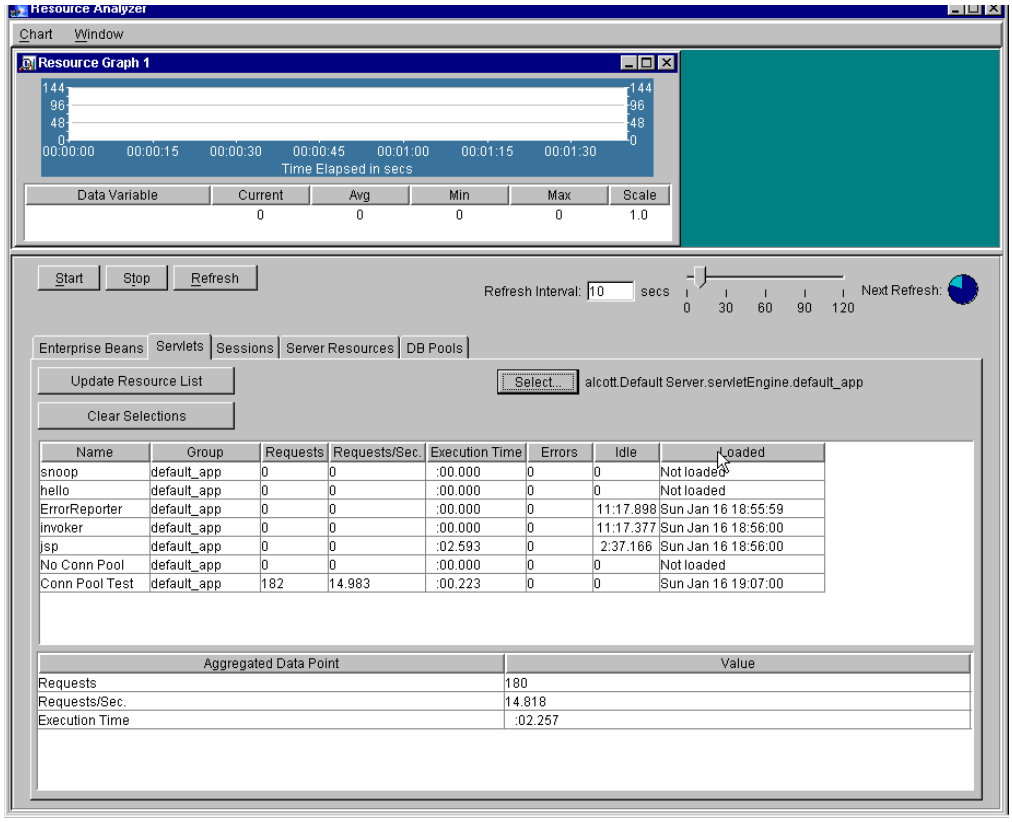


Figure 64. Resource Analyzer: Servlets

Name

Indicates the servlet instance being monitored. If the table lists more than one instance with the same name and you need to tell the instances apart, drag the cursor over an instance to see the servlet group with which the instance is associated.

Group

Indicates the servlet group (Web application) to which this servlet belongs. If you are monitoring servlets from multiple Web applications, this value helps you distinguish the servlets from one another.

Requests

Indicates the number of requests this instance of the servlet handled during the most recently completed polling interval.

Requests/Sec.

Indicates the average number of requests per second for the servlet instance, as measured during the most recent polling interval.

Execution Time

Indicates the average time the servlet instance spent performing services such as service(), doGet(), or doPost(), measured from the time the administrative server was started, to the end of the most recently completed polling interval.

Errors

Indicates the number of errors associated with this servlet instance, measured from the time the administrative server was started, to the end of the most recently completed polling interval.

Idle

Indicates the number of seconds the servlet instance has been idle, if the servlet is currently idle. If the servlet instance is active, this value will be 0.

Loaded

Indicates the time at which the class file for the servlet instance was most recently loaded. The value is 0 if the servlet class is not currently loaded.

10.1.2.1 The Aggregated Data Point

Requests

Indicates the total number of requests all servlets handled during the most recently completed polling interval.

Requests/Second

Indicates the average number of requests per second for all servlets during the most recently completed polling interval.

Execution Time

Indicates the average time the servlet spent during individual executions. It is the total time spent executing, divided by the number of executions of the servlet.

The average is measured from the time the administrative server was started, to the end of the most recently completed polling interval.

10.1.3 Sessions

The third tab on the Resource Analyzer is for monitoring of session resources. Clicking the **Select** button results in a selection dialog, which allows you to select the application server for which you wish to monitor session resources. This is shown in Figure 65 on page 99.

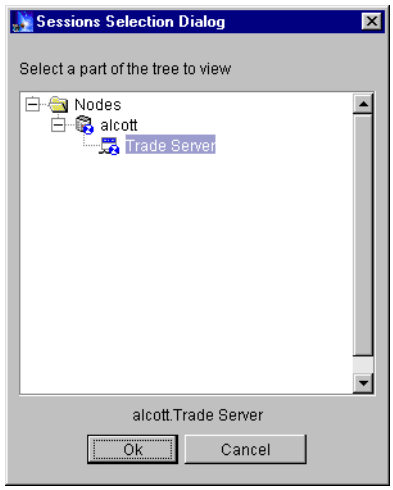


Figure 65. Sessions Selection Dialog

Unlike other resources such as EJBs and servlets, there are no statistics available for individual sessions, only aggregate data at the server level.

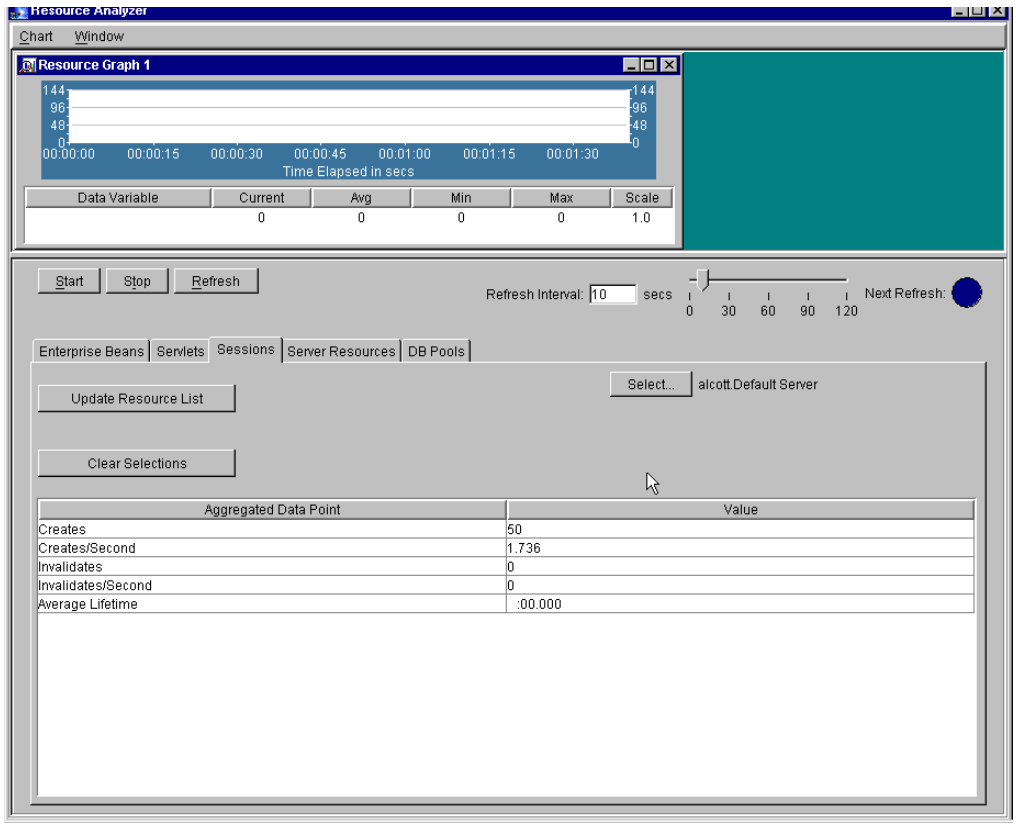


Figure 66. Resource Analyzer: Sessions

Creates

Indicates the number of sessions created during the most recently completed polling interval. This includes sessions that were available, active, or invalidated.

Creates/Second

Indicates the average number of sessions created per second during the most recently completed polling interval.

Invalidates

Indicates the number of sessions invalidated (removed from available or active state) during the most recently completed polling interval. This value is valid only in terms of in-memory sessions, not persistent sessions.

Invalidates/Second

Indicates the average number of sessions invalidated (removed from available or active state) per second during the most recently completed polling interval. This value is valid only in terms of in-memory sessions, not persistent sessions.

Average Lifetime

Indicates the average number of seconds sessions have remained alive in memory. The average is measured from the time the administrative server was started, ending with the most recently completed polling interval. The value is

valid only in terms of in-memory sessions, not persistent sessions. Currently valid sessions are not included in the calculation because their life spans are not yet known.

10.1.4 System Resources

The System Resources tab allows you to monitor Java Virtual Machine (JVM) resource use for each JVM in use. Recall that in WebSphere V3 each application server process is a separate JVM. Clicking the **Select** button results in a dialog that prompts you to select the application server (JVM) that you wish to monitor. This is shown in Figure 67.

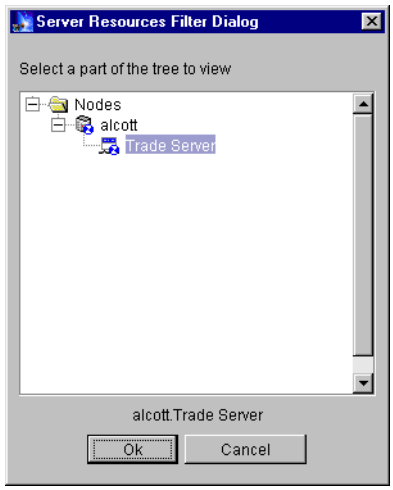


Figure 67. Server Resources Filter Dialog

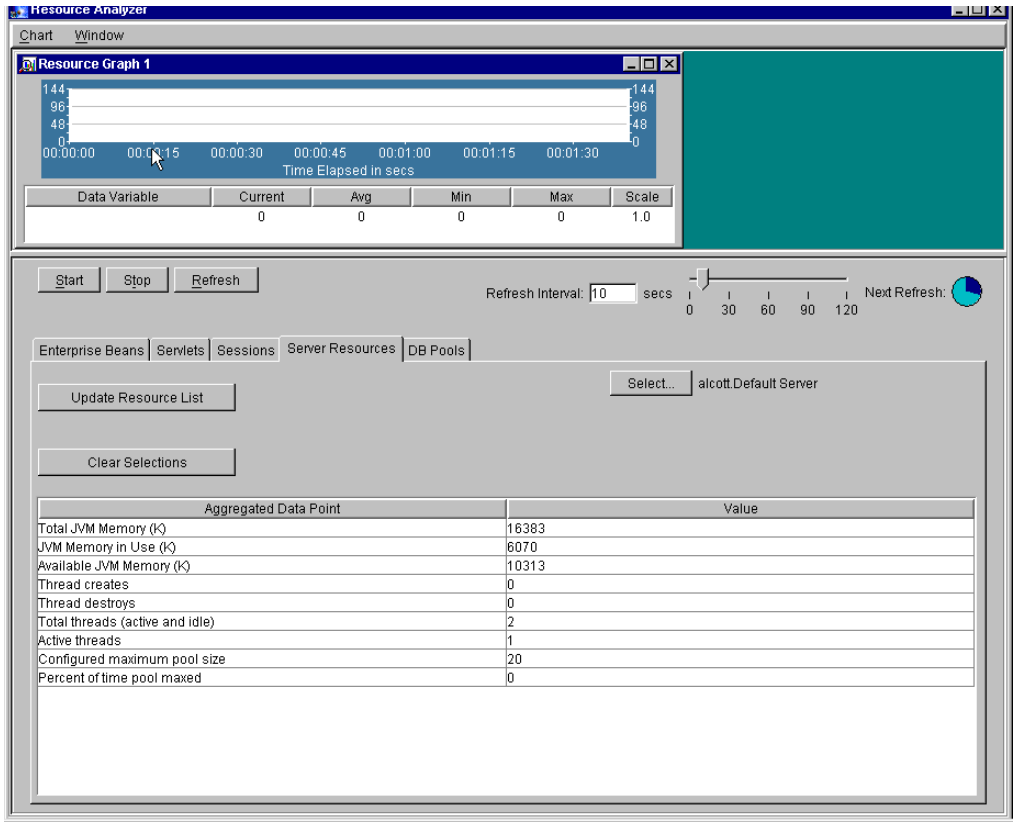


Figure 68. Resource Analyzer: System Resources

Total JVM Memory (K)

Indicates the total memory, in kilobytes, available to the Java Virtual Machine during the most recently completed polling interval. This value is the JVM heap size. This value will not necessarily match values reported on the Performance tab of the Windows Task Manager because this does not include the native component of the process.

JVM Memory in Use (K)

Indicates the memory in use by the Java Virtual Machine during the most recently completed polling interval. The value is in kilobytes.

Available JVM Memory (K)

Indicates the memory available (in kilobytes) to the Java Virtual Machine during the most recently completed polling interval. This is the difference between the total JVM memory and the memory in use. Because the Java virtual machine can increase the virtual memory in the storage heap as needed, this value can fluctuate.

10.1.4.1 Thread pool properties

The remaining properties on the Server Resources tabbed page of the Resource Analyzer correspond to the thread pools maintained by application servers. Each application server has its own thread pool or cache from which it uses threads to process remote method invocations.

The size of a server's thread pool varies throughout the server's lifetime. Threads are created when needed and destroyed when there are too many idle threads.

Thread creates

Indicates the number of threads created during the interval.

Thread destroys

Indicates the number of threads destroyed during the interval.

Total threads (active and idle)

Indicates the total number of threads that are currently either idle or active.

Active threads

Indicates how many threads are currently active.

Configured maximum pool size

Indicates the maximum size of the thread pool.

Percent of time pool maxed

Indicates how often the pool was "maxed out" during the most recently completed polling interval. The thread pool is considered "maxed out" when all of its threads are in use.

If this number is high, consider increasing the number of threads allocated to the server.

10.1.5 DB pools

The final tab on the Resource Analyzer allows you to monitor use of database pools. The Select button allows you to specify the application server for which you wish to monitor database pool resources. The selection dialog is shown in Figure 69.

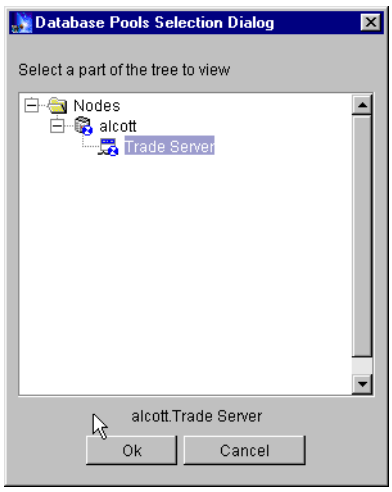


Figure 69. Database Pools Selection Dialog

Each table row on the DB Pools tabbed page represents one pool. Each row lists the data source and the user for which the pool maintains connections.

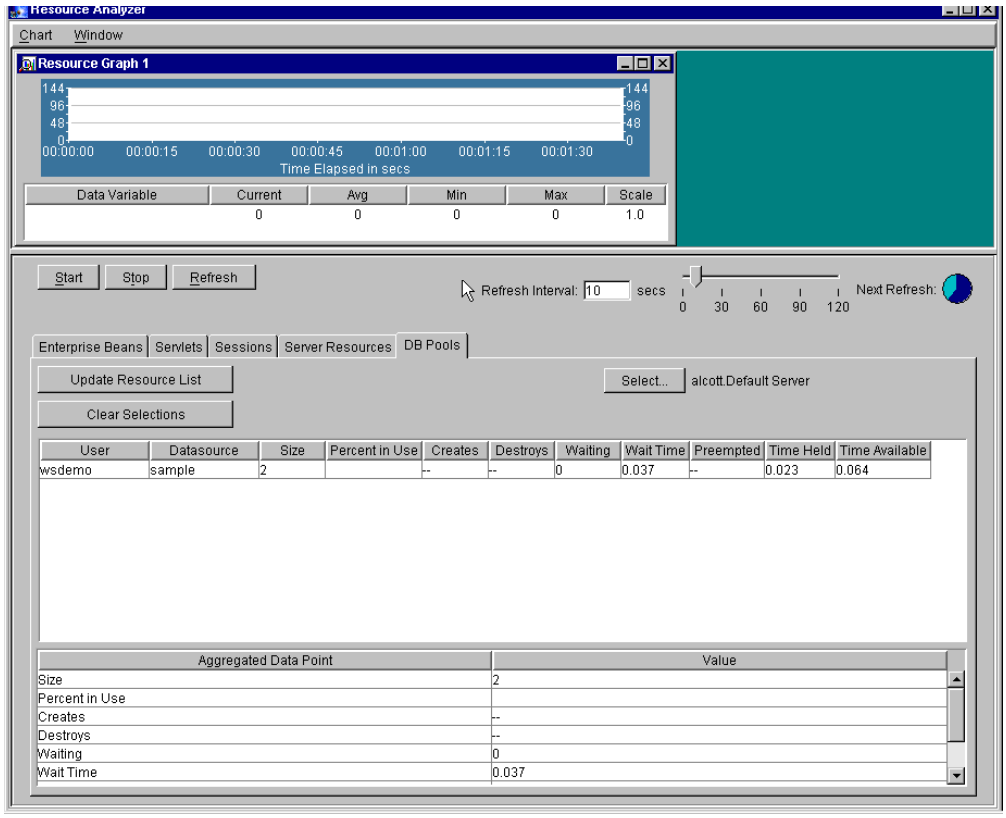


Figure 70. Resource Analyzer: DB Pools

User

Indicates the ID used to log into the database to which this pool maintains connections.

Datasource

Indicates the datasource object associated with the pool. A datasource object keeps a connection pool from which database connections can be borrowed.

Size

Indicates the average pool size, as measured from the time the administrative server was started, ending with the most recently completed polling interval.

Percent in Use

Indicates the percentage of the connections that are currently unavailable for servicing requests. The connections might be orphaned, or already servicing requests.

Creates

Indicates the number of connections created during the most recently completed polling interval.

Destroys

Indicates the number of connections destroyed by this pool during the most recently completed polling interval. For example, an orphaned connection

might be destroyed because its owning client has died or is otherwise unresponsive.

Waiting

Indicates the number of threads currently waiting for connections from this pool.

Wait Time

Indicates the average number of seconds threads must wait to gain connections from the pool, and whether or not the pool has available connections when requests are made. The average is taken from the time the administrative server was started, to the end of the most recently completed polling interval.

Preempted

Indicates the number of times a connection was returned to the pool before a request was finished using it. This might happen as a result of time outs on the client side, or for other reasons. A high value indicates a possible problem.

Time Held

Indicates the average number of seconds requests hold connections from this pool, measured from the time the administrative server was started, to the end of the most recently completed polling interval.

Time Available

Indicates the number of seconds the pool had one or more available connections, measured from the time the administrative server was started, to the end of the most recently completed polling interval.

10.1.5.1 The Aggregated Data Point

Size

Indicates the average pool size, as measured from the time the administrative server was started, and ending with the most recently completed polling interval.

Percent in Use

Indicates the percentage of the connections (in all the pools) that are currently unavailable for servicing requests. The connections might be orphaned, or already servicing requests.

Creates

Indicates the total number of connections (for all pools) created during the most recently completed polling interval.

Destroys

Indicates the number of connections destroyed by pools during the most recently completed polling interval. For example, an orphaned connection might be destroyed because its owning client has died or is otherwise unresponsive.

Waiting

Indicates the number of threads currently waiting for connections from the pools.

Wait Time

Indicates the average number of seconds threads must wait to gain connections from the pools, and whether or not the pools have available connections when requests are made. The average is taken from the time the administrative server was started, to the end of the most recently completed polling interval.

Preempted

Indicates the number of times connections were returned to the pool before requests were finished using them. This might happen as a result of timeouts on the client side, or for other reasons. A high value indicates a possible problem.

Time Held

Indicates the average number of seconds requests hold connections from the pools, measured from the time the administrative server was started, to the end of the most recently completed polling interval.

Time Available

Indicates the number of seconds the pools had one or more available connections, measured from the time the administrative server was started, to the end of the most recently completed polling interval.

10.2 AIX performance tools

In AIX a wide variety of tools are available to first identify and understand the work load, and then to help set up the system environment so that it is as close as possible to the ideal execution environment for the work.

Refer to Figure 43 on page 154 for an overview of the tuning commands and how they affect the various subsystems.

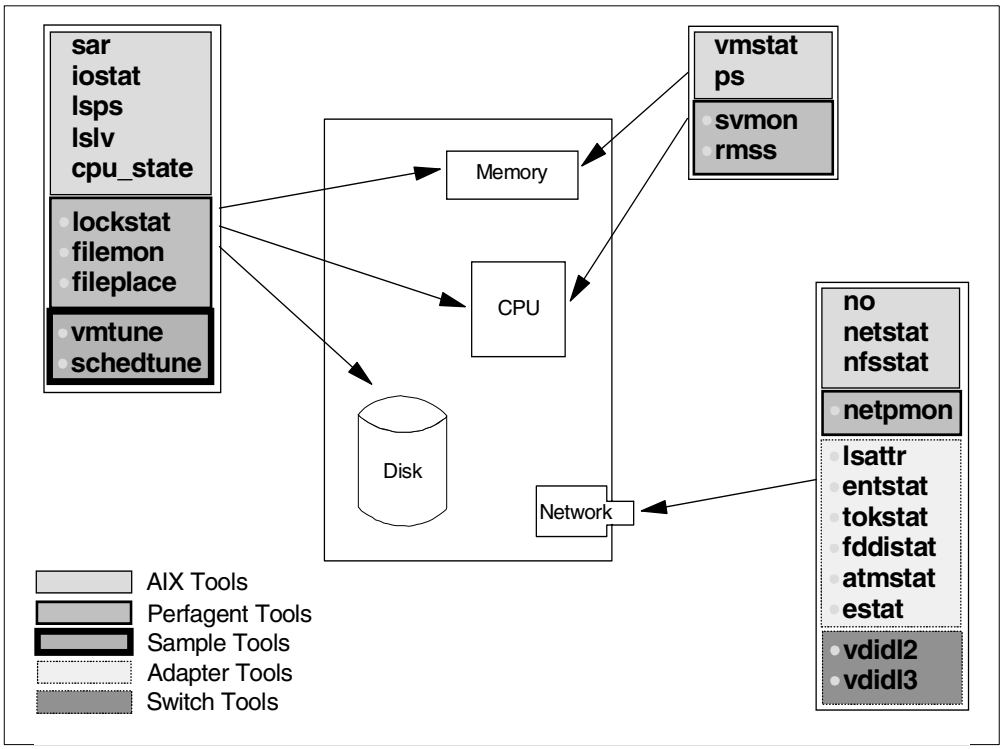


Figure 71. System tuning overview

10.3 Managing memory resources

Memory is a valuable and critical resource. Insufficient memory, or poor use of memory, results in serious performance problems.

The tools in this section are used to identify memory related performance problems, and to correct or minimize these.

10.3.1 Monitoring memory with vmstat

The `vmstat` command provides statistical information collected by AIX for the Virtual Memory Manager (VMM), Central Processing Unit (CPU), and process scheduler.

In this section we review aspects of this command that are associated with the VMM.

Use this command during periods when the system workload is representative of the system's expected workload. In some cases a system has several workload

patterns. It is important to gain an understanding of memory utilization during these periods.

The output shown in Figure 72 was captured using vmstat 5 10. Using this command we monitored the changing memory usage every five seconds for a total of 50 seconds. The first line is statistical information, collected by AIX when the system was last booted.

```
# vmstat 5 10
kthr     memory             page              faults        cpu
----- ------------ ----------------------- ------------ -----------
r  b    avm   fre  re  pi  po  fr   sr cy  in   sy   cs us sy id wa
0  0 219963   201   0   0   0  14   50  0 138  457   65 13 10 59 18
2  2 217167  3002   0   0   0  25   34  0 992 1638  238 30  8 49 13
3  2 221754   228   0   0   1 402 1736  0 959 1198  217 31  8 56  5
3  2 216648  5034   0   0   0  65   84  0 943 1383  194 31  7 60  1
3  2 221020   536   0   0   0   0    0  0 948 2697  181 31 17 52  0
3  2 222456   437   0   0   0 311  406  0 953 1730  187 35 12 52  0
3  2 214221  8633   0   0   0  13   19  0 960 2955  224 25 14 58  3
4  2 216900  5887   0   3   0   0    0  0 969 19824 381 31 17 48  4
3  2 216366  5902   0   0   0   0    0  0 949 29553 193 30  9 59  2
3  2 216054  4472   0   0   0   0    0  0 973 2570  274 26 11 46 17
```

Figure 72. vmstat output

If the fre column (number of pages in the free list) is low (below 2 * MB of real memory - 8), and the pi column (page in rate/s) exceeds 5 per second, memory is overcommitted.

A high page scan (sr) to page steal (fr) ratio indicates that the memory subsystem is overactive. The higher this ratio, the more time the VMM is spending searching for available memory to allocate. Further investigation of this should be undertaken.

Other column headings are as follows:

- r - the number of runnable processes during the interval
- b- the number of processes blocked waiting for high-speed I/O
- avm - active virtual memory in units of 4 KB pages
- fre - size of the list of free RAM pages
- pi - the number of pages per second that have been paged
- po - the number of pages per second that have been paged out since the last report
- fr - page steal rate
- cs- average context switches per second (an idle system will average 20 to 60 per second)
- us - user mode cpu average
- sy -system mode cpu average
- id - idle time average
- wa - disk I/O wait average

In AIX Version 4 and onward, the page reclaim column (re) is always 0. Page reclaims (a page that is released by the VMM and then reclaimed by the same process before allocation to another process) are no longer recorded.

10.3.2 Monitoring memory with sar

This command reports the values of the operating system activity counters, which are a quick and easy way to check on the system work.

Information for the VMM is shown by the paging activity counters.

Figure 73 on page 109 shows a system that is lightly loaded. The VMM has no difficulty fulfilling page requests. On average, 203 page faults per second were generated. The VMM maintained a large number of free memory slots throughout the time monitored. It was not necessary for the VMM to cycle through memory searching for free pages, and very little paging I/O was required.

If this system's performance were considered, we would conclude that memory is not a contributing factor to any problems.

In Figure 73 there is a 3-second peek in paging. This indicates an uneven workload distribution. In a heavily loaded system these peeks need to be investigated. A multiuser environment needs a balanced workload to maintain a consistent response time for users. In our experience users do not remember 999 subsecond responses. They remember the one that took 5 seconds.

```
# sar -r 1 10

AIX sp3en0 2 4 004008966700    09/29/98

00:25:01   slots cycle/s fault/s  odio/s
00:25:02   63326    0.00  248.18    1.82
00:25:03   63326    0.00    0.88    0.00
00:25:04   63325    0.00    0.91    0.00
00:25:05   63325    0.00    0.00    0.00
00:25:07   63529    0.00  212.73    3.64
00:25:08   63446    0.00  941.46    8.13
00:25:09   63333    0.00  526.32    4.39
00:25:10   63333    0.00    0.00    0.00
00:25:11   63333    0.00    0.00    0.00
00:25:12   63333    0.00    0.00    0.00

Average    63361       0     203       2
```

Figure 73. Monitoring paging with sar

10.3.3 Monitoring memory with lsps

This command provides information about the paging space. Use it to check how much virtual memory is used. It is useful to know how much real memory is extended when considering a memory upgrade. If there is heavy paging, and memory is only extended by a small amount, additional memory is a very cost-effective upgrade.

Figure 74 shows an example of the paging area for our system. We see that real memory is currently extended by 110 MB.

The system paging area is spread across two drives. Paging to disk is occurring evenly across the two disks.

```
lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
paging01    hdisk2            rootvg         144MB      38     yes   yes    lv
paging00    hdisk1            rootvg         128MB      38     yes   yes    lv
paging00    hdisk2            rootvg          16MB      38     yes   yes    lv
```

Figure 74. Viewing paging space

10.3.4 Monitoring memory with ps

This command lists the current processes and their status. By examining the processes, we obtain an overview of how much memory each process uses. We also obtain information of the VMM overhead for each process.

This command takes a snapshot of the system showing a set of statistics for each process.

Figure 75 is the output of `ps gvc` on our system. This example has been reduced to 10 lines.

```
# ps gvc
  PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
  31124      - A     0:01 1058  1332    96 32768    35    84  0.0  0.0 db2sysc
  31474 pts/13 A     0:05  704   516    52 32768   295    40  0.0  0.0 tcsh
  33146      - A     0:00    0   112   232 32768    49    80  0.0  0.0 xlC_r
 267064      ? A     0:00  185   384    12 32768   261     0  0.0  0.0 aixterm
 268156      - A     0:10    0 15116 17572 32768  2862  2408  4.0  2.0 xlCentry
 273814      - A     0:10   92  1036   148 32768   274   104  0.0  0.0 db2bp_s
 274866  pts/3 A     0:00    0   112   232 32768    49    80  0.0  0.0 xlC_r
 306432      - A     0:21 1507  1268   224 32768   485    16  0.0  0.0 dtwm
 306722 pts/19 A     0:00    3   176   232 32768   195   220  0.0  0.0 ksh
 307380  pts/3 A     0:10    0 12928 15384 32768  2862  2408  9.6  2.0 xlCentry
```

Figure 75. ps gvc output

When using this command to review memory usage, examine:

- `PGIN`: Number of memory frames paged in
- `%MEM`: Percentage of system memory used

10.3.5 Monitoring memory with svmon

This command shows the current state and usage of memory.

With `svmon` memory page use can be viewed from the:

- System level
- Process level
- Segment level

10.3.5.1 Total memory usage

Always obtain an overview of memory usage from the system level. A performance problem caused by memory contention is unlikely when the system has sufficient memory available.

Figure 76 is the output of `svmon -G` on our system.

```
# svmon -G
               memory          in use        pin      pg space
size  inuse free pin   work  pers  clnt  work pers clnt  size   inuse
65536 62724 2812 3508  41482 21242 0     3347 161  0     131072   25555
```

Figure 76. Global memory view

Note: The values are displayed as memory pages. A memory page is 4069 bytes.

Interpretation of the `svmon` report:

- `memory`: System memory usage

  `size`: Total size of real memory

  `inuse`: Amount of memory in use

  `free`: Amount of free memory

  `pin`: Pinned memory (memory pages that cannot be swapped out)
- `inuse`: Expands the column memory `in use`

  `work`: The system working set (data and stack regions)

  `pers`: Pages that are persistent on file

  `clnt`: Client allocated memory (network clients)
- `pin`: Expands the column `pin`

  (Refer to preceding description of in use columns.)
- `pg space`: Size of the paging space

  `size`: Size of paging area

  `inuse`: Amount of page space in use (size of real memory extension)

10.3.5.2 Process memory usage

When memory has been identified as a performance issue, isolating the cause requires details on how memory is used by the processes.

Using `ps gvc | grep app` (refer to 10.3.4, “Monitoring memory with ps” on page 110 for more detail on the use of this command), we identified a process running `app` (we are using `app` to represent an application). The process ID was 51994.

Figure 77 is an example of using `svmon` to view a process’s memory usage.

```
# svmon -P 51994
  Pid                          Command        Inuse        Pin      Pgspace
51994                             find         4450        479          968

Pid:  51994
Command:  app

Segid  Type  Description          Inuse    Pin  Pgspace  Address Range
380e  work  sreg[5]               1069    478      959  0..65535
48d0  work  lib data                15      0        0  0..358
4411  work  shared library text    3260      0        9  0..65535
6b7e  work  private                  98      1        0  0..43 : 65304..65535
62d8  pers  code,/dev/hd2:49266       8      0        0  0..7
```

Figure 77. Process memory view

The app process has 4450 pages of memory allocated. A shared library which this process uses accounts for 3260 of these memory pages. Every additional copy of app therefore requires an additional 1190 pages of memory:

Number of memory pages used minus the number of memory pages shared.

Further investigations of other processes executing on this system would determine whether the shared library is used by other applications. Using this information we could determine whether the memory overhead for the shared libraries should be considered a system overhead or an application overhead.

Using svmon, memory usage can be classified as:

- System
- Application shared
- Application
- Application instance

This information is important for:

- Assessing a system's memory requirements
- Evaluating application mixes for nodes
- Identifying candidate applications to be shifted

The Address Range column shows where in the allocated memory segment the process is referenced (the process's memory footprint). This allows determination of how much free physical memory is required for the application to prevent paging.

10.4 Managing CPU resources

It is difficult to draw a line when reviewing CPU utilization. "If the percentage of CPU utilization exceeds x we have a system constrained by CPU" is often a wild stab in the dark to provide a definition.

10.4.1 Monitoring the CPU with vmstat

We have already reviewed this command in 10.3.1, “Monitoring memory with vmstat” on page 107.

`vmstat` is used to monitor CPU usage. It provides a single line report, which shows a quick status of CPU utilization.

Figure 78 on page 113 is an example of using `vmstat 5` on a system with free CPU resources. While monitoring this system, a peak in the workload occurs.

```
# vmstat 5
kthr     memory             page              faults        cpu
----- ------------ ---------------------- ------------ -----------
r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
0  0 25565  5251   0   0   0   0    1   0 131  365  64  1  2 96  1
6  0 41263   124   0   0   0  56   56   0 724 2564 450 41 24 34  0
3  0 40600  1374   0   0   0  97  114   0 746 2097 393 40 21 38  0
3  0 40769  1212   0   0   0   0    0   0 698 2142 389 40 24 36  1
6  0 41019   871   0   0   0   0    0   0 719 2707 493 48 22 27  3
0  0 41301   529   0   0   0   0    0   0 610 1439 304 25 10 64  0
2  0 41517   479   0   0   0  64   71   0 628 2588 421 47 23 23  7
6  0 43293   147   0   0   0 360  368   0 691 3429 407 74 26  0  0
8  1 43636   190   0   0   0  86  114   0 724 3899 508 67 33  0  0
6  0 44093   175   0   0   0 108  109   0 689 2807 378 71 29  0  0
7  0 44416   415   0   0   0 120  120   0 721 3676 448 67 33  0  0
7  0 44169   853   0   0   0  25   25   0 693 2927 403 74 26  0  0
6  0 43075  2363   0   0   0   0    0   0 653 2907 436 85 14  0  0
2  0 42395  3087   0   0   0   0    0   0 732 3413 432 80 17  2  1
3  0 42635  2764   0   0   0   0    0   0 718 3020 415 59 19 19  3
2  0 42748  2647   0   0   0   0    0   0 752 2704 432 55 16 29  0
1  0 41626  4092   0   0   0   0    0   0 748 2506 452 52 17 21 10
0  0 41228  4610   0   0   0   0    0   0 730 2806 450 54 17 26  3
3  0 40773  5216   0   0   0   0    0   0 673 2782 445 67 21 12  0
```

Figure 78. CPU monitoring with vmstat output

Interpreting system activity from this vmstat report:

1. The CPU is lightly utilized:

   Idle average 40% (id)
   Users average 40% (us)
   System average 20% (sy)
   Low I/O wait (wa)

   Conclusion: CPU utilization of 40% is not due to an I/O bound system.

2. The system workload peaks:

   Idle average 0%
   User average 75%
   System average 25%
   Low I/O wait
   Free list low (fre)
   High number of page scans (sr)
   High number of page steals (fr)

Conclusion: VMM is searching for memory to replenish the free list. This overhead is constraining the user application CPU requirement.

3. Workload constrained by CPU:

   Idle average 0%
   User average 85%
   System average 15%
   No I/O wait
   Free memory available
   No paging

   Conclusion: The system is CPU bound. Users' CPU utilization increased as system CPU utilization decreased. This confirms the above conclusion that VMM constrained the user CPU utilization.

4. The system workload drops:

   Idle average 20%
   User average 50%
   System average 20%
   I/O wait 10%

   Conclusion: The workload peak is over and a backlog of work is being cleared in the I/O subsystem.

The peak in workload was a few lines on this report, each line representing 5 real-time seconds. The peak in workload therefore lasted 40 seconds. Once completed, the system response was still degraded while a backlog of work was cleared.

Further investigation of this will be required. Identify the program/application and determine its loading pattern (when and how often is this application executed).

10.4.2 Monitoring the CPU with sar

In 10.3.2, "Monitoring memory with sar" on page 109 we discussed using this command to check paging statistics.

This command provides a simple method to review CPU utilization.

10.4.2.1 CPU utilization report

To obtain an overview of CPU utilization, that is, how much time is spent doing actual work versus the time spent being idle or waiting on I/O, use the default `sar` options.

Figure 79 on page 115 is an example of using `sar` to report CPU utilization. The report is for a system where the CPU is a limiting factor and is 100% utilized.

```
# sar 1 10

AIX sp3en0 3 4 000081007000    09/28/98

07:48:32    %usr    %sys    %wio   %idle
07:48:33      77      23       0       0
07:48:34      82      18       0       0
07:48:35      81      19       0       0
07:48:36      79      21       0       0
07:48:37      81      19       0       0
07:48:38      80      20       0       0
07:48:39      79      21       0       0
07:48:40      81      19       0       0
07:48:41      77      23       0       0
07:48:42      67      33       0       0

Average       78      22       0       0
```

Figure 79. CPU Utilization report using sar

10.4.3 Monitoring the CPU with time

The `time` command gives CPU and real-time execution figures for commands and applications.

Use this command to determine an application's CPU utilization. A good example of when it should be used is given in 10.4.1, "Monitoring the CPU with vmstat" on page 113. An application was executed which used substantial CPU resources. `time` can be used to quantify the CPU requirement.

Figure 80 shows the result of using `time` to measure the CPU resource requirement to execute our application apprep. This is an unobtrusive method of collecting CPU utilization for applications.

```
# time apprep

real  3m12.76s
user  2m38.72s
sys   0m26.54s
```

Figure 80. Checking CPU utilization with time

10.4.4 Checking active CPUs using cpu_state

This command sets and displays which processors will be active when the system is restarted. It is only available for SMP RS/6000 models.

With `cpu_state`, processors are selectively enabled or disabled. The changes take effect when the node is rebooted.

Generally, disabling processors reduces performance. We do not recommend disabling CPUs on production systems.

When assessing performance of applications, there are situations when comparisons between an SMP and a uniprocessor are required, for example to determine the performance benefits of an SMP node.

We used this feature to perform low-level application tracing. With multiple CPUs it was difficult to paste execution streams into a sequenced execution flow.

Figure 81 is a report from an 8-way SMP machine after 7 of the 8 processors were disabled.

```
# cpu_state -l
Name    Cpu      Status         Location
proc0   0       enabled         00-0P-00-00
proc1   1       disabled        00-0P-00-01
proc2   2       disabled        00-0Q-00-00
proc3   3       disabled        00-0Q-00-01
proc4   4       disabled        00-0R-00-00
proc5   5       disabled        00-0R-00-01
proc6   6       disabled        00-0S-00-00
proc7   7       disabled        00-0S-00-01
```

Figure 81. Active CPU report

10.5 Managing network resources

Network communication is a combination of hardware and complex software. It is important that network parameters are set appropriately.

10.5.1 Monitoring the network with netstat

This command is used to assess the network load and the reliability of the network. It is traditionally used for problem determination. We have found this command to be useful for determining the network load, and whether the network is congested.

10.5.1.1 Determine the proportion of network traffic for an adapter

Use `netstat -I` to compare the network traffic on a selected adapter with the total volume of network traffic.

Figure 82 is an example of a `netstat -I` report. This report was produced while a large file was transferred using ftp (a file transfer application) between two nodes. The high-speed switch css0 was chosen to do this transfer.

```
# netstat -I css0 1
   input    (css0)      output            input    (Total)     output
 packets  errs  packets  errs colls  packets  errs  packets  errs colls
  125696     0   110803     0     0   356878     0   287880     0     0
     119     0      216     0     0      123     0      221     0     0
     117     0      222     0     0      120     0      224     0     0
     115     0      225     0     0      117     0      227     0     0
     115     0      202     0     0      117     0      204     0     0
     115     0      207     0     0      117     0      209     0     0
     116     0      201     0     0      118     0      203     0     0
     115     0      211     0     0      118     0      213     0     0
```

Figure 82. Viewing the network load using netstat

The report is divided into 5 columns for the adapter, and 5 columns for the total network utilization:

1. Incoming packets.
2. Number of error packets received. An error causes a retransmission of the packet.
3. Number of packets sent.
4. Number of error packets sent (the number of packets which were re-requested because of a transmission error).
5. Number of collisions (colls). When information is sent using a network adapter, a component of Transmission Control Protocol (TCP) is a collision detection algorithm. The algorithm works as follows:
    1. Before sending, check that the network connection is not in use.
    2. When the network connection is free, begin writing the packet onto the network.
    3. Check that in the very short amount of time between determining that the network connection was free and beginning to write the packet, no other network adapter also started to write a packet. When this happens, it is called a network collision.
    4. The adapters abort the write operation.
    5. The adapters wait for a random period of time before trying again.

Collisions affect network throughput; the adapters involved are forced to spend time waiting, and network bandwidth is wasted when the collision occurs.

A few collisions will happen in a network from time to time, and are acceptable. If a lot of collisions are detected, it indicates the network is overloaded.

Notes:

1. Adapters involved in a network collision can be in the same host or in different hosts.
2. Packet numbers do not directly translate into network utilization, because packet sizes vary. Packet numbers are only a guide to network load.

10.5.1.2 Network memory buffer allocation statistics

If memory buffer allocation requests fail, the request is lost and more memory needs to be allocated to the network memory pool.

Figure 83 is an example of using `netstat -m` to check the network memory buffer allocation statistics. Check the column `failed` to ensure that the communication subsystem has sufficient memory allocated.

```
# netstat -m

Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed    free   hiwat   freed
32              232       442      0      24     640       0
64              168       227      0      24     320       0
128             113       214      0      15     160       0
256             153    109491      0     327     384       2
512             240      4406      0       8      40       0
1024             54      2200      0       2     100       0
2048              1       400      0      99     100     143
4096             66       154      0     107     120       9
8192              4         7      0       0      10       0
16384             1        15      0      19      24       7
32768             1         1      0       0    2047       0

By type       inuse     calls failed  memuse  memmax  mapb

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

Figure 83. Network memory buffer allocation statistics

10.6 Tuning methodology example with changing JVM parameters

The parameters listed in previous chapters provide a starting point for your application and application server testing. Since your application and operating environment will differ from those used for our tests the recommended approach is to run application-specific performance tests.

The following example depicts the iterative approach required to determine the the appropriate Java heap size settings (-ms and -mx) for the Trade application run on an AIX Model 43P workstation with a 200 MHz processor and 256 MB of RAM.

> **Note**
>
> This example is for illustration of the performance tuning process and should *not* be used for capacity planning purposes.

10.6.1 Case 1: -mx64m

As a starting point we set the maximum heap size for the application server to 64 MB (-mx64m) on the command line arguments. We did not set the minimum heap size.

Figure 84 shows the output of `vmstat` at 5-second intervals (`vmstat 5`). Looking at the statistics for Page and CPU we can see that the test was started at the seventh row below. This is indicated by the increase in CPU use and a drop in free memory with an attendant increase in system paging.

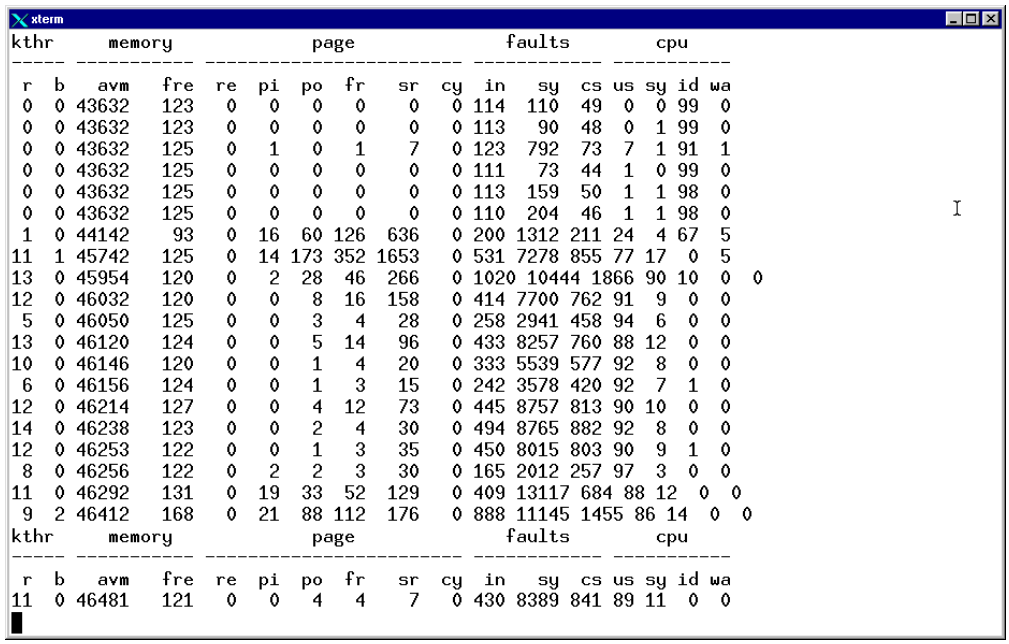


Figure 84. Output of vmstat (1/2) of case1 (-mx64m)

Within 20 seconds (4 records) though paging decreases for the most part, there continues to be some paging. At the bottom of the diagram, there is an increase in paging.

At the same time we can monitor the JVM heap size by using the WebSphere Resource Analyzer which we can access via **Tasks -> Performance -> Start -> Server Resources** tab.

From there you specify the node, and server and the information, shown in Figure 85, as displayed. To graph a particular resource such as JVM Memory in use, highlight the item of interest, then specify **Chart -> Add Selection to Chart**. You can also detach the Resource Analyzer task window by selecting **Window -> Detach Task Window**.

As can be seen, the JVM heap size has grown rapidly from 7550 KB (7.55 MB) to 15,976 KB (15.97 MB) before decreasing, then cycling up and down as garbage collection controls the size of the heap.

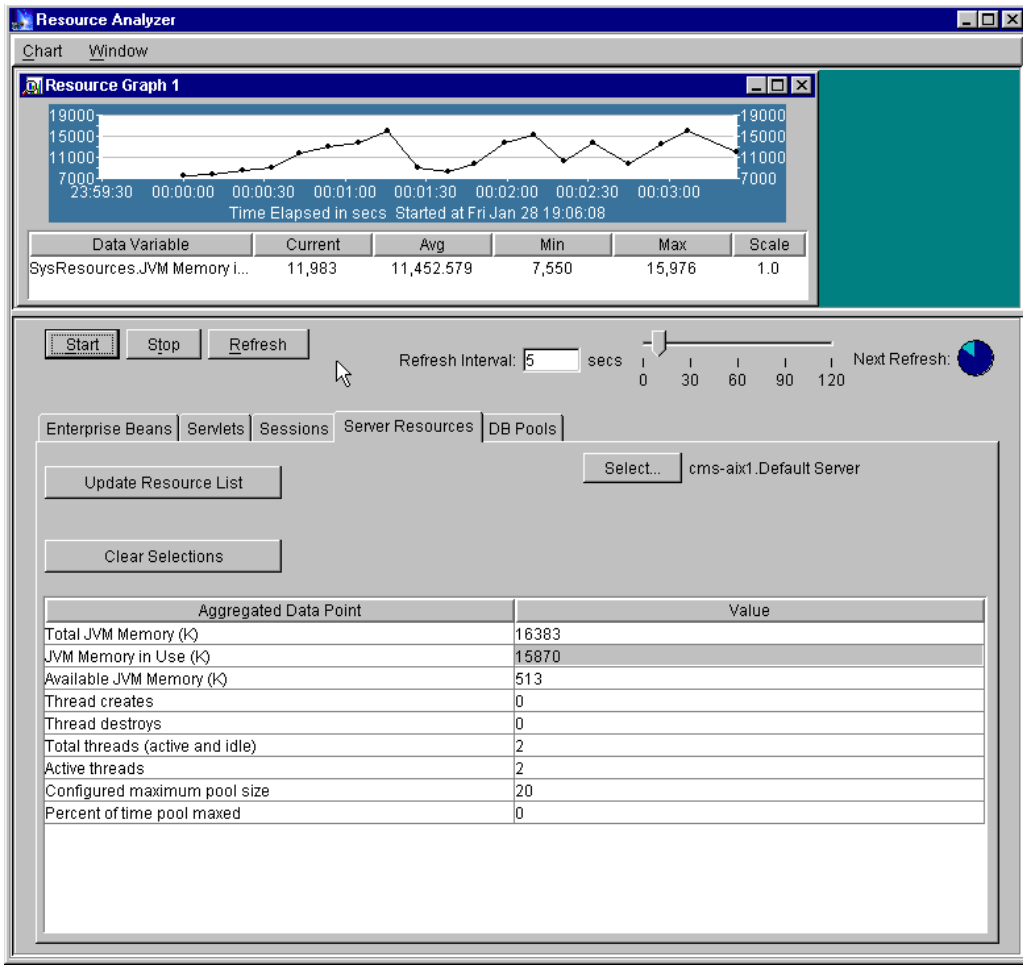


Figure 85. Resource Analyzer with case1 (-mx64m)

Examining `vmstat` again as the run progresses and we can see that CPU use has remained fairly constant, and that the memory use (paging) has settled down as shown in Figure 86.

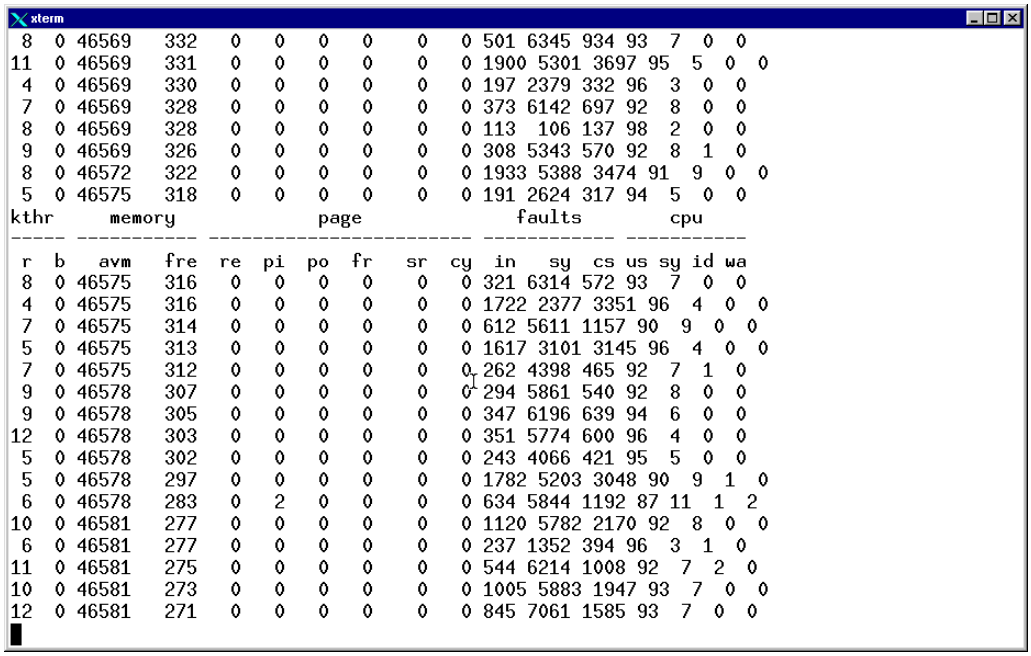


Figure 86. Output of vmstat (2/2) of case1 (-mx64m)

Turning our attention back to the WebSphere Resource Analyzer, we can see a continuation of the same oscillation pattern in JVM heap size as garbage collection occurs.

Note the values plotted are at the end of the Resource Analyzer polling cycle so the JVM most likely did increase to the max of 16 MB. Use of more finite polling would depict a more accurate picture of JVM use, as shown in Figure 87.

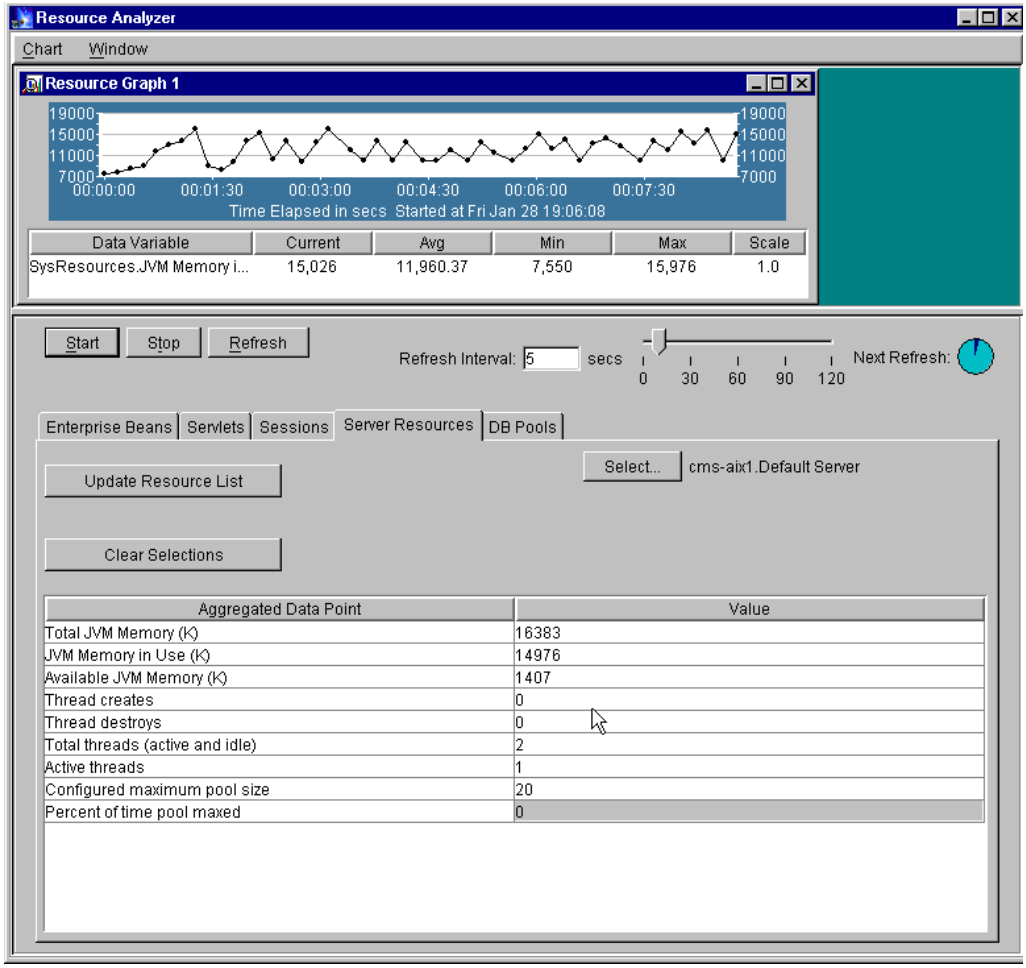


Figure 87. Resource Analyzer with case1 (-mx64m)

Of course we're all interested in the end result and for that we turn to AKstress, which we used for generating high load, at the end of our run.

```
AKstress Process Statistics

   Uptime:     0 hours 13 minutes 50 seconds
   Number of Threads:     10
   Pages Completed:     10000
   Pages To Be Completed:     10000
   Pages per second:      12.05
   Requests completed:     10000
   Requests per second:      12.05
   Failed Connections:    0
   Incorrect response codes:    0
   Content verification failed:     0
   Request write failures:     0
   Number of early closes:     0
   Number of early server closes:     0
   Number of request write failures:     0
   SSL handshake failures:     0


Request statistics for request /servlet/TradeServlet (Rec_Request_dHe8Ea)
   Successes:   10000
   Return Code Failures:   0
   Early Server Closes:   0
   Content Verification Failures:   0
   Min time (milliseconds):   0
   Max time (milliseconds):   13660
   Mean time (milliseconds):   786


akstress - execution complete
```

Figure 88. Output of AKstress of case1 (-mx6m)

Upon examination of the results it can be seen that we processed 12.05 requests per second with these particular JVM settings.

10.6.2 Case 2: -ms32m, -mx64m

For the second run a minimum heap size of 32 MB was specified. As before the maximum heap size remained set at 64 MB. (-ms32m -mx64m). The start of the run is shown in Figure 89 with `vmstat`. We can see that the run started at the eighth line from the top by looking at the change in CPU use which transitions from 93% idle to 35% idle to 0% idle as the test run was started. Also note that as we increased JVM size the amount of system paging increased when compared to the first test.

```
xterm
 0  0 48473   169   0   4   0   0    0   0 119  243  61  2  0 95  3
 0  0 48473   160   0   1   0   0    0   0 114  124  50  1  0 97  1
 0  0 48473   142   0   3   0   0    0   0 119  277  77 20  1 77  3
 0  0 48473   120   0   5   0   1    4   0 121  111  60  1  0 94  5
 0  0 48473   126   0   2   2   3    9   0 117  243  53  1  0 96  2
 0  0 48473   123   0   0   0   0    0   0 116  118  47  1  0 98  0
 0  0 48473   121   0   1   2   3   18   0 125  238  59  2  1 90  8
 0  0 48473   121   0   1   1   1    5   0 115  111  51  1  0 97  1
 0  0 48473   123   0   6   3   6   17   0 122  133  62  2  1 93  5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  1 49295   126   0  28 157 219  683   0 205  692 273  5  7 35 54
 0  1 50089   123   0  57 180 219  524   0 259 1414 347 20  8  0 72
 0  1 50349   124   0  69  85 121  403   0 262 1927 502 23  5  0 72
 0  1 50435   126   0  83  69 100  547   0 259 2236 538 24  6  0 70
 0  1 50525   122   0  78  62  96  737   0 263 2305 567 24  7  0 69
 2  0 50573   124   0  82  30  92  259   0 271 2221 604 21  9  0 70
 0  1 50582   122   0  84  42  84  201   0 287 1387 485 14  5  0 81
 0  1 50583   127   0  88  55  88 1496   0 241  120 219  3  3  0 94
 0  1 50584   123   0  80  55  81 1217   0 221  119 211 17  3  0 79
12  0 50702   127   0   3  16  28  303   0 386 7010 676 87 12  0  0
 8  0 50771   127   0   1  14  16  132   0 361 5841 648 84 12  1  3
 7  0 50789   124   0   6   8   9   39   0 340 6107 644 83 13  0  5
 6  0 50801   120   0   4  10  12   62   0 329 6036 564 79 11  0 10
18  0 50830   121   0   3   8   9   69   0 581 9602 1046 88 10  0  2
22  1 50860   125   0  49  43  60  564   0 801 13063 1404 85 14  1  0
```

Figure 89. Output of vmstat (1/2) of case2 (-ms32m -mx64m)

The corresponding data from the WebSphere Resource Analyzer is shown below. Since a larger JVM minimum heap size was specified, we observe a longer interval between garbage collection cycles.

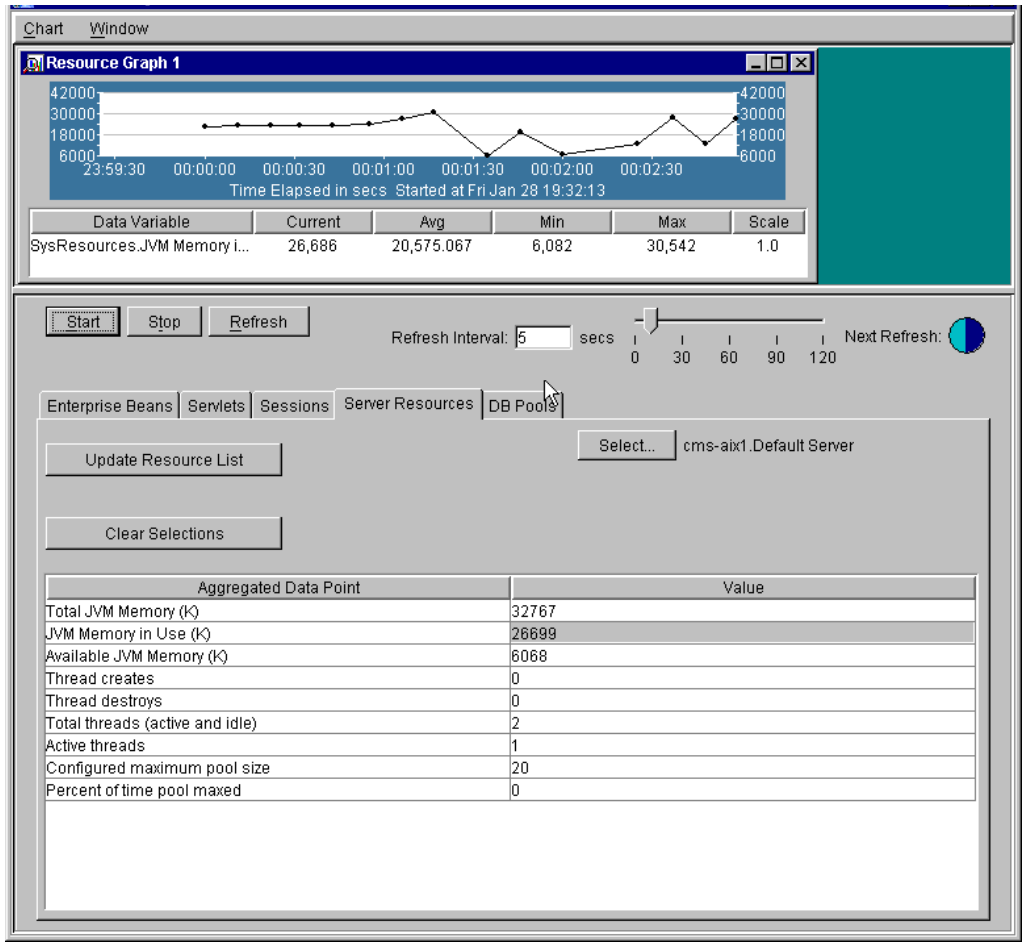


Figure 90. Resource Analyzer with case2 (-ms32m -mx64m)

The `vmstat` data during the middle of the run is shown in Figure 91. CPU use remains high and some memory paging is still occurring, though not as much as in the beginning of the run, but more than occurred during the first run as shown in Figure 86 on page 121.

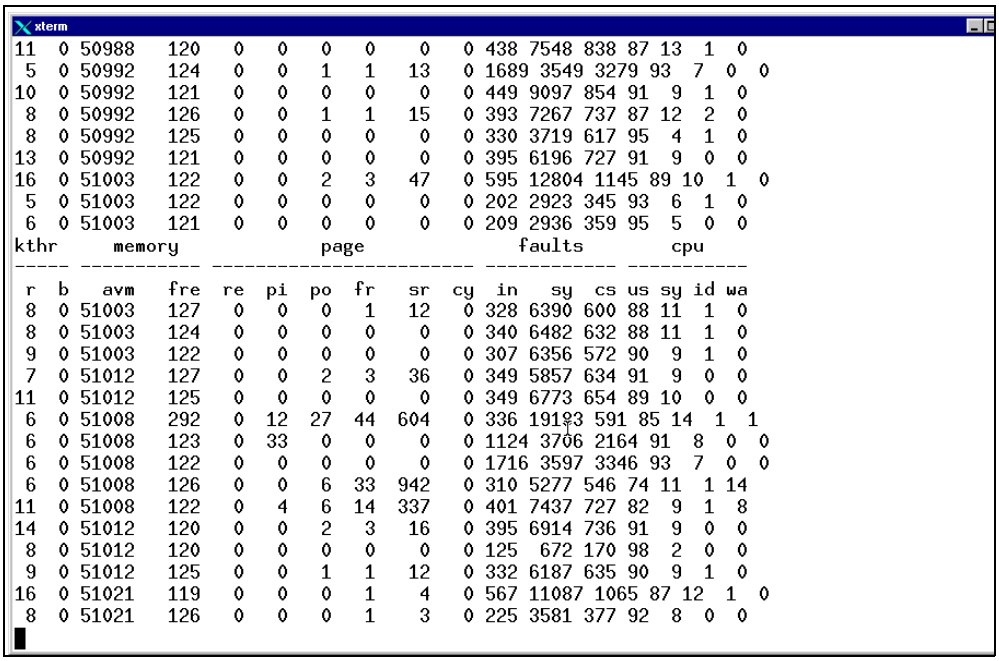


Figure 91. Output of vmstat (2/2) of case2 (-ms32m -mx64m)

The corresponding WebSphere Resource Analyzer data is shown in Figure 92.

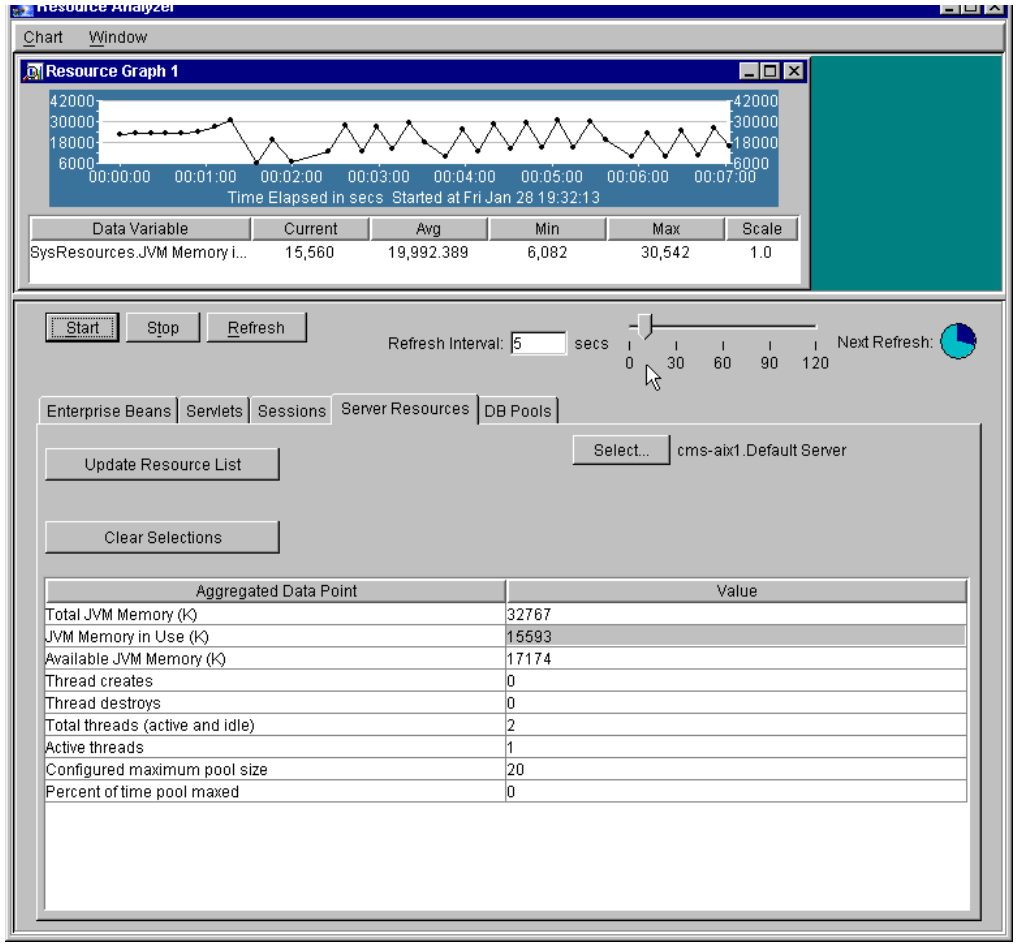


Figure 92. Resource Analyzer with case2 (-ms32m -mx64m)

Finally at the end of the run we see the results from AKstress as shown in Figure 93.

```
AKstress Process Statistics

   Uptime:     0 hours 10 minutes 50 seconds
   Number of Threads:     10
   Pages Completed:     10000
   Pages To Be Completed:     10000
   Pages per second:      15.38
   Requests completed:     10000
   Requests per second:      15.38
   Failed Connections:    0
   Incorrect response codes:    0
   Content verification failed:     0
   Request write failures:     0
   Number of early closes:     0
   Number of early server closes:     0
   Number of request write failures:     0
   SSL handshake failures:     0


Request statistics for request /servlet/TradeServlet (Rec_Request_dHe8Ea)
   Successes:   10000
   Return Code Failures:   0
   Early Server Closes:   0
   Content Verification Failures:   0
   Min time (milliseconds):   0
   Max time (milliseconds):   17084
   Mean time (milliseconds):   621


akstress - execution complete
```

Figure 93. Output of AKstress of case2 (-ms32m -mx64m)

Increasing the JVM minimum heap size has resulted in a significant performance improvement from 12.05 requests per second to 15.38 requests per second. This represents a 27% improvement in throughput. This result is no doubt due to the decreased amount of times that garbage collection had to run, despite the increased memory paging we observed.

10.6.3 Case 3: -ms64m, -mx64m

For our third test we again increase the minimum heap size, this time to 64 MB (-ms64m) so that it equals the maximum.

The starting `vmstat` output is shown in Figure 94. As can be seen by examining CPU use and paging, the test started at the seventh line from the top. While the second run incurred some CPU wait (far right column) as a result of the CPU waiting for resources due to paging, the wait for this the third run is far more pronounced in both the severity and duration then was observed in the previous runs. The same can be said of paging which is far more pronounced is the third run than in the second run.

.

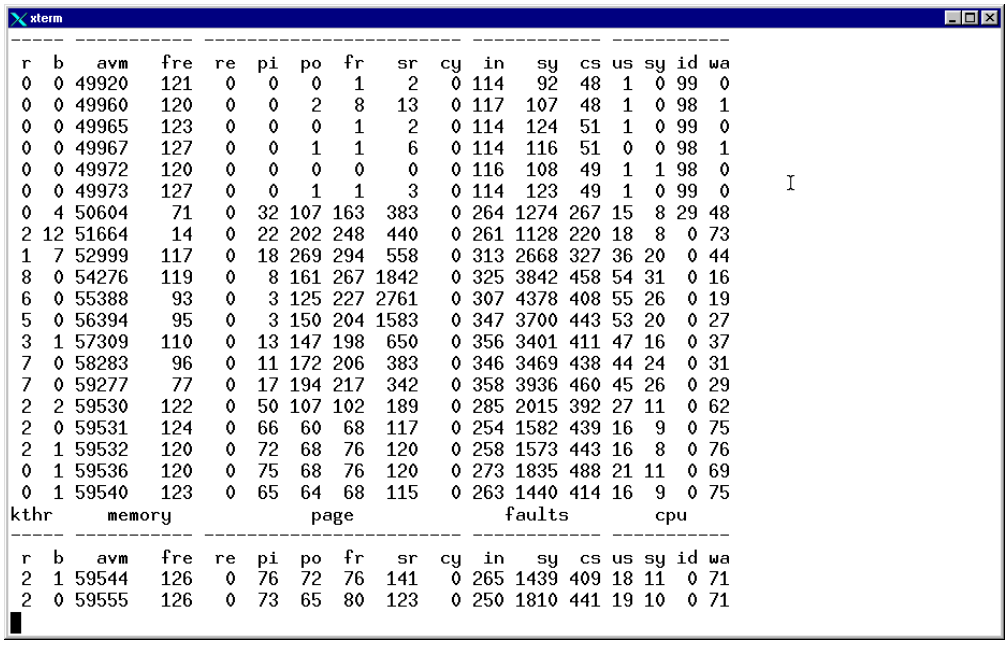


Figure 94. Output of vmstat (1/2) of case3 (-ms64m -mx64m)

The corresponding data from the WebSphere Resource Analyzer is illustrated in Figure 95. As can be seen, increasing the heap size results in a slow grow in the heap size without the oscillating pattern seen in the previous runs due to garbage collection.

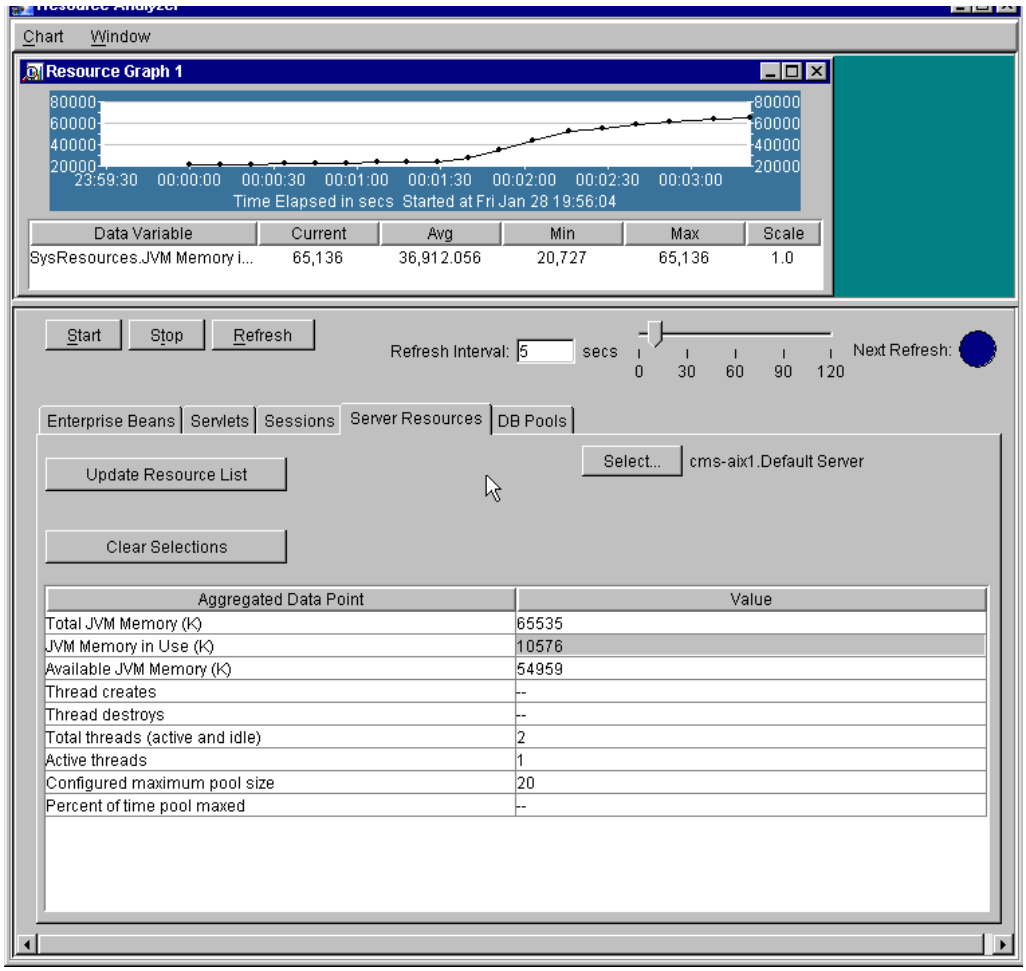


Figure 95. Resource Analyzer with case3 (-ms64m -mx64m)

As the run continues we examine the output from `vmstat` and see that we continue to experience a great deal of CPU wait state and memory paging as shown in Figure 96, neither of which is beneficial for performance.

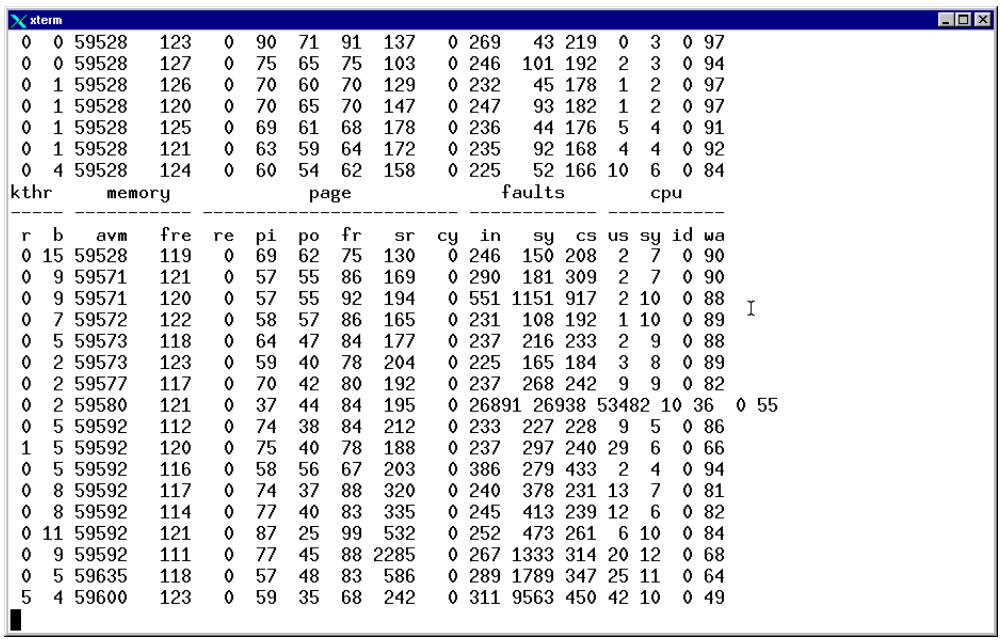


Figure 96. Output of vmstat (2/2) of case3 (-ms64m -mx64m)

In fact we saw that AKstress was running so slowly, as compared to the other runs, that we halt the run.

```
AKstress Process Statistics

   Uptime:     0 hours 22 minutes 11 seconds
   Number of Threads:     10
   Pages Completed:     4063
   Pages To Be Completed:     10000
   Pages per second:       3.05
   Requests completed:     4063
   Requests per second:       3.05
   Failed Connections:    0
   Incorrect response codes:    0
   Content verification failed:     0
   Request write failures:     0
   Number of early closes:     0
   Number of early server closes:     0
   Number of request write failures:     0
   SSL handshake failures:     0


Request statistics for request /servlet/TradeServlet (Rec_Request_dHe8Ea)
   Successes:   4053
   Return Code Failures:   0
   Early Server Closes:   0
   Content Verification Failures:   0
   Min time (milliseconds):   0
   Max time (milliseconds):   47904
   Mean time (milliseconds):   1511
```

Figure 97. Output of AKstress of case3 (-ms64m -mx64m)

We can see that throughput has fallen off to 3.05 requests per second, so whatever benefit we realized from garbage collection running was more than offset by the amount of time the system spent paging memory with the corresponding CPU wait.

Thus we've determined that the optimal heap size settings for this application in conjunction with this hardware environment are a maximum heap size of 64 MB (-mx64m) and a minimum of 32 MB (-ms32m). We can use this as a starting point for testing the effect of other JVM settings such as `-noclassgc`. A similar approach is then used for other settings such as connection poolsize.

Chapter 11. WebSphere Application Server Site Analyzer

The WebSphere Application Server Site Analyzer, part of the WebSphere family of products, provides basic Web site traffic measurements. Site Analyzer, at a high level, provides detailed analysis of Web content integrity, site performance, usage statistics, and a report writing feature to build reports from the content integrity and usage statistics.

You can get more information from the WebSphere site at `http://www.ibm.com/software/webservers/appserv/siteanalysis.html`.

11.1 What is WebSphere Application Server Site Analyzer?

Site Analyzer provides a set of tools to measure Web site traffic by analyzing content, measuring usage, and reporting. It assists the user in understanding *traffic volume* (hits and visits), identifying *traffic sources* (domains, subdomains, and referrers), and managing *site integrity* (link verification and site conformance). Site Analyzer may be installed on a single machine or over multiple machines in a client/server model.

Both administrator and non-administrator or client users need to define preferences prior to getting started. First, each user must go through the Startup wizard, also known as General Preferences. Second, each user must create a project to organize work. Project files are stored locally on the client machine, but link to elements on both the local client machine and on the server machine. When you have several Web sites to analyze, you should consider creating a separate project for each of the sites to keep your information separate. This will allow you to easily access information from different Web sites.

Certain roles and responsibilities are defined for the Site Analyzer administrator and client user as follows:

- Creating analysis (Administrator only)
- Editing an existing analysis (Administrator only)
- Deleting an analysis from the database (Administrator only)
- Running an analysis (Administrator only)
- Scheduling an analysis to run later (Administrator only)
- Creating report elements
- Creating reports
- Generating reports

WebSphere Site Analyzer comes with a set of wizards for defining report elements and reports. Report elements are a method to operate on an analysis. Report elements consist of types (for example a browser), measures (such as hits per visit), and operations (such as sum or average). After a report element is created, then you can build and generate a report. Wizards provide a very user friendly interface for understanding how your Web site is used.

In general, Site Analyzer performs the following:

- Analyzes Web content integrity and site performance, and provides usage statistics.
- Supports the team environment, with its flexible client/server architecture.
- Supports teams that may be divided into administrators and non-administrators with different levels of authority.
- Uses database technology to provide scalability and historical data for trend analysis.
- Supports customization and configuration to suit the user environment.
- Supports remote administration and reporting via the client interface.

11.2 Why do I need WebSphere Application Server Site Analyzer?

The WebSphere Application Server Site Analyzer provides the Web site owner which a rich set of tools to analyze usage and content of Web pages. Site Analyzer allows you to develop specific reports to provide feedback to different groups within your company. For example, an executive might ask the question *How is the money that I am spending on a Web site affecting my bottom line?* Or, *Am I reaching my customers?* Someone in sales or marketing might ask the question *Who are my customers? Are my marketing programs effective?* Or*, How effective is the Web site content?*

Site Analyzer provides the tools necessary to assist in the development, deployment, and management of a Web site. Data collected may provide different people within an organization with such information as measuring and optimizing usage, or tracking the effectiveness of marketing campaigns.

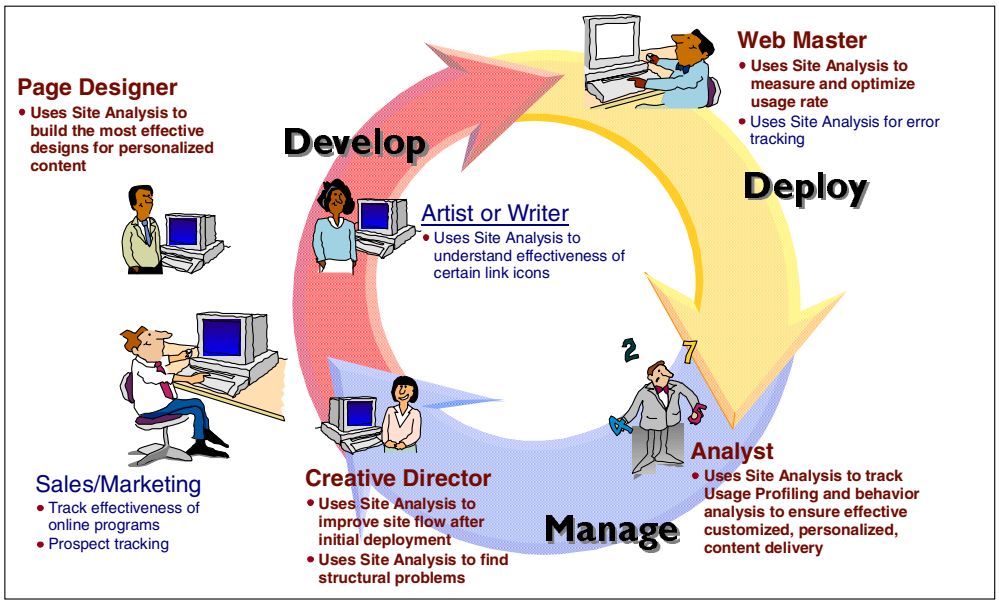


Figure 98. Site Analyzer feedback loop

11.2.1 Features of WebSphere Application Server Site Analyzer

Features of the site analysis component include:

- Content and structural analysis
- Usage analysis
- Modular reporting
- Scalability and trending
- Categorical analysis
- Dynamic content analysis
- Visitor analysis
- Additional technical features
- Flexible client/server configuration

11.2.2 Content analysis

Content analysis is a function of Site Analyzer, and provides information about the structure of your Web site, including information about duplicate pages, unavailable resources, broken links, and content with excessive load time. Content analysis provides the following functions:

- Explores secured servers with a fast Internet friendly crawler. A crawler is a program that, given an anchor URL, searches the Web site collecting information about links, and detailed information about resources within your Web site. The crawler obeys the Web server robot rule specified in the robot.txt file. If the robot.txt file indicates URLs *not* to explore, then the content analysis will honor that request and explore only the areas that are not excluded. If the Web page is password protected, then the crawler uses predefined user IDs and passwords to gain access.
- Provides domain or URL inclusion and exclusion from the detailed analysis. Users may include additional domains or URLs other than the root domain to the analysis. Likewise, they may exclude domains or URLs.
- Collects page attributes and site structure for mapping with the Site Surveyor visual tool. Site Surveyor maps out the results of a content analysis. With Site Surveyor, you can view the following:
  - A visual representation of your site based on links it contains
  - Information about the site including the structure of the Web site
  - Detailed information about resources within the Web site
  - Information about the links in the Web site
- Identifies duplicate and inactive files on the Web server.
- Detects unavailable resources such as broken links and missing files.
- Identifies content with excessive load sizes or too many objects on a page allowing the user to fine-tune the site performance.
- Identifies pages that do not conform to user-defined site policies, such as meta tags.
- Allows users to view sites as a whole or divided into small partitions for more detailed analysis.

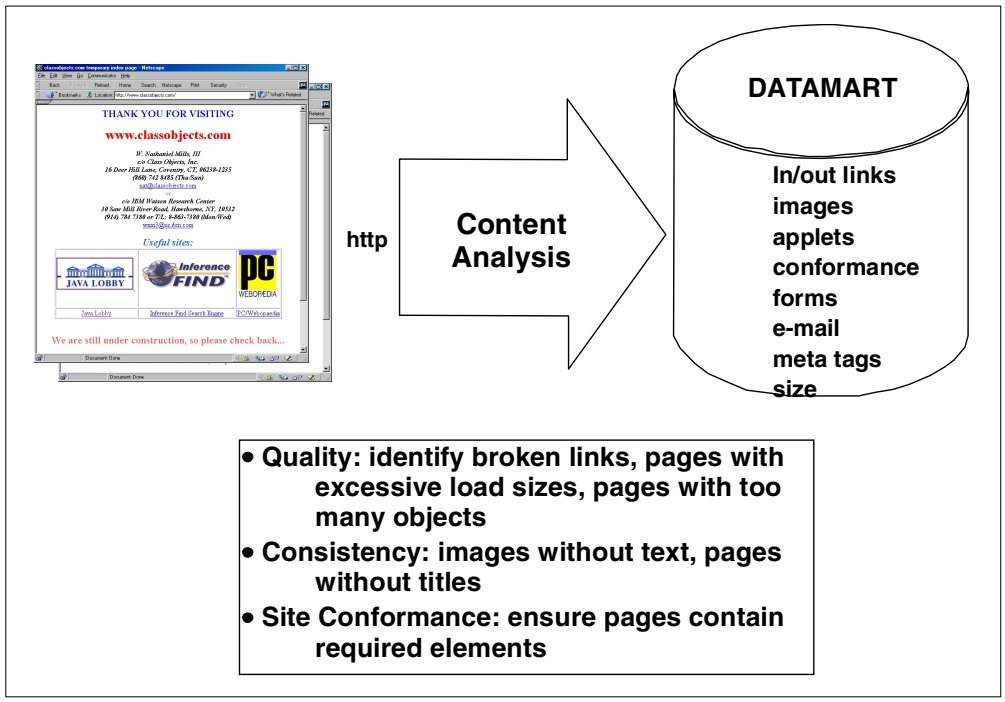


Figure 99. Content Analysis: measuring the integrity of Web sites

11.2.3 Usage analysis

Usage analysis reveals how your site is being used, including who is using it, where they enter it, and how they navigate within the site. You can view this information in reports. The usage analysis gathers site information by processing your Web server's log files. Usage analysis provides the following functions:

- Full heuristic (registration, cookie, IP, or referral-based) sessionization provides critical data about visits, entries, exits, etc. For example, you might be interested in hits per visit for different browsers.
- Supports National Center for Supercomputing Applications (NCSA) Combined, NCSA Common, and World Wide Web Consortium (W3C) Extended log file formats. The following is a list of Web servers and their supported log file formats:

Table 7. Site Analyzer supported log file formats

Apache Server Version 1.3.6	W3C Extended
Lotus Domino Go	NCSA Common or Combined
IBM HTTP Server Version 1.3.6	W3C Extended
Microsoft Internet Information Server (IIS) Version 4.x	W3C Extended
Netscape Enterprise Server Version 3.x	NCSA Common or Combined

- Automatically retrieves log files witha wildcard specification from servers. For example, filters include eliminating internal hits, and entering part of an address, such as 9.19.*, to exclude anything that begins with 9.19.
- Supports multi-threaded DNS resolution with customizable expiration rate, allowing reporting by readable names rather than by IP addresses. When the

usage analysis encounters an unrecognized IP address, it performs a DNS look up to resolve the host name so that reports will contain the host name.

- Supports customizable analysis process using categories, allowing users to include new items or categories of items by patterns.

This feature is crucial for extensibility. For example, users may specify a file name pattern to indicate a new category of advertisements. Usage analysis matches the pattern with the hit records during analysis and treats the matches as advertisements. This is useful, if for example the marketing group was creating a new marketing campaign associated with a Web page, and they want to capture how effective it is by counting the visits to that page.

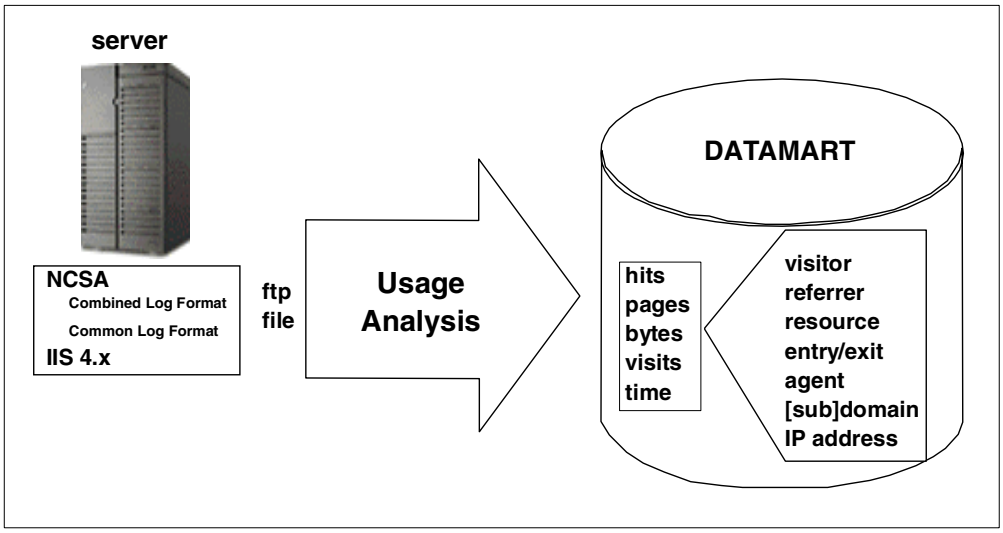


Figure 100. Usage Analysis: discover usage patterns

11.2.4 Visualization and reports

The Site Analyzer provides custom reports from the results of either the content analysis or usage analysis.

The Site Analyzer comes with predefined custom reports. Key features and customization are as follows:

- Provides visualizers that allow a user to view the site structure and quickly locate pages with problems via color schemes and icons.
- Provides on-demand searching for certain page attributes such as site policy conformance, author, expiration date, etc.
- Provides many predefined reports that are fully customizable including style, such as color and fonts to generate the report, and reporting range, such as beginning and ending dates and time.
- Allows the users to specify the beginning and ending day of the week. The default is Sunday to Saturday.
- Enables graphical 3-D charts for plotting data in a professional fashion.
- Allows users to request information in minute range (for example, last 15 minutes).

- Generates static or dynamic reports. Static reports use the currently available data; dynamic reports delay fetching data until requested via the browser. Site Analyzer contains a set of servlets that dynamically compose the report on the fly using most up-to-date data. Servlets run on the Web Server machine.
- Provides leading-edge tools for a user to create additional reports with no SQL knowledge required.

11.2.5 Usability

Site Analyzer provides key usability enhancements to assist the user in performing Site Analyzer tasks, such as:

- InstallShield to simplify the installation process
- Wizards to guide a user through various tasks
- Features Start-Up Wizard that quickly helps users to set up and get started
- Interfaces consistent with other WebSphere user interfaces, such as IBM WebSphere Studio

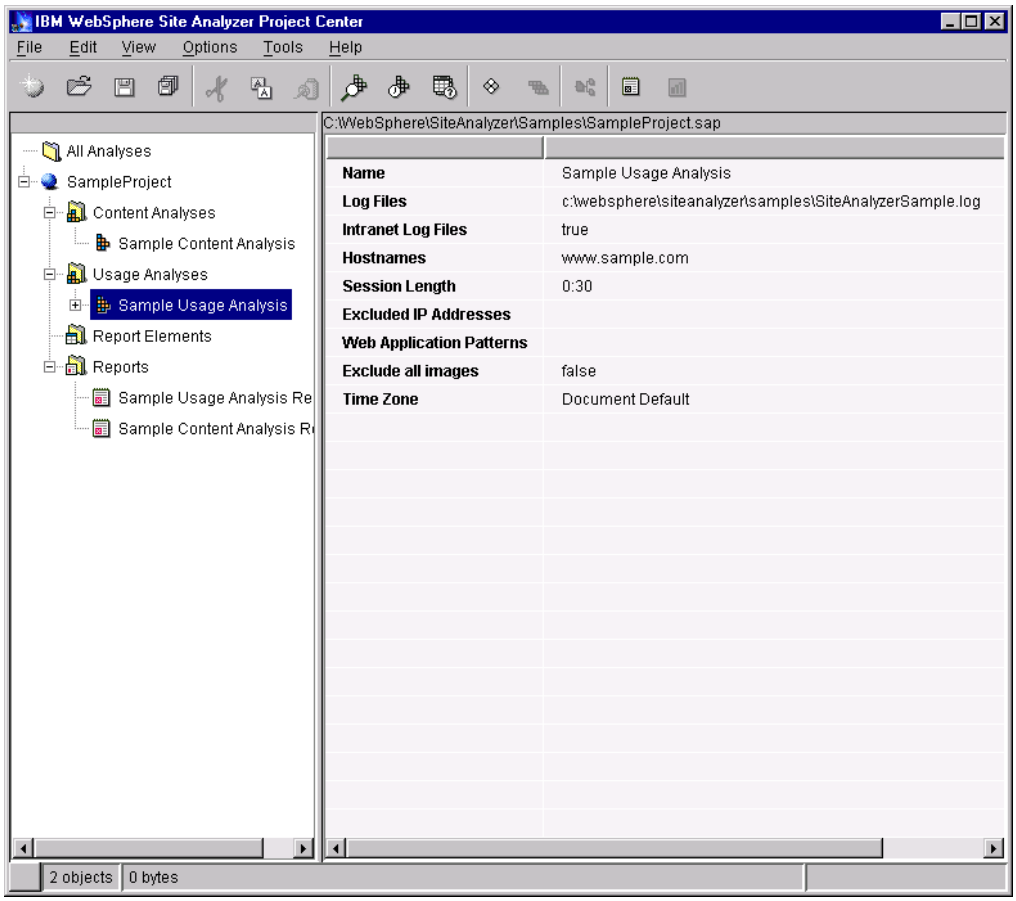


Figure 101. Usage analysis

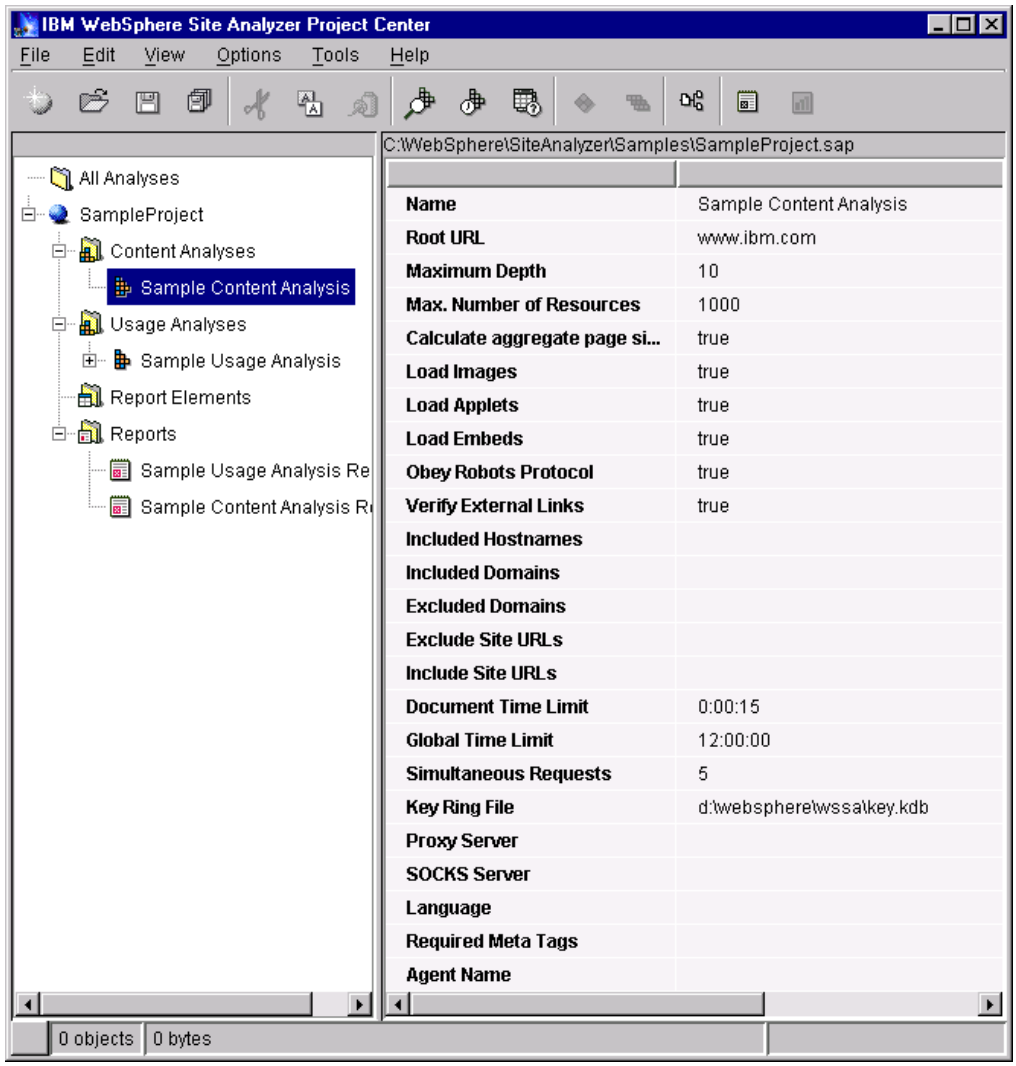


Figure 102. Content analysis

11.2.6 Technology

Site Analyzer covers a wide range of technologies and standards. It supports:

- Site Analyzer server support on AIX, Solaris and Windows NT and client support on all Windows platforms. In addition, it supports the use of HTTP log files from S/390 and AS/400 servers.
- Collection and storage of data in a DB2 Universal Database (UDB) included with the site analysis component.
- The site analysis component provides centralized task scheduling, allowing for scheduling of log file FTP, multiple analyses, and generation.
- Centralized task scheduling on the server also allowing the client to shut down without affecting scheduled tasks.
- Multiple sites and multiple servers.
- Administrators and clients with differing levels of authority.

11.2.7 Client/server configuration

Site Analyzer may be installed in a stand alone client/server configuration or in a multi-machine client/server configuration. See Figure 103. The server contains the analyzers whose responsibility is to transform the raw data into valuable information and store it in the database. The client interface provides administrative, visualizing, and report-generating functions. From the client, you can schedule analysis tasks to run at a specific time and/or time interval.

The progress status is broadcast to the interested clients and displayed as appropriate and necessary. Once the analysis is completed, you may generate reports or visualize the data. The system comes with about 30 predefined ready-to-use report elements.

The user may also install the entire system on one machine. In this case, the communications between the client and the server take place locally. The reports may be made available to other users by publishing them to a Web server using the provided publishing tool.

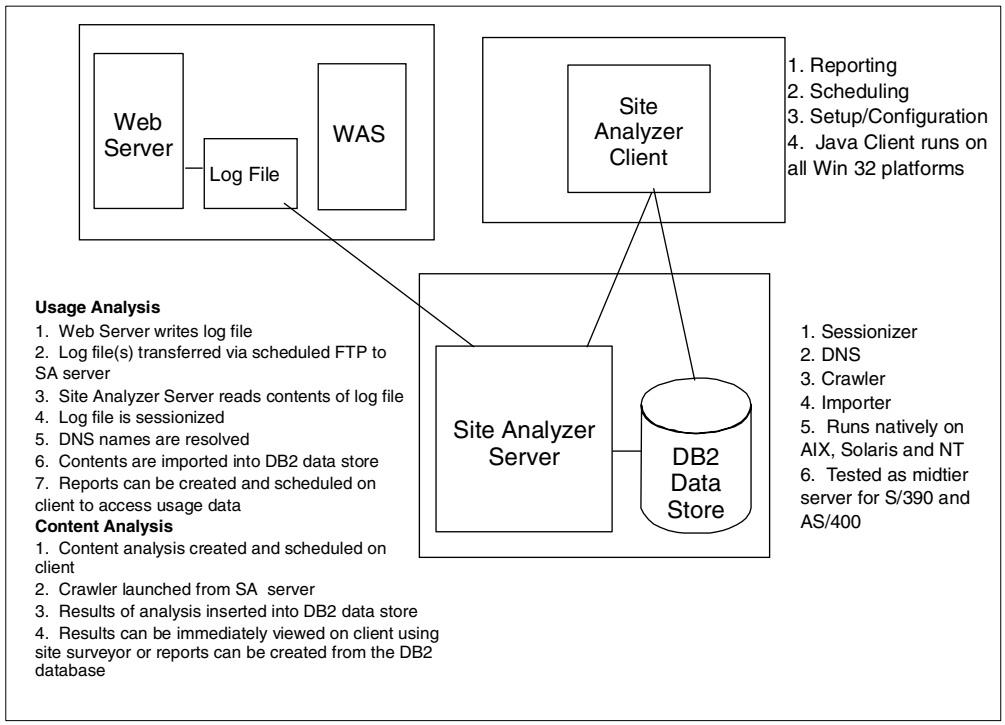


Figure 103. Site Analyzer architecture

Chapter 12. AFS performance tuning guide

AFS has provided scalable file administration and file sharing for large enterprises for many years and now it is a part of WebSphere Performance Pack. AFS is based upon its use of a virtual namespace to make naming and logical directory structures of files independent of their physical location. AFS clients and AFS servers are used to establish this virtual namespace capability. In a Web site, the AFS clients can be installed on HTTP servers to reduce the administrative effort associated with maintaining URL-to-file I/O mapping relationships. In addition, HTTP servers that are simultaneously AFS clients can significantly increase the connectivity capacity to Web server content and can provide local and geographically distributed access efficiency. Figure 104 shows the typical configuration of AFS with WebSphere.

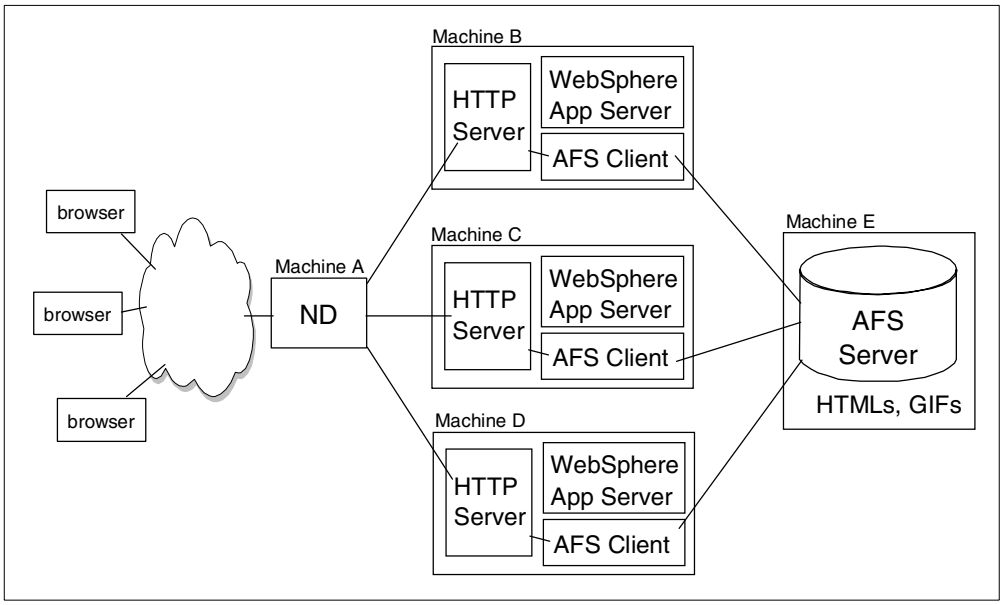


Figure 104. WebSphere clustering with AFS and ND

12.1 Overview

For documentary purposes, it should be noted that "File Server" refers to the machine serving files while "fileserver" refers to the individual process assigned with accomplishing this task in concert with other AFS processes.

This chapter is intended as an augmentation to the distributed AFS documentation, for use by persons who are knowledgeable with respect to the size of their cell in terms of approximate users and load conditions and wish to monitor and possibly fine tune their File Server. The concepts covered in this chapter will provide such a user with information regarding File Server parameters that can be used to help improve the performance of their File Server machines as well as discussions of debugging tools that can aid in the diagnosis and treatment of less than optimal File Server performance. We will not discuss these options in absolute terms as the needs and usage of File Servers, not to mention their performance capacity, varies greatly from cell to cell and prohibits

such a discussion. Other server processes are mentioned, but only in so far as their action affects the action of the File Server.

File Server tuning can usually be accomplished by the addition of selected parameters to aid the File Server in processing a user's requests in a timely manner. The impact of this tuning will obviously vary greatly depending on the dynamics of the cell and the situation encountered.

Scenarios are introduced in this chapter to educate the administrator in the recognition of situations where tuning of the File Server may improve performance. These scenarios are culled from real-life customer situations, but may not describe all aspects of the problems encountered, only those that relate to this subject. The AFS administrator should use these scenarios as an education tool on the road to better awareness of the general health of the File Servers in their cell and potential warning signs that intervention is needed. These scenarios are discussed in no particular order and are intended to provide AFS Administrators with a practical experience knowledge base in problems they may encounter and the methods to employ in determining the causes and solutions. As with any real-life scenario, your individual experiences, causes, and solutions may differ from these. The usage of the tools that are employed in this section are covered in the next section. Typically, the first sign that there is a File Server tuning or overload problem comes from the users who notice poor response times when their clients are requesting or storing a file. We will investigate a few of these situations and illuminate the steps needed to identify and correct the cause.

AFS is distributed with a host of debugging and monitoring tools that can greatly aid the AFS administrator in File Server tuning. These tools are often used by AFS Technical Support to diagnose reported problems by customers, but can also be effectively used by AFS administrators in an on-going basis to monitor these systems. Familiarity with the tools and their related output should help AFS administrators take a more proactive role in monitoring their systems.

This guide also contains information about performance improvements and parameter enhancements which have been added to the AFS 3.5 File Server. These changes are discussed in the 12.4, "Overview of AFS 3.5 File Server changes" on page 146, 12.5, "AFS 3.5 File Server performance improvements" on page 146, and 12.6, "AFS 3.5 File Server parameter changes" on page 151. However, only the changes from the AFS 3.4 File Server are discussed. Therefore, this entire document should be reviewed in order to gain a complete picture of the AFS 3.5 File Server.

This document is meant as a guide and will likely not touch upon all possible scenarios that could cause problems. As such, we recommend contacting AFS Technical Support in the event you experience problems or questions that are outside the scope of this document and are unable to resolve on your own.

12.2 Communications with the fileserver process

Before we discuss some of the more commonly used fileserver process parameters, a discussion of the steps taken during a "conversation" between a client and the fileserver process will help to highlight the importance of these parameters.

In this example, a user on a client machine makes an initial request to view the root directory of a volume from a File Server that it hasn't spoken to before. We will begin by assuming the client has already performed pathname lookups and is aware of the location, volume, vnode, and uniquifier of the directory. The client makes a fetchstatus RPC request to the fileserver process to gain information on the directory itself (length, last modified, creation date, etc.). The fileserver process has a listener thread listening on UDP port 7000 for any traffic from the clients. This thread receives the fetchstatus request from the client and establishes a connection with that client.

A server pool thread then looks at the request being made and makes the appropriate function calls to satisfy the request. In this example, the thread completes the request (by checking ACLs, obtaining the information from disk, etc.) and sends the reply back to the client. If there are no server pool threads available, the listener thread will create the call structure for the request and place it on a queue for the server pool threads to pick off when they are available. Otherwise the call is created and passed to a waiting server pool thread.

Since this is the first time this client machine has talked to this File Server, the fileserver also makes a call to the ptserver to check for any host-based ACL information based on the client machine's IP address. It then makes a second call to the ptserver to check for any ACL information based on the user ID making the fetchstatus request. The ptserver responds with the groups to which the IP address and/or user is a member. The fileserver stores this information in an internal structure for future comparison. The next time the fileserver hears from a different user on this same client, it will perform only the user ID call to the ptserver since it already has the host-based ACL information stored and needs only ACL information specific to this user. The user ID call to the ptserver is done once per connection with the client. This is why you need to `klog` to the cell again when your user ID ACL information changes; the `klog` command breaks this connection and establishes a new one with the fileserver process.

When a client is ready to actually fetch data for a file, it makes a fetchdata RPC to the fileserver's UDP port 7000 to request a copy of the file (or as much of the file that will fit into one "chunk" as defined in the client afsd parameters). With the default chunksize on the client set to 64 KB, the size of the response from the fileserver to this request will be up to 64 KB depending on the size of the file. The listener thread receives this fetchdata request from the client and creates a call structure for the request. It then passes this off to the next available server pool thread which compares the ACLs on the directory with those of the user stored in the internal structure and, assuming they allow access, will send up to the defined chunksize of data to the client.

When the fileserver sends either a file status or data back to the client, on the return there is an item called a callback. A callback is essentially a statement of how long the fileserver will remember that this user has some sort of access to this file. The fileserver also adds to its own list the fact that this client requested this file and how long the fileserver has told the client the callback is good for. If another user updates this file, the fileserver will look through its list of callbacks for this FID (fileid) for the clients (other than the update user) who have callbacks to this file and will send an RPC to these clients telling them to break the callback.

As alluded to previously, there are several different "kinds" of threads in the fileserver process. The bulk of these are server pool threads that actually handle

the RPCs and perform most of the operations relating to the file or directory manipulation. In addition to these threads, each fileserver process has 5 additional threads that perform the housekeeping tasks for that process:

Listener thread: This thread reads from UDP port 7000 listening for something to happen. When it receives a packet, it determines if there is already a running call that it is for and if so, passes it to that receive queue or if not, it creates the call structures for the request.

FiveMinute thread: This thread closes and reopens the log file every five minutes so that you can move it aside if desired, cleans up expired callbacks from the fileserver's list, resets disk usage stats, along with some additional cleanup tasks.

HostCheck thread: If the fileserver has not heard from a particular client for a while, this thread will send a check to see if the client is still alive. If it is not, this thread will remove the data structures associated with that client.

IOMGR thread: This thread runs when the fileserver process has handled all the RPCs and is waiting for a new RPC. The listener thread has registered its interest in reading from the socket for port 7000. The IOMGR thread waits in select() for data to read from this socket or for a signal to occur.

Signal thread: This transient thread is created by the IOMGR thread to do the work of a received signal.

12.3 Commonly used parameters

[-p (number of processes)]

This parameter sets the number of lightweight processes, or threads, that the File Server will use to service requests. The original default values are 6 threads for "small", 9 threads for "medium", and 12 threads for "large" (based upon the File Server size designation). Use this option in addition to the size option (it sets other parameters) if you want to increase the threadcount on a fileserver process. The fileserver process will automatically default to the "medium" setting of 9 threads if no size is specified.

Prior to AFS 3.4a build 5.53, the maximum "safe" number of threads you could use was 24. Anything beyond that level would eventually panic the machine. Build 5.53 introduced Delta 10083 to increase the maximum thread value to 59. The total max number of threads on the File Server is actually 64, but 5 are used for housekeeping tasks.

It should be noted that changes in this parameter will have varying results depending on such things as the machine's speed, size, and memory and therefore ability to make the best use of additional File Server threads. Adding many threads to a very small machine will obviously have a negligible effect.

It should also be noted that the File Server not only services users' store and fetch requests, but also in the course of these actions, consults with other server processes. Increasing the "size" of the File Server with this parameter without monitoring and adjusting the effect on other server processes may not result in the desired performance gain expected.

[-spare (number of spare blocks)]

Used to set the amount of space (in K blocks) by which users are allowed to exceed their quotas when storing in a volume. 0 indicates no grace amount.

While this parameter is not specifically a tuning parameter per se, it is useful in controlling the overall operation of the File Server machine as full File Server partitions could result from incorrect settings in your environment.

[-pctspare (percentage spare)]

This parameter is similar to “-spare” in that it allows the administrator to set the amount of space that the users of this File Server are allowed to exceed their quotas by when storing a volume. In this case, the “grace” amount is expressed as a percentage of the user’s quota and is therefore variable depending upon the individual quotas. 0 indicates no grace amount.

Again, it should be noted that while this parameter is not specifically a tuning parameter per se, it is useful in controlling the overall operation of the File Server machine.

[-busyat (redirect clients when queue > n)]

Under extremely heavy loads, it is possible for a large number of RX File Server calls to be queued while the data associated with those calls has been discarded. When these calls come to the front of the pending call queue and are scheduled, the thread waits for the rest of the data until the client retransmits. If this occurred to a large number of server threads, a situation referred to as “meltdown” occurs -- servers handling as few as five to ten calls per second, instead of, for example, five hundred.

With this flag, the fileserver will return VBUSY to the requesting client if the number of pending calls is higher than the value set by the flag.

In cells that may experience brief bursts of high activity on a semi-regular basis, this parameter can be used to help control these situations without affecting the overall function of the File Server.

[-rxpck (number of RX extra packets)]

"-rxpck N" increases the size of the File Server by about 1650 * N bytes.

This parameter controls the size of the RX packet pool. This pool is used for storing data for calls currently being handled, pending calls, and previous replies that are not yet complete. Using this option can increase the amount of data the File Server can store while processing requests, which will reduce the amount of retransmissions by the client. However, use of this parameter must be initiated in metered steps until the desired result is achieved. The more RX packets, the longer the list the File Server has to process and the more pings to clients it needs to make etc.

The current default is 200 packets. We recommend starting with a value of 400 and if an AFS administrator perceives a need to increase this amount, we recommend doing so in 200 packet increments until the optimal result is achieved.

[-udpsize (size of socket buffer in bytes)]

This parameter sets the UDP buffer size. The default size is 64 KB.

Since the File Server is a single-threaded (OS-speaking) process, all packets go through a single UDP port (7000), whose input buffer is 64 KB (set by the

RX layer). This parameter is useful if clients are hitting the File Server with many requests and UDP packets are subsequently being dropped. To determine this, the AFS administrator monitors `netstat` output on the File Server machine (to be discussed later in the Tools section). This option became available in the 5.24 build for the fileserver instance and the 5.26 build for the volserver instance.

[-nojumbo]

This option, when added to the File Server instance, turns off the File Server's ability to send "jumbograms" to the client.

Jumbograms are RX packets which are from 2 to 4 MTUs long. The MTU is the smallest MTU of either the server or client. So, if both client and server have the same MTU size, the RX packet will be that size. For example, if both are on FDDI then the packets will be FDDI-sized (4352), not Ethernet-sized (1500). If however, the client is on Ethernet (1500) and the server is on FDDI (4352), the RX packet will be 1500 (the smaller MTU of the two).

Jumbograms were introduced in AFS 3.4. AFS 3.3a functioned in the manner which is referred to above. In other words, we would use the MTU size of whichever is smaller (the client or server) and not increase beyond this point (by employing jumbograms). In AFS 3.4 we start with this initial idea of setting the MTU size based upon the smaller MTU between the two, but then we "ramp up" to RX packets that have 2 - 4 MTUs amount of data in them. With the "-nojumbo" option, you will revert to the 3.3a method wherein we pick the smaller MTU of the two and stick with it for that transaction.

12.4 Overview of AFS 3.5 File Server changes

Many aspects of the 3.5 File Server have remained the same as they were in 3.4. For example, the communications process between the client and File Server, the type and number of File Server housekeeping threads used, and the optional parameters and switches have, for the most part, remained the same as in 3.4 and are described in other sections of this document.

This addendum to the original *AFS 3.4 File Server Operations Tuning and Troubleshooting Guide* seeks to describe only those changes that have been made in the File Server functioning and available options in AFS 3.5. Please read this entire document to fully understand the File Server options and available tuning parameters.

The changes to the AFS 3.5 File Server can be grouped into two general categories: Performance Improvements and Parameter Changes, which are discussed in the sections that follow.

12.5 AFS 3.5 File Server performance improvements

The 3.5 release of AFS includes enhancements that significantly improve the throughput of AFS File Servers. Performance benchmarks run at IBM Transarc to compare AFS 3.4 with AFS 3.5 indicate AFS 3.5 File Servers can process three to four times more requests per second than AFS 3.4 File Servers. These changes, made to improve the AFS 3.5 File Server performance, are described below.

12.5.1 POSIX threads

The AFS 3.4 File Server runs with a light weight process (LWP) package that implements something resembling threads using setjmp and longjmp context switching library routines. In AFS 3.4, any thread that makes a blocking system call causes the other File Server threads to “sleep” while that thread waits for the kernel to complete the call. Disk I/O is a good example of a blocking system call in the File Server. So, for example, a 3.4 File Server thread that wants to write to or read from the disk is going to make a blocking system call to the kernel which will in turn cause the other File Server threads to sleep while waiting for this operation to complete. This has the negative effect of not only creating a bottleneck for disk I/O, but also reducing the potential operating performance of the other File Server threads by causing them to “sleep” when one is blocked in a system call.

In AFS 3.5, POSIX threads have been implemented in place of LWP. POSIX threads are scheduled independently, so the other File Server threads keep running even when one of the threads is blocked in a system call. In reference to the example cited in the preceding paragraph, we no longer have the single disk access I/O bottleneck seen in AFS 3.4 File Servers nor do the other threads “sleep” during this process, thus increasing the performance of the File Server. It should be noted, however, that any thread which is blocked while waiting for its system call to complete will still need to wait for a response from the kernel before continuing. For example, when a thread makes a disk I/O system call, that thread must wait for the kernel to complete the call, the speed of which will vary depending upon the speed of the disks and processor among other things.

12.5.2 RX slow start

AFS 3.4 File Servers used an aggressive retransmission policy that in practice performs poorly on congested networks. AFS 3.5 implements slow start, fast retransmit, congestion detection, and congestion avoidance as described in RFC 2001. If you would like to learn more about RFC 2001, you may obtain a copy from the following location:

```
ftp://ftp.isi.edu/in-notes/rfc2001.txt
```

12.5.3 File descriptor caching

The AFS 3.4 File Server did not make use of file descriptor caching. The AFS 3.5 File Server keeps a Least Recently Used (LRU) cache of file descriptors to reduce the overhead of opening and closing files between fetchdata or storedata RPCs sent by the client. This cache is hard-coded with a static size of approximately 2000 file descriptors.

As an example, suppose you are reading a 40 MB file. The client is going to be fetching chunks of that file one at a time by sending fetchdata RPCs to the File Server for each chunk.

When the AFS 3.4 File Server receives one of these fetchdata RPCs for, say, chunk 0, it will open the file, lseek to chunk 0, read in the data, close the file, and then send the data to the client. It would then get the next fetchdata RPC for (in this example), chunk 1, and again open the file, lseek to chunk 1, read in the data, close the file, and then send the data to the client.

When the AFS 3.5 File Server receives one of these fetchdata RPCs for, say, chunk 1 of a file, it will first search an internal hash table of handles to see if it has this file open already. If so, it then looks at the handle to obtain the list of file descriptor(s) open for this file. So, by saving the least recently used file descriptors, we can avoid opening and closing frequently used files as often between these RPCs.

12.5.4 Reduced lock contention

Using a profiling software package against the File Server binary we found and fixed several places where locks were being held longer than necessary, thus increasing the optimization of the File Server. For example, the CreateFile routine was holding a write lock on the parent directory while breaking callbacks, and the StoreData routine was holding a read lock on the parent directory while transferring data.

Further analysis of the 3.4 code also indicated that the RX code was one of the biggest sources of lock contention. This code was reworked with RX fine grain locking to add locks to protect individual data structures instead of one global lock for all of RX.

12.5.5 Overload processing

AFS File Servers have a busyat threshold that sets an upper limit on the number of calls that can be queued waiting for a thread. The default is three times the number of RX packets, divided by two. So, for example, if you have 400 RX packets, then the busyat threshold defaults to 600. If the number of calls waiting for a thread exceeds the busyat threshold, then the File Server starts rejecting calls with a VBUSY error. This error tells the client to sleep for 15 seconds and try the call again or, if a ReadOnly exists for that volume on another File Server, the client will instead contact the other File Server. If no ReadOnly exists on another File Server, the user will see a "Waiting for busy volume" message or, if it is an application making the request, the application will not fail unless the VBUSY condition exists for approximately 30 minutes.

AFS 3.4 does this check only after a call has been assigned to a server thread, not when the first packet for a new call is received. Assume for example that you currently have 600 calls waiting for a thread and you have the default threshold of 600 for the busyat parameter. When the next call comes in, it is the 601st call waiting for a thread. That 601st call goes through the queue of 600 calls, gets assigned to a thread, then when the thread picks it up and starts working on it, it realizes that there are already more than the busyat threshold of calls waiting for a thread, and sends the client a VBUSY error response. In the meantime, more calls are coming in and stacking up waiting for an available thread, thus increasing the overall count of waiting calls. If this condition persists and more calls are coming in than the File Server can process, the File Server can enter a state known as “meltdown”. File Server meltdown occurs when the File Server cannot keep up with the incoming call queue, and all calls get rejected with VBUSY.

AFS 3.5 fixes this by checking the number of queued calls when the first packet for a new call is received. Only calls that would cause the call queue to overflow are rejected so the File Server can continue to process the calls it currently has queued and can accept new calls as it services and removes calls from this queue.

The following is a graphical representation of the change made in how the File Server handles calls that are above its busyat threshold. Assume that the “|” indicates clock time incrementing as you move down the chart and arrows indicate the direction of data with the client on the left and the File Server on the right.

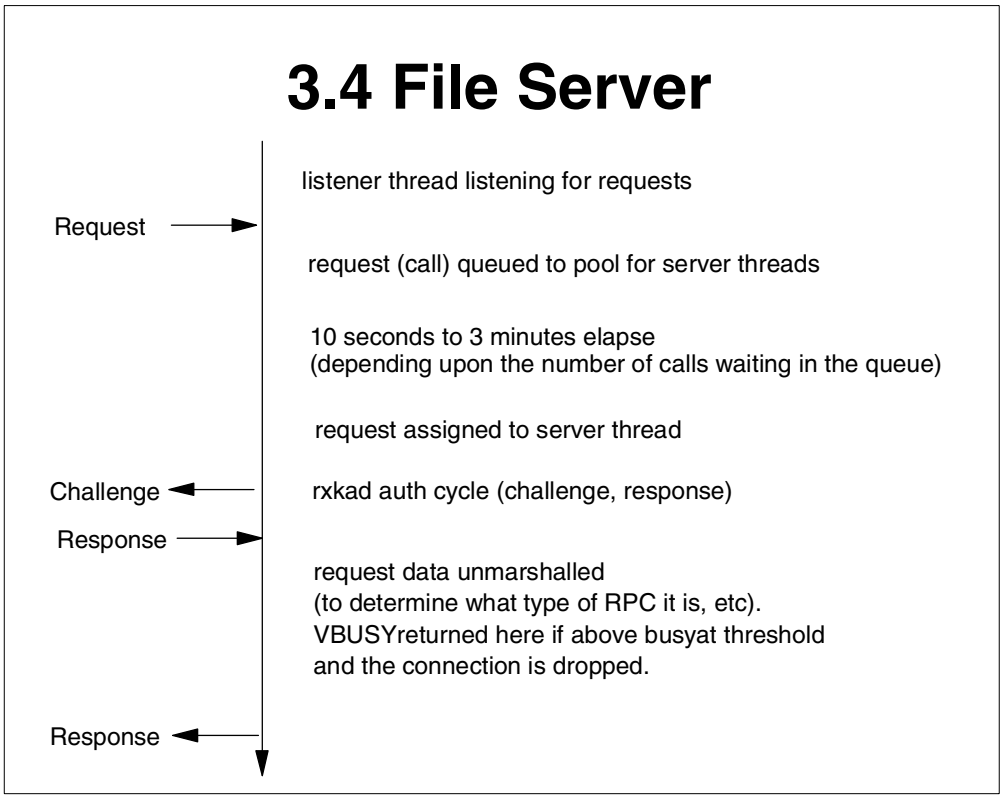


Figure 105. How 3.4 File Server handles calls

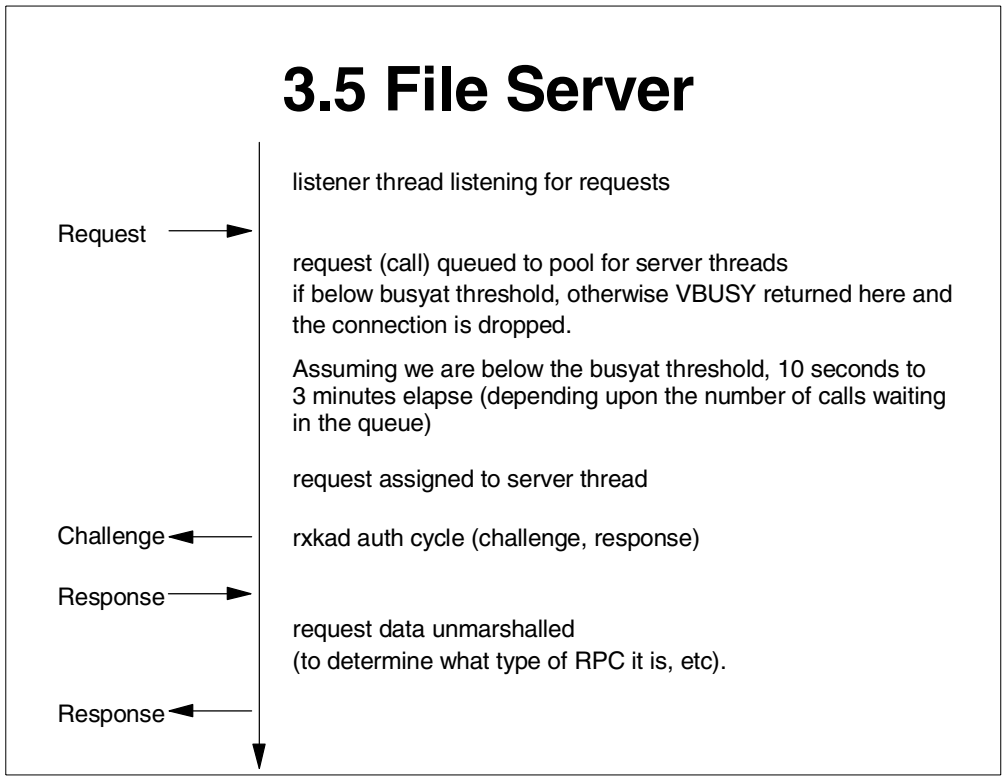


Figure 106. How 3.5 File Server handles calls

Note

The 3-minute upper boundary indicated above in both graphs assumes you have 600 new calls (and a busyat threshold of 600). If these are all new connections, they would all have to be authenticated, which means the File Server would have to check with the ptserver for each one of these calls. This is obviously a “worst case” scenario, but the potential delay is worth noting. If the 3.4 File Server is in this situation, then the 601st call coming in could potentially wait this long until the File Server performs the authentication check with the ptserver and since this is the 601st call, only then is it rejected with the VBUSY error. The connection is then dropped due to the VBUSY error, but the File Server and ptserver processing on this call up to this point has been wasted.

12.5.6 Buffer management

AFS 3.4 File Servers will delete packets from the receive queues of calls waiting for a server thread when the File Server runs out of RX packet buffers (as indicated by the NOBUFS field in rxdebug output – see 12.8, “Debugging tools and example output” on page 160 for more information). If, for example, a thread is processing a call and needs more packets but the File Server is out of packets, it will find a call that has packets and is waiting for a thread and usurp some of those packets. Once packets have been deleted from a call's receive queue, new packets for that call are not accepted until the call is assigned to a thread.

This scenario created two problems. First, the client would now need to retransmit the packets taken when the call, now without some of its packets, eventually was processed by a thread. This results in additional network traffic. Second, instead of having the full data for the thread to start working on, some or all of the packets would be missing and the thread would have to wait for them to be retransmitted before it could start processing the call. These delays negatively impact the performance of the File Server and, in extreme cases, can result in a File Server meltdown because once the File Server starts reclaiming packets from receive queues, the throughput of the File Server drops significantly. The service time on a cleared call is longer because the File Server has to wait for the client to resend the packets that were discarded. The File Server also has additional overhead because the clients do not stop retransmitting when the server clears the receive queue.

The AFS 3.5 File Server relies on the busyat threshold to indirectly limit the number of packet buffers used and can also allocate packet buffers as needed so threads no longer need to usurp packets from waiting calls when they need more. In this way, we can avoid the condition described above.

12.6 AFS 3.5 File Server parameter changes

[-p (number of processes)]

Similar to the 3.4 File Server, this parameter sets the number of threads that the File Server will use to service requests. However, the 3.5 File Server uses POSIX threads in place of the lightweight process threads that were implemented in 3.4. The original default values are 6 threads for “small”, 9 threads for “medium”, and 12 threads for “large” (based upon the File Server size designation) and have remained unchanged at this time. Use this option in addition to the size option (it sets other parameters) if you want to increase the threadcount on a File Server process. The fileserver process will automatically default to the “medium” setting of 9 threads if no size is specified.

The maximum number of threads available in the 3.5 File Server has been increased from 59 (3.4a build 5.53 and later) to 123. Similar to the 3.4 File Server, the total maximum number of threads is actually 128, but 5 are reserved for housekeeping tasks.

As in 3.4, it should be noted that changes in this parameter will have varying results depending on such things as the machine’s speed, size, and memory and therefore ability to make the best use of additional File Server threads. Adding many threads to a very small machine will obviously have a negligible effect.

It should also be noted that the File Server not only services users’ store and fetch requests, but also in the course of these actions, consults with other server processes. Increasing the “size” of the fileserver process with this parameter without monitoring and adjusting the effect on other server processes may not result in the desired performance gain expected.

[-busyat (redirect clients when queue > n)]

With this flag, the fileserver will return VBUSY to the requesting client if the number of pending calls is higher than the value set by the flag. This flag existed in the AFS 3.4 File Server and is unchanged in terms of syntax;

however, the point at which it is implemented has changed in AFS 3.5. Instead of performing this check only after a call has been assigned to a thread (as is done in AFS 3.4), the AFS 3.5 File Server checks the number of queued calls when the first packet for a new call is received. Please review to 12.5.5, “Overload processing” on page 148 for more details on the new implementation of this parameter.

12.7 Scenarios

12.7.1 Scenario #1

Problem:

Users are complaining about poor response times or timeouts when they request information from a certain File Server. Other File Servers in the cell seem to be functioning normally with no delays in responding to users’ requests. In extreme cases, the troubled File Server can move to a “meltdown” situation if diagnosis and corrective action are not taken.

Possible cause:

There may be some I/O-intensive volserver operations taking place that are not allowing the File Server to service its own requests. We refer to this situation as the volserver being I/O bound.

What to look for:

- Several clients attempting to access files from the same File Server are experiencing poor response times.
- The disk I/O on this File Server will be high.

> **Note**
>
> Methods of determining this vary by operating system. The AFS administrator needs to be familiar with the steps to determine this value on the operating systems they employ.

- A `vos status` command on this File Server machine shows one (1) or more vos operations in progress. The following vos operations can cause heavy disk I/O activity and potentially trigger this problem:
  - vos remove
  - vos backup (essentially a clone operation)
  - vos release
  - vos move

If the above symptoms are present, also check `rxdebug` output on the File Server to see if there are RX packets backing up:

```
# ./rxdebug -servers zeus -port 7000 -rxstats
Trying 158.98.19.134 (port 7000):
Free packets: 328, packet reclaims: 91, calls: 323297, used FDs: 7
not waiting for packets.
```

Figure 107. Sample output of rxdebug

If RX packets are backing up, the above-noted section of `rxdebug` output will indicate the amount.

What to try:

Since it appears that the volserver has forgotten how to share the disk with his neighbors, we can tell the volserver to "sleep" for a couple seconds and let the fileserver have a chance at the disk too.

To do this, you add the hidden option `-sleep x/y` to the volserver process and restart the fs instance. The x value refers to the length of time the volserver should sleep and the y value refers to the length of time it should run between these sleep intervals, for example:

```
volserver -sleep 2/15
```

This means the volserver should sleep for two (2) seconds after fifteen (15) seconds of work.

Note

- Do not make the sleep (x) number greater than ten (10).
- Two to three (2 - 3) is usually optimal for the x value.
- The runtime (y value) should equal two or three (2 or 3) times a minute.
- If “vos status” shows a LOT of activity, you can use three or four (3 or 4) in the x value.
- This option will NOT cause the volserver to sleep every y seconds all of the time. It will only come into effect when one of the operations that is listed above is taking place.
- The presence or absence of RAID has no bearing on whether you will see this problem.
- You will need to restart the fs instance in order to pick up the sleep option.
- The use of this flag could be a potential performance hit - use it only if needed.
- Use this flag on a machine by machine basis - not as a cell-wide immediate solution to one machine's problems.

12.7.2 Scenario #2

Problem:

Users are complaining about clients hanging when storing or fetching a fairly large file. The command doesn't complete, and the user gets a "communication timeout failure". This problem should occur on a fairly repeated basis -- not a one-time occurrence.

Possible cause:

The File Server may be attempting to send the data in jumbogram form to the client and something between the client and the server is inhibiting this action.

What are jumbograms?

As mentioned previously, when sending data to or from a client, if the size of the data is sufficient, we begin to "ramp up" the size of the RX packets and eventually send essentially very large RX packets across the wire. We are trusting that the machines and routers on the wire between this server and client will properly break the packets into IP fragments and reassemble them.

When this becomes a problem:

Some of these packets can be dropped if the routers are filling to capacity, or if some of the networks simply cannot handle fragments or are set to disallow them (“DF” flag). Aside from the explicit network restriction, what typically happens is a router or client is collecting all of the IP fragments waiting to get them all so that it can reassemble them and pass them to the next layer above. If it is missing one of the pieces, this device will just continue to collect all of the other fragments and will eventually fill up its input buffer before it can reassemble the packet.

What to look for:

Collect `tcpdump` data on the source and destination machines (the server and client), however do not run `tcpdump` on these machines themselves -- rather, on another machine with each of their subnets. `tcpdump` should be run on the piece of wire that the host is sitting on as well as on each piece of wire between the server and client (including the subnet the client is on). This will allow us to see if packets are being dropped somewhere along the way between the server and the client.

If you are unable to obtain a `tcpdump` of the problem, an alternate (although less revealing) method would be to perform a KDUMP on the client while it has a failing call in progress. In the KDUMP output, you can get the call MTU size and this will at least point to (or refute) the possibility of a jumbograms issue.

In the `tcpdump` output, look for the word “frag” and then check to see if the same frag pieces are being retransmitted over and over again. Also, check to see if the “DF” (or equivalent) flag is set which would prevent fragmentation of these multiple-MTU RX packets.

An example of this problem in `tcpdump` output:

The following is an example of what to look for in the `tcpdump` output to see if jumbograms are the problem. In this example, the File Server was attempting to

send jumbograms. The identifier in the `tcpdump` output is where we see the File Server sending 1444-byte packets and then attempting to send 1448-byte packets which never get through. Remember that 1472 + 8 (UDP header) + 20 (IP header) = 1500 bytes. In this example, most of the line to the left of the area of interest has been truncated and only the rightmost portion is included to highlight what we are looking for:

```
      < fileserver1.transarc.com.fs:  1.0006 data .... 1444 (DF)
      < fileserver1.transarc.com.fs:  2.0007 data .A.. 1444 (DF)
      < fileserver1.transarc.com.fs:  3.0008 data .... 1444 (DF)
      < fileserver1.transarc.com.fs:  4.0009 data .A.. 1444 (DF)
      < fileserver1.transarc.com.fs:  5.000a data .... 1444 (DF)
      > fileserver1.transarc.com.fs:  2.0000 ack   2 (DF)
      < fileserver1.transarc.com.fs:  6.000b data .A.. 1444 (DF)
      < fileserver1.transarc.com.fs:  7.000c data .... 1444 (DF)
      < fileserver1.transarc.com.fs:  8.000d data .A.. 1444 (DF)
      < fileserver1.transarc.com.fs:  9.000e data .... 1448 (DF)
      > fileserver1.transarc.com.fs:  4.0000 ack   4 (DF)
      < fileserver1.transarc.com.fs:  a.000f data .A.. 1448 (DF)
      < fileserver1.transarc.com.fs:  b.0010 data .... 1448 (DF)
      > fileserver1.transarc.com.fs:  6.0000 ack   6 (DF)
      < fileserver1.transarc.com.fs:  c.0011 data .A.. 1448 (DF)
      < fileserver1.transarc.com.fs:  d.0012 data .... 1448 (DF)
      > fileserver1.transarc.com.fs:  8.0000 ack   8 (DF)
      < fileserver1.transarc.com.fs:  e.0013 data .A.. 1448 (DF)
      < fileserver1.transarc.com.fs:  f.0014 data ..L. 244 (DF)
      < fileserver1.transarc.com.fs:  9.0015 data .A.. 1448 (DF)
      < fileserver1.transarc.com.fs:  a.0016 data .A.M 1448 (DF)
      < fileserver1.transarc.com.fs:  b.0017 data .A.. 1448 (DF)
      < fileserver1.transarc.com.fs:  c.0018 data .A.M 1448 (DF)
```

Figure 108. Sample output of tcpdump

From the above, there are several items to note:

- `fileserver1.transarc.com.fs` is the fileserver process.
- In the column that has numbers like:

  1.0006

  2.0007

  3.0008

  4.0009

  5.000a

The number to the left of the decimal is the packet number. The number to the right is the packet number from the beginning of the transmission -- but it is unimportant -- focus on the one to the left of the decimal.

- In the column that has "data" or "ack", this tells you whether it is a data packet being sent or an acknowledgment (other values are possible and mentioned in the Tools section of this document).
- The next column over (with "....") is a sort of “message” area. The possible “messages” are:

A = Acknowledgement requested

M = More data packets to follow

L = Last data packet in this series

- The next column tells you the size of the data. In this case, it starts out at 1444, which is normal for Ethernet, and then the fileserver process tries to increase it to 1448 (the start of jumbograms).
- The final column may or may not exist in your `tcpdump` trace. The “DF” means “Don't Fragment” and is set by MTU discovery.

So, with our new knowledge in hand, we see that the fileserver process first sends packets 1 and 2 with an ACK requested on packet 2. It then sends packets 3 and 4 with another ACK requested on packet 4. It then sends packet 5 and then receives its first ACK back from the client for packet 2 (essentially what the client tells him is “I'm done with packet 1 and have received packet 2”). The fileserver process then sends packet 6 with an ACK requested, then packet 7, then packet 8 is sent with an ACK requested also. The fileserver then sends packet 9, but notice, it is being sent with 1448 bytes of data! When you add the 56 bytes for IP information added to the packet, this totals 1504, which is 4 bytes more than this network will accept (remember the DF flag). As you look down the rows, you'll see that the fileserver process attempts to proceed with this larger size and continues to attempt to send these packets to the client. In particular, notice that packets 9, a, b, c, d, etc. are being retransmitted over and over. This is also a good sign that the packets are being lost and RX is continuing to try to send them (RX ignores the error it gets from the router saying that it won't accept the packet). Thus, we have identified that jumbograms are likely the cause of the user’s complaints.

What to try:

Add the "-nojumbo" option to the fileserver process and restart the fs instance.

Where this option exists:

- fileserver – added to the 5.00 build.
- volserver – added to the 5.00 build.
- vlserver (for vos listvldb jumbograms problems) – added in the 5.28 build.

We have already turned them off by default on the bosserver, kaserver, buserver and ptserver (added to the 5.55 build).

This command is "hidden" on the volserver and fileserver; it is visible on the vlserver.

Note

- RXTUNE (a.k.a. "Twiddle"), which is available from AFS technical support, is sometimes used to test for the presence of a jumbograms problem by causing the client to inform the servers it cannot accept jumbograms (thereby effectively turning them “off” for this client). Keep in mind that this program should only be run on the AFS client and will only affect the kernel. It has *no* effect on user-space commands such as bos, vos, kas, backup, etc.
- There is performance degradation (from optimal) when the jumbograms are turned off.
- Even if you see a lot of packets resent in the `tcpdump` output, that does not necessarily mean that it is a jumbograms problem. You can use RXTUNE to help isolate the problem.

12.7.3 Scenario #3

Problem:

Users are complaining about sluggish response times from the File Server and the clients are showing a lot of resends (from RXDEBUG output).

Possible cause:

The UDP buffer on the File Server may be overflowing, causing dropped packets which have to be resent by the client.

What to look for:

Look at the output from a `netstat -s` command run on the File Server. This command will give you statistics for all socket connections from that machine. Look at the "udpInOverflows" (or "buffer overflows" or equivalent) value. If this value is high or increasing over multiple iterations of the `netstat -s` command (one run before and one run after the test), then this buffer is overflowing.

What is happening:

All AFS packets are UDP packets which are sent over socket connections. When a File Server makes a socket to send packets over, it allocates a queue (buffer) for that socket in the kernel. This queue has a default size of 64 KB (set by the kernel). When the clients send packets, they go into the queue for the socket on the receiving end and the fileserver process picks them up from there. When a lot of clients are sending packets across to one File Server, and that process is reading packets off this queue, if the process is not able to keep up with the level of packets coming in, this queue will eventually fill and overflow (drop packets) causing the packets to have to be resent by the clients.

What to try:

If you see the udpInOverflows (or equivalent) buffer overflows value increasing over multiple iterations of the `netstat -s` command, you can instruct the fileserver (or volserver) to request a larger queue (buffer) at the time of creation. This is done with the UDPSIZE option added to either the fileserver or volserver

instance. To decide what value to set this option to, first determine the maximum value that the particular operating system supports, and then set a larger (> 64 KB) size within that boundary. You may need to adjust this several times to find the optimal value for your machine.

What this does:

With this option set, the File Server then reads this value when it opens a socket, and it sets the queue size to this value. Currently, you can have a buffer size of up to 2 GB on Solaris 2.5. Solaris 2.6 and HP10.20 currently set the default queue size to 1 MB. As mentioned previously, consult your operating system documentation or vendor for specifics.

Note

- The UDPSIZE option is available in AFS on every platform we support. Each of these sockets has a different size value depending on the operating system.
- As an example, to determine the system-wide maximum allowable socket buffer size for Solaris or AIX:

  Solaris: `ndd -get /dev/udp udp_max_buf`

  AIX: `no -o sb_max`
- To set the system-wide maximum allowable socket buffer size to 1 MB:

  Solaris: `ndd -set /dev/udp udp_max_buf 1048576`

  AIX: `no -o sb_max=1048576`

12.7.4 Scenario #4

Problem:

Users are complaining about extremely slow response times from a particular File Server.

Possible cause:

If the File Server is under heavy load during this time, it may be "melting down". Use the “meltdown” script shown in Figure 109 on page 161 to help make this determination.

What to look for:

In the RXDEBUG output, look at the "wproc" column and the "nobufs" column. If the "wproc" column is increasing in value (or spiking) and the "nobufs" value is increasing over several iterations of the RXDEBUG, then the server may be overloading or melting down. You will also see a high resend rate on the clients during this time (due to packets dropped by the fileserver).

Look at the size option on this File Server:

Table 8. The size option

Size	Default Threads	Default Packets
small	6	100
medium	9	150
large	12	200

There is a ratio between the number of extra RX packets that are set up and the size selected. This ratio is 16 2/3 RX packets per thread. This number is based on having 16 packets available to a thread (there is an 8-packet transmit window for each call the thread receives) which allows the thread to work on another call while waiting on the first call to finish with the first 8 packets.

Why it is happening:

This can happen if the File Server is having a hard time sending packets to the clients (packets being dropped somewhere along the network), unable to keep up with the demand given its current “size” settings, or if it is generally slow for some other reason (slow disk, other non-fileserver CPU-intensive processes on the same box, overloaded machine, etc.).

What to try:

- Make sure the File Server (via the fileserver instance) is set to at least the size option of "L".
- Add the File Server option "-rxpck" (via the fileserver instance) and set it to 400 as a start.
- Make sure the number of threads is sufficient for the rxpck value.
- You may alternatively want to set the thread value to something larger than 12 (if it is not sufficient) and then make sure the rxpck value is large enough (multiply the number of threads by 17).
- If the File Server has a high resend rate during this period, you should look for any network problems that would cause the File Server to need to resend packets as this would cause it to hold onto these packets while resending and can fill the rxpck pool.

Note

- "-rxpck N" increases the size of the fileserver process by about 1650 * N bytes.
- Q. When adding the "-rxpck" switch to the fileserver process, does this indicate the absolute number of RX packets for the fileserver process, or is it added on to an existing amount determined by the fileserver size switch? In other words, does adding "-rxpck 400" to a “large” fileserver process mean that it now has space for 400 RX packets, or is it 400 + ?? RX packets (the ?? is a number initially allocated to the fileserver process by the size switch).
- A. The fileserver code adds 16 2/3 packets per thread. RX itself adds 18 packets for each server thread started. So, there are two things to be aware of:
    1. If more threads are explicitly started in the fileserver process using the “-p” option, make sure that “-rxpck” is used and is at least 16 2/3 times the number of threads.
    2. -rxpck governs how big the RX packet pool is. Aside from the calls currently being handled, packets are used to hold data for pending calls and for finishing up previous replies. Increasing this value allows the fileserver process to hold onto data for calls not yet processed. This can help in meltdown or overload situations if the currently running threads do not have all the data they need to finish the call. Or if there is a slow or busy network where re-sending the packets to a busy File Server only tends to slow things down even further. In any of these cases, if the “nobufs” values from RXDEBUG are non-zero and increasing, and the File Server is not slow as a result of disk contention, increasing the number of RX packets is likely useful.

12.8 Debugging tools and example output

12.8.1 RXDEBUG incorporated in the meltdown script

The following is commonly (albeit somewhat incorrectly) referred to as the “meltdown script”. It utilizes the RXDEBUG diagnostic tool and formats the output in an easy-to-read table. It is suggested that you use this script as a monitoring tool when you:

- Perceive less than optimal performance on a File Server machine
- Experience rapid cell growth in terms of users and/or activity
- Are interested whether the File Server is reaching load capacity under its current configuration

The output from this script can help you determine if tuning changes are needed for a particular File Server machine due to usage and load variants (for example, large builds), or if the machine is in the process of melting down and unable to recover.

```
#!/bin/ksh
#

servername=$1

echo ""
echo Server $servername
echo "wproc   nobufs  wpack   free pack   calls       data
resends"
while [ 1 ] ; do
  /usr/afsws/etc/rxdebug $servername 7000 -rxstats |  \
     awk ' BEGIN { wproc = 0; wpack = 0; fpack = 0; calls =0; data =
0; resends = 0; nobufs = 0} \
           /waiting_for_process/ {wproc++} \
           /wait_packet/ {wpack++}         \
           /Free packets/ {fpack = $3; calls = $8} \
           /other send counters/ {data = $7; resends = $11} \
           /noBuffers/ {nobufs = $12} \
           END {printf "%5d %8d %6d %11d %7d %10d %13d\n", wproc,
nobufs, wpack, fpack, \ calls, data, resends } '
sleep 3
done

#     END { print " ", wproc, "    ", nobufs, "  ", wpack, "     ",
fpack, "    ", calls, "  ", data, "     ", \ resends }
```

Figure 109. The meltdown script

The following is sample output during a File Server meltdown, which is a worst-case scenario wherein the File Server is unable to keep up with incoming requests to such a point that it eventually is unable to recover at all. Notice that the nobufs number (a cumulative count) is increasing, the number of calls is staying the same, data is staying the same or not growing, and wproc (waiting for process) is growing.

```
wproc    nobufs  wpack    fpack     calls       data        resends
 272      5742,    0       74,      814081,    1134776       25810,
 272      5920,    0       74,      814081,    1134776       25810,
 278      7297,    0       74,      814081,    1134776       25810,
 278      7578,    0       74,      814081,    1134776       25810,
 281      8933,    0       74,      814081,    1134776       25810,
 284      9242,    0       74,      814081,    1134776       25810,
```

The following is sample output from a File Server that had a load spike at some point, but was able to recover and is now operating normally. Notice that the nobufs number is non-zero but not changing and the number of calls is increasing, however the wproc value is 0.

```
  wproc   nobufs  wpack    fpack     calls       data        resends
    0      5265,    0       182,     1501650,   1661183       53199,
    0      5265,    0       182,     1501657,   1661192       53200,
    0      5265,    0       184,     1501659,   1661193       53201,
```

12.8.2 tcpdump

tcpdump is a useful tool for monitoring networks and decoding protocols when you suspect communications-related problems between client and server. Some possible command lines could be:

```
./tcpdump -w <outfile> -s 150 host <hostname> and port 7000
./tcpdump -w <outfile> -s 150 host <hostname> and host <hostname2>
./tcpdump -w <outfile> -s 150 host <hostname>
```

We show you the tool syntax and usage as follows:

- The <outfile> should be located in local disk space rather than AFS. This is especially true when you are trying to isolate a communications problem between client and server.
- The -s refers to the snap length. This allows us to pick up the first 150 bytes of the UDP packet.
- The port is optional, but if used will most likely be 7000 (fileserver) or 7001 (client).
- tcpdump should not be run on the source or destination machines.
- tcpdump should be run on the piece of wire that the host is sitting on.
- If there are two wires between the fileserver and client, then run tcpdump on both wires to allow us to see if packets are being dropped in between the two.
- tcpdump usually doesn't work well on a switched hub - you don't get the packets that are going between other machines. In those instances, you may want to try running it on the machine that you're trying to sniff. Or try "snoop" if you are working with Solaris.
- If it's not Solaris and you're unable to obtain a tcpdump, you can alternatively perform a kdump on the client while you have a call that's failing in progress. While this is limited information, it can sometimes point to the problem.
- If you suspect an issue involving jumbograms (for example, Scenario #2), it's best to not filter on the port number at all as this will filter out the fragments of the jumbograms.

To read the raw tcpdump output file that is created, use the same tcpdump command with the -r switch on the outfile. You can also add filters to this command to narrow the scope of the output you are studying:

```
./tcpdump -r <outfile>
./tcpdump -r <outfile> host <hostname> and port 7000
./tcpdump -r <outfile> host <hostname> and host <hostname2>
./tcpdump -r <outfile> host <hostname>
```

```
11:24:12.044201 client1.transarc.com.cm.7b5de95c.0001 > fileserver1.transarc.com.fs:  1.0006 data .... 1444 (DF)
11:24:12.047017 client1.transarc.com.cm.7b5de95c.0000 < fileserver1.transarc.com.fs:  2.0007 data .A.. 1444 (DF)
11:24:12.047881 client1.transarc.com.cm.7b5de95c.0000 < fileserver1.transarc.com.fs:  3.0008 data .... 1444 (DF)
11:24:15.045200 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  4.0009 data .A.. 1444 (DF)
11:24:18.305831 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  5.000a data .... 1444 (DF)
11:24:20.422469 client1.transarc.com.cm.7b5de95c.0001 > fileserver1.transarc.com.fs:  2.0000 ack   2 (DF)
11:24:20.424691 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  6.000b data .A.. 1444 (DF)
11:24:20.426924 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  7.000c data .... 1444 (DF)
11:24:20.428027 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  8.000d data .A.. 1444 (DF)
11:24:20.429147 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  9.000e data .... 1448 (DF)
11:24:20.430624 client1.transarc.com.cm.7b5de95c.0001 > fileserver1.transarc.com.fs:  4.0000 ack   4 (DF)
11:24:20.431472 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  a.000f data .A.. 1448 (DF)
11:24:20.431651 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  b.0010 data .... 1448 (DF)
11:24:20.431927 client1.transarc.com.cm.7b5de95c.0001 > fileserver1.transarc.com.fs:  6.0000 ack   6 (DF)
11:24:20.432221 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  c.0011 data .A.. 1448 (DF)
11:24:20.432687 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  d.0012 data .... 1448 (DF)
11:24:20.433101 client1.transarc.com.cm.7b5de95c.0001 > fileserver1.transarc.com.fs:  8.0000 ack   8 (DF)
11:24:20.433533 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  e.0013 data .A.. 1448 (DF)
11:24:20.436192 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  f.0014 data ..L. 244 (DF)
11:24:20.437001 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  9.0015 data .A.. 1448 (DF)
11:24:20.437443 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  a.0016 data .A.M 1448 (DF)
11:24:20.439192 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  b.0017 data .A.. 1448 (DF)
11:24:20.440298 client1.transarc.com.cm.7b5de95c.0001 < fileserver1.transarc.com.fs:  c.0018 data .A.M 1448 (DF)
```

Figure 110. tcpdump output

Table 9. tcpdump output descriptions

Field No.	Description
1	Timestamp
2	Host name and process (in this example, .cm means cache manager), connection ID and call number.
3	The “<” or “>” indicates the direction in which the packet is traveling.
4	Host name and port designator (in this example, fs means fileserver), the connection ID and call number are identical at both ends of a connection.
5	Packet sequence number on the call, and serial number. Retransmitted packets will have the same sequence number, but different serial numbers.
6	Holds values of “data”, “ack”, “busy”, or “abort” and indicates what this packet is.
7	Flags indicating: C = Packet originated at the RPC’s client. A = Acknowledgement requested. M = More data packets to follow. L = Last data packet in this series.
8	The size of the data in bytes.
9	This field may or may not exist in your `tcpdump` trace. In this example, the “DF” means “Don’t Fragment” and is set by MTU discovery.

12.8.3 netstat

`netstat` output varies in format and field names depending upon the platform on which it is run. For this example, we look at the output from a SunOS 5.5 machine. If we were debugging Scenario #3, we would concentrate on the UDP section and specifically on the udpInOverflows field. The command we issue is:

```
./netstat -s
```

The following is a sample output of the command:

```
UDP
   udpInDatagrams      =2954751 udpInErrors         =     0
   udpOutDatagrams     =5738548

TCP
   tcpRtoAlgorithm     =     4 tcpRtoMin           =   200
   tcpRtoMax           = 60000 tcpMaxConn          =    -1
   tcpActiveOpens      = 20974 tcpPassiveOpens     =   179
   tcpAttemptFails     =    19 tcpEstabResets      =     9
   tcpCurrEstab        =     5 tcpOutSegs          =637145
   tcpOutDataSegs      =544819 tcpOutDataBytes     =200221356
   tcpRetransSegs      = 22034 tcpRetransBytes     =212746
   tcpOutAck           = 92316 tcpOutAckDelayed    = 47300
   tcpOutUrg           =    46 tcpOutWinUpdate     =     0
   tcpOutWinProbe      =     0 tcpOutControl       = 42303
   tcpOutRsts          =    23 tcpOutFastRetrans   =    14
   tcpInSegs           =622151
   tcpInAckSegs        =504560 tcpInAckBytes       =200242345
   tcpInDupAck         = 42609 tcpInAckUnsent      =     0
   tcpInInorderSegs    =299043 tcpInInorderBytes   =38379387
   tcpInUnorderSegs    =     0 tcpInUnorderBytes   =     0
   tcpInDupSegs        =     0 tcpInDupBytes       =     0
   tcpInPartDupSegs    =     0 tcpInPartDupBytes   =     0
   tcpInPastWinSegs    =     0 tcpInPastWinBytes   =     0
   tcpInWinProbe       =     0 tcpInWinUpdate      =     0
   tcpInClosed         =     5 tcpRttNoUpdate      =   811
   tcpRttUpdate        = 39477 tcpTimRetrans       =  1248
   tcpTimRetransDrop   =     3 tcpTimKeepalive     =   450
   tcpTimKeepaliveProbe=   342 tcpTimKeepaliveDrop =     0
IP
   ipForwarding        =     2 ipDefaultTTL        =   255
   ipInReceives        =5457657 ipInHdrErrors       =     0
   ipInAddrErrors      =     0 ipInCksumErrs       =     0
   ipForwDatagrams     =     0 ipInForwProhibits     =     0
   ipInUnknownProtos   = 27624 ipInDiscards        =     0
   ipInDelivers        =5154339 ipOutRequests       =6401650
   ipOutDiscards       =     0 ipOutNoRoutes       =     0
   ipReasmTimeout      =    60 ipReasmReqds        = 15914
   ipReasmOKs          = 15914 ipReasmFails        =     0
   ipReasmDuplicates   =     0 ipReasmPartDups     =     0
   ipFragOKs           =  4232 ipFragFails         =     0
   ipFragCreates       =  9525 ipRoutingDiscards   =     0
   tcpInErrs           =     0 udpNoPorts          =246892
   udpInCksumErrs      =     0 udpInOverflows      =     1
   rawipInOverflows    =     0
ICMP
   icmpInMsgs          =  1239 icmpInErrors        =     0
   icmpInCksumErrs     =     0 icmpInUnknowns      =     0
```

```
    icmpInDestUnreachs  =  1221 icmpInTimeExcds     =     9
    icmpInParmProbs     =     0 icmpInSrcQuenchs    =     0
    icmpInRedirects     =     0 icmpInBadRedirects  =     0
    icmpInEchos         =     7 icmpInEchoReps      =     2
    icmpInTimestamps    =     0 icmpInTimestampReps =     0
    icmpInAddrMasks     =     0 icmpInAddrMaskReps  =     0
    icmpInFragNeeded    =     0 icmpOutMsgs         =   208
    icmpOutDrops        = 27623 icmpOutErrors       =     0
    icmpOutDestUnreachs =   201 icmpOutTimeExcds    =     0
    icmpOutParmProbs    =     0 icmpOutSrcQuenchs   =     0
    icmpOutRedirects    =     0 icmpOutEchos        =     0
    icmpOutEchoReps     =     7 icmpOutTimestamps   =     0
    icmpOutTimestampReps=     0 icmpOutAddrMasks    =     0
    icmpOutAddrMaskReps =     0 icmpOutFragNeeded   =     0
    icmpInOverflows     =     0
IGMP:
             0 messages received
             0 messages received with too few bytes
             0 messages received with bad checksum
             0 membership queries received
             0 membership queries received with invalid field(s)
             0 membership reports received
             0 membership reports received with invalid field(s)
             0 membership reports received for groups to which we belong
             0 membership reports sent
```

From the above example, we see that the UDP buffer on this File Server overflowed once, resulting in one dropped packet which had to be resent by the client. A higher value in this field would help point to an ongoing problem in this area and suggest that the udpsize option on the File Server be used.

12.9 Summary

File Server tuning is a function that we recommend be performed on a somewhat consistent basis with a new cell and then at minimum every time the cell's usage patterns or machines change. There are no specific rules to follow in determining when exactly this function should be performed and it is expected that the AFS administrator is familiar with the cell and will be able to make a judgement call in this regard.

As we have outlined in this document, the overall tuning of the File Server is a two-step process beginning with analysis of current conditions and subsequent changes or additions to the parameters and switches available to the fileserver instance. This tuning can be either proactive or reactive depending upon the current conditions in your cell and will employ for the most part the same data-gathering techniques and analysis outlined here.

Use this document as a guide to help you better understand your cell and the loads placed on the File Servers within it. Please keep in mind that this document is intended to provide guidance and as such will not necessarily solve every problem or need you encounter. If you experience questions or problems that this guide does not answer, please contact AFS technical support for further assistance.

Appendix A. TCP/IP overview and tuning

In this appendix, we give an overview of the TCP/IP and performance tuning points.

For more in depth knowledge on AIX performance tuning, we recommend that you read the *AIX Performance Tuning Guide*, SR28-5930 and *RS/6000 SP System Performance Tuning*, SG24-5340.

A.1 TCP/IP overview

Before we discuss AIX TCP/IP network tuning, we should get a general understanding of how the different layers of the TCP/IP protocol stack interact.

TCP/IP consists of several communication layers. There are parameters that impact the different protocol layers. They can best be understood by breaking them down into categories:

- The `no` parameters are the initial network options that affect TCP, UDP and IP and are independent to the adapter type.
- The MTU, or maximum transmission unit, is the largest possible packet size that can be sent on a specific physical medium (Ethernet, token-ring, SP switch, and so on).
- The adapter queues specify the number of packets that can be queued on a specific adapter while it is sending or receiving data. These are specific to an adapter even if there are other adapters of the same type.

For a review of the TCP/IP layer model and to clarify the interrelationships, let's break this down further, step by step:

1. An application performs a write request. Data is copied from the application's working buffer to the socket send buffer.
2. The socket layer passes the data to TCP or UDP.
3. For remote networks, if the data is larger than the maximum segment size (MSS), TCP breaks the data into fragments that comply with the MSS.
4. For local networks, if the data is larger than the MTU, TCP breaks the data into fragments that comply with the MTU.
5. UDP leaves the fragmentation to the IP layer.
6. The interface layer makes sure that no packet exceeds the MTU.
7. The packets are then placed on the adapter output queue, and transmitted to the receiving system.
8. The receiving host places the incoming packets on the adapter's receive queue. They are then passed up to the IP layer.
9. The IP layer then determines if any fragmentation has taken place due to the MTU. If so, it puts the fragments back to their original form and passes the packets to TCP or UDP.
10. TCP reassembles the original segments and puts them on the socket receive buffer in kernel memory or UDP passes the data on to the socket receive buffer in kernel memory.

11. The application's read request causes the appropriate data to be copied from the socket receive buffer to the buffer in the application's working area.

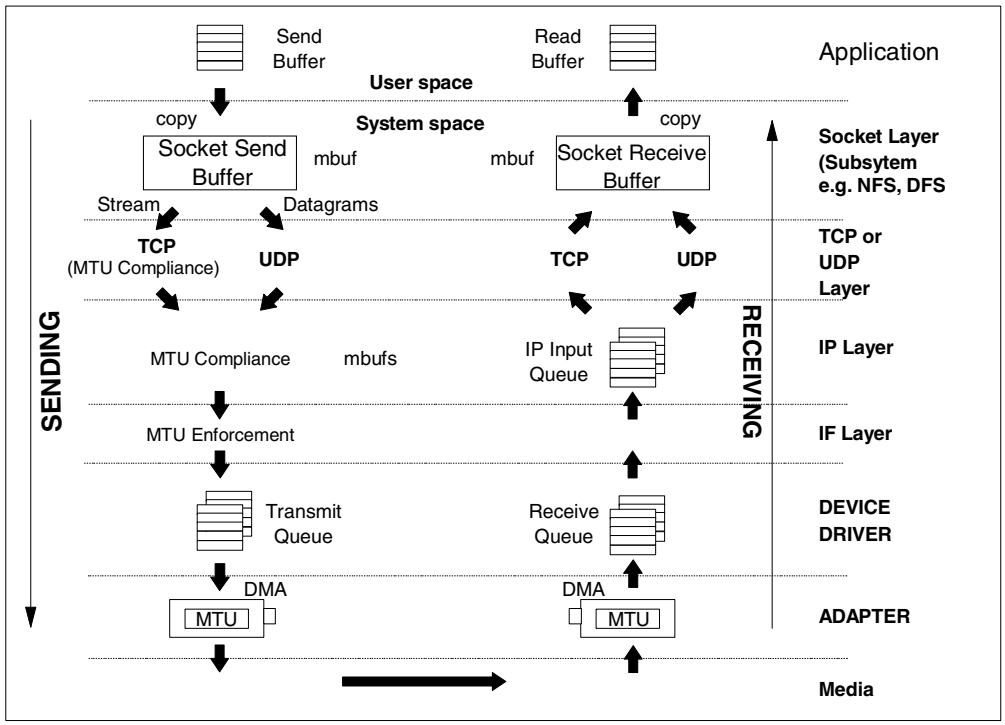


Figure 111. TCP/UDP/IP data flow

There are many parameters that can affect your network performance.

At the device driver layer you have your transmit queue size marked by the parameters xmt_que_size. You also have your receive queue size marked by rec_que_size.

At the interface layer you have enforcement of the MTU or segment size as it pertains to what type of network media is being used, Ethernet, token-ring, or others.

At the transport layer your performance parameters are set by tcp/udp send/recvspace.

You also have the socket layer, between the transport and application layers, the parameter sb_max, which determines the maximum amount of memory or mbuf space that can be used by TCP or UDP for socket buffers for each socket.

Lastly, the parameters listed above all impact system memory. `thewall` parameter determines the maximum amount of buffer space that can be used across the entire communication subsystems.

Now, we will discuss these in more detail.

A.2 Maximum Transmission Unit (MTU)

The Maximum Transmission Unit (MTU) specifies the maximum size of packets (including all the protocol headers) that can be transmitted on a network. For an overview see Figure 116 on page 175. All systems on the same physical network must have the same MTU. The MTU can be displayed using the `netstat -i` command. Table 10 on page 169 gives an overview of common network adapters and their related MTU sizes.

The MTU value can be changed per adapter using the `ifconfig` command or via SMIT. Because all systems on the same physical network should have the same MTU, any changes made should be made simultaneously. The change is effective across system boots.

Table 10. Maximum Transmission Units

Network Type	Default MTU	Maximum MTU	Optimal
Ethernet	1500	1500	1500
Token Ring	1492	17284	4096
FDDI	4352	4352	4352
ATM	9180	65530	9180
Gigabit Ethernet	9000		9000

A.3 Adapter queue size

There is a fixed number of adapter queue slots to stage packets in each network adapter device driver for traffic to that network. The transmit adapter queue length specifies the maximum number of packets for the adapter. The send and receive pools are separate buffer pools as shown in Figure 112 on page 170.

If the adapter queue size is exceeded, subsequent packets are discarded by the adapter device driver, resulting in dropped packets. This results in a transmit time out in the TCP layer, which leads to a rollback of the TCP window and the resending of data. For UDP the result is lost packets.

Adapter queue overflows can be detected by looking at the errors logged in the adapter counters as "S/W Transmit Queue Overflows". For Ethernet, token ring, FDDI and ATM the adapter statistics can be seen by using the `entstat`, `tokstat`, `fddistat` and `atmstat` commands respectibelly.

Most communication drivers provide a set of tunable parameters to control transmit and receive resources. These parameters typically control the transmit queue and receive queue limits, but may also control the number and size of buffers or other resources. They limit the number of buffers or packets that may be queued for transmit or limit the number of receive buffers that are available for receiving packets. For an example see Table 11 and Table 12 on page 170. These parameters can be tuned to ensure enough queueing at the adapter level to handle the peak loads generated by the system or the network.

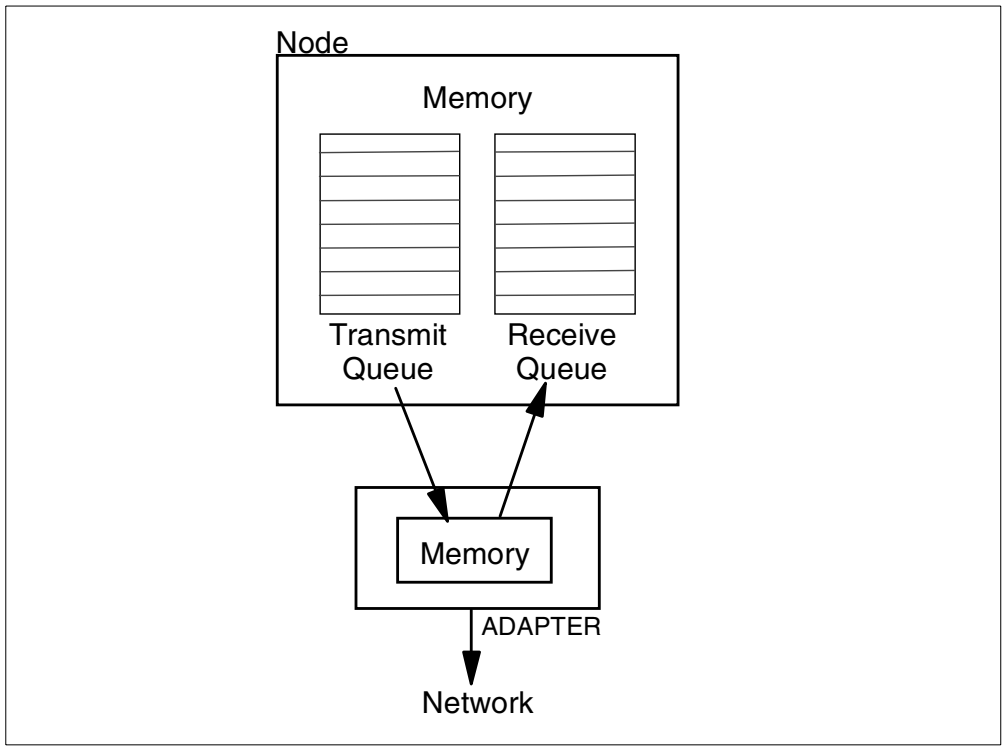


Figure 112. Adapter queue overview

Table 11. Transmit queue size examples: MCA

Adapter	Default	Range
Ethernet	512	20 - 2048
10/100 Ethernet	64	16, 32, 64, 128, 256
Token Ring	99 or 512	32 - 2048
FDDI	512	3 - 2048
ATM/155 ATM	512	0 - 2048

Table 12. Transmit queue size examples: PCI

Adapter	Default	Range
Ethernet	64	16, 32, 64, 128, 256
10/100 Ethernet	256 - 512	16, 32, 64, 128, 256
Token Ring	96 - 512	32 - 2048
FDDI	30	3 - 250
155 ATM	100	0 - 4096

A.3.1 Transmit and receive queues

For transmit, the device drivers may provide a "transmit queue" limit. There may be both hardware queue and software queue limits, depending on the driver and adapter. Some drivers have only a hardware queue, some have both hardware and software queues. Some drivers control the hardware queue internally and

only allow the software queue limits to be modified. Generally, the device driver will queue a transmit packet directly to the adapter hardware queue. If the system CPU is fast relative to the speed of the network, or on an SMP system, the system may produce transmit packets faster than they can be transmitted on the network. This will cause the hardware queue to fill. Once the hardware queue is full, some drivers provide a software queue and subsequent packages will be queued to it. If the software transmit queue limit is reached, then the transmit packets are discarded. This can affect performance because the upper level protocols must then retransmit the discarded packets.

A typical example would be that you set your adapter queue length to 30. Assuming that the MTU of that adapter is 1500, you have set the maximum amount of data which that adapter can hold to 45,000 bytes. Figure 113 illustrates the different MTU ratios. If you try and stage more packets to an adapter, then the packets that arrive when this queue is full get thrown away.

Receive queues are the same as transmit hardware queues.

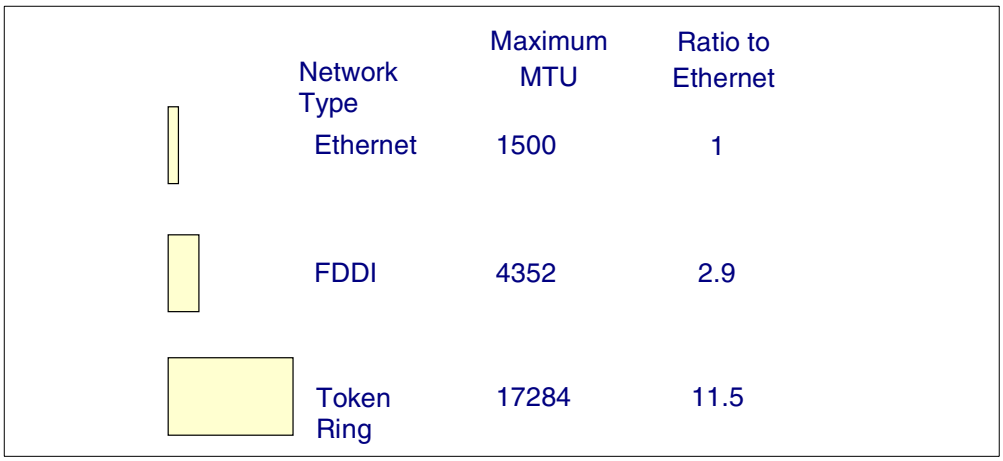


Figure 113. MTU ratio

A.3.2 Adapter queue settings

To show the adapter configuration settings, you can use the `lsattr` command or SMIT. For example, to display the default values of the settings, you can use the command:

```
lsattr -D -l <adapter - name>
```

and to display the current values, you can use:

```
lsattr -E -l <adapter - name>
```

Finally, to display the range of legal values of an attribute (for example, `xmt_que_size`) for a given adapter (for example, Token Ring), you can use the command:

```
lsattr -R -l tok0 -a xmt_que_size
```

Different adapters have different names for these variables. For example, they may be named `sw_txq_size`, `tx_que_size`, `xmt_que_size` to name a few for the transmit queue parameter. The receive queue size and/or receive buffer pool

parameters may be named `rec_que_size`, `rx_que_size`, or `rv_buf4k_min`, for example.

The easiest way to change the adapter settings is by using SMIT. The other method is to use the `chdev` command.

For example, to change `tx_que_size` on `en0` to `1024`, use the following sequence of commands. Note that this driver only supports four different sizes, so it is better to use SMIT to see the valid values.

```
ifconfig en0 detach
chdev -l ent0 -a tx_que_size=1024
ifconfig en0 up
```

A.3.3 Adapter tuning recommendations

If you consistently see output errors when running the `netstat -i` command, increasing the size of the `xmt_que_size` parameter may help. Check also the adapter transmit average overflow count. As a rule of thumb, always set `xmt_que_size` to the maximum.

One way to tune IP to prevent exceeding the adapter queue size is to reduce the aggregate TCP window size or `udp_sendspace` so that it is less than the transmit queue size times the segment size (MTU) on the network. This usually results in optimal throughput and slightly higher system overhead for network traffic. If multiple connections are sharing an adapter at the same time, the aggregate TCP window size across all connections should be slightly less than the transmit queue size times the segment size for the network media type.

A.4 TCP maximum segment size (MSS)

The TCP protocol includes a mechanism for both ends of a connection to negotiate the maximum segment size (MSS) to be used over the connection. In other words, the MSS is the largest segment or “chunk” of data that TCP will send to a destination. Each end uses the Options field in the TCP header to advertise a proposed MSS. The MSS that is chosen is the smaller of the values provided by the two ends.

The purpose of this negotiation is to avoid the delays and throughput reductions caused by fragmentation of the packets when they pass through routers or gateways and reassembly at the destination host.

The value of MSS advertised by the TCP software during connection setup depends on whether the other end is a local system on the same physical network (that is, the systems have the same network number) or whether it is on a different, remote, network.

A.4.1 Subnetting and the subnetsarelocal

Several physical networks can be made to share the same network number by subnetting. The easiest way to understand subnet addressing is to imagine that a site has a single class B IP network assigned to it, but has two or more physical networks. Only local routers know that there are multiple physical nets and how to route among them.

Conceptually, adding subnets only changes the interpretation of the IP address slightly. Instead of dividing the 32-bit IP address into a network prefix and a host suffix, subnetting divides the address into a network portion and a local portion. The interpretation of the network portion remains the same as for networks that do not use subnetting. The interpretation of the local portion is left up to the site.

The example in Figure 114 shows subnet addressing with a class B address that has a 2-octet internet portion and 2-octet local portion. In the example one octet of the local portion identifies a physical network and the other octet identifies a host on that network.

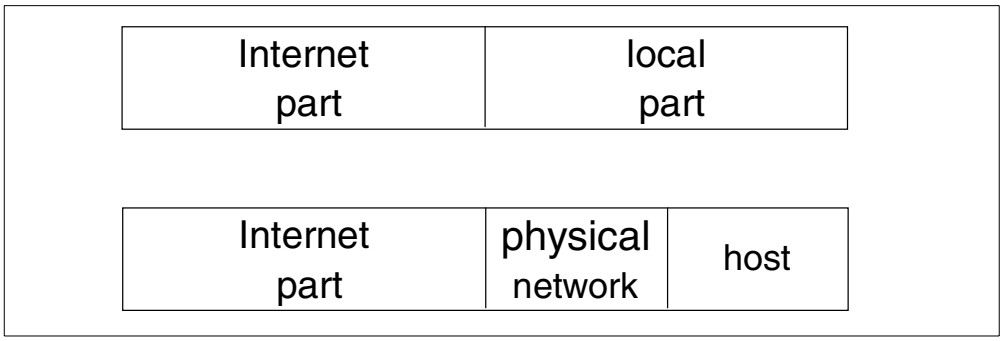


Figure 114. Subnet addressing

In AIX the `no` option subnetsarelocal specifies, on a system-wide basis, whether subnets are to be considered local or remote networks. With subnetsarelocal=1 (the default), Host A on subnet 1 considers Host B on subnet 2 to be on the same physical network.

The consequence of this is that when Host A and Host B establish a connection, they negotiate the MSS, assuming they are on the same network. Each host advertises an MSS based on the MTU of its network interface. This usually leads to an optimal MSS being chosen.

This approach has several advantages:

- It does not disable or override the TCP MSS negotiation, so that small differences in the MTU between adjacent subnets can be handled appropriately.
- In order for small differences between adjacent subnets in the MTU to be handled appropriately, it does not disable or override the TCP MSS negotiation.

Figure 116 on page 175 shows the relations between MSS and MTU. The value of MSS is determined as follows:

1. If the destination is local, that is, if the network ID and the subnet ID of the destination IP address are the same as the local ones, then the MSS value is calculated based on the MTU value of the outgoing interface, as follows:

   MSS = MTU - (20 + 20)if rfc1323 = 0

   MSS = MTU - (20 + 20 + 12)if rfc1323 = 1

   since the TCP header is 20 bytes, and the IP header is also 20 bytes long. Furthermore, enabling rfc1323 costs an additional 12 bytes.

2. If the destination address is remote, that is, if the network ID of the destination IP address is different from the local one, then TCP uses a global variable that determines the MSS.
3. If the destination has the same network ID as the local one but with a different subnet ID, then the destination could be either local or remote. The no option of subnetsarelocal lets you specify whether subnets on the same network are local or remote. If local, you would follow item (1) above. If remote, you would follow item (2) above.

Since segmentation occurs at the TCP level, if the total packet is less than the MTU, then IP does not do anything to the packet other than preappend a header and send the data to the interface layer. TCP on the receiving side will reassemble the packet in the correct sequence and deliver the data to the application. Figure 115 illustrates this.

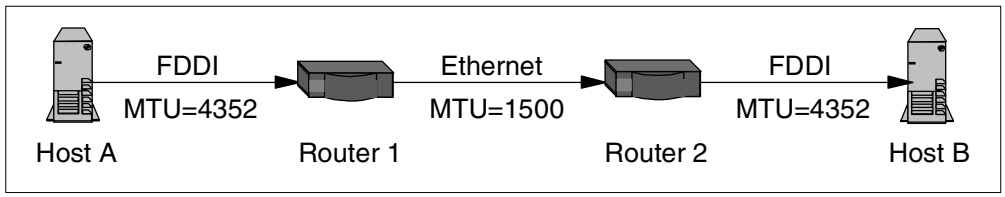


Figure 115. Inter-subnet fragmentation

In this scenario, Hosts A and B would establish a connection based on a common MTU of 4352. A packet going from A to B would be fragmented by Router 1 and defragmented by Router 2, and the reverse would occur going from B to A. Source and destination must both consider subnets to be local.

A.4.2 TCP data flow

TCP breaks the data into smaller pieces called segments to comply with the MSS. See also Figure 116 on page 175. The resulting IP datagram is 40 or 52 bytes larger: 20 bytes for the IP header and 20 bytes for the TCP header and an optional 12 bytes for rfc1323.

When a connection is established, each end has the option of announcing the tcp_sendspace it is willing to receive depending upon its buffer space. Since the moving-window technique requires that the two systems be able to buffer the same amount of data, the effective window size is set to the lesser value in both directions. The nominally available extra space for buffering output shown in Figure 117 on page 175 is never used.

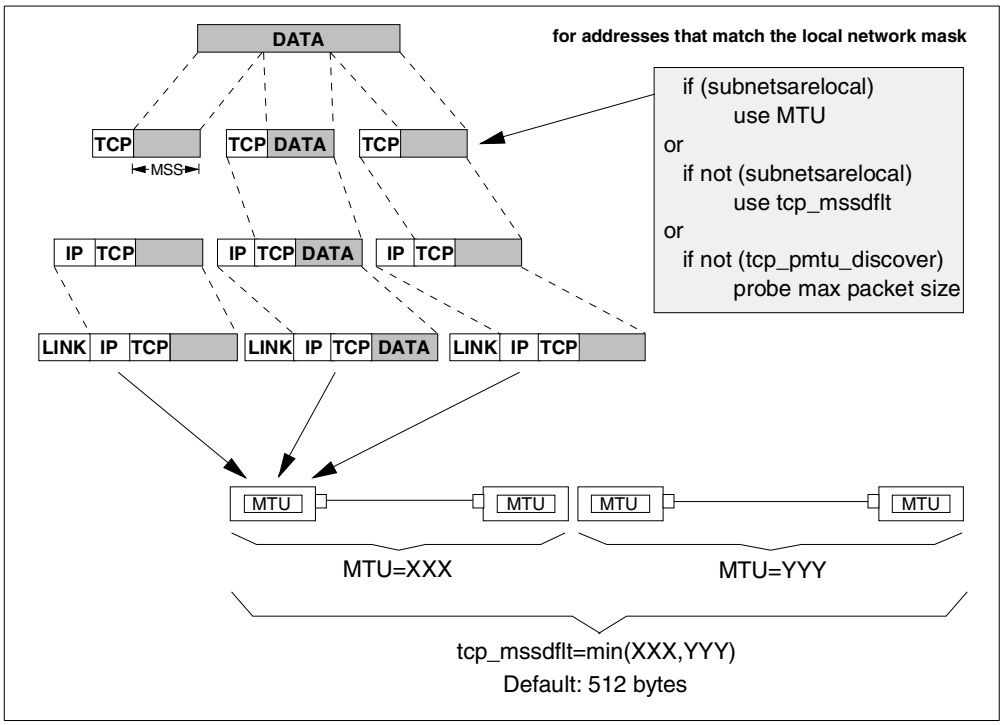


Figure 116. TCP data flow

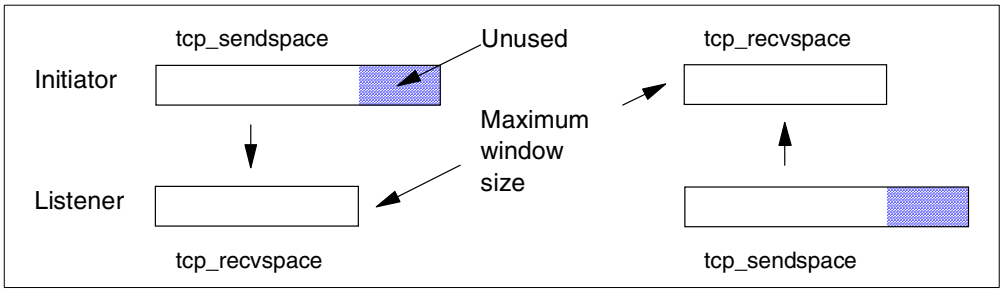


Figure 117. TCP window size

In general, the larger the MSS the better, as long as it is not so large that it causes fragmentation at the IP layer.

A.5 TCP sliding window

TCP enforces flow control of data from the sender to the receiver through a mechanism referred to as sliding window. This helps ensure delivery to a receiving application. The size of the window is defined by the tcp_sendspace and tcp_recvspace values.

The window is the maximum amount of data that a sender can send without receiving any ACK segment. This obviously contributes to performance improvement. A receiver always advertises its window size in the TCP header of the ACK segments.

In the example in Figure 118, the sending application is sleeping because it has attempted to write data that would cause TCP to exceed the send socket buffer space (that is, tcp_sendspace). The sending TCP has sent the last part of rec5, all of rec6 and rec7, and the beginning of rec8. The receiving TCP has not yet received the last part of rec7 or any of rec8.

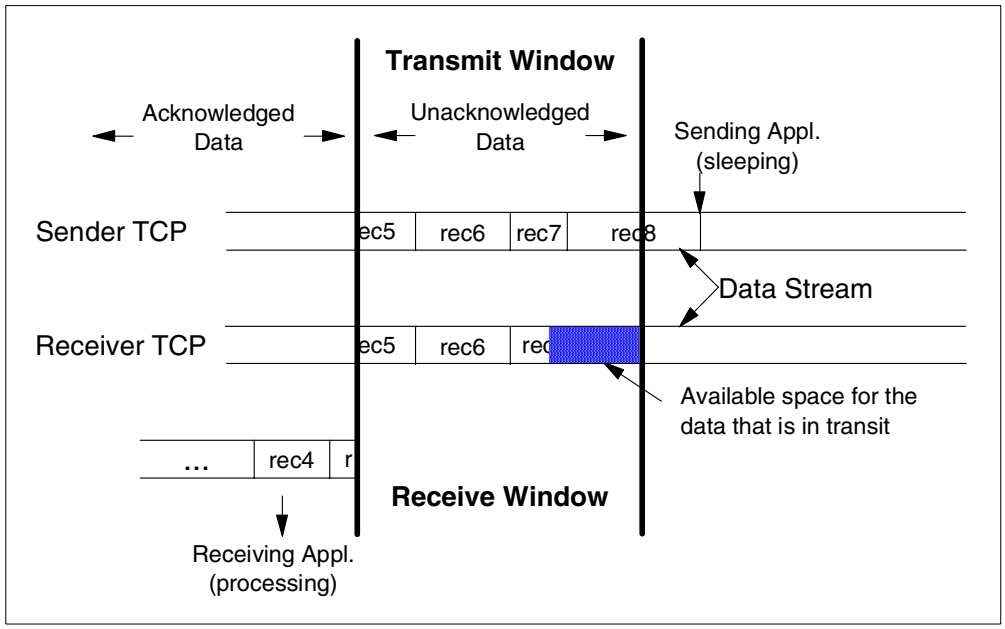


Figure 118. TCP sliding window

The receiving application got rec4 and the beginning of rec5 when it last read the socket, and it is now processing that data. When the receiving application next reads the socket, it will receive (assuming a large enough read), the rest of rec5, rec6, and as much of rec7 and rec8 as has arrived by that time.

In the course of establishing a session, the initiator and the listener converse to determine their respective capacities for buffering input and output data. The smaller of the two sizes defines the size of the effective window. As data is written to the socket, it is moved into the sender's buffer. When the receiver indicates that it has space available, the sender transmits enough data to fill that space (assuming that it has that much data). It then informs the sender that the data has been successfully delivered. Only then does the sender discard the data from its own buffer, effectively moving the window to the right by the amount of data delivered. If the window is full because the receiving application has fallen behind, the sending thread will be blocked.

Nowadays we have a lot of high-speed network media and memory for a workstation. The maximum of 64 KB for a window may not be big enough for such an advanced environment. TCP has been enhanced to support such situations by RFC 1323, TCP Extensions for High Performance.

If the rfc1323 parameter is 1, the maximum TCP window size is 4 GB (instead of 64 KB). Figure 119 on page 177 illustrates this TCP enhancement.

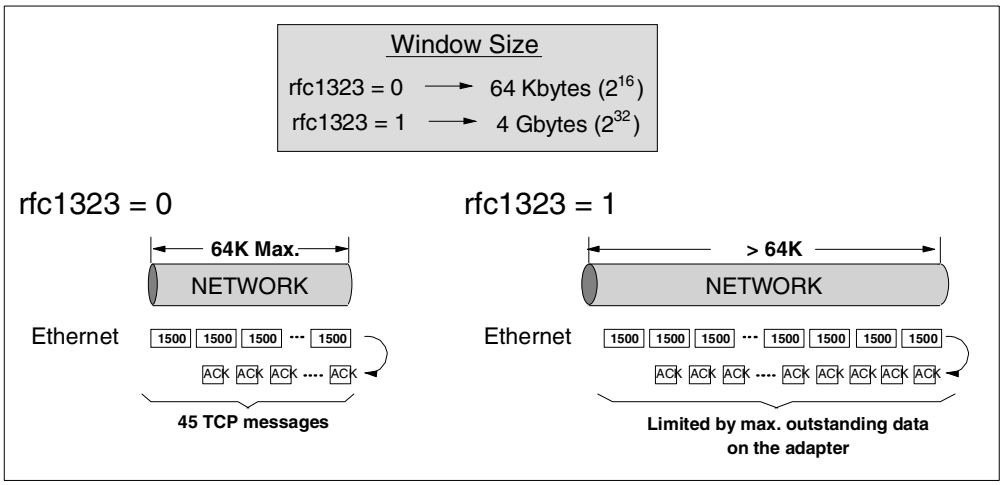


Figure 119. rft1323 - TCP extension

There is, of course, no such thing as free function. The additional operations performed by TCP to ensure a reliable connection result in about 7 to 12% higher CPU time than in UDP.

A.6 Socket layer

Sockets provide the application program interface (API) to the communication subsystem. There are several types of sockets that provide various levels of service by using different communication protocols. Sockets of type SOCK_DGRAM use the UDP protocol. Sockets of type SOCK_STREAM use the TCP protocol. See Figure 120 on page 178 for on overview.

The semantics of opening, reading, and writing to sockets are similar to those for manipulating files.

The sizes of the buffers in system virtual memory (that is, the total number of bytes from the mbuf pools) that are used by the input and output sides of each socket are limited by system-wide default values (which can be overridden for a given socket by a call to the setsockopt() subroutine):

- udp_sendspace and udp_recvspace

  The buffer sizes per datagram socket.

- tcp_sendspace and tcp_recvspace

  The buffer sizes per stream socket.

Note

Socket send or receive buffer sizes are limited to no more than *sb_max* bytes, because *sb_max* is a ceiling on buffer space consumption. The two quantities are not measured in the same way, however.

The socket buffer size limits the number of bytes that can be held in the socket buffers. *sb_max* limits the amount of space in network memory pool buffers that can be allocated to a socket at any given time.

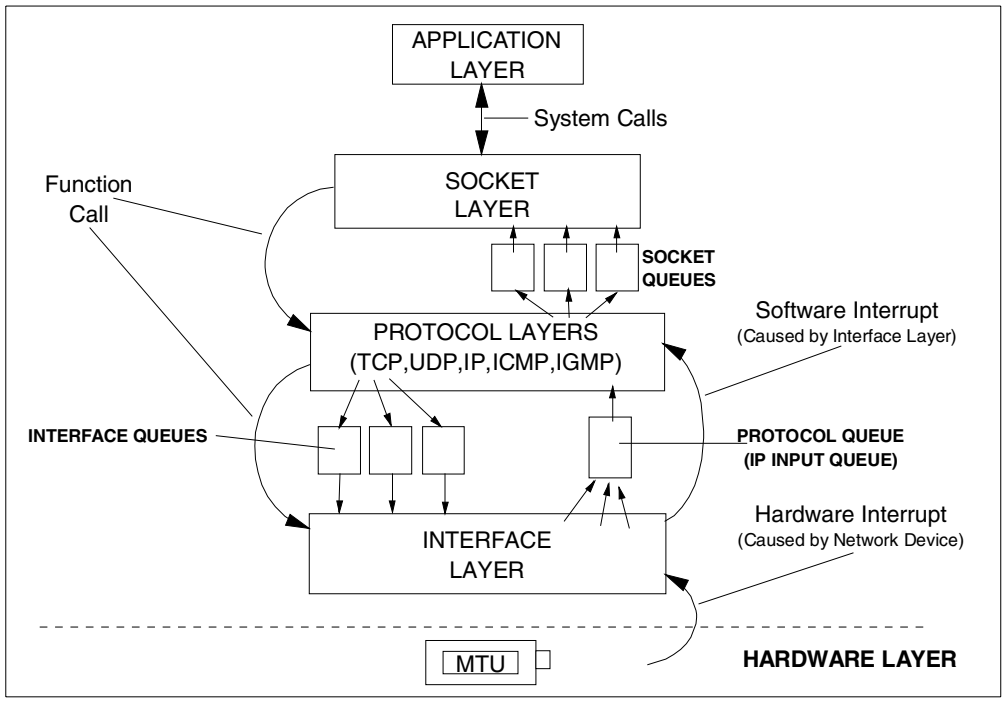


Figure 120. Socket layer

A.7 Communication subsystem memory management

AIX can efficiently allocate and reclaim pinned (physical) memory within the communication subsystem. This is achieved through the use of mbufs and additional buffers called clusters.

Use `netstat -m` to get on overview of cluster usage and to which subsystems clusters are allocated. See Figure 121 on page 179 for an overview.

As you can see from the command output, each CPU in an SMP system has a dedicated network memory pool. This is to improve performance by having pools dedicated per CPU. This eliminates the need to look at mbuf pool access.

```
Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed    free   hiwat   freed
32              135     17392      0     121     640       0
64               77      1826      0      51     320       0
128              52      2678      0      12     160       0
256              42   3129745      0      86     384       0
512              51      3594      0      29      40       3
1024             22     19845      0      62     100       0
2048              0       768      0       4     100       0
4096              2       664      0       5     120       0
16384             1      1026      0      18      24       7
32768             1         1      0       0    2048       0

******* CPU 1 *******
By size       inuse     calls failed    free   hiwat   freed
32               19     12593      0     109     640       0
64               14      2522      0      50     320       0
128               6      2345      0      26     160       0
256              59   2980466      0      69     384       0
512              23      3946      0      41      40       0
1024              3     15438      0      65     100       0
2048              0       716      0       6     100       0
4096              1       641      0       1     120       0
16384             0       916      0      18      24       8

******* CPU 2 *******
By size       inuse     calls failed    free   hiwat   freed
32                9      2903      0     119     640       0
...

By type       inuse     calls failed  memuse  memmax  mapb

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

Figure 121. netstat -m output

By setting the extendednetstat parameter with the `no` command, you will get more detailed information, for example:

```
no -o extendednetstat=1
```

Figure 122 on page 180 shows the output of the `netstat -m` command after the modification. In this screen you will notice that the netstat command now gives more information about the use of the various buffers.

The inuse column shows how many pinned pieces of kernel virtual memory are currently used, which means that they always reside in physical memory and are never paged out.

```
Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed    free   hiwat   freed
32              135     17394      0     121     640       0
64               77      1827      0      51     320       0
128              53      2760      0      11     160       0
256              42   3130406      0      86     384       0
512              51      3596      0      29      40       3
1024             22     19847      0      62     100       0
2048              0       768      0       4     100       0
4096              2       664      0       5     120       0
16384             1      1026      0      18      24       7
32768             1         1      0       0    2048       0

******* CPU 1 *******
By size       inuse     calls failed    free   hiwat   freed
32               19     12598      0     109     640       0
64               14      2526      0      50     320       0
...

By type       inuse     calls failed  memuse  memmax  mapb
mbuf             21       452      0    5376   10240     0
socket          321         8      0    1348    1120     0
pcb             728         4      0      78     128     0
fragtbl           0         4      0       0      32     0
...

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

Figure 122. netstat -m with extendednetstat=1

In addition to avoiding duplication, sharing the mbuf and cluster pools allows the various layers to pass pointers to one another, reducing mbuf management calls and copying of data.

The only network option used to tune the network maximum memory pool is thewall. Since the new scheme is self-tuning, there is no need to tune any other parameter. If the system memory requirements exceed thewall, then it will start to drop packets.

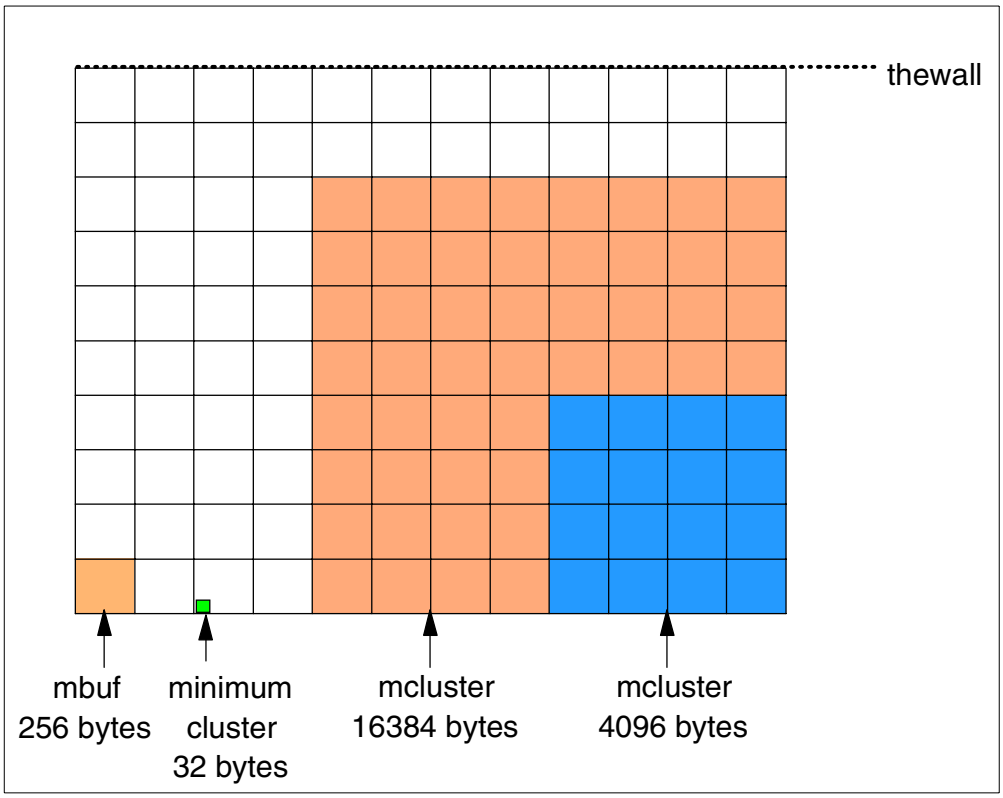


Figure 123. The network memory pool

A.8 Interface specific network options for AIX 4.3.3

Many AIX systems have multiple network interfaces combining traditional and high-speed TCP/IP interfaces on a single system. Up to Version 4.3.3, AIX provides a single set of system-wide values, set using the `no` command, for the key IP interface network tuning parameters making it impossible to tune a system that has widely differing network adapter interfaces.

As new high-speed (100 Mbps or more) network options became available, AIX system administrators learned these TCP/IP interfaces must be specially tuned to achieve good, high-speed performance.

So system administrators faced performance trade-off decisions; perhaps tuning for one TCP/IP network interface while sacrificing performance on the other, or balancing the two with options that were not totally suited for either interface type. Ideally, system administrators should be able to tune for each TCP/IP interface individually for best performance. This feature is now included in AIX V 4.3.3.

Starting with Version 4.3.3 AIX offers a feature called Interface Specific Network Options (ISNO) which allows IP network interfaces to be custom tuned for the best performance. Values set for an individual interface take precedence over the system-wide values set with the network option (`no`) command. The feature is enabled (the default) or disabled for the whole system with the `no` command's `use_isno` option. This single point ISNO disable option is included as a diagnostic

tool to eliminate potential tuning errors if the system administrator needs to isolate performance problems.

A.8.1 Implementation overview

The AIX V4.3.3 programmers and performance analysts should note that the ISNO values will not show up in the socket; this means they cannot be read by `getsockopt()` until after the TCP connection is made. The interface this socket will actually be using is not known until the connection is complete, so the socket reflects the system `no` defaults. Once the connection is accepted, ISNO values are put into the socket.

Five new parameters:

- rfc1323
- tcp_nodelay
- tcp_sendspace
- tcp_recvspace
- tcp_mssdflt

have been added for each supported network interface. When set for a specific interface, these values override the corresponding `no` option values set for the system. These parameters are available for all of the mainstream TCP/IP interfaces we checked -- token ring, FDDI, 10/100 Ethernet, and Gigabit Ethernet -- except the css# IP interface on the SP switch. As a simple workaround, SP switch users can set the tuning options appropriate for the switch using the system-wide `no` command, then use the ISNOs to set the values needed for the other system interfaces.

Note

These options are set for the TCP/IP interface such as en0 or tr0 but not for the network adapter interfaces such as ent0 or tok0.

A.8.2 How to use the new options

The five new ISNO parameters cannot be displayed or changed using SMIT. Here are commands that can be used to first verify system and interface support and then set and verify the new values.

1. Verify general system and interface support using the `no` and `lsattr` commands.

   Make sure the `use_isno` option is enabled using the following command or a variation:

   ```
   $ no -a | grep isno
   use_isno=1
   ```

   Make sure the interface supports the five new ISNOs using the `lsattr -El` command:

   ```
   $ lsattr -E -l en0 -H
   attribute       value     description
    rfc1323                 N/A
   ```

```
tcp_nodelay             N/A
tcp_sendspace           N/A
tcp_recvspace           N/A
tcp_mssdflt             N/A
```

2. Set the interface specific values, using either the `ifconfig` or `chdev` commands. The `ifconfig` command sets values temporarily so it is good for testing. The `chdev` command alters the ODM, so custom values return after system reboots.

   For example, to set the `tcp_recvspace` and `tcp_sendspace` to 64 KB and enable `tcp_nodelay` use one of the following methods:

```
$ ifconfig en0 tcp_recvspace 65536 tcp_sendspace 65536 tcp_nodelay 1
$ chdev -l en0 -a tcp_recvspace=65536 -a tcp_sendspace=65536 -a \
  tcp_nodelay=1
```

   Or, assuming the `no` command reports an rfc1323=1 global value, user root can turn rfc1323 off for all connections over en0 with the following commands:

```
$ ifconfig en0 rfc1323 0
$ chdev -l en0 -a rfc1323=0
```

   Verify the settings using the `ifconfig` or `lsattr` commands.

```
 $ ifconfig en0
en0:flags=e080863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT
,64BIT>  inet 9.19.161.100 netmask 0xffffff00 broadcast 9.19.161.255
 tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1 rfc1323 0
 $ lsattr -El en0
 rfc1323    0          N/A                   True
 tcp_nodelay1          N/A                   True
 tcp_sendspace65536    N/A                   True
 tcp_recvspace65536    N/A                   True
 tcp_mssdflt           N/A                   True
```

Note

In our simple tests, we were pleased to see that changes made via `chdev` were reflected in the `ifconfig` output. Specifically, our test machine was already set to 64 KB send and receive space, so we used `ifconfig` to set a 16 KB value. We ran `ifconfig en0` to verify that setting. Next, we set the 64 KB value using `chdev`, executed the `ifconfig en0` command and discovered that the `ifconfig` output reflected the new 64 KBvalue we'd just set using `chdev`.

A.8.3 References for the ISNO

More information about the interface specific network options (ISNO) can be found in:

- *AIX Version 4.3 Differences Guide*, SG24-2014
- *AIX Commands Reference*, available at:

  `http://www.austin.ibm.com`.

The following is an excerpt from the `ifconfig` command page. Here are two clarifications to the command page text:

1. The main purpose of RFC 1323 is to allow TCP to increase its window size larger than 64 KB (controlled by tcp_recvspace) for large MTU adapters.

Without RFC 1323, a 64K MTU adapter could only have one packet outstanding, which results in very poor performance.

2. The tcp_sendspace option only affects send space buffering in the kernel. The tcp_recvspace value on the receiver system is the only value that affects TCP maximum window size.

In AIX Version 4.3.3 and later versions, the following network options, commonly known as ISNO (Interface Specific Network Options), can be configured on a per interface basis as noted below.

Table 13. Network options for AIX V4.3.3

rfc1323 [0 | 1]	Enables or disables TCP enhancements as specified by RFC 1323, TCP Extensions for High Performance. A value of 1 specifies that all TCP connections using this interface will attempt to negotiate the RFC enhancements. A value of 0 disables rfc1323 for all connections using this interface. The SOCKETS application can override this ISNO and global behavior on individual TCP connections with the setsockopt subroutine.
-rfc1323	Removes the use of ISNO for rfc1323 for this network. A SOCKETS application can override the global behavior on individual TCP connections using the setsockopt subroutine.
tcp_mssdflt Number	Sets the default maximum segment size used in communicating with remote networks. If communicating over this interface, a socket uses Number as the value of the default maximum segment size.
-tcp_mssdflt	Removes the use of ISNO for tcp_mssdflt. The global value, manipulated via /usr/sbin/no, is used instead.
tcp_recvspace	Specifies the default socket buffer size for interface sockets receiving data. The buffer size affects the window size used by TCP. (See the no command for more information.)
-tcp_recvspace	Removes the use of ISNO for tcp_recvspace. The global value is used instead.
tcp_sendspace	Specifies the default socket buffer size for interface sockets sending data. The buffer size affects the window size used by TCP. (See the no command for more information.)
-tcp_sendspace	Removes the use of ISNO for tcp_sendspace. The global value is used instead.
tcp_nodelay [0 | 1]	Specifies that sockets using TCP over this interface follow the Nagle algorithm when sending data. By default, TCP follows the Nagle algorithm.
-tcp_nodelay	Removes the use of ISNO for the tcp_nodelay option.

Appendix B. Special notices

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

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

AFP
AIX
AS/400
DB2
DB2 Universal Database
IBM
MQSeries
Netfinity
RS/6000
S/390
SP
System/390
TXSeries
WebSphere
Wizard

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Appendix C. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 189.

- *RS/6000 SP System Performance Tuning*, SG24-5340
- *AIX Version 4.3 Differences Guide*, SG24-2014
- *IBM HTTP Server Powered by Apache on RS/6000*, SG24-5132

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

C.3 Other resources

These publications are also relevant as further information sources:

- *DB2 Universal Database V6.1 for UNIX, Windows, and OS/2 Certification Guide*, by Jonathan Cook, Robert Harbus, and Tetsuya Shirai, ISBN 0-13-086755-1
- *AIX Performance Tuning Guide*, SR28-5930
- *AIX Commands Reference*, available at http://www.austin.ibm.com
- *WebSphere Application Server Advanced Edition: Getting Started* (shipped with the product)
- *Apache: The Definitive Guide*, by Ben Laurie, Peter Laurie, and Robert Denn, ISBN 1565925289
- *Apache Server Bible*, by Mohammed J. Kabir, ISBN 0764532189

C.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- http://www.redbooks.ibm.com
- http://www.spec.org
- http://www.apache.org
- http://www.zeustech.net
- http://www.radview.com
- http://www.merc-int.com
- http://www.rational.com
- http://www.mindcraft.com
- http://www.ibm.com/software/webservers/appserv/siteanalysis.html
- http://www.austin.ibm.com
- http://w3.itso.ibm.com
- http://w3.ibm.com

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

E

F

G

H

I

J

U

V

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5657-00 WebSphere V3 Performance Tuning Guide
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. http://www.ibm.com/privacy/yourprivacy/

SG24-5657-00

Printed in the U.S.A.

WebSphere V3 Performance Tuning Guide

SG24-5657-00