Linux on System z

WebSphere on IBM System z 64-bit and 31-bit studies with J2EE workloads

Linux on System z

WebSphere on IBM System z 64-bit and 31-bit studies with J2EE workloads

Note

Before using this information and the product it supports, read the information in “Notices” on page 49.

Contents

Figures

Tables

About this publication

Authors

Dr. Juergen Doelle

Paul V. Sutera

Acknowledgements

Thank you to these people for their contributions to this project:

Eugene Ong
Robert Wisniewski

The benchmarks were performed at the IBM System z World Wide Benchmark Center in Poughkeepsie, NY.

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this publication, send your comments using IBM Resource Link at http://www.ibm.com/servers/resourcelink. Click **Feedback** on the navigation pane. Be sure to include the name of the document and the specific location of the text you are commenting on (for example, a page number or table number).

Chapter 1. Introduction

This study measures performance and throughput for the WebSphere® Application Server (both 64-bit and 31-bit) on Linux® for IBM® System z® with Java™ 2 Platform, Enterprise Edition (J2EE) workloads.[1]

There are many customer workloads based on Web applications, including those that use Web servers, J2EE-based middleware, and backend databases. For these tests, a workload that heavily exercises all the major J2EE components, but does not make heavy use of resources on the backend database, was chosen. Using this workload, various performance studies of IBM WebSphere 6.1.0.15 on Linux for IBM System z with a DB2® database using Java-based clients, including servlets and Java Server Pages (JSPs), were conducted.

The advantage of 64-bit WebSphere is its ability to make use of a much larger Java Virtual Machine (JVM) heap than the 31-bit version. In theory, a 64-bit machine can address up to 16 exabytes of storage, millions of times more than the physical memory typically found on most computers. Of the available memory dedicated to a Linux system, a JVM heap can occupy a large percentage of that memory, but not all of the memory. Even on a system that only runs WebSphere and Java, there are still memory requirements outside of the heap for thread-level program stack allocation.[1]

Objectives

The objective of these tests was to use a representative workload to test different WebSphere Application Server environments and settings for a Java-based application on Linux for IBM System z. The environment was three-tiered, consisting of:

- Up to three J2EE clients
- A WebSphere Application Server in 31-bit or 64-bit mode, both running on 64-bit Linux for IBM System z
- An IBM DB2 Universal Database™ (UDB) on Linux for IBM System z

The objectives include:

- Study JVM heap size in 64-bit mode.
- Compare 64-bit versus 31-bit performance on WebSphere 6.1.0.
- Study Central Processing Unit (CPU) scaling.

This information will help IT architects design and choose the correct size for middleware and database resources for Web-based J2EE applications that make heavy use of J2EE middleware components such as:

- Servlets
- Java Server Pages (JSPs)
- Enterprise Java Beans (EJBs)

[1] This paper is intended to provide information regarding performance of environments using WebSphere Application Server 6.1. It discusses findings based on configurations that were created and tested under laboratory conditions. These findings may not be realized in all customer environments, and implementation in such environments may require additional steps, configurations, and performance analysis. The information herein is provided 'AS IS' with no warranties, express or implied. This information does not constitute a specification or form part of the warranty for any IBM products.

- EJB Container Managed Persistence
- Java Messaging Services (JMS)
- Transactions
- Database connections

Executive summary

This study explores the performance of a WebSphere Application Server 6.1 system under a customer-like J2EE application workload. This study includes a very detailed description of how the test environment was set up and how the systems were configured. The difference in performance behavior of the 31-bit and the 64-bit WebSphere versions was compared, and the impact of heap size and garbage collection was analyzed.

CPU scaling studies explored the maximum workloads using one, two, four, and eight Central Processing Units (CPUs) and showed a very good linear scaling, making it easy for a system administrator to plan the resources needed for scaling this workload.

Results show that with a special scenario at the highest workload level, which used eight CPUs, the 64-bit WebSphere version with a large heap produced the best results. With the high computing power of the IBM System z10™, the network bandwidth becomes a critical factor. At the highest workload submission rate (the rate that workload transactions are sent to WebSphere), a 10 Gb Ethernet card was needed to manage the traffic from the workload generating clients.

The results show also that the bandwidth of the interconnect between WebSphere and the database has a significant impact on performance. The HiperSockets™ connection under LPAR is a very appropriate connectivity type for this.

Summary

The IBM WebSphere Application Server helps drive business agility by providing developers and IT architects with an innovative, performance-based foundation to build, reuse, run, integrate, and manage Service Oriented Architecture (SOA) applications and services. IBM DB2 Universal Database (UDB) provides a database management system for mission-critical enterprise data. WebSphere as an interface to DB2 provides a means for the Web-enablement of data and business logic. WebSphere introduced support for 64-bit platforms early in Version 6, while continuing to provide a 31-bit WebSphere able to run in compatibility mode.

Java Virtual Machines (JVMs) in 64-bit mode have provided key advantages, including a significantly larger Java heap size, as well as Java code optimizations. The IBM System z 31-bit WebSphere provides a maximum heap size of approximately 1 GB, while the 64-bit WebSphere heap can address as much as 256 TB, far exceeding the available physical memory of the current platforms.

This study set up and used a system that provides a high throughput for a workload using all major J2EE components on the WebSphere Application Server. The characteristics of the workload used in this study are that throughput depends primarily on the submission rate selected when installing the workload. When sufficient resources are available to support a certain submission rate, more resources do not increase the throughput. The first indicators of processor over-utilization are increasing response times and increasing CPU utilization.

While there are advantages of running on a 64-bit JVM WebSphere, there are also disadvantages, including 8-byte pointers and a greater memory requirement overall. This study showed that 64-bit WebSphere could equal or exceed the performance of 31-bit WebSphere with a more conventional middleware-intensive workload. However, in some tests, 64-bit WebSphere showed a slightly degraded performance when compared to 31-bit WebSphere.

While throughput data from this workload is fairly identical across both architectures, Central Processing Unit (CPU) percent-busy statistics showed that 64-bit WebSphere is, in scenarios with high workloads, able to use its much larger heap size to provide efficiencies of scale at the higher workload submission rates.

At lower submission-rate workloads, 31-bit WebSphere still had a performance advantage in terms of shorter response times, because the number of stored objects (EJBs, JSPs) was smaller and fit comfortably in the 31-bit 1 GB (1024 MB) heap. Garbage collection durations were short, even if frequent. The comparison studies were prepared by first studying the optimal heap size for the 64-bit WebSphere JVM. It was found that for the workload used in this study, a Java heap size ranging between 70% and 75% of the system memory resulted in the best trade-off between maximum throughput, short response times, and low CPU utilization.

A CPU utilization rate greater than 90% was not observed for the higher workloads, indicating an unidentified bottleneck. This bottleneck might be the HiperSockets connection between the WebSphere Application Server and the database. Additional investigation would be required. The high CPU utilization rate of 97% for the run with one CPU on 64-bit WebSphere becomes a critical threshold for a system running with HiperSockets. This high utilization is very likely the reason for the high response times observed on the run with one CPU.

The CPU load of the DB2 LPAR is generally quite light for the transaction workload. A full CPU is rarely used unless there are already eight CPUs used on the WebSphere system. The workload generally uses short data records and does not use complex SQL statements, so the backend database load is expected to be light. The higher cost on the 64-bit WebSphere at a submission rate of 110 strongly suggests that the WebSphere CPU load is too high for the HiperSockets connection, because missing CPU resources on the middleware will cause overhead on the database.

The workload also shows good linear scaling until the highest workload. At the highest workload, throughput declines, falling away from a linear relationship. Here, the 64-bit version performs better than the 31-bit version. The poor response times at the highest workload are an indication of resource contention. It is assumed that the critical resource here is again the interconnect to the database. The database shows a decline in throughput and CPU at a fixed ratio (that is, the CPU cost per transaction stays the same), indicating that the database waits to receive requests from the application sever. Because there is no contention for CPU, this strongly suggests that the interconnect is the cause of the bottleneck.

The high response times with the workload submission rate of 600 in LPAR mode could be easily attributed to network I/O traffic from the clients reaching the limit of the 1 Gb Ethernet from the WebSphere system being tested. In this study, the limiting factor was the amount of packages, where the number of 80 000 packages could be considered as close to the upper limits of a 1 Gb Ethernet interface or card. Using a 10 Gb Ethernet card improved that situation.

Chapter 2. Hardware and software configuration

To perform the 64-bit and 31-bit WebSphere study with J2EE workloads, a customer-like environment was created.

Server hardware and software - LPAR

The server side of the 64-bit and 31-bit WebSphere study with J2EE workloads used two LPARs, one for the WebSphere application server and one for the DB2 database server.

IBM WebSphere Version 6.1 host and IBM DB2 Universal Database host

There are two LPARs on a 56-way IBM System z10 Enterprise Class (z10 EC), 4.4 GHz, model 2097-E56. They are equipped as described in Table 1.

Table 1. Server hardware

LPAR	Description
LPAR 1 for WebSphere Application Server	• Between one and ten physical CPUs dedicated • 8 GB central storage • One 1 Gb OSA card for client connectivity on an isolated performance LAN • One 10 Gb OSA card for client connectivity, also on an isolated performance LAN • One 1 Gb OSA card for administration • HiperSockets connection to the DB2 UDB LPAR
LPAR 2 for DB2 database server	• Four physical CPUs dedicated • 8 GB central storage • One 1 Gb OSA card for administration • HiperSockets connection enabled between WebSphere and DB2 UDB LPARs

Storage setup

The server side of the 64-bit and 31-bit WebSphere study with J2EE workloads used disks for the operating system, applications, and databases. The server storage is described in Table 2.

Table 2. Server storage

Disks	Description
Operating system and applications on two Linux host systems	Four disks on an IBM DS8000® configured with 16 Logical Control Units (LCUs).
Database data disks and DB2 log files	41 ECKD™ mod 9 disks employed using Logical Volume Manager (LVM) with striping over 16 LCUs.

Server software

The server side of the 64-bit and 31-bit WebSphere study with J2EE workloads used software for the operating system and applications. The server software is described in Table 3.

Table 3. Server software

Product	Version/Level
IBM DB2 Universal Database Enterprise Server	Version 9.5 fixpack 1
Novell SUSE Linux Enterprise Server	SLES 10, SP2 64-bit
Retailing customer workload	N/A
WebSphere Application Server	Version 6.1.0 fixpack 15, 64-bit and 31-bit

Client hardware and software

The client side of the 64-bit and 31-bit WebSphere study with J2EE workloads used IBM System x® processors, a operating system, and applications.

Client hardware

The client hardware consists of these processors:

- IBM System x X336 2-way 3.06 GHz Intel® with 8 GB RAM (workload generator running WebSphere Version 6.0)
- Two IBM X335 Intel XEON 2-way CPU 2.40 GHz (workload generators)

Client software

The client software consists of these operating systems and applications. Table 4 describes the client software.

Table 4. Client software

Product	Version/Level
Red Hat Linux AS	Release 4 Update 6
Internal workload driver	N/A
WebSphere Application Server	Version 6.1.0.0

Chapter 3. System setup

A detailed system setup for the 64-bit and 31-bit WebSphere study with J2EE workloads is described.

Environment

A customer-like environment was used for the 64-bit and 31-bit WebSphere study with J2EE workloads.

To emulate a customer environment, these components are used:

- A workload driver that emulates a retail and manufacturing workload
- A single WebSphere V 6.1 Application Server to host the application
- A DB2 UDB V9.5 database

Figure 1 on page 8 shows the configuration used for the testing.

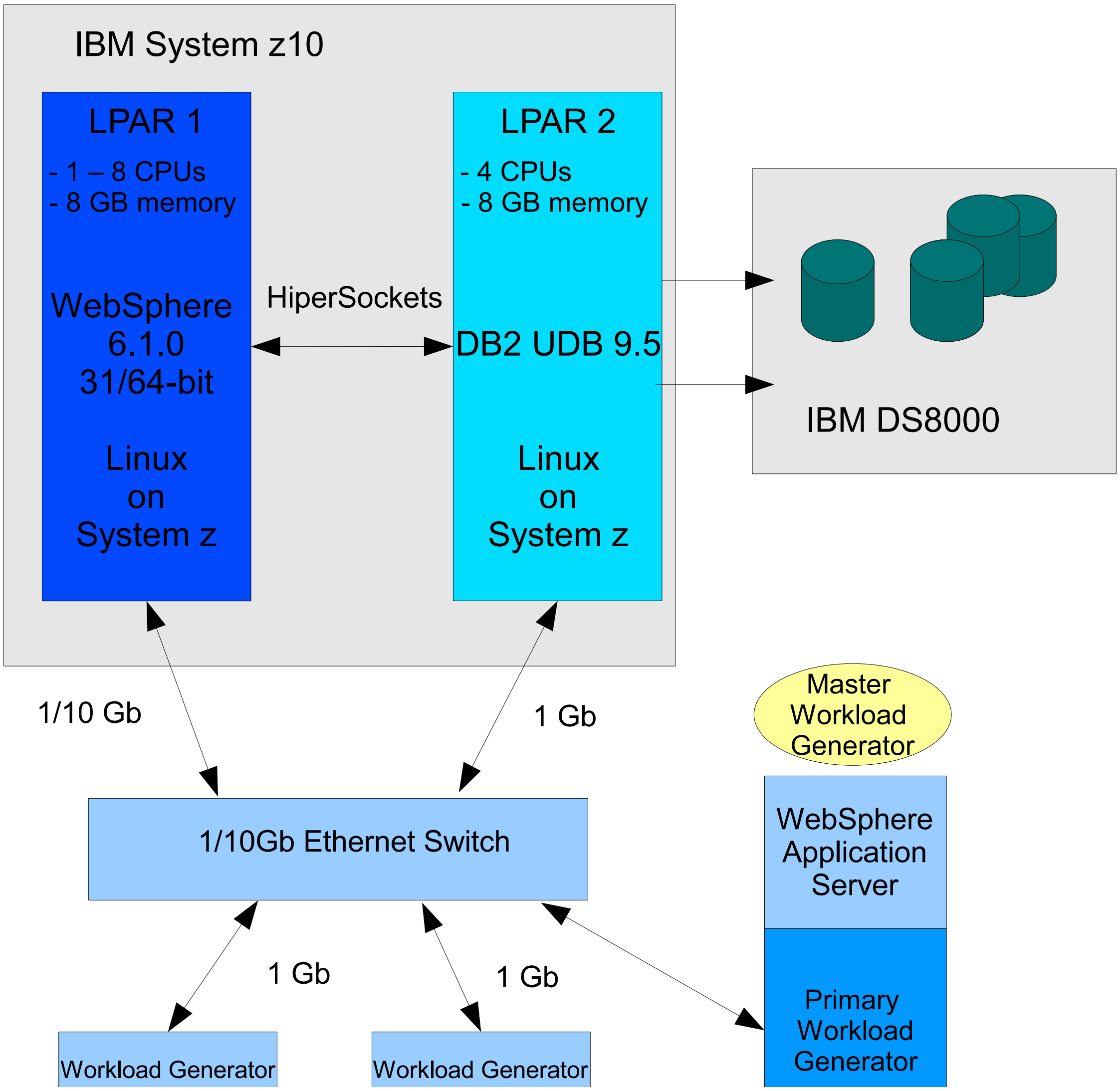


Figure 1. System configuration for the customer workload

Network setup

The network setup for the 64-bit and 31-bit WebSphere study with J2EE workloads uses a three-tiered workload.

The network setup for the three-tiered workload is as follows:

- The IBM System x Linux-based guests each had one 1 Gb network interface to the Ethernet for communication with the WebSphere system on the server.
- The IBM System z was connected with either a 1 Gb or 10 Gb Ethernet card to the Ethernet network.
- The two Logical Partitions (LPARs) were connected with HiperSockets with a 16 KB frame size for a fast isolated network connection.

The network setup is shown in Figure 1.

Workload generator systems

Up to three IBM System x systems are used to generate the workload for the 64-bit and 31-bit WebSphere study with J2EE workloads.

A WebSphere Application Server is present on the master workload generator, but not on the helper machines. These helpers only need a working Java Runtime Environment (JRE).

A workload generator consisting of multithreaded independent Java processes was also used. These processes run on all the clients, including the master client.

At workload submission rates greater than 500, two additional client systems are needed to create the workload.

Chapter 4. Linux kernel settings

Different Linux kernel settings were used to optimize performance on the WebSphere Application Server, workload generator, and the DB2 Universal Database systems.

WebSphere Application Server Linux kernel settings

To specify the Linux kernel settings for the WebSphere Application Server, enable the use of a 1 GB JVM, and set the swappiness parameter to zero.

To configure the WebSphere Application Server kernel settings, perform these tasks:

- “Enabling 31-bit WebSphere Application Server on IBM System z to use 1 GB JVM”
- “Setting swappiness parameter to zero”

Enabling 31-bit WebSphere Application Server on IBM System z to use 1 GB JVM

A 31-bit WebSphere Application Server on IBM System z must be changed to use a 1 GB Java Virtual Machine (JVM), in order to improve performance.

Normally, you cannot define more than 768 MB of JVM heap on a 31-bit distribution of WebSphere Application Server. However, with Linux, you can use the mapped_base support to enable your system to have up to 1 GB of heap. Unpredictable results occur if you exceed the 1 GB value. To enable this capability, place this line into the `startServer.sh` startup script:

```
echo 268435456 >/proc/self/mapped_base
```

The `startServer.sh` script now looks like this (**bold** is added for emphasis only):

```
nwas3:/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin # cat startServer.sh
#!/bin/sh
echo 268435456 >/proc/self/mapped_base
WAS_USER_SCRIPT=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin/setupCmdLine.sh
export WAS_USER_SCRIPT
/opt/IBM/WebSphere/AppServer/bin/startServer.sh "$@"
```

Setting swappiness parameter to zero

Setting the swappiness parameter to zero ensures that application pages will not be moved to swap space.

The swappiness parameter influences the kernel preference to move memory pages from applications to swap page, versus reclaiming memory from the cache. After system restart, set the swappiness parameter to zero. This ensures that if memory is constrained, the page cache is reduced in an attempt to recover memory before application pages are moved to swap space:

```
echo 0 >/proc/sys/vm/swappiness
```

This setting might improve or degrade the performance of an application. Because there is adequate memory already dedicated to this workload, large amounts of memory would not need to be swapped to disk. Because precautionary (early) swapping is now avoided, the study results are free of the effects of this kind of swapping.

Linux kernel settings for the workload generator systems

Some Linux kernel settings must be permanently increased to run the driver workload.

To change the Linux kernel settings, add a shell script named `performance.sh` to directory `/etc/profile.d`. The processing of `/etc/profile` calls any shell scripts in the `/etc/profile.d` directory.

Shell script `performance.sh` should have these lines:

```
        echo "20000" > /proc/sys/net/core/netdev_max_backlog
        echo "20000" > /proc/sys/net/core/somaxconn
        echo "30"    > /proc/sys/net/ipv4/tcp_fin_timeout
        echo "20"    > /proc/sys/net/ipv4/tcp_syn_retries
        echo "20"    > /proc/sys/net/ipv4/tcp_synack_retries
        echo "3"     >/proc/sys/net/ipv4/conf/all/arp_ignore
        echo "2"     >/proc/sys/net/ipv4/conf/all/arp_announce
    ulimit -n 50000
```

The `ulimit -n 50000` command sets the maximum number of open file descriptors (and therefore the maximum number of open files) to 50000. The default value is 30000.

The echo commands establish the network settings that produce the largest throughput without excessive time outs or retries. These are the echo command arguments:

netdev_max_backlog

Specifies the maximum number of incoming packets that can be enqueued for upper-layer processing. This is a global variable.

somaxconn

Specifies the maximum number of pending connection requests queued for any listening socket. These are set to high values that reflect the expectation of a high initial and high sustained number of both connections and incoming packets.

tcp_fin_timeout

Specifies how long to keep sockets in the state FIN-WAIT-2 if the socket is being closed. A longer timeout means that socket structures are held in memory longer, while the current timeout value of 30 seconds assumes reasonable completion times and conserves memory.

tcp_syn_retries

Specifies how many times to try to retransmit the initial SYN packet for an active TCP connection attempt. The current setting is 20, which means that there are 20 retransmission attempts before the connection times out. This can take several minutes, depending on the length of the retransmission attempt.

tcp_synack_retries

Specifies how many times to try to establish a passive TCP connection that was started by another host. This parameter is also set to a fairly small value for the same reasons that the **tcp_syn_retries** parameter was set to a fairly small value.

arp_ignore

Defines different modes for sending replies in response to received ARP requests that resolve local target IP addresses. An **arp_ignore** value of 3 means to not reply for local addresses configured within the host, but to reply only for global and link addresses.

arp_announce

Defines different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface. An **arp_announce** value of 2 means to always use the best local address for this target, even if it means ignoring the source IP address and choosing the preferred IP address for the destination host.

Linux kernel settings for the DB2 Universal Database system

These are the DB2 Universal Database (UDB) Linux kernel settings.

The kernel semaphore settings recommended for DB2 are used. To use these settings, the kernel.sem setting is placed in the file `/etc/sysctl.conf`.

```
kernel.sem="250 32000 32 4096"
```

The values in this line are:

- The first value is the number of semaphores per array: 250.
- The second value is the maximum number of semaphores system wide: 32000.
- The third value is the maximum number of operations per semop call: 32.
- The fourth value is the maximum number of semaphore arrays: 4096.

The use of these values also implies a maximum number of system-wide semaphores, which is calculated by multiplying the maximum number of semaphore arrays by the number of semaphores per array (4096 multiplied by 250). In this case, the value for system wide semaphores was capped at 32 000.

This setting is activated automatically at system start time when the `boot.sysctl` service is enabled, or at run time by issuing this command:

```
sysctl -p /etc/sysctl.conf
```

Also, the kernel swappiness parameter is set to zero, as described in “Setting swappiness parameter to zero” on page 11.

Chapter 5. Setting up WebSphere and DB2 Universal Database

To optimize the performance of WebSphere and DB2, tuning scripts are defined and used.

The scripts included in “Configuration, tuning, and performance scripts,” on page 47 were used to tune WebSphere and DB2 Universal Database (UDB) for the workload. The WebSphere scripts are jacl-based and run only one time. The DB2 tuning scripts are run each time that the database is created. Adapting the buffer pools and configuration settings improves DB2 performance. Modifications of the WebSphere settings are needed in addition to the jacl scripts, as well as changes to settings that are part of the tests.

WebSphere V6.1.0 configuration

Settings are chosen to improve the performance of WebSphere when running this workload. While many of these settings are applicable to other workloads, the settings and their chosen values might not be appropriate for production environments.

Java DataBase Connectivity connection pools

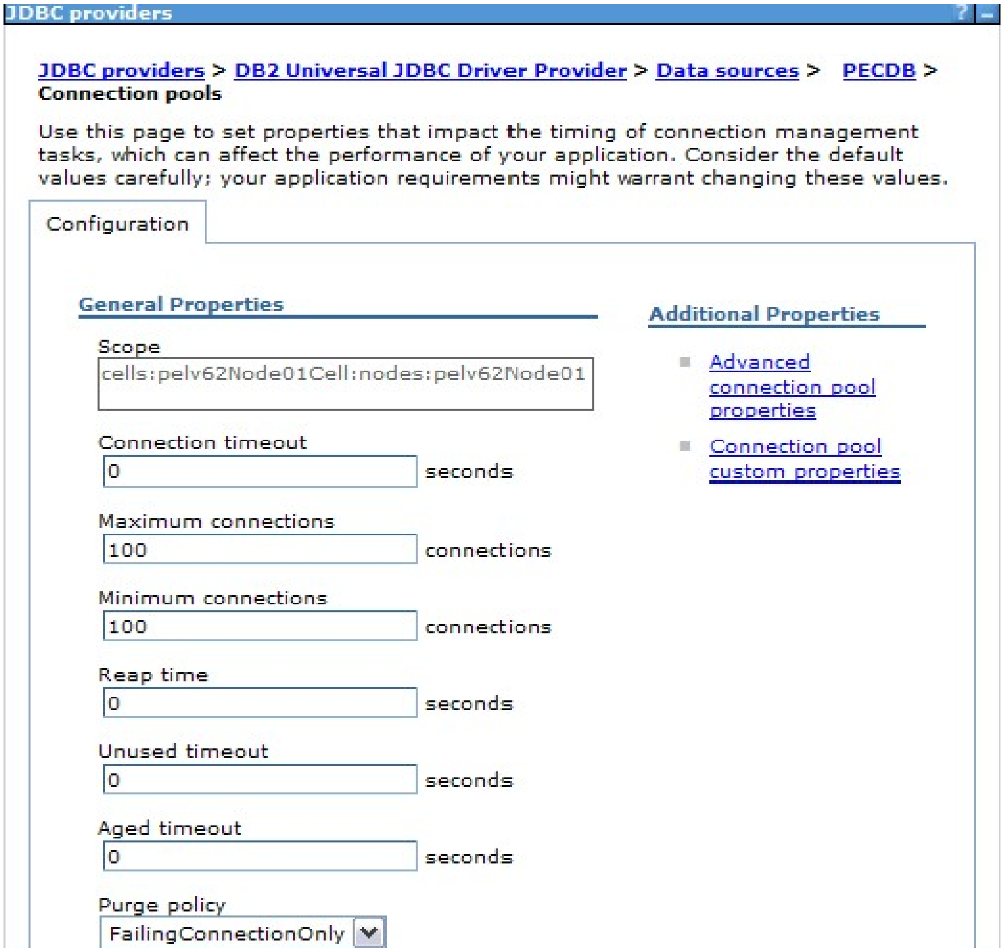
JDBC connection pools are established to allow multithreaded applications to request resources from a backend database without having to incur the connect and disconnect overhead on every database request.

Using larger Java DataBase Connectivity connection pools has the advantage of holding previously-used connections active in the pool, so that applications and application threads can share (or pool) their backend database connections. Subsequent requests by different threads can use an existing connection and avoid the overhead of new connection creation and teardown, as well as the overhead of being queued.

Increasing the size of the connection pool increases the memory requirements on WebSphere and DB2. Therefore, the connection pool requirements must be balanced against available server memory. Users from Web-based applications often issue short-lived transactions, where reducing connect and disconnect times can provide greatly improved response times.

Figure 2 on page 16 displays the values for the Java DataBase Connectivity connection pools. These are the selected settings:

- A minimum and maximum connection pool value of 100.
- A Connection timeout value of zero, which means that when there are no available connections (all 100 are in use), the connection request waits for an indefinite amount of time.
- A Reap time value of zero, which disables the pool maintenance thread so that the other time-outs related to pool maintenance become irrelevant.
- All other values set to zero.



JDBC providers

JDBC providers > DB2 Universal JDBC Driver Provider > Data sources > PECDB > **Connection pools**

Use this page to set properties that impact the timing of connection management tasks, which can affect the performance of your application. Consider the default values carefully; your application requirements might warrant changing these values.

Configuration

General Properties

Scope: cells:pelv62Node01Cell:nodes:pelv62Node01

Connection timeout: 0 seconds

Maximum connections: 100 connections

Minimum connections: 100 connections

Reap time: 0 seconds

Unused timeout: 0 seconds

Aged timeout: 0 seconds

Purge policy: FailingConnectionOnly

Additional Properties

- Advanced connection pool properties
- Connection pool custom properties

Figure 2. Setting JDBC connection pools

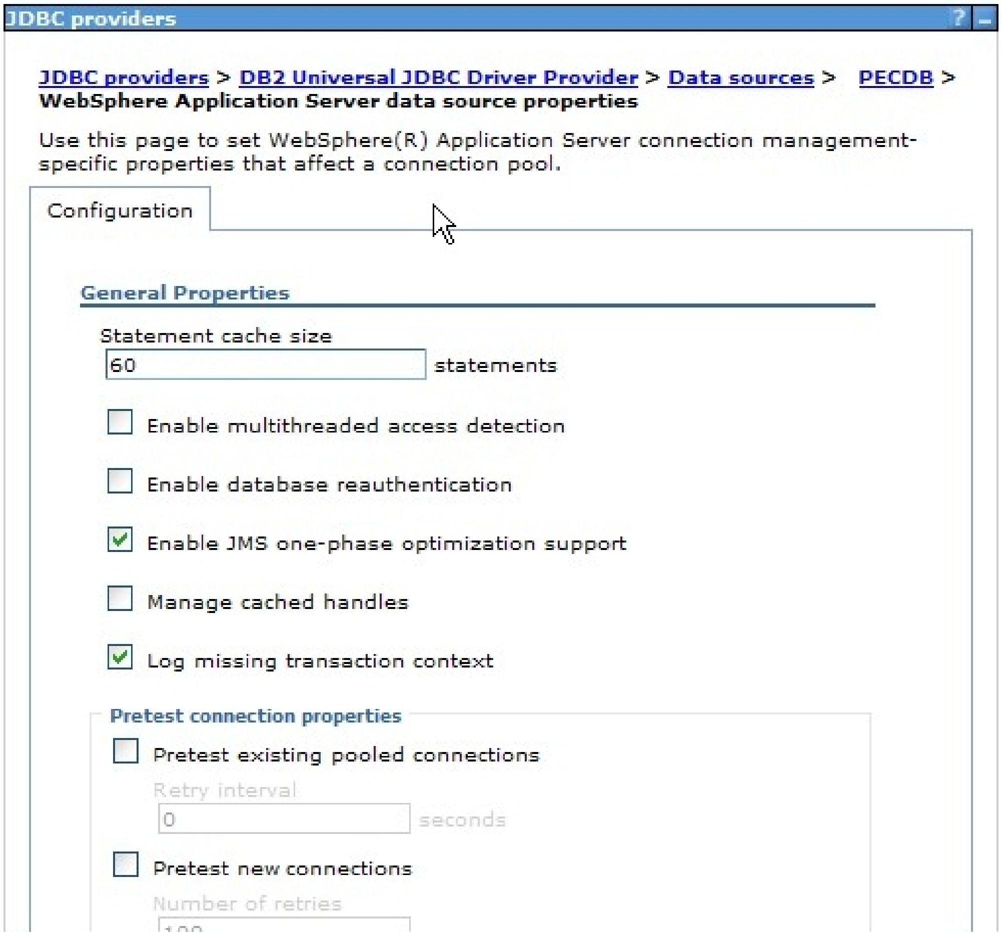
Java DataBase Connectivity data source properties

The settings used to configure the Java DataBase Connectivity (JDBC) data source properties were chosen to maximize performance.

Figure 3 on page 17 displays the settings for the JDBC data source properties. These are the selected settings:

- A statement cache size of 60 statements was selected because the workload produces approximately 60 unique SQL statements that are then stored as precompiled SQL objects.

  Caching these statements improves performance because each cached statement can then be run multiple times without the cost of additional statement preparation. Increasing the cache size to be greater than the number of needed statements is related to higher memory use, which is probably not needed. Too small a cache size causes cached statements to be discarded, and these statements might have to be recreated in the future.
- The setting to enables JMS one-phase optimization support allows JMS to obtain optimized connections from the data source. The workload uses message queues for access to the database; therefore this setting was enabled.



JDBC providers > DB2 Universal JDBC Driver Provider > Data sources > PECDB > **WebSphere Application Server data source properties**

Use this page to set WebSphere(R) Application Server connection management-specific properties that affect a connection pool.

General Properties

Statement cache size
60 statements

☐ Enable multithreaded access detection

☐ Enable database reauthentication

☑ Enable JMS one-phase optimization support

☐ Manage cached handles

☑ Log missing transaction context

Pretest connection properties

☐ Pretest existing pooled connections

Retry interval
0 seconds

☐ Pretest new connections

Number of retries

Figure 3. JDBC data source configuration

Object Request Broker thread pool

The Object Request Broker (ORB) is a service that handles requests from clients over the Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP). One part of the workload driver connects to the workload application using its Enterprise Java Beans (EJBs) with Remote Method Invocation of Java methods (RMI) using RMI/IIOP.

The ORB thread pool values are adapted to the expected amount of RMI/IIOP traffic. Figure 4 on page 18 displays the setting for the ORB thread pool. These are the selected settings:

- The size of the ORB pool is set to 15 for the minimum and maximum number of threads.
- The thread inactivity timeout is set to 3500 milliseconds.

WebContainer thread pool

The WebContainer thread pools are used for HTTP requests that come from the client.

Figure 4 displays the WebContainer thread pools settings. These settings were chosen due to the high HTTP traffic generated by the workload, to prevent requests from being queued in the transport chain when all threads are busy. These are the selected settings:

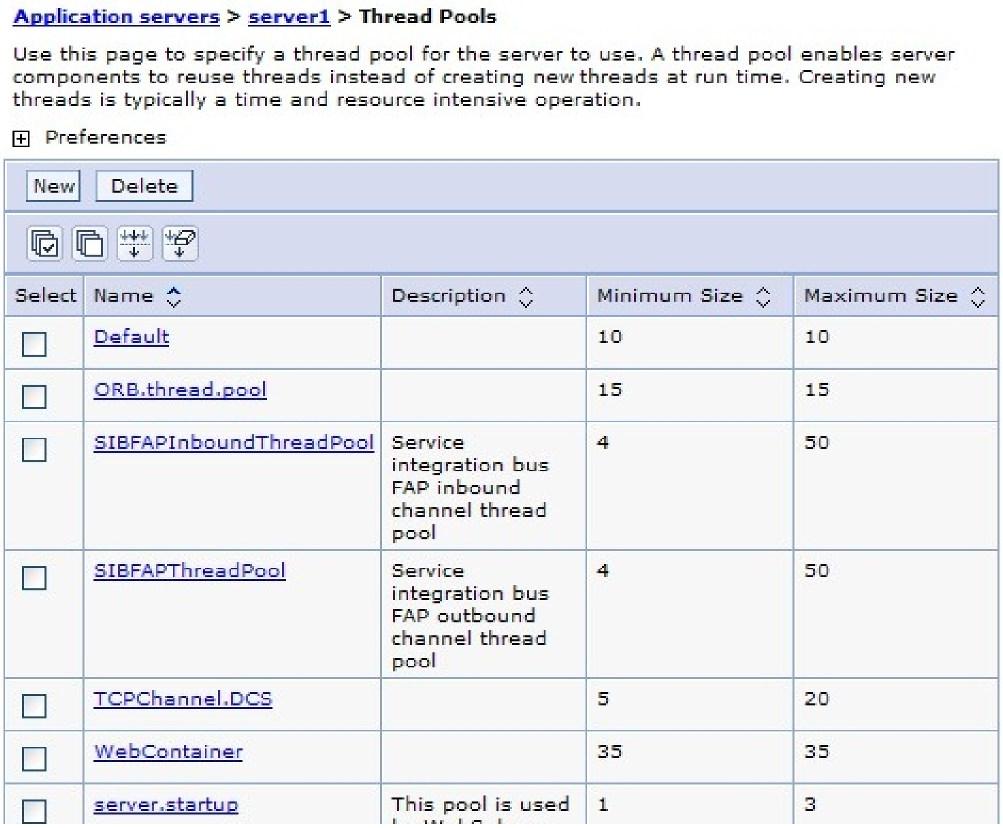
- The size for the WebContainer pool is set to 35 minimum and 35 maximum threads.
- A timeout value is set to 3500 milliseconds.

Default thread pool

The parameters associated with the default thread pool are modified to reuse threads instead of creating new ones.

Figure 4 displays the default thread pools settings. These are the selected settings:

- The size for the default thread pool is set to 10 minimum and 10 maximum threads.
- A timeout value is set to 3500 milliseconds.



Application servers > server1 > Thread Pools

Use this page to specify a thread pool for the server to use. A thread pool enables server components to reuse threads instead of creating new threads at run time. Creating new threads is typically a time and resource intensive operation.

Select	Name	Description	Minimum Size	Maximum Size
□	Default		10	10
□	ORB.thread.pool		15	15
□	SIBFAPInboundThreadPool	Service integration bus FAP inbound channel thread pool	4	50
□	SIBFAPThreadPool	Service integration bus FAP outbound channel thread pool	4	50
□	TCPChannel.DCS		5	20
□	WebContainer		35	35
□	server.startup	This pool is used	1	3

Figure 4. Thread pool

HTTP transport settings

The default values for the HTTP transport channels associated with a transport chain are modified to improve performance.

For the workload, the parameters associated with the chain using port number 9080 are relevant. In the WCInboundDefault window, select the HTTP Inbound Channel (HTTP_2) link.

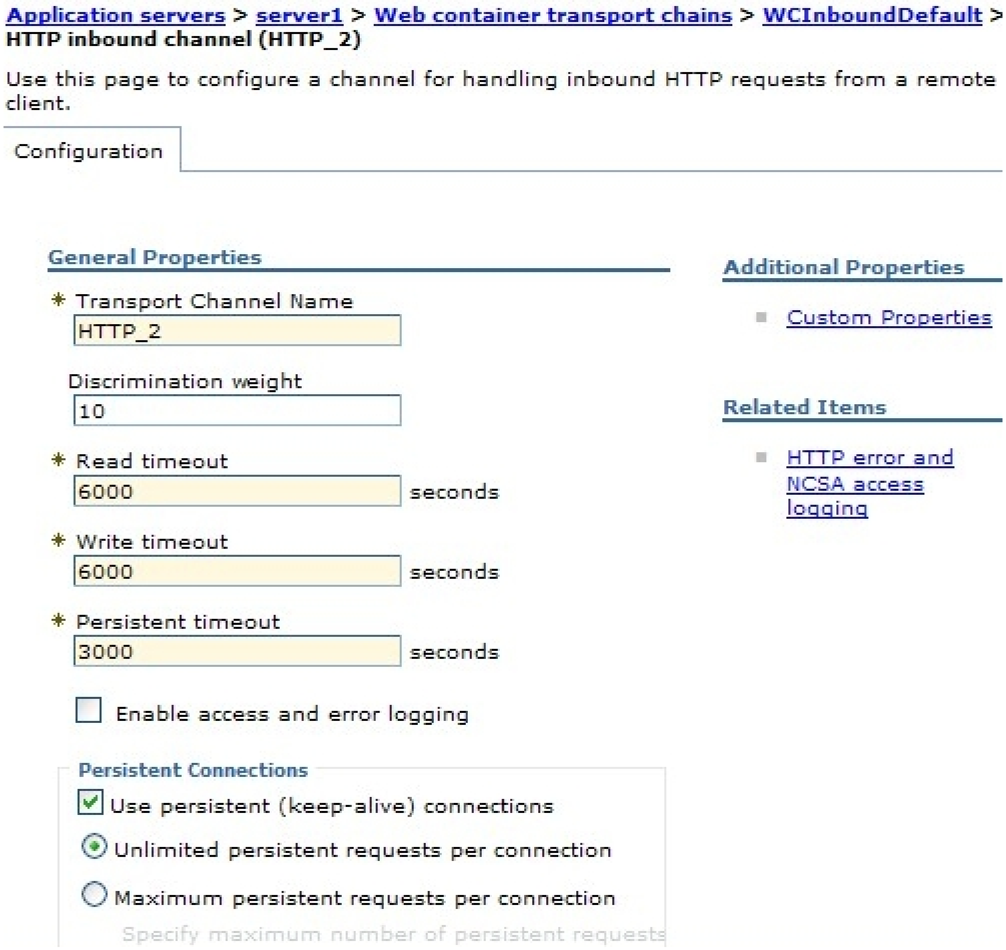
Figure 5 displays the HTTP transport settings. These are the selected settings:

- The read timeout is set to 6000 seconds.

  This ensures that clients sending data have enough time to send their data.
- The write timeout is set to 6000 seconds.

  This ensures that a client can receive all the data being sent (written) to them.
- The HTTP keep-alive setting is modified by increasing the value of the persistent timeout selection from 30 to 3000 seconds.

  This ensures that even when there is no activity, a connection is kept open for 3000 seconds for clients needing a connection to make a request.
- The Maximum persistent requests per connection selection, which would prevent more than a certain number of requests, is disabled. This sets the radio button for Unlimited persistent requests per connection.



Application servers > server1 > Web container transport chains > WCInboundDefault > HTTP inbound channel (HTTP_2)

Use this page to configure a channel for handling inbound HTTP requests from a remote client.

General Properties

* Transport Channel Name: HTTP_2

Discrimination weight: 10

* Read timeout: 6000 seconds

* Write timeout: 6000 seconds

* Persistent timeout: 3000 seconds

□ Enable access and error logging

Persistent Connections

☑ Use persistent (keep-alive) connections

● Unlimited persistent requests per connection

○ Maximum persistent requests per connection

Specify maximum number of persistent requests

Figure 5. HTTP transport settings

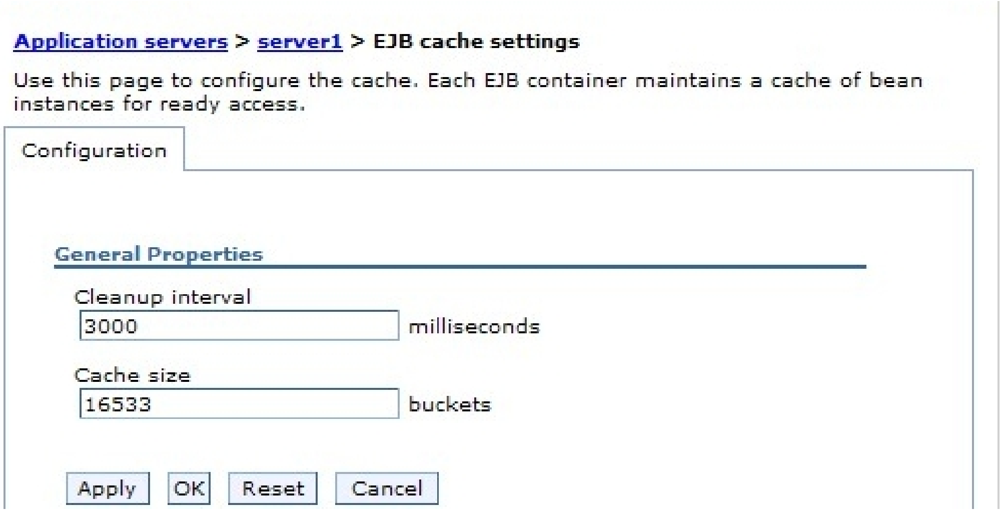
Enterprise Java Beans cache settings

This value specifies the number of buckets in the active instance list available for Enterprise Java Beans (EJBs).

A bucket can contain multiple enterprise bean instances. This ensures that more entity beans can be kept active in the EJB cache, improving EJB performance. The expected number of active enterprise beans determines the best value for the EJB cache setting, and a larger EJB cache setting means more buckets are allocated.

When the number of instances exceeds the available space in the buckets, some beans are made passive to allow new beans to become active instances. Because EJB buckets are allocated from memory, increasing EJB cache settings comes at the expense of additional memory.

Figure 6 displays the EJB cache settings. For the workload, the EJB cache setting is changed from the default value to 16533.



Application servers > server1 > EJB cache settings

Use this page to configure the cache. Each EJB container maintains a cache of bean instances for ready access.

Configuration

General Properties

Cleanup interval: 3000 milliseconds

Cache size: 16533 buckets

Figure 6. EJB cache settings

Tune the Java Virtual Machine properties

The Java Virtual Machine (JVM) heap is the amount of available memory allocated to the Java virtual machine. The JVM minimum and maximum values are set based on heap size studies, with different values for 64-bit WebSphere running a 64-bit JVM, and 31-bit WebSphere running a 31-bit JVM.

Figure 7 on page 22 displays the JVM properties. The selected settings are:

- EJB cache setting is changed from the default value to 16533.
- Both the heap minimum and maximum values are set to the optimum values for the workload, 6144.
- Verbose garbage collection is selected in order to record that information.
- Under the Generic JVM arguments, these settings are added:

```
-Dcom.ibm.ws.pm.batch=true
-Dcom.ibm.ws.pm.deferredcreate=true
-Dcom.ibm.CORBA.FragmentSize=3000
```

The Generic JVM arguments:

`com.ibm.ws.pm.batch=true`

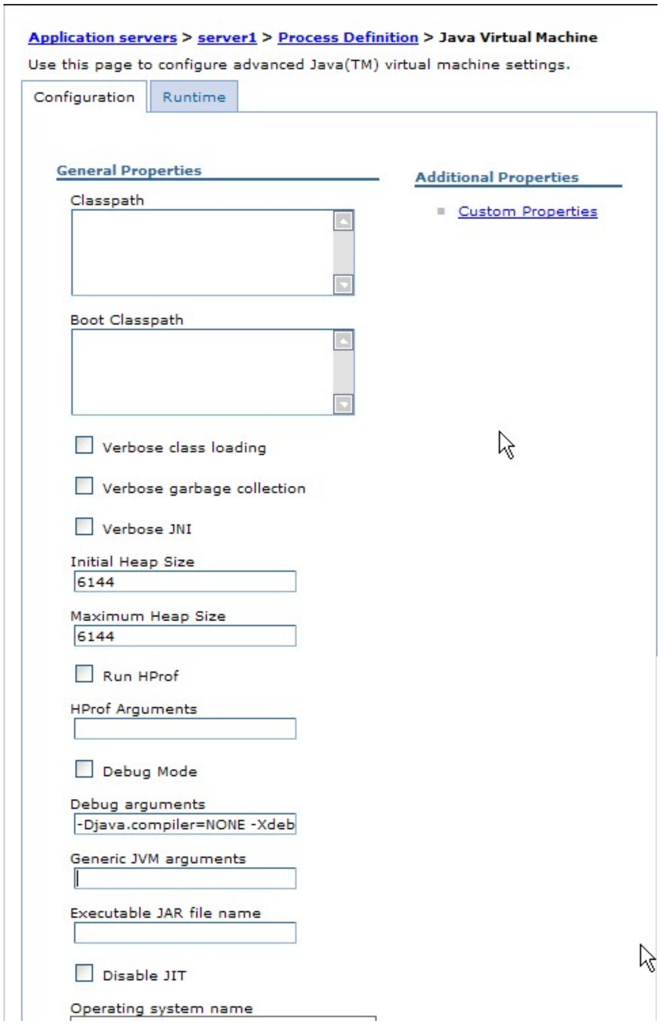
The flag is used when an application updates multiple container-managed persistence (CMP) beans inside of a single transaction. This flag can be used to allow batching of the update so that one transaction containing both update transactions is presented to the database instead of two or more transactions. This flag is used only when the database supports batching of update operations, and the application regularly accesses multiple CMP beans for update. This saves round trips to the database, improving performance on these operations.

`com.ibm.ws.pm.deferredcreate=true`

The flag also provides a performance benefit on an ejbCreate() method call. The default behavior is to immediately insert an empty row into the database with only the primary key. Most transactions then modify fields within the bean, so the insertion of the row into the database can be deferred until the data in the rows is actually present, saving a database call.

`com.ibm.CORBA.FragmentSize=3000`

The setting makes the size of the Object Request Broker (ORB) fragment 3000 bytes. The ORB separates messages into fragments to send over the ORB connection. The 3000 byte size was calculated to be a good size for this workload.



Application servers > server1 > Process Definition > Java Virtual Machine

Use this page to configure advanced Java(TM) virtual machine settings.

General Properties

Classpath

Boot Classpath

□ Verbose class loading

□ Verbose garbage collection

□ Verbose JNI

Initial Heap Size: 6144

Maximum Heap Size: 6144

□ Run HProf

HProf Arguments

□ Debug Mode

Debug arguments: -Djava.compiler=NONE -Xdeb

Generic JVM arguments

Executable JAR file name

□ Disable JIT

Operating system name

Additional Properties

- Custom Properties

Figure 7. Java Virtual Machine settings

Other Java Virtual Machine arguments

These Java Virtual Machine (JVM) arguments were identified as beneficial to the workload used here, and other benchmarks with similar characteristics to the workload. The values are specific to the workload and should not be expected to necessarily benefit other workloads or production environments.

```
-noclassgc -Xss128k  -Xgcpolicy:gencon -Xmo768m -Xcodecache16m -Xgcthreads4
-Djava.net.preferIPv4Stack=true -Dsun.net.inetaddr.ttl=0
```

-Xgcpolicy:gencon
Directs the JVM to manage its heap using the generational concurrent or split heap. This benefits applications with many short-lived objects, and is discussed in Chapter 3, “System setup,” on page 7.

-Xmo768m
Specifies that 768 MB of memory is to be dedicated to the tenured part of the split heap, with the remaining part of the JVM heap dedicated to buffers and new object or nursery areas. This value is changed for some 64-bit measurements.

-Xnoclassgc
Directs the JVM to not perform garbage collection on the classes that reside in the permanent or tenured space in the split heap.

Unloading of these classes means that future references require a costly load from the file system, instead of an efficient memory load. This is a commonly used setting for workloads with repetitive use of the same classes.

-Xgcthreads4
Instructs the JVM to start four threads for garbage collection. The default value is to have as many garbage collection threads as there are available processors.

-Xss128k
Allocates 128 KB of memory for the native stack area of each thread. The native stack area is used for native library loads by Java from the C/C++ environment, which is the native layer.

-Xcodecache16m
Sets the size of each block of memory that is allocated to store native code of compiled Java methods to 16 MB. By default, this size is selected internally according to the CPU architecture and the capacity of your system.

Transaction service properties

The values of the total transaction lifetime timeout and the client inactivity timeout are set to zero.

Total transaction lifetime timeout
Sets the number of seconds that a transaction can remain inactive before it is ended by the transaction service. A value of 0 indicates that there is no timeout limit.

Client inactivity timeout
Sets the number of seconds for which a transaction started by, or propagated into, this application server can run before it is ended by the transaction service. A value of 0 indicates that there is no timeout limit.

To set these parameters on the WebSphere Administration console, where server1 stands for the name of the application server:

1. Click **Application Servers** → **server1** → **Container Settings** → **Container Services**.
2. Click **Transaction Service link**.
3. Set **Total transaction lifetime timeout** to 0.
4. Set **Client inactivity timeout** to 0.

Disable Java 2 security

Java 2 security is disabled because security policy files are not available for the workload.

Note: This is *not* recommended for production environments.

To reset these parameters on the WebSphere Administration Console:

1. Click **Security** → **Global security**. The **Global security** panel is displayed
2. Clear the **Enforce Java 2** security option.

DB2 V9.5 configuration

Settings for DB2 Version 9.5 used in the 64-bit and 31-bit WebSphere study with J2EE workloads include settings for the kernel, log files, and buffer pools.

These settings are illustrated in the script contained in "Initial database setup."

Kernel settings

The recommended kernel semaphore settings are used, as described in "Linux kernel settings for the DB2 Universal Database system" on page 13.

Log files

Before creating a new database, scripts are run to define the DB2 log files and an appropriate buffer pool. The first three commands in "Initial database setup" specify these options:

- DB2 logs will each have 65 MB of space
- A value of 40 logs, for a total of 2.6 GB of log file space
- The `/db2log` directory contains these 40 log files

Restarting DB2 causes the logs to be put to use, so the available space in the `newlogpath` directory then decreases, which reflects the allocation and use of the DB2 logs.

Buffer pools

The size of the default buffer pool is dependent on available memory, and is later readjusted by the DB2 autoconfigure commands. The size of the default buffer pool is set initially to 975 MB. This size then changes dynamically during execution, increasing or decreasing depending on the workload.

Initial database setup

The initial database setup specifies details about log files and buffer pools.

Use this script for the initial setup of the database used in the 64-bit and 31-bit WebSphere study with J2EE workloads.

```
db2 update db cfg for <dbname> using logfilsiz 65535
db2 update db cfg for <dbname> using logprimary 40
db2 update db cfg for <dbname> using newlogpath /db2log
db2 connect to <dbname>
db2 -v alter bufferpool ibmdefaultbp size 975000 automatic
db2 connect reset
db2stop;db2start
db2 connect to <dbname>
db2 connect reset
```

Tuning the populated database

After the initial setup, the database is populated using a DB2 restore operation.

Before the database is used, it is tuned with these commands:

```
db2 connect to <dbname>
db2 autoconfigure using mem_percent 80 workload_type simple num_stmts 60 tpm 2000
is_populated yes num_local_apps 0 num_remote_apps 100 isolation rs bp_resizeable yes
apply db and dbm
```

```
db2 terminate
db2 connect to <dbname>
db2 reorgchk update statistics on table all
db2 terminate
```

See “DB2 autoconfigure command” for a detailed explanation of each argument.

DB2 autoconfigure command

These are the arguments for the DB2 autoconfigure command used to optimize database performance.

These arguments are specified on the DB2 autoconfigure command displayed in “Tuning the populated database” on page 24.

mem_percent 80
Allocates 80% of the memory as a buffer pool.

num_stmts 60
Indicates that there are approximately 60 unique SQL statements with this workload.

workload_type simple
Indicates that the workload uses simple SQL statements, those most commonly used by WebSphere.

tpm 2000
Indicates the maximum expected number of transactions per minute, which is estimated at 2000.

num_remote_apps 100
Specifies the number of connections to the database from WebSphere, which is set in this WebSphere configuration.

rs
Indicates an isolation level of read stability, which is required by most of the Java beans. Isolation level rs means that database rows that are read by one activation bean are only read when they are not being updated by another bean. However, subsequent reads can reflect intervening updates by different beans.

bp_resizeable yes
Tells DB2 to use its autoconfiguration capabilities to adjust the size of the buffer pool from its original size during future database operations, in a dynamic manner.

The `reorgchk update statistics on table all` command updates the statistics information from the catalog table to find the best plan to access table data. It is important that this information is current for the optimizer to be able to make the right decisions here. The reorgchk command runs a RUNSTATs operation on every table, which improves the query performance for each DB2 table.

Chapter 6. Workload description

A benchmark emulating a customer-like workload was used for the 64-bit and 31-bit WebSphere study with J2EE workloads.

The workload was chosen to stress the middleware and J2EE components using an end-to-end Web application. The applications are a collection of

- Java classes
- Java Servlets
- Java Server Pages
- Web Services
- Enterprise Java Beans (EJBs) built to open J2EE APIs

All major components of J2EE technologies are exercised, including:

- The Web container (servlets and JSPs)
- The EJB container
- EJB 2.0 Container Managed Persistence
- Java Messaging Service (JMS)
- Message Driven Beans (MDBs)
- Transaction management
- Database connectivity

The workload exercises all parts of the infrastructure, such as hardware, JVM software, database software, Java DataBase Connectivity (JDBC) drivers and the system network. The workload implements a Web layer and makes extensive use of JMS and MDB technology.

The workload is a retail and manufacturing application implementing a set of user services such as login and logout, stores, buying, selling, account details, and so on, using standards-based HTTP and Web services protocols. In addition to the retail domain, a manufacturing work order domain is also simulated to drive other high-volume transactions. The retail domain uses Web layer connections to access the applications, with the manufacturing domain connecting to the application with the EJBs with the RMI protocol using RMI/IIOP.

These server connection modes are used:

EJB Database access uses EJB 2.1 methods to drive retail and manufacturing operations.

Direct Database and messaging access using direct JDBC and JMS code.

Type 4 JDBC connectors are used with EJB containers.

The workload is driven by the workload generator machines at a certain submission rate. Data must be present in the database before running the customer's workload simulation. The number of retailers, number of customers, and manufacturing data varies with the intended submission rate.

Scripts to load the database use an argument representing the amount of data to load. For submission rates greater than 100, they are a multiple of 100. For example, at submission rate 110 the database scripts are given an argument of 200. The database size is therefore roughly correlated with the submission rate.

Chapter 7. Results

The results of the 64-bit and 31-bit WebSphere study with J2EE workloads are presented, along with observations and conclusions.

Results are observed and analyzed according to these categories:
- “Heapsize for the 64-bit Java Virtual Machine”
- “Comparing 64-bit WebSphere versus 31-bit WebSphere” on page 31
- “CPU scaling study” on page 35
- “Database LPAR analysis” on page 39
- “Network study – 1 Gb Ethernet versus 10 Gb Ethernet” on page 42

Heapsize for the 64-bit Java Virtual Machine

To compare the performance of 64-bit WebSphere to 31-bit WebSphere, an attempt was made to identify the optimal percentage of available memory to dedicate to the JVM heap for the workload.

A starting value of 50% of available memory was used, and the value was increased to 80%. Throughput, workload operations per second, and CPU utilization are recorded. Response times are also recorded as another possible measure of optimization.

The data is collected at two different workload submission rates, 300 and 500, with either four or eight CPUs, and with either 4 GB or 8 GB of memory configured on the WebSphere LPAR. Within each submission rate, WebSphere’s JVM minimum and maximum heap settings are set to identical values corresponding to different percentages of the LPAR’s configured memory.

To analyze the results, two comparisons were done:
- “Throughput and CPU utilization”
- “CPU utilization and response times” on page 31

Throughput and CPU utilization

The results for throughput and CPU utilization are summarized in Table 5, and displayed graphically in Figure 8 on page 30.

Table 5. Heapsize studies of 64-bit WebSphere: throughput, CPU load, and response time using workload submission rate 300 and 500, and different JVM heap percentages of total available memory

Workload submission rate	Number of CPUs	Memory in GB	Heap percentage of memory	Normalized workload throughput	CPU utilization	Response time (ms)
300	4	4	33%	100%	346%	349
300	4	4	60%	100%	322%	334
300	4	4	69%	100%	315%	343
300	4	4	75%	100%	333%	359
500	8	8	59%	165%	633%	740
500	8	8	70%	166%	542%	751

Table 5. Heapsize studies of 64-bit WebSphere: throughput, CPU load, and response time using workload submission rate 300 and 500, and different JVM heap percentages of total available memory (continued)

Workload submission rate	Number of CPUs	Memory in GB	Heap percentage of memory	Normalized workload throughput	CPU utilization	Response time (ms)
500	8	8	75%	165%	554%	706
500	8	8	80%	165%	573%	706

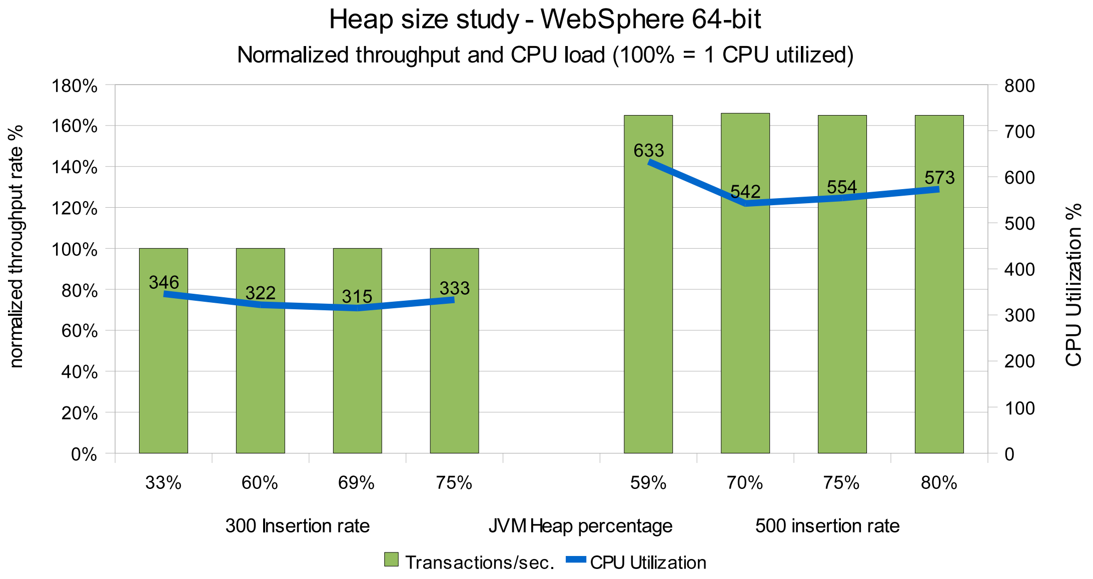


Figure 8. Heap size studies of 64-bit WebSphere: throughput and CPU load at workload submission rate 300 and 500, and different JVM heap percentages of total available memory

Observation

The measures of throughput are generally fairly constant with this workload. The CPU utilization generally is lowest at values between 68% and 75% of the available memory of the processor. When memory of 8 GB is dedicated to the Linux system, the 75% of available memory value provides the lowest CPU utilization. When memory of 4 GB is dedicated to the Linux System, the optimal heap size tends to be closer to the 70% of available memory value.

Conclusion

The throughput seems to be dependent only on the submission rate. When sufficient resources are available to support a certain submission rate, more resources do not increase the throughput. The CPU utilization seems to be a better parameter to determine the best JVM heap size. For this workload, a value of between 70% and 75% from the main memory is the best, where smaller heap sizes are better for smaller memory sizes.

CPU utilization and response times

Another parameter to observe is the response time between the simulated Web operation, such as a purchase or a browse operation, and the turnaround time of the request from the WebSphere system. The results for CPU utilization and response time are summarized in Table 5 on page 29 and displayed graphically in Figure 9. Response times are shown as a bar chart, and unlike throughput measures, smaller response time values are considered optimal.

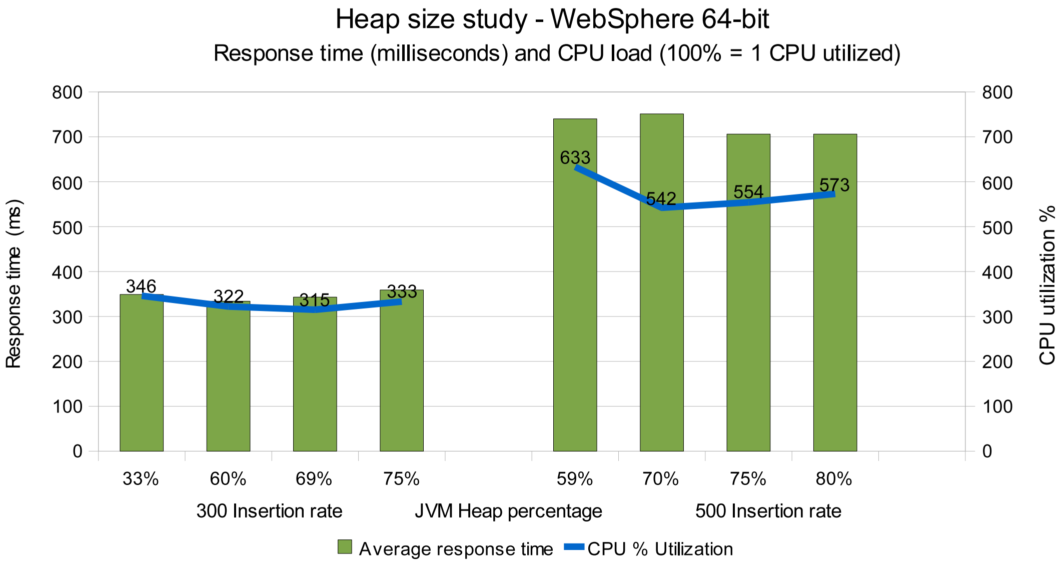


Figure 9. CPU utilization and corresponding average response times at different JVM heap percentages

Observation

In this workload, the CPU utilization is lowest, and the response times shortest, at between 68% and 75% of the available memory of the processor. Mainly it seems that the response time follows the CPU utilization.

Conclusion

The correlation of response time with CPU utilization when scaling the Java heap size indicates that both have a common dependency. A larger heap can avoid some garbage collection, which leads to a decreasing CPU utilization. And it seems that a heap that is too large increases the CPU utilization (user space CPU), probably caused by an increased garbage collection duration that degrades performance when compared with a smaller heap size. For the other studies with 8 GB memory, a heap size of 75%, which is 6144 MB, was used.

Comparing 64-bit WebSphere versus 31-bit WebSphere

This study compares the performance of 64-bit WebSphere to 31-bit WebSphere .

Performance of 64-bit WebSphere

The workload stresses all of the major J2EE components in the middleware layer, but does not stress the database or client software layers. The intent was to explore how deploying on a WebSphere Application Server running the 64-bit JVM versus deploying on the 31-bit version affected performance. This workload uses most of the J2EE components, including some components that other applications are only beginning to exploit, such as message queues and the service oriented architecture (SOA). Here data points are used where expected CPU utilization would not be maximized as it was in the CPU scaling studies, so CPU usage is not a limiting resource.

Costs and advantages of using 64-bit WebSphere

Significant performance gains are expected in applications capable of taking advantage of 64-bit WebSphere features. For example, reducing database requests by leveraging a large heap space to cache database data can provide significant gains. However, there is also a disadvantage for 64-bit WebSphere applications. All address references are 64-bits wide, roughly double the size of address references in 31-bit deployments. This results in an increased memory footprint and can reduce hardware cache efficiency. Therefore, applications might actually see a performance loss. The 64-bit processors also provide hardware support for double-precision numbers and wider 64-bit integers.

Heapsize and other factors

As with other studies, a 75% heap size value is chosen for all variations of the 64-bit WebSphere test, and used the 1024 MB (1 GB) heap size variation of the 31-bit WebSphere test. For 31-bit WebSphere, a mapped_base kernel setting is used to reduce the kernel memory footprint and provide more space for the 1 GB heap to live inside the 2 GB 31-bit address space.

At a workload submission rate of 600, the response times were unacceptably long. Reports generated by the sar command show that the network traffic between the client systems and the WebSphere system was exceeding the capacity of the network card on the WebSphere Application Server LPAR. Therefore, the 1 Gb OSA card is replaced with a 10 Gb OSA card for the test cases with a workload submission rate of 600. The network issue is discussed in detail in “Network study – 1 Gb Ethernet versus 10 Gb Ethernet” on page 42.

Garbage collection

The **-Xgcpolicy:gencon** garbage collection option specifies a choice of either Split Heap or Generational Concurrent garbage collection mode. The heap is split into two areas: the tenured area for long-lived objects, and a *nursery area*, where new and recently used objects are stored. This garbage collection features a movement of longer-lived objects (ones that have survived between five and ten scavenger garbage collections) to the tenured heap. The global garbage collection is a garbage collection of the tenured area with the longest-lived objects. This global garbage collection is associated with pause times, during which objects managed by the JVM are held exclusively (locked) and not available to the application. The Generational Concurrent option was chosen for this workload due to the existence of very short-lived objects coexisting with longer-lived J2EE objects that survive beyond the transactional commit phase.

The tenured heap size is set with the **-Xmo<sizeM>** JVM option. For example, **-Xmo768m** defines a 768 MB tenured area. The remainder of the heap is then dedicated to a nursery area. The **-Xmo2048m** option was chosen for the tenured area on the 64-bit runs using 8 GB of memory. The option **-Xmo768m** was chosen for the 31-bit JVM because the total heap for the JVM is only 1024 MB. It is possible to increase the tenured area to an even larger size on 64-bit WebSphere, but at the value of 2048m for the tenured area, only one global garbage collection occurs during the 10 minute steady-state interval, and that frequency of tenured area maintenance is considered acceptable. For more information about these options, see "Other Java Virtual Machine arguments" on page 22.

Methodology

CPU utilization, workload throughput, and response times were measured during the steady-state phase with workload submission rates of 300, 500 and 600. For the 300 workload submission rate, four dedicated CPUs were assigned to the system being tested. For the 500 and 600 workload submission rates, eight dedicated CPUs were used. The DB2 image continued to run with four CPUs and 8 GBs of memory. The submission rates reflect a high but not fully-utilized CPU load.

CPU utilization, throughput, and response time measurements are summarized in Table 6 and displayed graphically in Figure 10 on page 34 and Figure 11 on page 34.

Table 6. 64-bit and 31-bit WebSphere comparison: CPU utilization, workload throughput, and response time measurements

31-bit or 64-bit JVM	Workload submission rate	Normalized workload throughput	Number of CPUs	CPU utilization	Response time (ms)	gc* sec/sec
31-bit	300	100%	4	302%	323	147/0
64-bit	300	100%	4	333%	337	61/0
31-bit	500	165%	8	551%	651	246/0
64-bit	500	165%	8	538%	709	76/0
31-bit	600	200%	8	612%	562	276/9
64-bit	600	200%	8	641%	518	127/1
*gc = nursery heap garbage collection in seconds divided by tenured heap garbage collection in seconds						

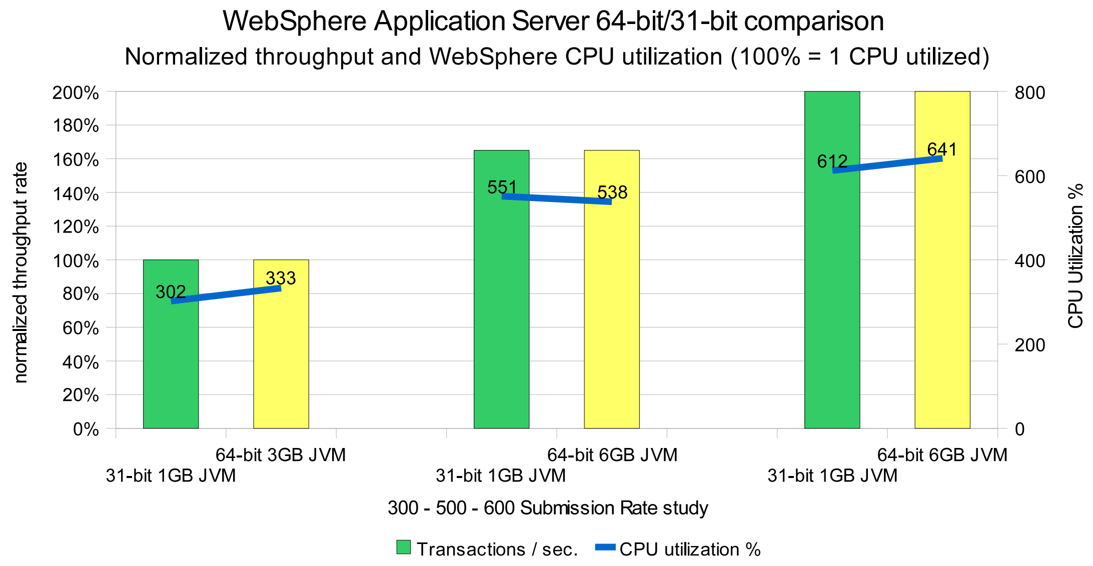


Figure 10. 64-bit and 31-bit WebSphere comparison: CPU utilization and throughput

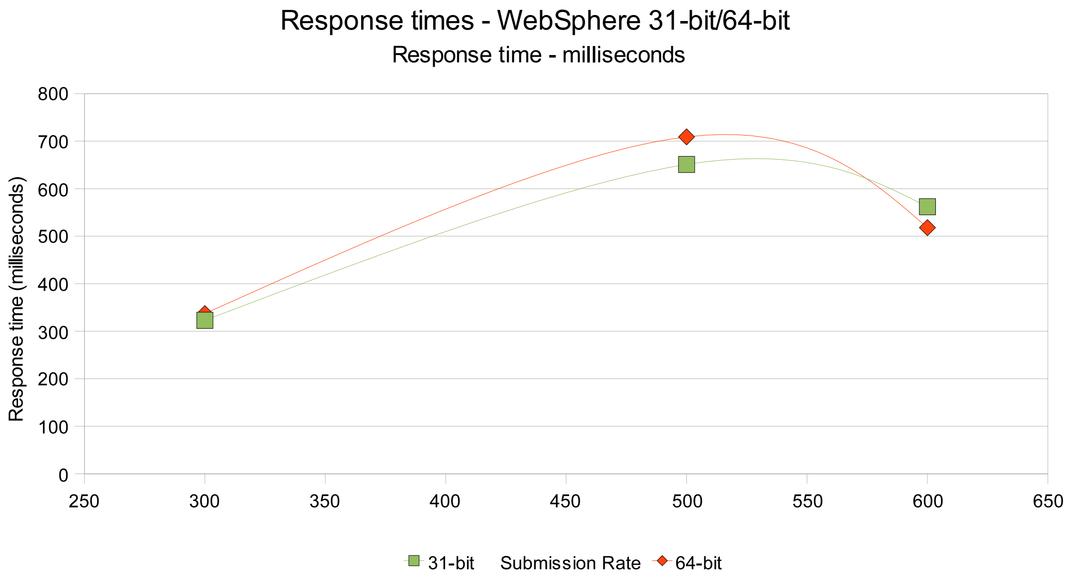


Figure 11. 64-bit and 31-bit WebSphere comparison: Response time (in milliseconds)

Observations

The same throughput is observed for 31-bit and 64 bit WebSphere Application Servers. Also, the CPU utilization is very similar. There is a difference in the response time, which is slightly slower on 64-bit WebSphere Application Servers, until the 600 submission rate test, and then the response time on 64-bit WebSphere Application Servers is slightly better than 31-bit WebSphere Application Servers.

The Java garbage collection statistics were added to show the effect of the larger heap sizes of 64-bit WebSphere on the total amount of time spent in garbage collection. More total time spent in garbage collection is observed on the 31-bit WebSphere than on the 64-bit version. On 31-bit, the average duration of garbage collection intervals is much shorter, but the frequency of garbage collection is much higher than on 64-bit. The total garbage collection time is used, which takes into account both garbage collection frequency and duration.

Using total garbage collection time, the time spent in garbage collection on 64-bit WebSphere was, approximately between 30% and 46% of the garbage collection time spent on the 31-bit version, significantly lower. At the highest workload submission rate of 600, the garbage collection time of the tenured heap area (Global garbage collection) was up to ten times longer on 31-bit WebSphere than on 64-bit WebSphere.

Conclusions

Under this workload, the behavior of the WebSphere Application Server is very similar for both 31-bit and 64-bit versions. With the larger workload, the 64-bit WebSphere Application Server has better response time, probably because garbage collection behaves differently. The **-Xgcpolicy:gencon** option was specified on the JVM command line for 64-bit WebSphere as well as the 31-bit WebSphere installation. The design of Generational Concurrent garbage collection is to minimize the time spent in global garbage collection of the tenured heap area by doing some concurrent cleanup of new object areas called *nurseries*.

Although pause times from exclusive locks held by global garbage collection are minimized by the JVM, they still consumed 9 seconds of total time on 31-bit WebSphere versus 1 second of total time on 64-bit WebSphere. This is one parameter that causes the slightly better response times observed on 64-bit WebSphere at the higher workloads with the larger 6 GB JVM heap. The larger tenured area (2048 MB versus 768 MB) on 64-bit WebSphere showed that fewer *stop the world* global garbage collections of the tenured area are needed on 64-bit WebSphere. Another reason for better response time is that the large nursery area afforded by the large 64-bit heap allows for the storage of more short-lived objects in memory for longer durations of time, resulting in fewer scavenger garbage collections and fewer memory allocation failures in the nursery area of the split heap.

These differences should result in improved performance on 64-bit WebSphere. However, this improvement is probably offset by the generally higher CPU requirements of 64-bit WebSphere observed for the workload.

CPU scaling study

This study shows how the workload scales when workload submission rates are increased while available dedicated Central Processing Units (CPUs) are scaled.

Introduction to CPU Scaling

CPU scaling is a measure of how much workload can be driven when the CPU resources are increased. An increase of workload can occur when the number of total transactions or the transaction rate are increased. For this workload, the workload submission rate (the rate at which work is submitted to the J2EE middleware layer), has to be increased. However, increased workload in this study also requires a larger database, which means that not only the workload must be scaled, but the whole environment. Scaling the whole environment might have other effects on the performance than just doing more work with the same data.

Maximizing CPU utilization

To determine the performance characteristics of the workload, measurements are taken using one, two, four, and eight dedicated CPUs on the WebSphere system. A workload entry rate is chosen that is high enough to drive the CPUs to near full utilization. The results can be used to gain a better understanding of the scalability of the workload, and can be used as a way to measure differences in the performance of the same workload on 64-bit WebSphere versus 31-bit WebSphere.

In all 64-bit WebSphere measurements, the heap settings for the JVM are set to 75% of the 8 GB available memory. This is the optimum percentage derived from the study “Heapsize for the 64-bit Java Virtual Machine” on page 29. That worked out to a 64-bit WebSphere JVM heap settings of **-Xms6144m -Xmx6144m**. A memory size of 8 GB is also configured for the DB2 LPAR, which runs with four configured CPUs for all of the tests.

10 Gb Ethernet chosen for highest workload

The workload submission rate of 600 was found to exceed the capacity of the 1 Gb Ethernet network. This causes network saturation, dampening throughput and providing additional work for error handling. The 600 workload submission rate tests are therefore run using a 10 Gb Ethernet, to remove the effects of a network bottleneck on the results. A submission rate higher than 600 would have required a larger restructuring of the environment, because of the higher resource usage from the clients to WebSphere and up to the database. This would have exceeded the scope of the study.

CPU Scaling

Dedicated CPUs are assigned to the WebSphere System being tested. The experiments use one, two, four, or eight dedicated CPUs. The workload is then varied until a CPU utilization close to or greater than 90% is observed. The workload is adjusted by changing the workload submission rate. When eight CPUs are dedicated to the WebSphere image, only approximately 80% total CPU utilization at a workload submission rate of 600 was observed. This is because, as explained in “10 Gb Ethernet chosen for highest workload,” different WebSphere or client tuning values would have been needed for a submission rate greater than 600.

Transaction scaling and response time measurements

The transaction rate is the throughput as reported by the client-side summary reports. Response time measured is the observed response time of a simulated Web operation (such as an online Web purchase or a Web browse operation) and the turnaround of the Web request from the WebSphere system after the completion of some business logic. These response times are averaged with response times for manufacturing operations. The performance of the DB2 subsystem is also represented in this data. Table 7 on page 37 summarizes these results in tabular format. Figure 12 on page 37, Figure 13 on page 38, and Figure 14 on page 38 are graphical representations of the results.

With this workload, the throughput measurements stay fairly constant and performance degradations are first indicated by increasing response times and CPU utilization.

Table 7. CPU scaling study: WebSphere V6.1 CPU Scaling results for one, two, four, and eight CPUs at high CPU utilization

31-bit or 64-bit JVM	Workload submission rate	Workload throughput	Number of CPUs	CPU utilization	Response time (ms)
31-bit	110	101%	1	87%	432
64-bit	110	100%	1	97%	793
31-bit	190	174%	2	175%	385
64-bit	190	174%	2	189%	654
31-bit	350	322%	4	353%	361
64-bit	350	321%	4	371%	404
31-bit	600	551%	8	612%	562
64-bit	600	550%	8	641%	518

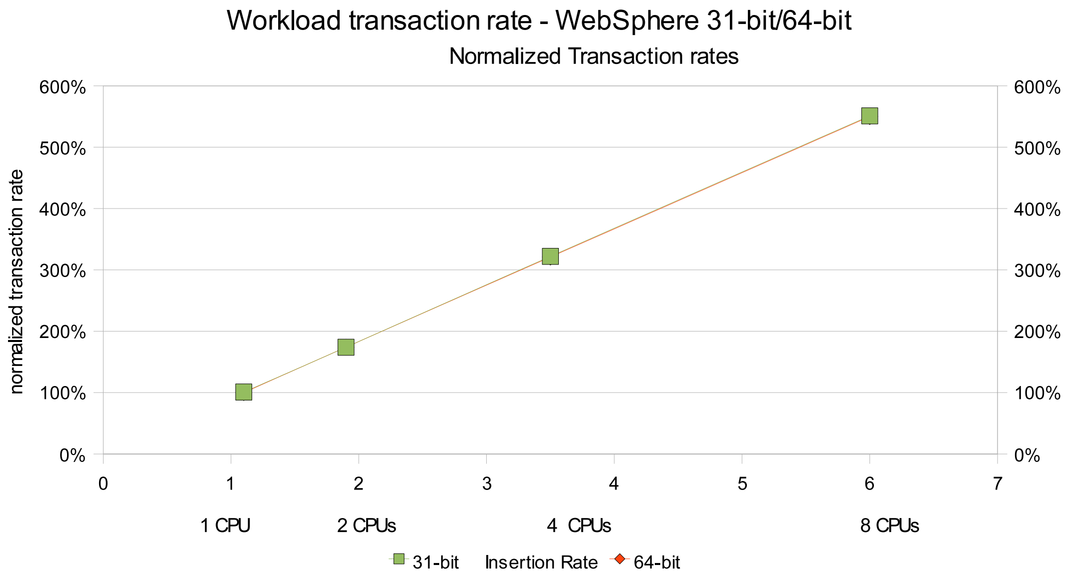


Figure 12. CPU scaling study: Workload transaction rates with 31-bit and 64-bit WebSphere

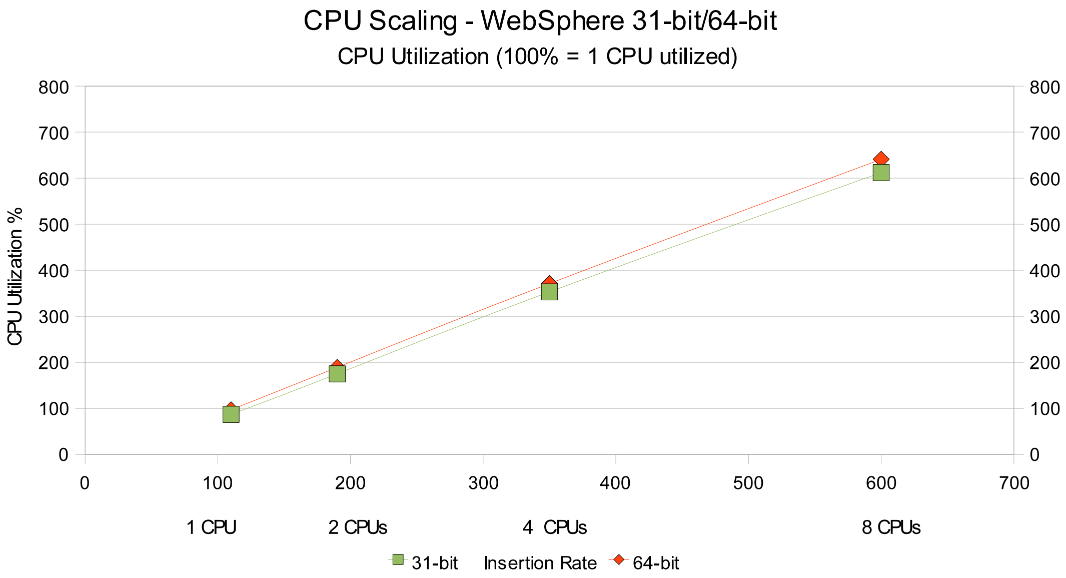


Figure 13. CPU scaling study: CPU utilization with 31-bit and 64-bit WebSphere

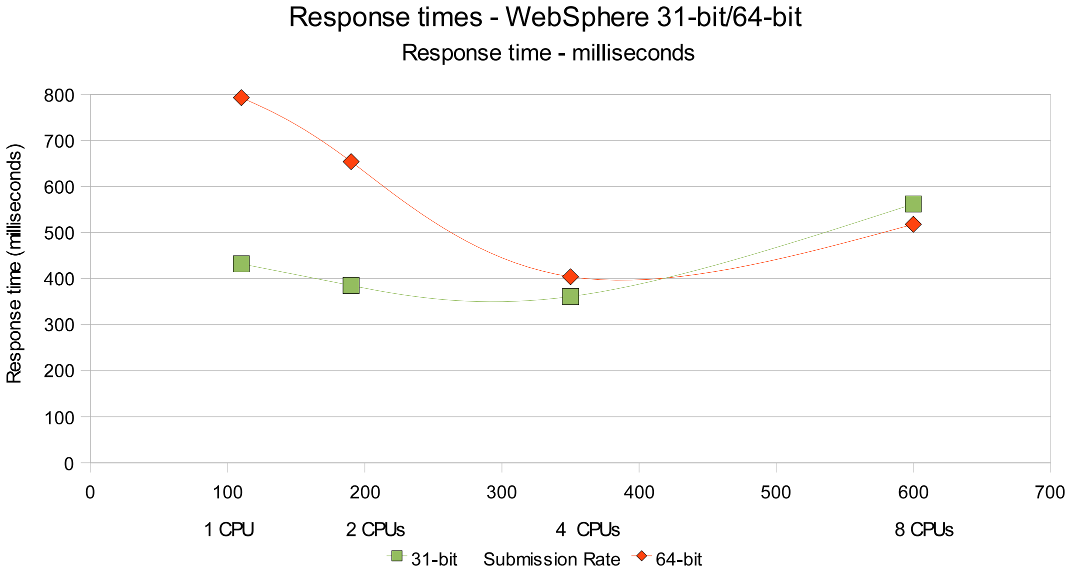


Figure 14. CPU scaling study: Response times with 31-bit and 64-bit WebSphere

Observations

The workload scales very linearly for both 31-bit and 64-bit WebSphere Application Servers. The 31-bit version requires a little less CPU at higher workloads than the 64-bit version. The utilization of the CPUs also scales very linearly for both 31-bit and 64-bit WebSphere.

An unexpected behavior is shown by the response time. The response time becomes shorter with the higher workloads when using a larger number of CPUs, and increases again on the last scaling step with the highest workload. Here, the 31-bit WebSphere Application Server differs significantly from the 64-bit WebSphere Application Server; the response time with one CPU is much shorter, but the gap decreases with the scaling. At a submission rate of 600 with eight CPUs, the 64-bit WebSphere Application Server's response time becomes shorter.

Conclusions

The very good linear scaling in throughput and CPU utilization makes scaling of this workload easy for a system administrator. The difference between the 31-bit WebSphere and the 64-bit WebSphere is small. The more efficient garbage collection of the 64-bit version seems to compensate for the drawback of the larger memory addresses, as seen in other studies. See http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux390/perf/ZSW03030-USEN-00.pdf.

At higher workload submission rates, the advantages of a larger heap on 64-bit WebSphere result in an improving response time curve. A more detailed analysis of garbage collection can be found in "Comparing 64-bit WebSphere versus 31-bit WebSphere" on page 31.

A CPU utilization greater than 90% was not observed for the higher workloads, indicating an unidentified bottleneck, which might be the HiperSockets connection between the WebSphere Application Server and the database. Additional investigation would be required to determine the cause of this bottleneck. The high CPU utilization of 97% of the one CPU run with the 64-bit WebSphere becomes critical for a system running with HiperSockets, and is very likely the reason for the high response times there.

Large heaps provide more space for both long-lived and newly-created objects. It seems that the Generational Concurrent garbage collection option works very efficiently, even for this workload, which was designed to have a high load and resource utilization on the WebSphere Application Server.

Database LPAR analysis

This study measures the effect on the DB2 relational database of scaling the workload and the CPUs on the WebSphere system.

Even though the transactional workload stresses the DB2 backend database system only slightly, this stress shows the effects resulting from scaling the workload and the CPUs on the WebSphere system.

Methodology

The DB2 snapshot tool is used to collect database snapshot data before and after the steady state portion of the workload. The Dynamic SQL statements attempted during the 10 minute steady state were added to the static SQL statements, and a DB2 workload throughput rate is established. CPU utilization data was also obtained. The CPU cost per unit is calculated as **sql/sec/CPU * 100**, meaning that the number of transactions driven with 100% CPU (== 1 Integrated Facility for Linux (IFL)). This is the value shown in the **sql/sec/CPU * 100** column. It is a measurement of the throughput that takes into account the CPU cost.

Table 8 summarizes these results in tabular format. Figure 15, Figure 16 on page 41, and Figure 17 on page 41 are graphical representations of the results.

Table 8. Database LPAR analysis: DB2 UDB CPU utilization and throughput

Workload throughput	31-bit or 64-bit WebSphere	CPU utilization	SQL statements per second	sql/sec/CPU * 100
101	31-bit	23%	4193	18326
100	64-bit	25%	4282	17350
174	31-bit	38%	7263	19013
174	64-bit	39%	7233	18700
322	31-bit	65%	13500	20692
321	64-bit	65%	13476	20732
551	31-bit	115%	21853	19057
550	64-bit	120%	22973	19176

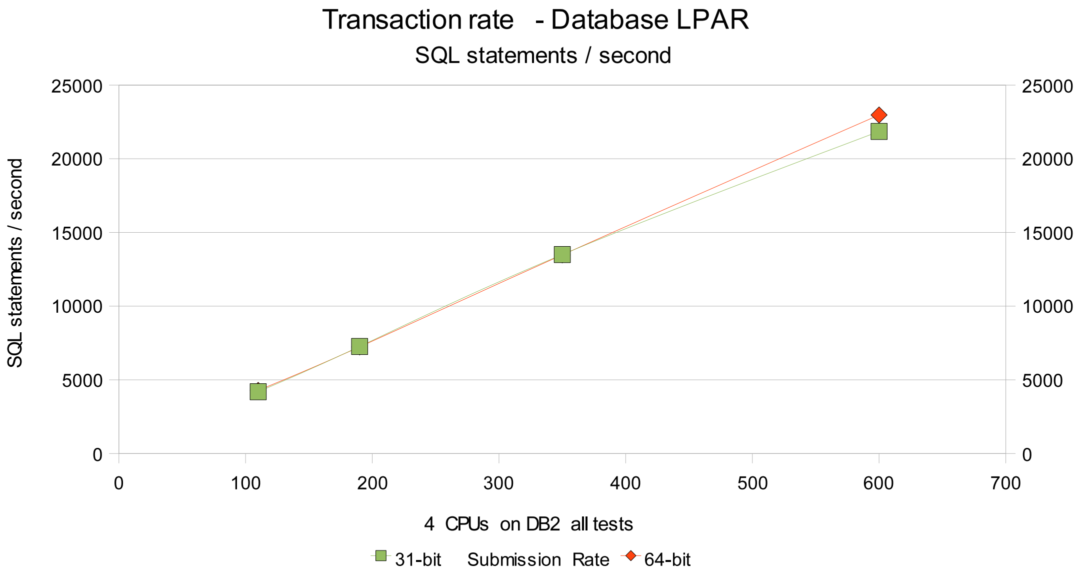


Figure 15. Database LPAR analysis: Database transaction rate for scaling WebSphere Application Server CPUs and workload submission rate.

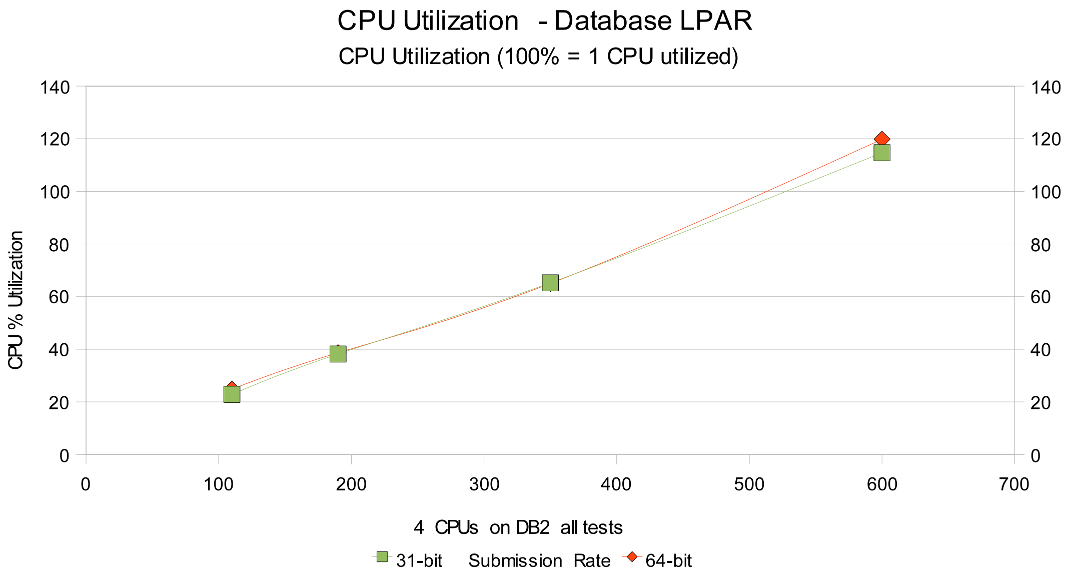


Figure 16. Database LPAR analysis: CPU utilization on Database LPAR

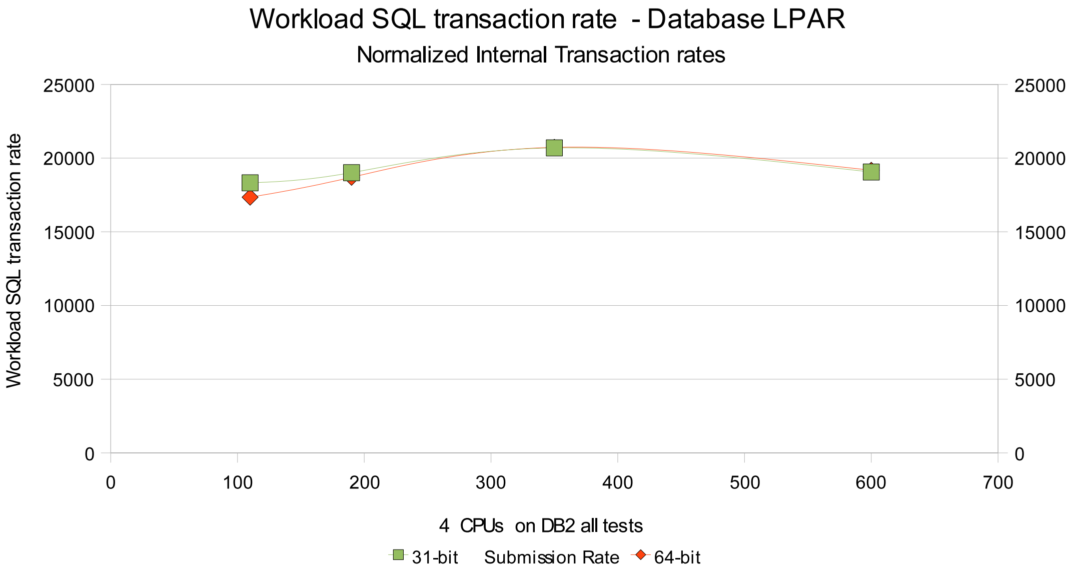


Figure 17. Database LPAR analysis: Normalized internal SQL transaction rate

Observations

The number of SQL statements per second for the 64-bit WebSphere Application Server is very linear, while the 31-bit WebSphere Application Server causes fewer statements per second at a submission rate of 600. The CPU utilization scales linearly up to a submission rate of 350, with a higher slope to the submission rate of 600, where it increases faster with a 64-bit WebSphere Application Server.

Looking at the CPU cost in terms of how many statements are driven with one CPU, it is shown that the cost is independent of the addressability from the application server, except for the submission rate of 110. It is also shown that submission rate 350 achieves the largest number of SQL statement processed with the database CPUs.

Conclusions

The CPU load of the DB2 LPAR is generally quite light for the transaction workload. A full CPU is rarely used unless there are eight CPUs used on the WebSphere system. The workload generally uses short data records and does not use complex SQL statements, so the amount of backend database load is expected to be light.

It should be remembered that in an end-to-end transaction, the performance of the middleware will affect the DB2 CPU utilization and workload throughput. The DB2 load originating from 31-bit or 64-bit WebSphere was nearly identical. The higher cost on the 64-bit version at a submission rate of 110 strongly suggests that the WebSphere CPU load is too high for the HiperSockets connection, because missing CPU resources for the HiperSockets on the middleware will cause overhead on the database for re-sending packets.

Network study – 1 Gb Ethernet versus 10 Gb Ethernet

This is a study of network traffic between the WebSphere Application Server and the workload generating clients. Network traffic is measured during workload runs, and the issue of network saturation is discussed.

Network considerations

In analyzing the first data from the workload submission rate 600 runs, as shown in "Comparing 64-bit WebSphere versus 31-bit WebSphere" on page 31, response times increased dramatically from sub-seconds to approaching 5 seconds. With CPU utilization at less than 75%, little or no swapping has occurred, indicating that there is no contention for resources. While the configuration of WebSphere and DB2 was designed to take advantage of the available memory, the near absence of swapping to disk indicates that constrained memory did not cause the poor response time.

Examination of the sar and netstat command data from submission rate 500 runs showed good response time, so it had to be determined if the workload submission rate of 600 created network traffic exceeding the bandwidth of the 1 Gb OSA Express2 Ethernet card. Some benchmark measurements indicated some reduced throughput at a workload submission rate of 600.

Methodology

Several key measurements are taken to establish network traffic rates at different data points. Network traffic bits per second are found in the sar command report under these two fields:

rxkB/s Represents kilobytes per second read.

txkB/s Represents kilobytes per second transmitted.

To use these values to obtain a total number of network bits per second, perform this calculation:

1. Add the **rxkB/s** and **txkB/s** values together.
2. Multiply this sum by 1024. This produces the total number of bytes.
3. Multiply the total number of bytes by eight. This produces the total number of network bits per second.

The total number of network bits per second is a good overall measurement of network traffic.

Two measurements of network congestion consistent with an I/O-bound workload were obtained. The first measurement, segments retransmitted, came from the netstat command report. A netstat snapshot before and after the workload is taken, because the ramp-up or warmup phase of the benchmark would also accumulate network data and that should not be included in the measurements.

To calculate the number of segments retransmitted during the steady-state phase, subtract the number of segments retransmitted found in the first netstat command report (that included ramp-up data) from the number displayed in the second report.

The second measurement, **txdrop/s** is the number of packets dropped per second, because of resource constraints. It is found in the sar report.

Table 9 summarizes these results in tabular format. Figure 18 on page 44 and Figure 19 on page 44 are graphical representations of the results.

Table 9. Network study: Data read and written per second, segment retransmits, and packets dropped

31-bit/64-bit WebSphere	Workload submission rate	Network card link speed	Megabits read or written per second	Segments retransmitted	txdrop/sec
31bit	500	1 Gb Ethernet	746	N/A	0.46
64-bit	500	1 Gb Ethernet	749	N/A	0.29
31bit	600	1 Gb Ethernet	770	N/A	241
64-bit	600	1 Gb Ethernet	783	158,828	249
31bit	600	10 Gb Ethernet	901	4232	0.75
64-bit	600	10 Gb Ethernet	902	198	0.07

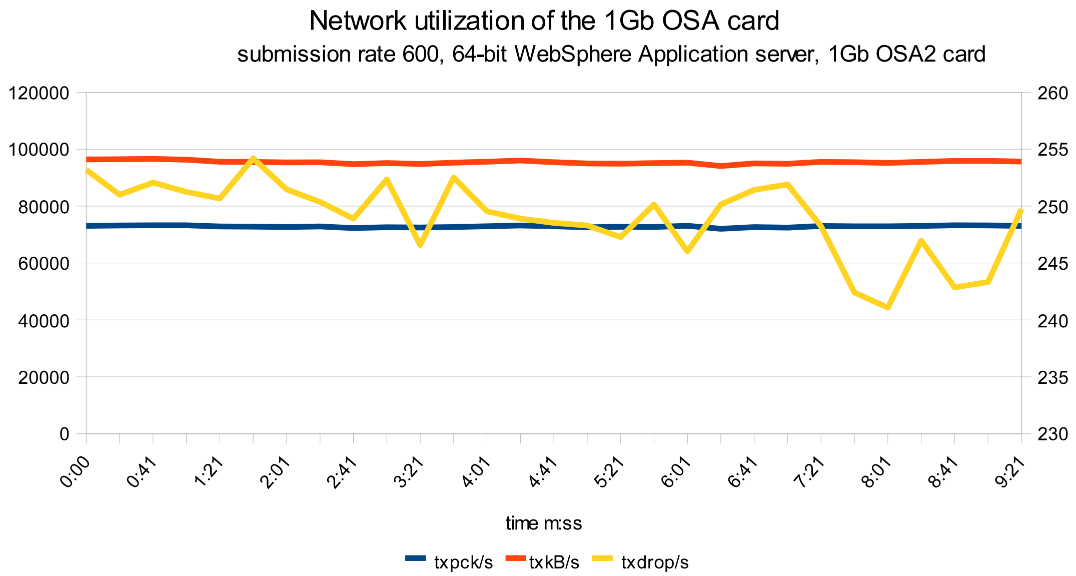


Figure 18. Network study: Utilization of the 1 Gb OSA card from the WebSphere Application Server from the traffic to the clients

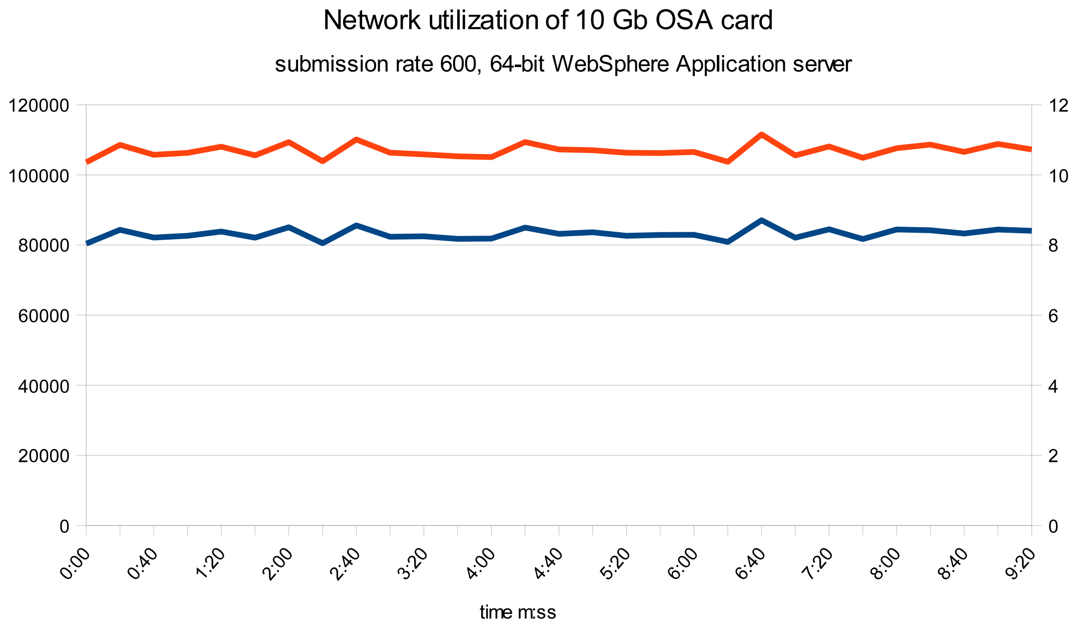


Figure 19. Network study: Utilization of the 10 Gb OSA card from the WebSphere Application Server from the traffic to the clients

Observations

The number of packets dropped per second was a high value (greater than 240) for both 64-bit and 31-bit WebSphere. Related to these same high drop rates is a high number of segment retransmits, which additionally increase the load on the network card. With the 10 Gb Ethernet card, the total throughput increased by 12%, and the amount of packages increased by 14%.

Conclusions

The poor response times with the workload submission rate of 600 could be easily attributed to network I/O traffic reaching the limit of the 1 Gb Ethernet on the WebSphere system being tested. The results also show that for this workload, the practical limit for good response time on a 1 Gb Ethernet might be approximately 780 Mb per second, which indicates that the throughput itself is not the major limiting factor.

Another factor is the number of packages, where the number of 80 000 packages or more could be considered as close to the upper limits. There is seen, on average, a very small package size here (less than 100 bytes). In these cases, the maximum throughput will not be reached because the high number of I/O requests per second are the limiting factor. This might also reach limits on other resources such as the network switch.

Appendix. Configuration, tuning, and performance scripts

These configuration, tuning, and performance scripts were used in the 64-bit and 31-bit WebSphere study with J2EE workloads.

Script for first time database creation run before database load

```
db2 update db cfg for mydb using logfilsiz 65535
db2 update db cfg for mydb using logprimary 40
db2 update db cfg for mydb using newlogpath /db2log
db2 connect to mydb
db2 -v alter bufferpool ibmdefaultbp size 675000 automatic
db2 connect reset
db2stop;db2start
db2 connect to mydb
db2 connect reset
```

Tuning script for DB2 after database load

```
# DB2 Auto Configurator Example
#
# Using this command can help you tune DB2 automatically for your environment
#
# 1. Manually adjust your log file locations
# 2. Specify the values that is needed for DB2 Auto Configurator
#
#
# Example:
#
# db2 autoconfigure using mem_percent 80 workload_type simple num_stmts 60
tpm 2000 is_populated yes
#             num_local_apps 0 num_remote_apps 65 isolation rs bp_resizeable
yes apply db and dbm
#
#  mem_percent - the percentage of system memory you would like to use for
bufferpool memory.
#
#  Workload_type - for our purpose, WebSphere generally uses simple SQLs.
#
#  num_stmts - number of statements involved in the workload
#
#  tpm - estminate the transaction per minute that you want DB2 to handle
#
#  is_populated - is the database populated with data?
#
#  num_local_apps - number of applications connecting locally.
If you setup is a two tier config with WebSphere on a separate machine,
then enter '0'
#
#  num_remote_app - number of applications that will connect to DB2 remotely.
This number should be the summation of all the connection pool sizes from
WebSphere. For our workload, it is the size of Connection Pool size of the DB.
If it is a WebSphere cluster, multiply it by the number of WebSphere nodes.
#
#  isolation - specify the highlest isolation level your Application demands.
#
#  bp_resizable - bufferpool resizable or not. Generally is 'yes'
#
#  apply db and dbm - apply the settings on your behalf.
#
#  I have a dedicated DB2 machine that is in a 2-tier setup separated from the
WebSphere machine.
#  - use 80% of the memory as my buffer pool.
# - workload has about 60 statements,
# - simple workload,
#       - tpm is dictated by workload rate, so my guess it will hit 2000 max
for my work on this machine.
# - Data is already loaded. I have 65 connections coming in from Websphere,
# - most of the beans require isolation level rs.

db2 connect to mydb
db2 autoconfigure using mem_percent 80 workload_type simple num_stmts 60
tpm 2000 is_populated yes num_local_apps 0 num_remote_apps 100 isolation
rs bp_resizeable yes apply db and dbm
#Run the following db2 terminate or else the changes won't take.
db2 terminate
db2 connect to mydb
db2 reorgchk update statistics on table all
db2 terminate
```

Shell script to capture DB2 performance statistics

```
#invocation:   ./startgatherDB2 testname
db2 -v connect to mydb
db2 -v update monitor switches using bufferpool on
db2 -v get snapshot for database on MYDB >snapA$1.out
db2 -v get snapshot for tablespaces on MYDB >tablesnapA$1.out
db2 -v get snapshot for all applications >appsnapA$1.out
db2 -v get snapshot for all bufferpools >buffersnapA$1.out
date >>snapA$1.out
############## Beginning of steady state – sleep through the whole thing!
sleep 600
db2 -v get snapshot for database on MYDB >snapC$1.out
date >>snapC$1.out
db2 -v get snapshot for all applications >appsnapC$1.out
#############
db2 -v reset monitor all
```

Gather performance statistics on WebSphere and DB2

```
echo 'in startmeas'
mkdir $1
cd $1
#First argument is Directory off the current directory
#Second argument is total seconds to run - choose 600 for 10
#Third argument is seconds between sadc snapshots - typically 20
let totsamps=$2/$3
echo "created directory /results/$1"
echo "Running for a total of $2 seconds"
echo "Total sar snapshots is $totsamps"
/usr/lib/sa/sadc $3 $totsamps sadc.out.$HOSTNAME &
netstat -as >netstat.$1.out1.$HOSTNAME
sleep $2
netstat -as >netstat.$1.out2.$HOSTNAME
```

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

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

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

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

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

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
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Edition notices

Trademarks

IBM, the IBM logo, ibm.com®, DB2, DB2 Universal Database, DS8000, ECKD, HiperSockets, Resource Link™, System x, System z, System z10, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Adobe®, the Adobe logo, PostScript®, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine™ is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside®, Intel Inside logo, Intel® Centrino®, Intel Centrino logo, Celeron®, Intel® Xeon®, Intel SpeedStep®, Itanium®, and Pentium® are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of the manufacturer.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of the manufacturer.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any data, software or other intellectual property contained therein.

The manufacturer reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by the manufacturer, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

THE MANUFACTURER MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THESE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA