September 2007

z/VM Virtualization Performance

Table of Contents

Preface

Level of understanding

Knowledge and understanding of enterprise-level computer environments, WebSphere® Application Server, and z/VM® will help you understand the results detailed in this paper.

Objectives

There are new virtualization techniques in the market for various platforms. Our objective in these tests was to show the scalability of WebSphere Application Server environments using the Trade benchmark running under the virtualization product z/VM. One environment consisted of an IBM HTTP Server, a WebSphere Application Server, and an IBM DB2 Universal Database™ (UDB) all installed on separate Linux® guests. This setup is referred to a triplet in this document.

A special feature of a virtualized environment is that it is possible to assign the guests more virtual resources (memory, processors) than physically available; this is called overcommitment. The possibility to overcommit makes virtualization very attractive because it allows a much better utilization of the hardware resources.

We wanted to see:

- *The effects on the total throughput and CPU load when the number of triplets is scaled*
- *The effects of overcommitting resources (processors, memory)*
- *A comparison of the performance (based on the Trade 6 workload) between using Secure Sockets Layer (SSL) encryption and no encryption in our configuration.*
- *The effects of the z/VM quick dispatch option on our environment*

This information will help differentiate the value of an IBM System z™ solution against likely competitors.

Executive summary

This chapter provides a short summary of our test results. Our test results and recommendations are specific to our environment. Parameters useful in our environment might be useful in other environments, but are dependent on

application usage and system configuration. You will need to determine what works best for your environment. For our detailed test results information, see Results.

The following are our summary results:

- *z/VM manages resources very efficiently so throughput did not degrade in the overcommitted scenarios. We overcommitted the CPUs up to 150% and the memory up to 25%. On z/VM, the CPU overcommitment had only a slight impact as long as the system CPUs were not fully utilized.*
- *z/VM optimizations regarding memory handling were very efficient for our scenarios. When comparing the defined virtual machine size with the physical memory we had a memory overcommitment of 25%. However, only about 10 GB of the 25 GB of memory allocated was used. Surprisingly we did not have a memory shortage.*
- *Using the quick dispatch (QUICKDSP) option for z/VM guests can have a benefit when there are resource (CPU, memory) overcommitments.*
- *Because response times on z/VM are very short even though the system is already overloaded, it is safe to assume that z/VM would support several more triplets until very bad response times of about half a second or more are reached. When making comparisons, it is important to not only compare the number of supported users or workload, but also the response times the users see.*
- *Be aware that the hardware platform also contributed to our results. This means that z/VM takes advantage of System z hardware, which is designed to support virtualization products using hardware in a shared mode, and is stable with short response times under a heavy load.*

Hardware equipment and software environment

This chapter provides details on the hardware and software used in our testing. Topics include:

- *Server and client hardware used*
- *Server and client software used*
- *The test environment*
- *A description of the workload used*

Hardware and software

The following section details the hardware and software we used for our z/VM test runs.

Server hardware

Host

One LPAR on a 16-way IBM System z9™ Enterprise Class (EC), Model 2094-S18, equipped with:

- *8 physical CPUs, dedicated*
- *20 GB central memory*
- *2 GB expanded memory*
- *OSA Express 2 Ethernet card*
- *2 FICON® Express Channels*
- *5 FCP Channels 2 Gb/sec*
- *1 CEX2C Crypto card*

Network setup

- *Web Servers and client workstations on a 1 Gb Ethernet LAN*
- *All Linux guests also used a z/VM HiperSockets™ guest LAN, 24 K frame size*

Storage server setup

IBM System Storage™ DS8000™, Disk Drive Modules = 73 GB each/15000 RPMs

For the operating system and applications on 15 Linux guest systems:

- *30 ECKD™ mod9 spread over one rank/LCU*
- *2 FICON paths*

For the database data disks:

- *5 SCSI disks over 5 FCP channels/ranks. 1 channel per database system. 10 GB each from the same DS8000 as the ECKD disks.*

Server software

Table 1. Server software used

Product	Version/Level
IBM DB2 Universal Database Enterprise Server	8.2, fixpack 4
IBM HTTP Server	6.0.2.0
SUSE Linux Enterprise Server	SLES9, SP3, 64-bit
Trade	6.02
WebSphere Application Server	6.0.2.1, 31-bit
z/VM	5.2.0

z/VM guests setup:

15 Linux guests

Table 2. z/VM guests setup

Hostname	lnweb(1-5)	lnwas(1-5)	lnudb(1-5)
Function	Web Server	WebSphere Application Server	DB2® UDB Server
Number of virtual CPs	1	2	1
Defined memory	1 GB	2 GB	2 GB
Crypto enabled	Yes	No	No

Client hardware

- *1 x 2.4 GHz H20 Intel® Blade with 4 GB RAM (works as the Trade workload generator)*
- *IBM eServer™ xSeries® model x335, 2x2.8 GHz, Intel Xeon® 2 GB, 72 GB (works as a Trade controller and to gather the Trade statistics from the workload generators)*

Client Software

Table 3. Client software Used

Product	Version/Level
H20 Intel Blade	
Microsoft® Windows®	XP SP2
WebSphere Studio Workload Simulator	controller level - PQ76210
xSeries Model x335	
Red Hat Enterprise Linux	RHEL 4 AS (Trade client)
WebSphere Studio Workload Simulator	engine level - iwl-0-03309L

Environment

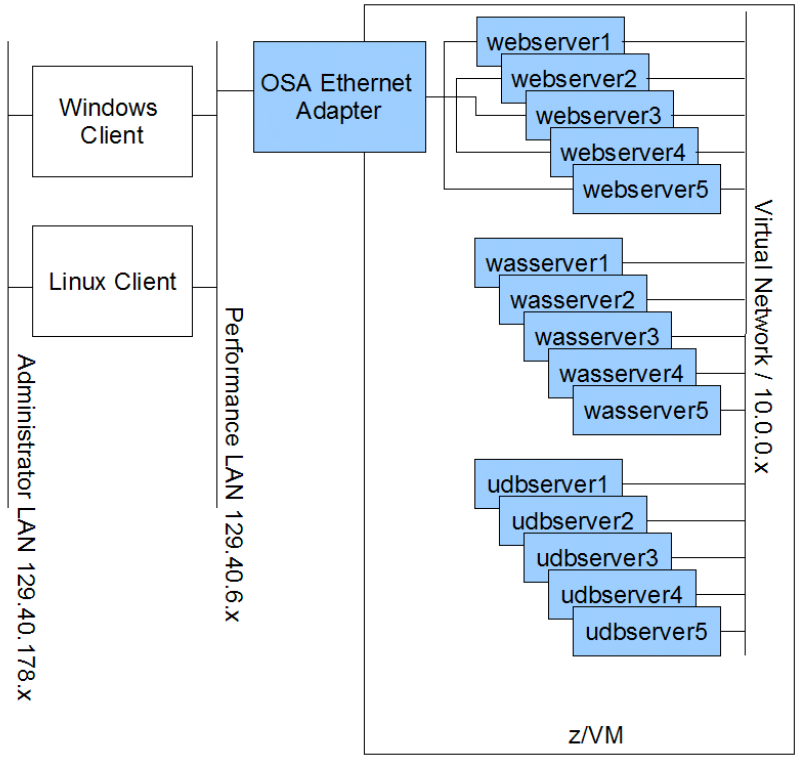

Figure 1. z/VM test environment

Workload description

The Trade performance benchmark is a workload developed by IBM for characterizing performance of the WebSphere Application Server. The workload consists of an end to end Web application and a full set of primitives. The applications are a collection of Java classes, Java Servlets, Java Server Pages, Web Services, and Enterprise Java Beans built to open J2EE APIs. Together these provide versatile and portable test cases designed to measure aspects of scalability and performance. Figure 2 shows an overview of the Trade J2EE components.

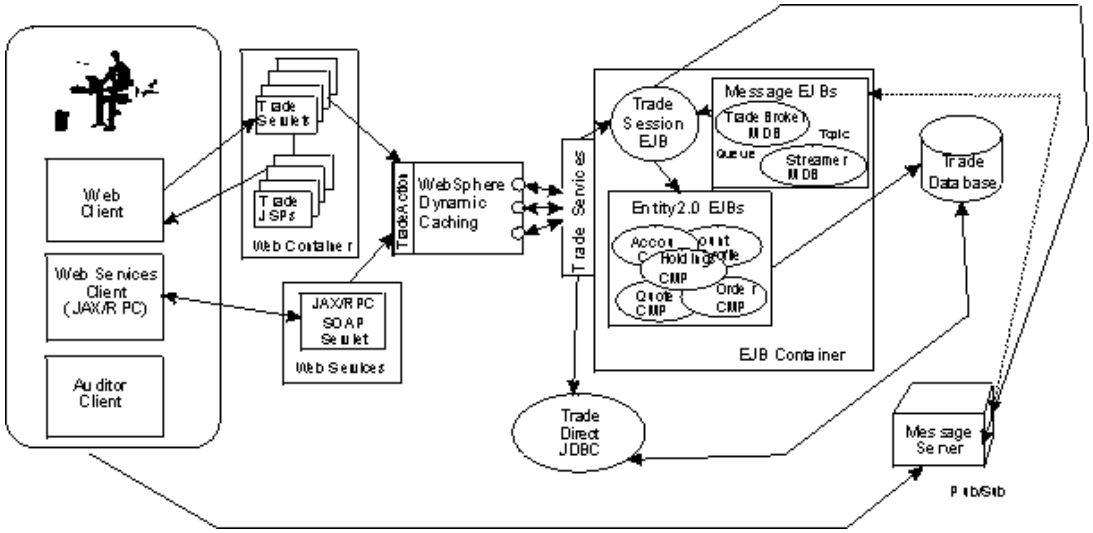

Figure 2. Trade J2EE components

The new Trade benchmark has been re-designed and developed to cover WebSphere's significantly expanding programming model. This provides a more realistic workload driving WebSphere's implementation of J2EE 1.4 and Web Services including key WebSphere performance components and features.

Trade's new design spans J2EE 1.4 including the new EJB 2.1 component architecture, Message Driven beans (MDBs), transactions (1-phase, 2-phase commit) and Web Services (SOAP, WSDL). Trade also highlights key WebSphere performance components such as DynaCache, WebSphere Edge Server, and Web Services.

Trade is modeled after an online stock brokerage. The workload provides a set of user services such as login/logout, stock quotes, buy, sell, account details, and so on, through standards-based HTTP and Web services protocols.

Trade provides the following server implementations of the emulated Trade brokerage services.

- *EJB - Database access uses EJB 2.1 technology to drive transactional trading operations.*
- *Direct - This mode uses database and messaging access through direct JDBC and JMS code*

Type 4 JDBC connectors are used with EJB containers. See Figure 2 for details.

To learn more about Trade, or to download the latest package, go to http://pulsar.raleigh.ibm.com.

EJB 2.1

Trade 6 continues to use the following features of EJB 2.0 and leverages EJB 2.1 features such as enhanced Enterprise JavaBeans Query Language (EJBQL), enterprise Web services and messaging destinations.

- *Container-Managed Relationships - One-to-one, one-to-many, and many-to-many object to relational data managed by the EJB container and defined by an abstract persistence schema. This feature provides an extended, real-world data model with foreign key relationships, cascaded updates and deletes, and so on.*
- *EJBQL - Standardized, portable query language for EJB finder and select methods with container-managed persistence.*
- *Local and Remote Interfaces - Optimized local interfaces providing pass-by reference objects and reduced security overhead.*

WebSphere Application Server provides significant features to optimize the performance of EJB 2.1 workloads. Trade uses access intent optimization to ensure data integrity while supporting the highest performing and scalable implementation. Using access intent optimizations, entity bean run-time data access characteristics can be configured to improve database access efficiency which includes access type, concurrency control, read-ahead, collection scope, and so forth.

The J2EE programming model provides managed, object-based EJB components. The EJB container provides declarative services for these components such as persistence, transactions, and security. The J2EE programming model also supports low-level APIs such as JDBC and JMS. These APIs provide direct access to resource managers such as database and message servers. Trade provides a Direct implementation of the server-side trading services using direct JDBC. This implementation provides a comparison point to container-managed services that details the performance overhead and opportunity associated with the EJB container implementation in WebSphere Application Server.

All the measurements done in this study used EJB.

Trade provides two order processing modes: asynchronous and synchronous. The order processing mode determines the mode for completing stock purchase and sell operations. Asynchronous mode uses MDB and JMS to queue the order to a TradeBroker agent to complete the order. Asychronous_2-Phase performs a two-phase commit over the EJB database and messaging transactions. Synchronous mode, on the other hand, completes the order immediately.

All the measurements done in this study used synchronous order processing mode.

Trade provides the following access modes to the server-side brokerage services.

- *Standard - Servlets access the Trade enterprise beans through the standard RMI protocol*
- *WebServices - Servlets access Trade services through the Web services implementation in WebSphere Application Server. Each trading service is available as a standard Web service through the SOAP Remote Procedure Call (RPC) protocol. Because Trade is wrapped to provide SOAP services, each Trade operation (login, quote, buy, and so on) is available as a SOAP service.*

All the measurements done in this study used the Standard access mode.

For all measurements in this study, the Trade database was populated with 500 users (uid:0 - uid:499) and 1000 quotes (s:0 - s:999).

Trade can run with any one of three WebSphere Application Server caching modes.

- *No cache - No caching is used.*
- *Command caching - This caching feature was added to WebSphere Application Server V5.0 for storing command beans in the dynamic cache service. Support for this feature was added in Trade 3 and carried over to Trade 6.*
- *DistributedMap - This feature is new in WebSphere Application Server V6.0, providing a general API for storing objects in the dynamic cache service.*

All the measurements done in this study used no caching.

Detailed information on the Trade workload can be found at http://pulsar.raleigh.**ibm.com**. Information on how Trade works with WebSphere Application Server can be found at http://www-306.**ibm.com**/software/webservers/appserv/was/performance.html.

Encryption

The communication between the workload generator and the Web server occurs either in clear text using the HTTP protocol or encrypted using the HTTPS protocol with SSL.

WebSphere Studio Workload Simulator

The Trade workload was driven by the WebSphere Studio Workload Simulator and the WebSphere Studio Workload Simulator script provided with the Trade distribution. Parameters for this script can be changed via a configuration file. You can set up several different configuration files and then tell the script which file to use. The configuration changes we made are detailed in WebSphere Studio Workload Simulator configuration.

We used different copies of the modified WebSphere Studio Workload Simulator script to perform runs that were intended to stress anywhere from one to five application servers.

All the measurements done in this study used a one-to-one relationship between the WebSphere Studio Workload Simulator script and the application server.

System setup

In this chapter we detail changes we made to our system setup for our test runs.

WebSphere environment

To emulate a customer-like environment, one WebSphere Application Server environment consisted of an IBM HTTP Web server, the WebSphere Application Server, and a DB2 UDB database server. This environment is referred to as a triplet in this document. In our tests, the number of triplets was scaled from 1, 2, 4, to 5.

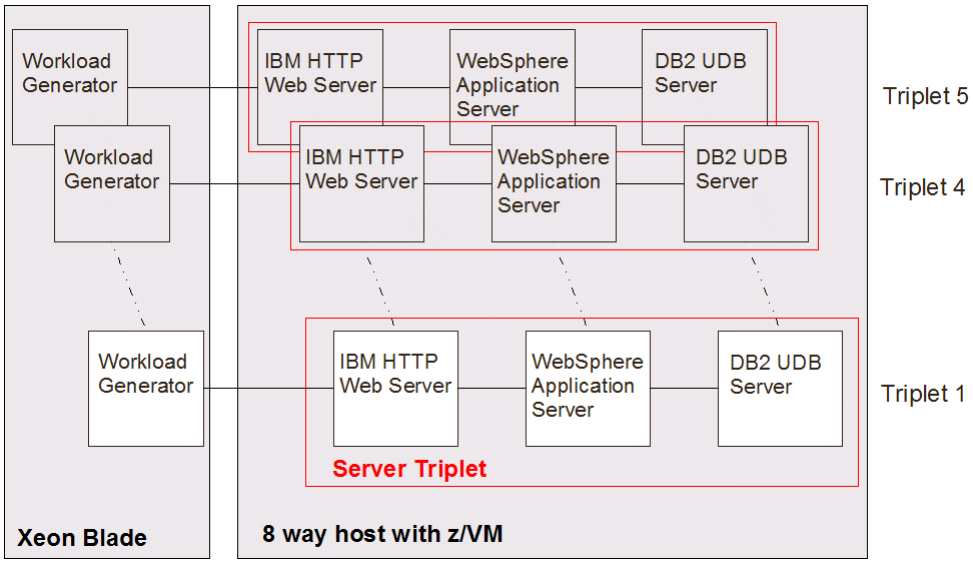

Figure 3. WebSphere environment – server triplets

System setup changes

The following system changes were made for our test runs.

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

Normally, you cannot define more than 768 MB of JVM heap on a 31-bit distribution of WebSphere Application Server. However, with SUSE, you can use the mapped base support to enable your system to have up to 1 GB of heap. Note that unpredictable results occur if you go over the 1 GB mark. To

enable this capability, you need to place the following line into the startServer.sh startup script:

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

The startServer.sh script now looks like the following (note that **bold** is added for emphasis only):

```
lnwas3:/opt/IBM/WebSphere/AppServer/profiles/default/
bin # cat startServer.sh
#!/bin/sh
echo 268435456 >/proc/self/mapped_base
WAS_USER_SCRIPT=/opt/IBM/WebSphere/AppServer/profiles
/default/bin/setupCmdLine.sh
export WAS_USER_SCRIPT
/opt/IBM/WebSphere/AppServer/bin/startServer.sh "$@"
```

Create a self-signed certificate

We used a self-signed certificate for the authentication. For creating a self-signed certificate we used the gsk7ikm. The 31-bit JRE was needed so the Java environment variable and path to the 31-bit Java could be set by using the following:

export JAVA_HOME=/opt/IBMJava2-s390-142/jre/
export PATH=/opt/IBMJava2-s390-142/jre/bin:$PATH

To make the above settings persistent over reboots, the above commands need to be added to the /etc/profile.

Configure IBM HTTP Server to use certificates

Enabling the use of gsk7ikm GUI (the 31-bit JRE is needed for this)

Set the Java environrment variable and path by doing the following:

export JAVA_HOME=/opt/IBMJava2-s390-142/jre/
export PATH=/opt/IBMJava2-s390-142/jre/bin:$PATH

To make the above settings persistent over reboots, the above commands need to be added to the /etc/profile.

Create a key database and self-signed certificate. Configure IBM HTTP server.

We used the following steps to enable HTTPS on the IBM HTTP Server.

Using the gsk7ikm utility we:

- *Created a key database*
- *Created a self-signed certificate (using lnweb1:lnweb1 as the name)*
- *Placed the certificate under the /opt/IBMIHS/security directory (we had to create this directory)*

Add the following to the IHS configuration file /opt/IBMIHS/conf/httpd.conf:

```
Listen 443
<IfDefine SSL>
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
</IfDefine>
<IfDefine SSL>
     <VirtualHost 0.0.0.0:443>
     DocumentRoot /opt/IBMIHS/htdocs/en_US
     SSLEnable
     SSLClientAuth none
     SSLServerCert lnweb1:lnweb1
     Keyfile /opt/IBMIHS/security/key.kdb
     </VirtualHost>
</IfDefine>
```

Restarted the Web server normally with /opt/IBMIHS/bin/apachectl restart.

WebSphere Studio Workload Simulator configuration

Configuration file changes

Using the configuration file, we made changes to the default WebSphere Studio Workload Simulator script that was provided with the Trade distribution. The changes we made were:

- *The number of simulated clients, which are executed by worker threads, was varied. Runs were made with 3, 4, 5, and 6 simulated clients. These clients were varied in order to reach optimal throughput and, at the same time, keep the overall processor utilization at around 90%. We found that four simulated clients achieved this.*
- *The time limit (length of our runs) was set to 20 minutes.*
- *The "element delay" (or "think time") was kept at 0.*
- *The "xml_interval" is the interval in minutes when WebSphere Studio*

Workload Simulator will take a snapshot of its output. This output can be customized. We paid attention to pages per second, transactions per second, and response time. Our xml_interval was set to five minutes.

A sample configuration file can be found in WebSphere Studio Workload Simulator sample configuration file.

Script file changes

Along with the configuration file changes we made, we also made changes to the WebSphere Studio Workload Simulator script file itself. The changes we made were:

- *Hostnames lnweb[1 thru 5] were added to each of the five sample scripts.*
- *For SSL, every occurrence of getpage and postpage were changed to getpage_ssl and postpage_ssl respectively.*

A portion of the script, showing the changes we made, can be found in WebSphere Studio Workload Simulator sample script file.

Enabling quick dispatch (QUICKDSP)

When the quick dispatch option is assigned to a virtual machine, that virtual machine is added to the dispatch list immediately, whenever it has work to do, without waiting in the eligible list. We assigned it to all guests via the z/VM user directory (see z/VM directory templates for Linux guests for details). For detailed information on quickdsp see:

http://www2.marist.edu/htbin/wlvtype?LINUX-VM.30359

Gather CPU values

The Performance Toolkit for z/VM™ was used to determine the CPU load.

The CPU utilization was normalized in a way that 100% CPU utilization means that eight CPUs are fully utilized.

Collecting the output

This chapter details the steps we took to gather the output for our test runs.

WebSphere Studio Workload Simulator

Before making any performance runs, several things were done to tune the systems. The steps we took were:

1. *Run the WebSphere tuning script*
   - *Take the application server down*
2. *Backup your WebSphere system by issuing the following command from the /opt/IBM/WebSphere/AppServer/profiles/default/bin/ directory:*

   ```
   backup.sh <backup name> -nostop
   ```

   - *FTP the trade_tune.jacl file to your WebSphere Application Server system*
3. *Start your WebSphere Application Server*
4. *Run the WebSphere tuning script by issuing the following shell script with the -f flag from the /opt/IBM/WebSphere/AppServer/profiles/default/bin # ./ directory:*

   ```
   wsadmin.sh -f /home/trade_tune.jacl server server1
   ```

Note that your path will point to wherever you FTP'd the script. The script also needs the keyword "server" and the name of your application server.

5. *Run the Trade DB2 tuning script*
   - *Backup your DB2 UDB system by issuing the following commands:*

     db2 connect to tradedb
6. *db2 get db cfg for tradedb > <some file name>*
7. *db2 terminate*
   - *Run the Trade tune database script as db2inst1*

Note:
You only need to run the WebSphere and DB2 tuning scripts once (at the beginning) during your testing. Once run, any subsequent reboots will not change the values.

The following sequence of events was used for each measurement:

1. *Start DB2*
2. *Start WebSphere*
3. *Start the IBM HTTP Server*
4. *Run the WebSphere Studio Workload Simulator reset Trade DB script*
5. *Run the WebSphere Studio Workload Simulator repopulate Trade DB script*
6. *Run runstats on the DB2 UDB system. To do this, enter the following commands while under your DB2 userid:*

   db2 connect to tradedb
7. *db2 reorgchk update statistics*
8. *db2 terminate*
9. *db2stop force*
10. *db2start*
11. *Reboot the DB2 UDB, WebSphere Application Server, and IBM HTTP Server Linux guests*
12. *Start DB2 UDB*
13. *Start WebSphere*
14. *Start IBM HTTP Server*
15. *Start the VM performance monitor*
16. *Run the WebSphere Studio Workload Simulator script*

    Have it run for 20 minutes. The first five minutes will be used as a "warmup" period.
17. *Issue the following command at the beginning and end of each run:*

    `run netstat -s`
18. *Stop the VM performance monitor after 20 minutes*
19. *Stop the measurement collections*
20. *Stop all servers, DB2, WebSphere, and IBM HTTP Server*
21. *Gather measurement data and WebSphere Studio Workload Simulator results*

Below is a sample of the output produced by WebSphere Studio Workload Simulator. The key items that were used for throughput are Page element

throughput, Transactions throughput, and average Page Element response time.

Cumulative statistics are printed every 5 minutes and at the end of the WebSphere Studio Workload Simulator execution time interval, which was set to 1200 seconds for each of our measurements.

```
===============Cumulative Statistics================
IWL0038I Run time = 00:15:02
IWL0007I Clients completed = 0/30
IWL0059I Page elements = 629404
IWL0060I Page element throughput = 697.325 /s
IWL0059I Transactions = 521099
IWL0060I Transaction throughput = 577.332 /s
IWL0059I Network I/O errors = 0
IWL0059I Web server errors = 0
IWL0059I Num of pages retrieved = 629404
IWL0060I Page throughput = 697.325 /s
IWL0060I HTTP data read = 5657.374 MB
IWL0060I HTTP data written = 240.236 MB
IWL0060I HTTP avg. page element response time = 0.041
IWL0060I HTTP avg. page element response time =
0.041
        (with all clients concurrently running)
======== S h a r e d   V a r i a b l e  R e p o r t =======
int curClient = 358
=== E n d S h a r e d V a r i a b l e R e p o r t ===
```

Results

This chapter provides our detailed test results and conclusions. Results from our test runs are shown.

Scaling triplets with and without SSL with software encryption

Test case description

The purpose of these test runs was to scale the number of triplets in use. We wanted to see how z/VM was able to handle scalability as it reached a significant overcommitment of available resources. Also, we wanted to see the cost of using software encryption with SSL compared to using no encryption. We expected that encryption would cause either more CPU cost or less throughput than without using encryption.

Overcommitment

If each guest of a triplet were to be given one virtual CP, a z/VM system with eight physical CPs will have its CPs overcommitted when three triplets are reached, that is, nine CPs. Changing the number of CPs per triplet will only lower or raise the virtual number of CPs that the Linux guest will think it has defined to it. In other words, the physical aggregate cannot be greater than eight CPs. The following table shows which scenarios were running with more virtual resources than physical resources available (overcommitment) and the grade of that overcommitment.

Table 4. Ratio of virtual to physical resources (overcommitment)

Triplets	CPU Virtual/CPU physical * 100%	Memory Virtual/Memory Physical * 100%
2	100%	-
4	200%	100%
5	250%	125%

When we specify the overcommitment in this document, we specify the part the virtual resources exceed the physical resources. For example, with four triplets we have the CPUs overcommitted by 100% and no memory overcommitment.

The environment with 4 triplets uses all memory for the guests. Because there is some space needed for the z/VM operating system itself, this scenario is actually running with memory overcommitted. In any run with five triplets, the virtual CPU and memory exceeds the physical resources significantly. Be aware that the z/VM system has a very fast paging device with the 2 GB expanded storage.

Charts

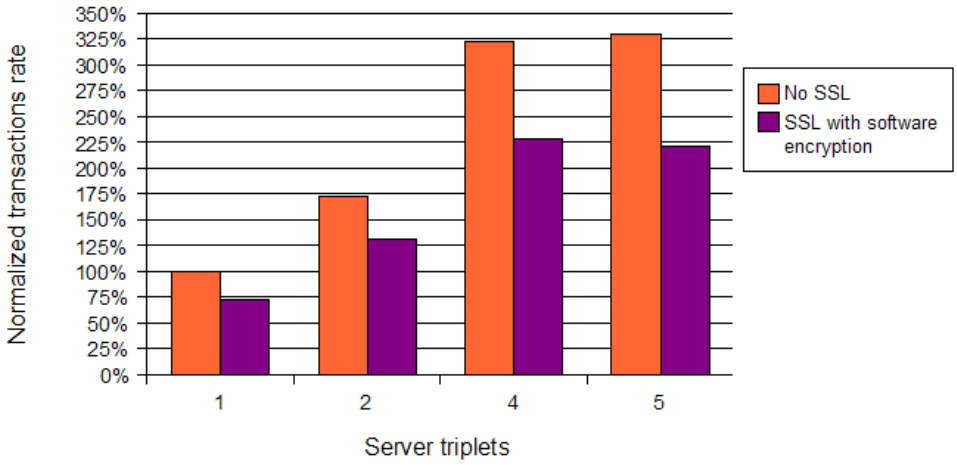

Figure 4. Transaction throughput for scaling triplets and encryption settings

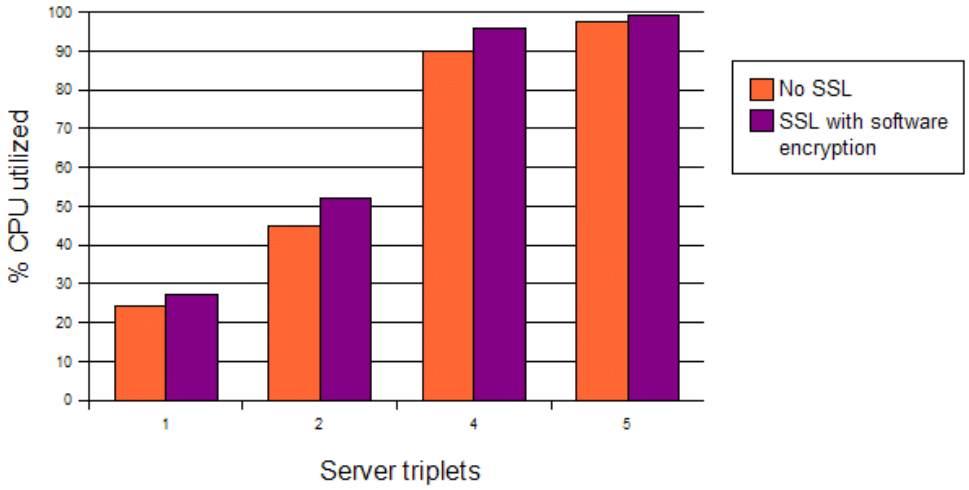

Figure 5. CPU utilization for scaling triplets and encryption settings

Observations

The throughput continues to increase from one triplet to four triplets, even through the overcommitted case with five server triplets. Comparing the runs without SSL and the runs with encryption, we see that the additional effort for the encryption costs CPU and reduces the throughput between 25% and 30% (see Figure 4 and Figure 5).

With four triplets the system is already utilized above 90%. Surprisingly, the throughput with five triplets does not degrade, which would be expected. With five triplets and encryption, the CPUs were utilized to 100%.

Table 5. Throughput scalability

Triplets	No SSL	SSL with software encryption
1	1.0	1.0
2	1.7	1.8
4	3.2	3.1
5	3.3	3.0

The workload scaled well for two triplets. Considering that four triplets already has a CPU overcommitment by 100% and there was a slight memory overcommitment (all memory was used for the guests and some was needed for the z/VM control program), the scalability was still good. The result for five triplets is very good because the system ran completely overcommitted (CPU by 150%, memory by 25%) at nearly the same throughput level as four triplets.

Conclusions

z/VM manages resources very efficiently so the throughput does not degrade in the overcommitted scenarios. When exceeding the point where the system becomes overloaded, the throughput does not degrade. Here it would have been interesting to know how many more triplets would be needed until the total throughput goes down.

Response Times

Figure 6 shows the response times for the cases with and without software encryption.

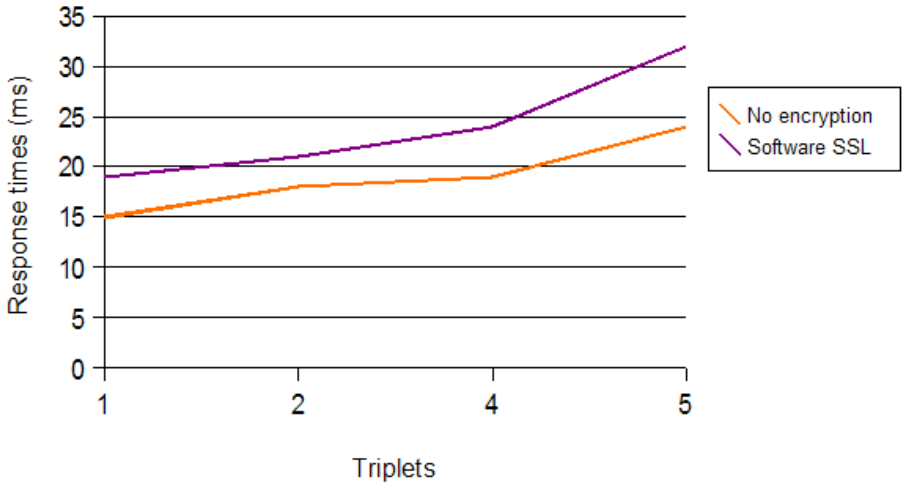

Figure 6. Response times for scaling triplets with and without encryption

Observations

The response times with encryption linearly increase with the number of triplets up to four triplets. Response times increase faster in the overcommitted case with five triplets.

Conclusions

The knee of the curve for response time is about four triplets. This is the point at which the system is saturated and more load can only be handled by using queuing. However, even in the overcommitted case, response times are very good.

Quick dispatch (QUICKDSP)

Test case description

The purpose of these tests was to see the impact of the quick dispatch option. When the quick dispatch option is assigned to a virtual machine, that virtual machine is added to the dispatch list immediately, whenever it has work to do, without waiting in the eligible list. We assigned it to all guests.

Charts

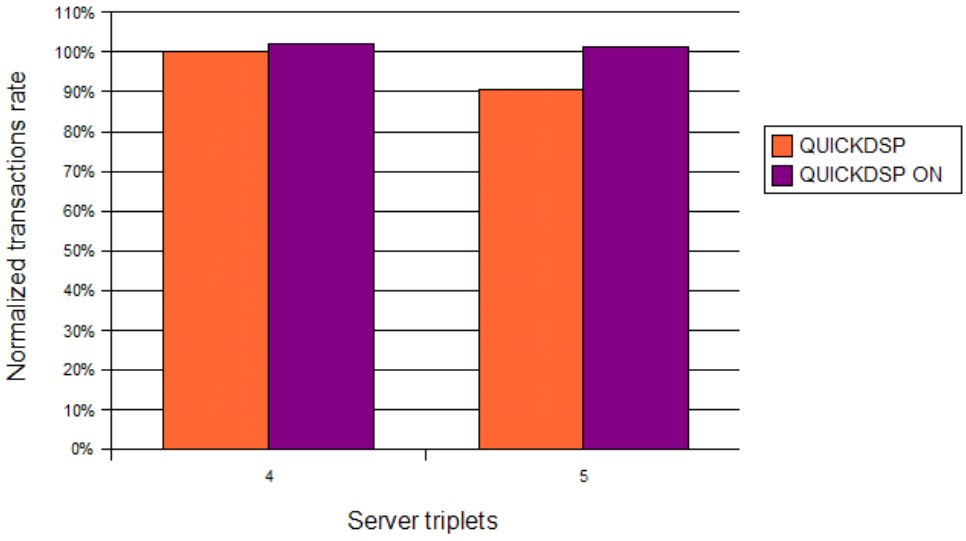

Figure 7. Impact of the z/VM quick dispatch option on transaction throughput

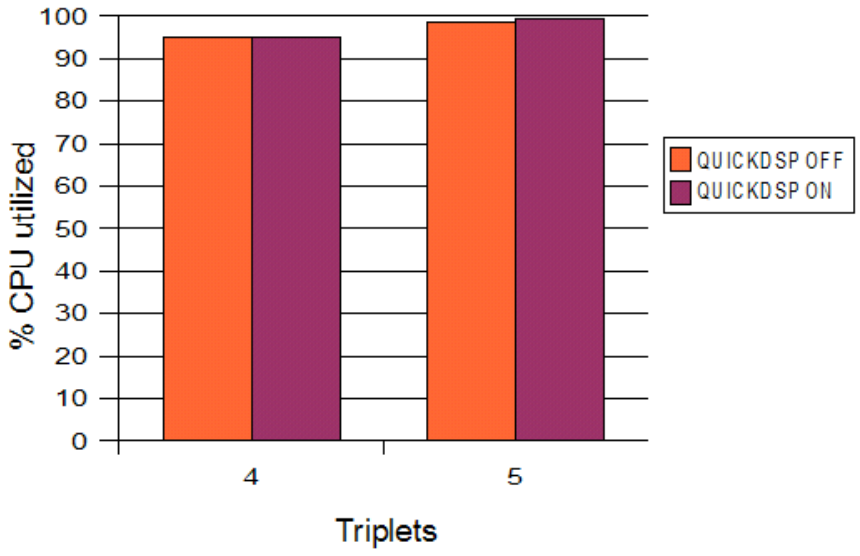

Figure 8. Impact of the z/VM quick dispatch option on CPU utilization

Observations

Looking at Figure 7 and Figure 8 we see that using the QUICKDSP option with Linux z/VM guests provided a significant benefit when the resources are overcommitted, like CPU and memory for five triplets and there is a resource constraint. Otherwise, there was no benefit to using QUICKDSP.

The CPU costs also remained nearly the same, regardless of whether there was CPU overcommitment.

Conclusions

Using QUICKDSP for the z/VM guests can bring a benefit when the resources (CPU, memory) are overcommitted and resources are constrained.

FCP Adapter Load

The data files of the DB2 UDB databases are residing on FCP disks. From the Performance Toolkit for z/VM we saw that the per adapter load on the FCP adapters using 5 triplets was 0.4 MB/sec per write and 64 KB/sec per read. This resulted in a per adapter load of 2 MB/sec per write and 320 KB/sec per read, which is very low.

z/VM guest statistics: working set size and CPU overhead

Beside the view on the virtual resources, another view on the system is what is physically allocated and what the costs are of all the virtualization. Concerning the CPUs, we saw what happens when the available physical CPUs are exhausted with five triplets. The total throughput stays nearly constant, even though there must be some overhead because of dispatching unique resources to the waiting guests. Concerning the memory, there are a lot of optimizations implemented in z/VM to keep the allocation small. This can be easily verified looking at the working set size (WSS), which represents the amount of memory really allocated from one guest (see Table 6). Additionally, it shows the ratio of total CPU to Virtual CPU spent for one guest. The total CPU consists of the Virtual CPU (where the guest is using CPU) and the CPU used from CP to manage that guest and its virtual devices.

Table 6. Working set size and CP cost per guest

Guest	WSS (MB)	CPU Total/Virtual
LNUDB1	398	1.04
LNUDB2	400	1.04
LNUDB3	390	1.04
LNUDB4	390	1.06
LNUDB5	393	1.07
LNWAS1	1434	1.07
LNWAS2	1442	1.07
LNWAS3	1443	1.07
LNWAS4	1433	1.12
LNWAS5	1445	1.12
LNWEB1	159	1.22
LNWEB2	169	1.22
LNWEB3	162	1.23
LNWEB4	160	1.23
LNWEB5	163	1.22
	Total: 9,981 MB	**Average: 1.12**

Observations

The WSS and the overhead depends heavily on the type of server. The lowest amount of memory is used by the Web server, the next lowest is the database server with more than twice the memory, and finally, the WebSphere Application Server with about 1.5 GB, which is still less than the maximum possible value of 2 GB. The ranking for the overhead is quite different. Here the database server and the WebSphere Application Server are very similar and are much lower than the Web server.

Conclusions

The reason for the small WSS size of the Web server is that it 'only' transmits the requests coming from the external network, it does no disk I/O. On the WebSphere Application Server the Java heap is set up to use 1 GB and it seems to be really used, while the database requires memory for buffer pools and disk I/O, which uses the page cache. The reason for the high CP overhead from the Web server is expected to be the access to the physical network interface, which generates interrupts, requiring z/VM to intercept. Using z/VM guest LAN has a much lower impact, shown by the overhead from the

database server and the WebSphere Application Server, even though the WebSphere Application Server has more CPUs than the others. Therefore, the usage of the z/VM guest LAN could be highly recommended, not only because the setup is so easy, but because it reduces the overhead significantly.

Actually, we did not really overcommit the memory. This shows that z/VM is very efficient in managing guest memory pages.

Appendix A. Detailed set up examples

This appendix contains detailed examples of configuration and sample scripts we used in our test runs.

Configure IBM WebSphere Application Server to accept SSL requests

If you are interfacing with WebSphere you will need to create a virtual host alias that will use port 443 for SSL connections. To do that, you must:

1. Go into the WebSphere Administrative Console (see Figure 9)

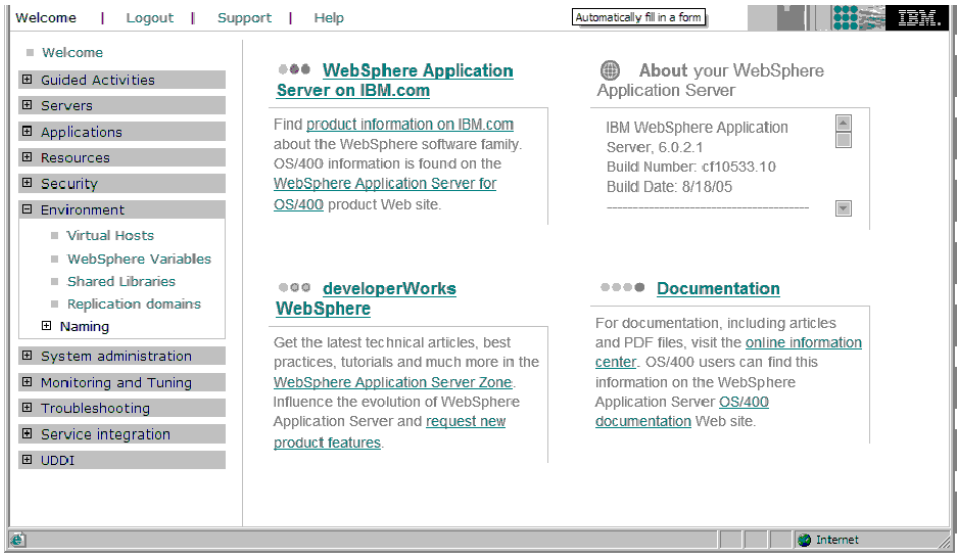

Figure 9. WebSphere Administrative Console

2. *On the left hand side of the screen, under Environment, choose Virtual Hosts (see Figure 10)*

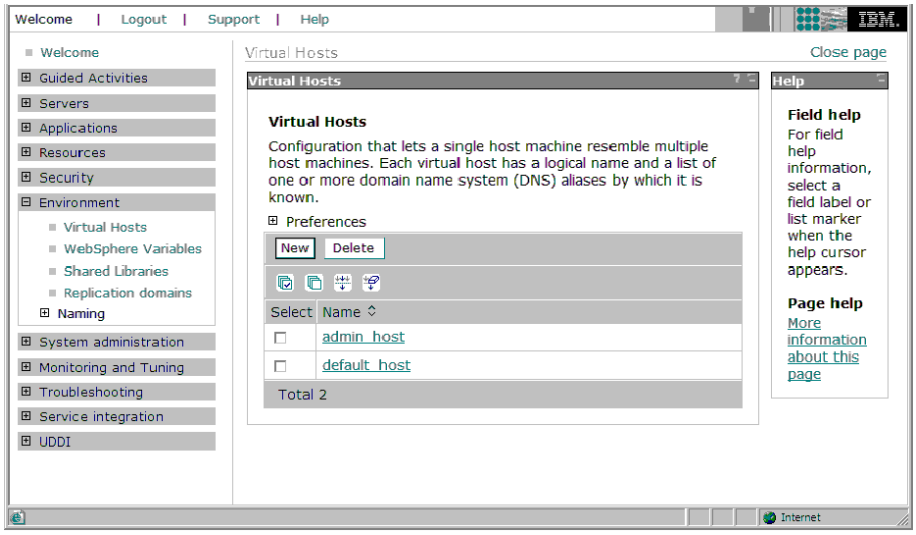

Figure 10. WebSphere Administrative Console – Virtual Hosts

3. *Choose "default host" (see Figure 11)*

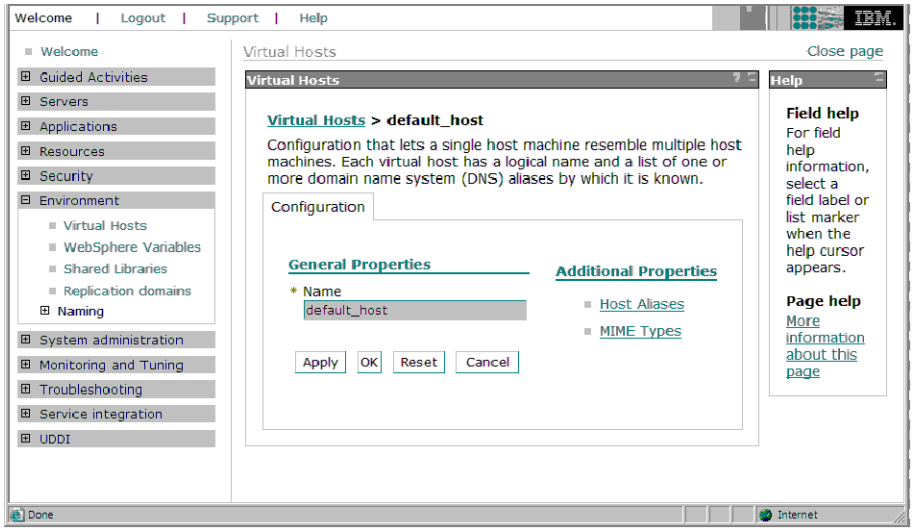

Figure 11. WebSphere Administrative Console – Default Host

4. Under Additional Properties, choose Host Aliases (see Figure 12)

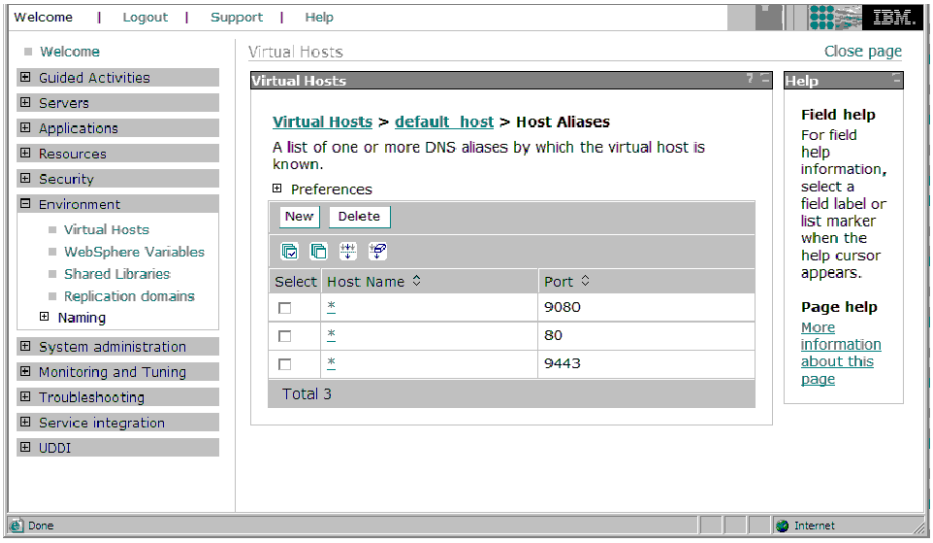

Figure 12. WebSphere Administrative Console – Host Aliases

5. Click New

6. Enter the General Properties information (see Figure 13)

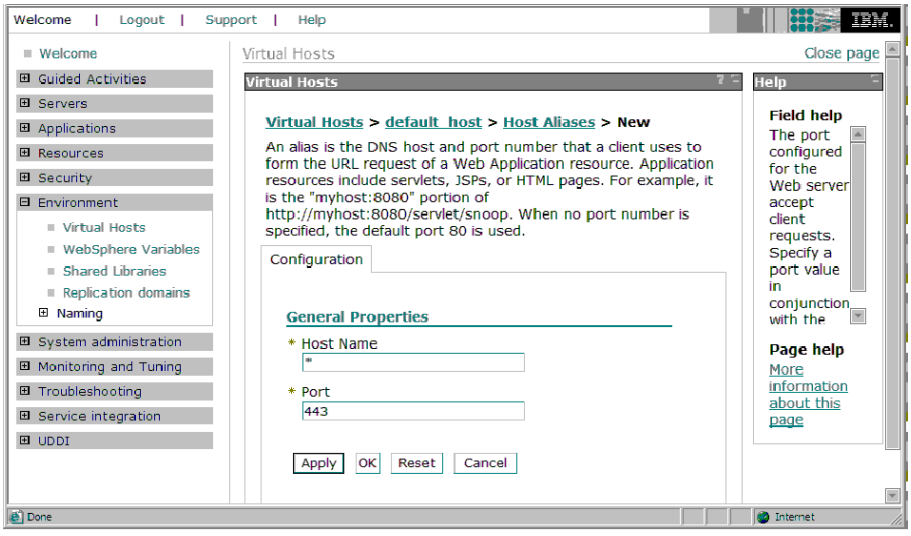

Figure 13. WebSphere Administrative Console – General Properties

7. *Click OK and save your changes.*
8. *From the left hand side of the screen, choose* ***Servers -> webservers***
9. *Generate the plug-in again*

   The plug-in will be stored in /opt/IBM/WebSphere/AppServer/profiles/default/cells/lnwas5Node01Cell/nodes/webserver1_node/servers/webserver1/plugin-cfg.xml.
10. *Copy the plug-in to your webserver in the /opt/IBM/WebSphere/Plugins/config/webserver1 directory.*
11. *Start IBM HTTP Server*

Note that you can verify your settings by going to https://<ip address>/trade.

Tips

The following are some tips to keep in mind when setting up your hardware encryption.

Be careful when setting your password. When you initialize your cryptographic token with pkcsconf -c 0 -l, the SP pin is set by you, but it is already expired. You have to change it using the pkcsconf -c 0 -P option. If the password you enter is incorrect, you might receive the following error when running gsk7ikm:

Figure 14. Error Message

If you do mess up the password, you need to do the following:

1. *Delete all the files (but not the directories) in the /etc/pkcs11 directory*
2. *Delete all the files (but not the directories) in the /etc/pkcs11/lite directory*
3. *Re-initialize the token with the pkcsconf command*

The password you enter on the sslstash command is the user pin set when you issued the pkcsconf-c 0 -P command.

Remember to create your key.sth file by checking off the "create stash file" box when prompted for a password in the gsk7ikm dialogs.

Remember that when a plug-in is regenerated, both the IBM HTTP Server and the WebSphere Application Server need to be restarted.

WebSphere Studio Workload Simulator sample script file

The following is a portion of the WebSphere Studio Workload Simulator script file that we used in our testing. Lines that we changed are in **bold** for emphasis only.

```
//
/*trade6 version 1.3*/
init_section
{
        string hostname =
getparm("hostname","lnweb1:443");
        int botClient = getparm_int("botClient",0);
        int topClient = getparm_int("topClient",499);
        shared int curClient = botClient - 1;
}
int html_percent = 10;
int gif_percent = 0;
int num_u = 500;
int num_s = 1000;
int num_stock, i;
string name, uid, add, email, holdid, stocks;
bool loop = true;
int sell_deficit = 0;
int flop;
HttpResponse r;
while (true)
{
        loop = true;
                start_transaction("login");
                startpage(4);
                thinktime(1000);
                        //uid =
URLEncode("uid:"+random(0,num_u - 1));
                        //TODO: Make this random but
better than above - must be random across all
           clients
                        int clientnum;
                        enter_critical_section();
                        curClient = curClient + 1;
                        if (curClient > topClient)
```

```
                        {
                                curClient =
botClient;
                        }

                        clientnum = curClient;
                        leave_critical_section();
                        uid = URLEncode("uid:" +
clientnum);

postpage_ssl(""+hostname+"","/trade/app",1,close,0,st
art,"",
                        "uid=" +uid+
"&passwd=xxx&action=login",
                        "Accept: image/gif, image/x-
xbitmap, image/jpeg, image/pjpeg, */*",
                        "Referer:
http://"+hostname+"/trade/app",
                        "Accept-Language: en-us",
                        "Content-Type: application/x-
www-form-urlencoded",
                        "Accept-Encoding: gzip,
deflate",
                        "User-Agent: Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.1)",
                        "Host: "+hostname+"",
                        "Connection: Keep-Alive");

                endpage;
        end_transaction("login");
        while (loop)
        {
                distribute
                {
                        weight 20:

start_transaction("home");
                                        startpage(5);

thinktime(1000);

getpage_ssl(""+hostname+"","/trade/app",1,close,0,sta
rt,"?action=home",

"Accept: image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, */*",

"Referer: http://"+hostname+"/trade/app",
```

```
"Accept-Language: en-us",

"Accept-Encoding: gzip, deflate",

"User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1)",

"Host: "+hostname+"",

"Connection: Keep-Alive");

                                        endpage;

end_transaction("home");
.
.
.
```

Note that this is just a portion of this script.

WebSphere sample tuning script

The following is the WebSphere tuning script (trade_tune.jacl) supplied with Trade that we used for our z/VM test runs.

Note:
The V symbol indicates that the text continues on the next line. These lines should be entered on one line, not broken into multiple lines.

```
#--------------------------------------------------
----------------
# Generic WebSphere Tuning Script
#--------------------------------------------------
----------------
#
# Author: Christopher Blythe
#
# This script is designed to modify some of the most
common
# WebSphere configuration parameters and tuning
knobs.
# In order to tune the config parameters, simply
change the values
# provided below. This script assumes that all server
names in a
# cluster configuration are unique.
```

```
#
# To invoke the script, type:
#   wsadmin -f tuneWAS.jacl <scope> <id>
#      scope      - 'cluster' or 'server'
#      id         - name of target object within
scope (ie. servername)
#
# Examples:
#   wsadmin -f tuneWAS.jacl server server1
#
#   wsadmin -f tuneWAS.jacl cluster TradeCluster
#
#
#-------------------------------------------------
----------------

$AdminConfig setValidationLevel NONE

set buildDate "09212004"

puts "Starting script..."
puts "Version: $buildDate"
puts "Reading config parameters..."


#-------------------------------------------------
----------------
# COMMON CONFIG PARAMETERS
# - Adjust these parameters based on the intended
target system
#-------------------------------------------------
----------------


# ORB properties (10,50,false)
set minORBPool     10
set maxORBPool     50
set noLocalCopies  true

# WebContainer Thread Pool (10,50)
set minWebPool     50
set maxWebPool     50

# HTTP KeepAlive settings (true, 100)
set keepAliveEnabled      true
set maxPersistentRequests -1

# Inactivity Timeouts for thread pools (3500)
```

```
set inactivity     3500

# EJB Cache properties (2053)
set ejbCacheSize   2053

# JVM properties
set minHeap        1024
set maxHeap        1024
set verboseGC      "false"
set genericArgs    ""

# OS Specific JVM options
set IBMJDKoptions ""
set SUNJDKoptions "-XX:MaxPermSize=64m -
XX:MaxNewSize=680m -XX:NewSize=680m V
    -XX:SurvivorRatio=16"
set HPJDKoptions  "-Xmn680m"

# SPECjAppServer2002 related generic JVM arguments
# -Dcom.ibm.ws.pm.batch=true
# -Dcom.ibm.ws.pm.deferredcreate=true
# -Dcom.ibm.CORBA.FragmentSize=0
# Trade3 related generic JVM arguments
# -
Dcom.ibm.websphere.ejbcontainer.noPrimaryKeyMutation=
true

# Transaction Service properties (120,60)
set txTimeout      120
set clientTimeout  60

# SystemOut and SystemErr log rollover type (SIZE)
set rollover       "NONE"

# TraceService settings {"*=all=disabled",20,1}
set traceSpec            "*=all=disabled"
set traceRolloverSize    100
set maxFiles             10

# Java2 Security (false for 5.1 and true for 6.0)
set j2Security     false

# PMI service
set PMIstatus      false

# Uninstall default applications
# Possibly uninstall applications -
DefaultApplication, ivtApp
```

```
set uninstallApps  true
set uninstallList  [list ivtApp DefaultApplication
Query]

# Parallel server startup
set parallelStart  false


#--------------------------------------------
# Check/Print Usage
#--------------------------------------------

proc printUsageAndExit {} {
   puts " "
   puts "Usage: wsadmin -f tuneWAS.jacl <cluster |
server> <name>"
   exit
}

#--------------------------------------------
# Misc Procedures
#--------------------------------------------

proc getName {objectid} {
   set endIndex [expr [string first "(" $objectid] -
1]

   return [string range $objectid 0 $endIndex]
}


#--------------------------------------------
# Parse command line arguments
#--------------------------------------------

puts "Parsing command line arguments..."

if {[llength $argv] < 2} {
   printUsageAndExit
} else {
   set scope [lindex $argv 0]
   puts "Scope:   ${scope}"

   if {$scope == "cluster"} {
      set clustername [lindex $argv 1]
      puts "Cluster: ${clustername}"
   } elseif {$scope == "server"} {
      set servername [lindex $argv 1]
```

```
      puts "Server:  ${servername}"
   } else {
      puts "Error: Invalid Argument ($scope)"
      printUsageAndExit
   }
}


#-------------------------------------------------
-----------------
# Base OS Specific JVM settings
#-------------------------------------------------
-----------------

if {[string first "Windows" $env(os.name)] >= 0 } {
   set genericArgs [concat $genericArgs
$IBMJDKoptions]
} elseif {$env(os.name) == "AIX"} {
   set genericArgs [concat $genericArgs
$IBMJDKoptions]
} elseif {$env(os.name) == "Linux"} {
   set genericArgs [concat $genericArgs
$IBMJDKoptions]
} elseif {$env(os.name) == "SunOS"} {
   set genericArgs [concat $genericArgs
$SUNJDKoptions]
} elseif {$env(os.name) == "HP-UX"} {
   set genericArgs [concat $genericArgs
$HPJDKoptions]
}


#-----------------------------------------
# Obtain server list
#-----------------------------------------

puts ""
puts "Obtaining server list..."

if {$scope == "cluster"} {
   set cluster [$AdminConfig getid
"/ServerCluster:${clustername}/"]
   set temp [$AdminConfig showAttribute $cluster
members]
   set memberList [split [string trim $temp "{ }"] "
"]
   foreach member $memberList {
      set memberName [getName $member]
```

```
      lappend serverList [$AdminConfig getid
"/Server:${memberName}/"]
   }
} else {
   set server [$AdminConfig getid
"/Server:${servername}/"]
   lappend serverList $server
}


#--------------------------------------------
# Print config properties
#--------------------------------------------

puts ""
puts "WebSphere configuration"
puts "-----------------------"
puts ""
puts "   Enforce Java2 Security:      ${j2Security} "
puts ""

puts "Servers:"
foreach server $serverList {
   puts "   [getName $server]"
}
puts ""
puts " EJB/ORB -------------------------------------
--"
puts "   Min ORB Pool Size:           ${minORBPool} "
puts "   Max ORB Pool Size:           ${maxORBPool} "
puts "   EJB Cache Size:              ${ejbCacheSize}
"
puts "   NoLocalCopies:
${noLocalCopies} "
puts " Web -----------------------------------------
--"
puts "   Min WebContainer Pool Size:  ${minWebPool} "
puts "   Max WebContainer Pool Size:  ${maxWebPool} "
puts " JVM -----------------------------------------
--"
puts "   Min JVM Heap Size:           ${minHeap} "
puts "   Max JVM Heap Size:           ${maxHeap} "
puts "   Verbose GC:                  ${verboseGC}"
puts "   Generic JVM Arguments:  "
puts "      ${genericArgs}"
puts " Transaction ---------------------------------
--"
puts "   Total Transaction Timeout:   ${txTimeout} "
```

```
puts "   Client Inactivity Timeout:
${clientTimeout} "
puts " Logging ------------------------------------
--"
puts "   System Log Rollover Type:    ${rollover} "
puts "   Trace Specification:         ${traceSpec} "
puts "   Rollover Size:
${traceRolloverSize} "
puts "   Max Backup Files:            ${maxFiles} "
puts " Misc ---------------------------------------
--"
puts "   Enable PMI Service:          ${PMIstatus} "
puts "   Pool Activity Timeouts:      ${inactivity} "
puts "   Parallel Startup:
${parallelStart} "
puts ""
puts "   Uninstall default apps:
${uninstallApps} "
puts ""


#-------------------------------------------
# Modify cell parameters
#-------------------------------------------

# Accessing cell based security config
puts "Accessing security configuration..."
set sec [$AdminConfig list Security]
set attrs [subst {{enforceJava2Security
$j2Security}}]
puts "Updating security..."
$AdminConfig modify $sec $attrs


#-------------------------------------------
# Modify server parameters
#-------------------------------------------

foreach server $serverList {
   set servername [getName $server]
   puts ""
   puts "Server: $servername"
   puts ""


   # Accessing server startup config
   puts "Accessing server startup configuration..."
   puts "Parallel Startup (old/new):  [$AdminConfig
```

```
showAttribute $server V
        parallelStartEnabled]/$parallelStart"
   set attrs [subst {{parallelStartEnabled
$parallelStart}}]
   puts "Updating server startup..."
   puts ""
      $AdminConfig modify $server $attrs


      # Accessing orb config
      puts "Accessing ORB configuration..."
      set orb [$AdminConfig list ObjectRequestBroker
   $server]
      set attrs [subst {{noLocalCopies
   $noLocalCopies}}]
      puts "ORB noLocalCopies (old/new):
   [$AdminConfig showAttribute $orb V
           noLocalCopies]/$noLocalCopies"
      $AdminConfig modify $orb $attrs
      set orbPool [$AdminConfig showAttribute $orb
   threadPool]
      puts "ThreadPool MaxSize (old/new):
   [$AdminConfig showAttribute $orbPool V
           maximumSize]/$maxORBPool"
      puts "ThreadPool MinSize (old/new):
   [$AdminConfig showAttribute $orbPool V
           minimumSize]/$minORBPool"
      puts "ThreadPool Inactivity Timeout (old/new):
   [$AdminConfig showAttribute V
           $orbPool inactivityTimeout]/$inactivity"
      set attrs [subst {{maximumSize $maxORBPool}
   {minimumSize $minORBPool}V
           {inactivityTimeout $inactivity}}]
      puts "Updating ORB..."
      puts " "
      $AdminConfig modify $orbPool $attrs


      # Accessing web container thread pool config
      puts "Accessing web container thread pool
   configuration..."
      set tpList [$AdminConfig list ThreadPool
   $server]

      set oI [lsearch -glob $tpList "*WebContainer*"]
      set webPool [lindex $tpList $oI]
      puts "ThreadPool MaxSize (old/new):
   [$AdminConfig showAttribute $webPool V
```

```
        maximumSize]/$maxWebPool"
   puts "ThreadPool MinSize (old/new):
[$AdminConfig showAttribute $webPool V
        minimumSize]/$minWebPool"
   puts "ThreadPool Inactivity Timeout (old/new):
[$AdminConfig showAttribute V
        $webPool inactivityTimeout]/$inactivity"
   set attrs [subst {{maximumSize $maxWebPool}
{minimumSize $minWebPool} V
        {inactivityTimeout $inactivity}}]
   puts "Updating web container thread pool..."
   puts " "
   $AdminConfig modify $webPool $attrs


   # Accessing HTTP keepalive config
   puts "Accessing HTTP KeepAlive
configuration..."
   set HTTPInbound [$AdminConfig list
HTTPInboundChannel $server]

   set oI [lsearch -glob $HTTPInbound "*HTTP_2*"]
   set http2 [lindex $HTTPInbound $oI]
   puts "KeepAlive Enabled (old/new):
[$AdminConfig showAttribute $http2 V
        keepAlive]/$keepAliveEnabled"
   puts "Max Persistent Requests (old/new):
[$AdminConfig showAttribute $http2 V

maximumPersistentRequests]/$maxPersistentRequests"
   set attrs [subst {{keepAlive $keepAliveEnabled}
{maximumPersistentRequests V
        $maxPersistentRequests}}]
   puts "Updating HTTP KeepAlives..."
   puts " "
   $AdminConfig modify $http2 $attrs


   # Accessing EJB cache
   puts "Accessing EJB cache..."
   set ejbCache [$AdminConfig list EJBCache
$server]
   puts "Cache Size (old/new): [$AdminConfig
showAttribute V
        $ejbCache cacheSize]/$ejbCacheSize"
   set attrs [subst {{cacheSize $ejbCacheSize}}]
   puts "Updating EJB cache..."
   puts " "
```

```
      $AdminConfig modify $ejbCache $attrs


      # Accessing Transaction Service
      puts "Accessing Transaction Service..."
      set txService [$AdminConfig list
   TransactionService $server]
      puts "Client Inactivity Timeout (old/new):
   [$AdminConfig showAttribute V
            $txService
   clientInactivityTimeout]/$clientTimeout"
      puts "Total Transaction Lifetime Timeout
   (old/new):[$AdminConfig showAttribute V
            $txService
   totalTranLifetimeTimeout]/$txTimeout"
   set attrs [subst {{clientInactivityTimeout
$clientTimeout} V
        {totalTranLifetimeTimeout $txTimeout}}]
   puts "Updating Transaction Service..."
   puts " "
   $AdminConfig modify $txService $attrs


   # Accessing JVM config
   puts "Accessing JVM configuration..."
   set jvm [$AdminConfig list JavaVirtualMachine
$server]
   puts "Initial Heap Size (old/new):  [$AdminConfig
showAttribute $jvm V
        initialHeapSize]/$minHeap"
   puts "Maximum Heap Size (old/new):  [$AdminConfig
showAttribute V
        $jvm maximumHeapSize]/$maxHeap"
   puts "VerboseGC Enabled (old/new):  [$AdminConfig
showAttribute $jvm V
        verboseModeGarbageCollection]/$verboseGC"
   puts "Generic Arguments (old/new):  [$AdminConfig
showAttribute V
        $jvm genericJvmArguments]/$genericArgs"
   set attrs [subst {{initialHeapSize $minHeap}
{maximumHeapSize $maxHeap} V
        {verboseModeGarbageCollection $verboseGC}
{genericJvmArguments V
        "$genericArgs"}}]
   puts "Updating JVM..."
   puts " "
   $AdminConfig modify $jvm $attrs
```

```
   # Accessing System log file config
   puts "Accessing System log file configuration..."
   set logList [$AdminConfig list StreamRedirect
$server]

   foreach log $logList {
       puts "[$AdminConfig showAttribute $log
fileName] Rollover Type (old/new):  V
            [$AdminConfig showAttribute $log
rolloverType]/${rollover}"
       set attrs [subst {{rolloverType $rollover}}]
       puts "Updating logs..."
       puts " "
       $AdminConfig modify $log $attrs
   }


   # Accessing Trace Service config
   puts "Accessing Trace Service configuration..."
   set traceService [$AdminConfig list TraceService
$server]
   set traceLog [$AdminConfig showAttribute
$traceService traceLog]
   puts "Trace Spec (old/new):          [$AdminConfig
showAttribute $traceService V
        startupTraceSpecification]/$traceSpec"
   puts "Rollover File Size (old/new):  [$AdminConfig
showAttribute $traceLog V
        rolloverSize]/$traceRolloverSize"
   puts "Max Backup Files (old/new):    [$AdminConfig
showAttribute $traceLog V
        maxNumberOfBackupFiles]/$maxFiles"
   set attrs  [subst {{startupTraceSpecification
$traceSpec}}]
   set attrs2 [subst { {rolloverSize
$traceRolloverSize} {maxNumberOfBackupFiles V
        $maxFiles}}]
   puts "Updating Trace Service..."
   puts " "
   $AdminConfig modify $traceService $attrs
   $AdminConfig modify $traceLog $attrs2


   # Accessing PMI service config
   puts "Accessing PMI service configuration..."
   set pmi [$AdminConfig list PMIService $server]
   puts "Enable (old/new):  [$AdminConfig
```

```
showAttribute $pmi enable]/$PMIstatus"
   set attrs [subst {{enable $PMIstatus}}]
   puts "Updating PMI..."
   puts " "
   $AdminConfig modify $pmi $attrs


   # Uninstalling default applications
   # Possibly uninstall applications -
DefaultApplication, ivtApp, UDDIRegistry, V
       ManagementEJB
   if {$uninstallApps} {
      puts "Uninstalling default applications..."
      set appList [$AdminApp list]
      foreach app $appList {
         set oI [lsearch -glob $uninstallList $app]
         if {$oI > -1} {
            puts "Removing application $app..."
            $AdminApp uninstall $app
         }
      }
   }
}


puts ""
puts "Script completed..."
puts "Saving config..."

$AdminConfig save
```

DB2 UDB sample tuning script

The following is the DB2 UDB tuning script supplied with Trade that we used for our z/VM test runs.

```
db2 -v "connect to tradedb"
#db2 -v "update db cfg for tradedb using DBHEAP 6992"
db2 -v "update db cfg for tradedb using DBHEAP 25000"
db2 -v "update db cfg for tradedb using CATALOGCACHE_SZ
282"
#db2 -v "update db cfg for tradedb using LOGBUFSZ 4096"
db2 -v "update db cfg for tradedb using LOGBUFSZ 8192"
db2 -v "update db cfg for tradedb using BUFFPAGE 366190"
db2 -v "update db cfg for tradedb using LOCKLIST 1000"
db2 -v "update db cfg for tradedb using SORTHEAP 642"
db2 -v "update db cfg for tradedb using STMTHEAP 2048"
db2 -v "update db cfg for tradedb using PCKCACHESZ 7500"
```

```
db2 -v "update db cfg for tradedb using MAXLOCKS 75"
db2 -v "update db cfg for tradedb using MAXAPPLS 500"
db2 -v "update db cfg for tradedb using LOGFILSIZ 5000"
db2 -v "update db cfg for tradedb using LOGPRIMARY 6"
db2 -v "update db cfg for tradedb using LOGSECOND 6"
#db2 -v "update db cfg for tradedb using newlogpath e:"
db2 -v "update db cfg for tradedb using SOFTMAX 70"
db2 -v "update dbm cfg using MAXAGENTS 500"
db2 -v "update dbm cfg using NUM_POOLAGENTS 250"
db2 -v "update dbm cfg using MAX_QUERYDEGREE 1"
db2 -v "update dbm cfg using FCM_NUM_BUFFERS 2048"
db2 -v "update dbm cfg using FCM_NUM_RQB 6500"
db2 -v "update dbm cfg using DFT_MON_LOCK OFF"
db2 -v "update dbm cfg using DFT_MON_BUFPOOL ON"
db2 -v "update dbm cfg using DFT_MON_STMT OFF"
db2 -v "update dbm cfg using DFT_MON_TABLE OFF"
db2 -v "update dbm cfg using DFT_MON_UOW OFF"
db2 -v "alter bufferpool ibmdefaultbp size 1000"
db2 -v "reorgchk update statistics on table all"
db2 -v "connect reset"
db2 -v "terminate"
db2 -v "connect to tradedb"
db2 -v "reorgchk update statistics"
db2 -v "terminate"
```

WebSphere Studio Workload Simulator sample configuration file

The following is one of the configuration files we used in our testing. This configuration file was used for testing of four clients. All other settings were the same in our other configuration files.

```
####
# IWL Option Configuration File for 4clients.conf
####

clients: 4
element_delay: 0
startup_delay: 0
repeat: 10000
xml_interval: 5
time_limit: 1200
max_clients: 30
```

z/VM directory templates for Linux guests

The following is the template we used to create our Linux directories lnweb1, lnwas1, and lnudb1 on our z/VM system.

```
USER LNWEB1  999999   1G 1G G
 INCLUDE CMSUSER
OPTION QUICKDSP
*                      Define internal comm. paths
 IUCV ANY
 IUCV ALLOW
*                      IPL LINUX automatically
 IPL CMS PARM AUTOCR
*                      Define CPs for user
 MACHINE ESA 8
 CPU 00 BASE
*CPU 01
*CPU 02
*CPU 03
 CRYPTO APVIRTUAL
*                      LINUX IPL DASD
 DEDICATE 8D00 8D00
 DEDICATE 8D01 8D01
 DEDICATE 8D02 8D02
*                      Gigabit Devices - Admin LAN
 DEDICATE 1964 1964
 DEDICATE 1965 1965
 DEDICATE 1966 1966
 DEDICATE 1967 1967
*                      Gigabit Devices - Performance
```

```
LAN
 DEDICATE 1104 1104
 DEDICATE 1105 1105
 DEDICATE 1106 1106
 DEDICATE 1107 1107
*                       Guest LAN setup
 SPECIAL 1000 HIPER 3 SYSTEM VMTEST1
*                       "A" disk
 MDISK 191  3390  541 50 V60M01 MR  ALL WRITE MULTI
*-------------------------------------------------
------
USER LNWAS1  999999   2G 2G G
 INCLUDE CMSUSER
OPTION QUICKDSP
*                       Define internal comm. paths
 IUCV ANY
 IUCV ALLOW
*                       IPL LINUX automatically
 IPL CMS PARM AUTOCR
*                       Define CPs for user
 MACHINE ESA 8
 CPU 00 BASE
*CPU 01
*CPU 02
*CPU 03
 CRYPTO APVIRTUAL
*                       LINUX IPL DASD
 DEDICATE 8D00 8D04
 DEDICATE 8D01 8D05
 DEDICATE 8D02 8D06
*                       Gigabit Devices- Admin LAN
 DEDICATE 1964 1968
 DEDICATE 1965 1969
 DEDICATE 1966 196A
 DEDICATE 1967 196B
*                       Gigabit Devices - Performance
LAN
 DEDICATE 1104 1108
 DEDICATE 1105 1109
 DEDICATE 1106 110A
 DEDICATE 1107 110B
*                       Guest LAN setup
 SPECIAL 1000 HIPER 3 SYSTEM VMTEST1
*                       "A" disk
 MDISK 191  3390 591 50 V60M01 MR  ALL WRITE MULTI
*-------------------------------------------------
------
USER LNUDB1  999999   2G 2G G
```

```
 INCLUDE CMSUSER
OPTION QUICKDSP
*                       Define internal comm. paths
 IUCV ANY
 IUCV ALLOW
*                       IPL LINUX automatically
 IPL CMS PARM AUTOCR
*                       Define CPs for user
 MACHINE ESA 8
 CPU 00 BASE
*CPU 01
*CPU 02
*CPU 03
 CRYPTO APVIRTUAL
*                       LINUX IPL DASD
 DEDICATE 8D00 8D08
 DEDICATE 8D01 8D09
 DEDICATE 8D02 8D0A
*                       SCSI device
 DEDICATE 0800 0800
*                       Gigabit Devices- Admin LAN
 DEDICATE 1964 196C
 DEDICATE 1965 196D
 DEDICATE 1966 196E
 DEDICATE 1967 196F
*                       Gigabit Devices - Performance
LAN
 DEDICATE 1104 110C
 DEDICATE 1105 110D
 DEDICATE 1106 110E
 DEDICATE 1107 110F
*                       Gigabit Devices
 SPECIAL 1000 HIPER 3 SYSTEM VMTEST1
*                       "A" disk
 MDISK 191  3390 641 50 V60M01 MR  ALL WRITE MULTI
```

Appendix B. Other Sources of Information

This section provides sources of additional information for the products, software, and hardware discussed in this paper:

- *For information on WebSphere Application Server see:*
  http://www.ibm.com/software/info1/websphere/index.jsp?tab=products/apptransaction
- *For information on Linux on System z see:*
  www.ibm.com/servers/eserver/zseries/os/linux/
- *For information on z/VM see:*
  www.vm.ibm.com
- *For detailed information on quickdsp see:*
  http://www2.marist.edu/htbin/wlvtype?LINUX-VM.30359
- *For performance information for Trade on WebSphere Application Server see:*
  http://www.ibm.com/software/webservers/appserv/was/performance.html
- *For information on IBM open source projects see:*
  www.ibm.com/developerworks/opensource/index.html

IBM Corporation
New Orchard Rd.
Armonk, NY 10504
U.S.A.

Produced in the United States of America
09/07

ZSW03020-USEN-00