z/VM and Xen Virtualization Performance

Table of Contents

Objectives

There are many virtualization techniques in the market for various platforms. For these tests, we wanted to compare the z/VM® virtualization product to the Xen virtualization product. These virtualization products provide the feature to run an operating system, for example a Linux®, on virtual hardware. This operating system instance is called a guest. With that virtualization feature it is possible to run many guests in parallel on the same physical hardware, completely isolated from each other.

Our objective was to evaluate:

1. *The effects of overcommitting resources (processors, memory) on the total throughput and processor load when the workload is scaled.*
2. *A comparison of performance numbers (based on the Trade 6 workload) between using Secure Sockets Layer (SSL) and hardware encryption in our configuration.*

To get comparable results we measured the scalability of the WebSphere® Application Server environment using the Trade benchmark running under the virtualization products z/VM and Xen. One environment consisted of an Apache HTTP Server, a WebSphere Application Server, and an IBM DB2® 9 for LUW, all installed on separate Linux guests. This setup is called a triplet.[1]

This information will help to better understand and differentiate the value and strengths of the z/VM solution on IBM System z®.

Executive summary

Businesses try to extract maximum business value from their IT investment, that's why server consolidation via virtualization has become a major market trend and has never made more sense.

We measured the effect of "resource overcommitment" – the possibility that all guests together have more virtual resources defined than the available physical resources – for the processor and the memory. This feature increases the utilization of the hardware and simplifies server infrastructure.

Our tests show that z/VM handles resource overcommitment well.

z/VM scales until the system is fully utilized.

- *Our guests under z/VM remained stable and throughput degradation was kept to a minimum, even with a CPU overcommitment ratio of 3:1, and a memory overcommitment ratio of 1.4:1.*
- *In the memory overcommitment test with z/VM, we could increase the memory overcommitment up to a ratio of 3.2:1 without any throughput degradation.*
  Note: z/VM is designed to provide an efficient memory management:
  - *z/VM only backs memory of guests that have actually been used. That is, if a page of memory for a guest machine is unused, z/VM does not keep a copy of it anywhere.*
  - *z/VM has the ability to move pages that are not currently in use out to the paging space and then page them back into real memory when needed. This results in a high effective overcommitment for these measurements*

The Xen system reaches lower throughput and stops scaling once the first CPU overcommitment step (1.5:1) was reached.

- *Increasing the level of CPU overcommitment further only increases the CPU load without increasing throughput.*
- *The version of Xen used version contained a feature for on demand guest memory paging (ballooning) but it did not work and therefore guest memory allocation was handled statically; whatever is defined for the guest is allocated, regardless of whether it is used or not. This makes memory overcommitment impossible. This is a significant limitation for the administrators when defining additional guests.*

Our results for the encryption test shows that encryption in general requires significantly more processor resources.

- *On z/VM leveraging the cryptographic hardware features available on System z, the encrypted workload resulted in an acceptable 18% throughput degradation compared to the same non-encrypted workload, with about the same processor utilization.*
- *On the Xen system with the faster processors, the throughput was degraded by 27%.*

- *In fact, the cryptographic hardware features available on System z provided more benefit than the more than two times faster processors in the Xen environment.*
- *Additionally the response times were two times longer as compared to the z/VM system.*

Based on our tests with different versions of the software components we highly recommend the upgrade to z/VM 5.3, WebSphere Application Server 6.1, and DB2 9.1 or the latest available version. In our test the performance could be doubled over prior versions if the listed software versions are used.

Summary for the z/VM and Xen virtualization performance tests

We measured the workload behavior for both virtualization environments while we increased the overcommitment for the processor and memory by scaling the number of guests. We compiled a summary of our results and recommendations.

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 for the virtualization tests.

z/VM 5.3 is an established product, which behaved as expected and we had no installation or configuration issues. Our experience was quite different with Xen. The version 3.1, used for this test, was ahead of the distribution and came from the SUSE Web site. We decided to take that, because it contains support for the Intel® virtualization technology. Several adjustments were needed to get it working.

- *We needed to specify the type of the console.*
- *It is recommended to specify Xen profiles with fully qualified pathnames.*
- *After booting the guests, we had to toggle off and on TX checksumming on the network interfaces to physical hardware, which resulted in a significant throughput improvement (> 10x).*

z/VM and Xen test results summary

Resource overcommitment is an important feature of server virtualization, describing the capability to define more virtual resources for all guests than physically resources are available. Typically overcommitment is done for processors, memory, network cards, etc. A good overcommitment capability is important and improves the system utilization rate and simplifies the management of the guests.

Results for overcommitment of processor resource

Both the z/VM and Xen environments scaled well, while from the throughput perspective z/VM scales better than Xen.

The Xen test cases started with lower throughput, but higher CPU utilization. The z/VM throughput continued to scale past the point at which Xen's throughput rate flattens out. This occurred after nine guests. At this scaling step, both environments are overcommitted on processors, but there was more than the capacity of one processor unused.

z/VM	Xen
Maximum throughput reached with 12 guests (4 triplets), it is 66% higher than at Xen.	Maximum throughput reached with 9 guests (3 triplets)
No to little throughput degradation, despite running in a processor overcommitted at a ratio of 3:1 at 18 guests (6 triplets)	Throughput flattens at the point after the processor overcommitment was reached, despite the processor was not being fully utilized.

We could scale the Xen system to 9 guests (three triplets) before the throughput degraded. That was the point when the system reached an overcommitment in processor, despite the processors have not being fully utilized. Scaling further on the Xen system increases only the processor load but not the throughput. In fact, physical processor resources could not exceed the low 90 percentage.

The throughput on z/VM does not start to degrade until we reach 15 guests (five triplets), and the system is by then fully utilized.

Scaling on z/VM is limited by the total processor utilization. We were able to scale the processor overcommitment on z/VM up to a ratio of 3:1 at 18 guests (six triplets) with only a small degradation in throughput. Finally, we saw 10% degradation at the processor overcommitment ratio of 3:1, that was due to the effort of managing the virtual processor and memory in an overcommitted scenario and no free processor cycles

Results for overcommitment of memory resource

When scaling memory overcommitment we found that z/VM managed its resources very efficiently, for example, the amount of memory allocated in the not-overcommitted scenario is already much below the memory size defined for the guest. While Xen, at the current level, was not able to handled memory overcommitment, because the feature for on demand guest memory paging (ballooning) did not work.

z/VM	Xen
Very effective memory management, up to a memory overcommitment of 3.2:1	No memory overcommitment possible up to level 3.1
	Unable to start more guests when memory limit is reached

z/VM managed the memory overcommitment up to a level of about 3.2:1 without degradation in performance. We found the first degradation at a memory overcommitment ratio of 3.8:1. Be aware that storage which does not fit into main memory has to be moved to the expanded memory or paging devices. This is critical when running with memory overcommitted. Paging devices with high I/O bandwidths are recommended. However, there is always a point in memory overcommitment where paging starts to degrade performance.

Xen, at the current level, was not able to overcommit memory. The static handling of memory in Xen 3.1 limited the amount of memory that could be allocated to the guests. The memory size of all guests must not exceed the physical memory minus the amount of memory for the Xen host (dom0), even if one or several guests do not use all their memory. Since scaling the guests also means increasing the memory footprint, a memory limit was reached and we were unable to start additional guests. This seems to be a minor limitation

because the memory on x86 architectures is much cheaper than on System z, but it limits the total number of guests. It is not possible to dynamically add guests even if it were known that several guests were not utilizing all of their memory.

Results for response time

Response time was also measured in both environments for our workload. When scaling the guests, the response times do increase in both environments. We see a slight increase for z/VM, but not as much as with Xen. This is another proof point that z/VM manages overcommitment of resources more efficiently than Xen.

Under z/VM, the response times is always lower compared to Xen. It starts flat in the beginning and increases slightly as the level of CPU overcommitment is increased. On Xen, response times begin to increase once we increase the number of guests/triplets even if there is no resource overcommitment at the beginning. Finally the response times on z/VM are nearly two times faster than on Xen.

Results of usage of encryption

Since encryption of data sent over the network is needed to protect a company's data, we studied the impact of using SSL encryption and how the System z cryptographic hardware devices can support this.

We identified the workload level which provided the highest throughput using the cryptographic hardware features and then did a comparison with no encryption and software encryption.

We did this for both virtualization environments where the Xen environment provided no hardware encryption support, but the Xen environment is based on processors which are twice as fast as the used System z9 processors. The question was what provided the higher benefit.

z/VM	Xen
Throughput degradation at 18%	Throughput degradation at 27%, also with the use of twice the fast processors
Hardware-accelerated encryption on System z greatly reduces impact of data encryption	Hardware-accelerated encryption not available in the Xen environment

Our results for the hardware-accelerated encryption and in software encryption show that encryption in general requires significant more processor resources.

On z/VM, the throughput degrades to less than half of what is seen with a non-encrypted workload. In a comparison between software encryption and hardware-accelerated encryption, the throughput is greatly enhanced in the hardware-accelerated case; the degradation is reduced to an acceptable 18%. The cryptographic hardware support on System z is a feature that greatly reduces the impact of data encryption.

On Xen, with an almost 50% lower throughput than on z/VM in the unencrypted case, we saw a 27% degradation in throughput when we ran with software encryption. Hardware-accelerated encryption was not available in the Xen environment. However, looking at the impact of encryption, hardware acceleration is highly recommended. But this is not only a question of additional cost, it also requires virtualization support from Xen and support from the corresponding crypto library (in this case OpenSSL).

The 80% higher throughput on z/VM versus the throughput on Xen, when using encryption, shows that offloading the cryptographic functions to hardware is a great advantage, which provides more benefit than the more than two times faster CPUs on Intel. Also the response times on z/VM are less than half of Xen.

Processor utilization on z/VM is 100%, showing that on System z it is possible to fully utilize the processors, while the Xen system does not exceed a utilization of about 92%.

Apache versus IBM HTTP server comparison

For our tests we used the Apache Web Server (see figure 2) because it supports all the cryptographic hardware features available on System z. We wanted to show if that has an impact on the non-encrypted workloads. On System z, we would recommend using Apache as the Web server, because it currently supports more crypto hardware accelerators than the IBM HTTP Web Server. We also verified that both performed similar.

Results for usage of latest software versions

We experienced another interesting performance improvement in regard to different software version/releases of the required software components in our tests. In a prior comparable project a software setup was used based on z/VM 5.2, WebSphere Application Server 6.0.1 and DB2 8.2. Comparing these prior results with the results in our current environment on z/VM 5.3, WebSphere Application Server 6.1, and DB2 9.1 showed a significant performance improvement, around a factor 2 for all scaling steps.

Bottom line, we highly recommend to migrate to/use the latest software levels for your installations.

Hardware and software configuration for the z/VM and Xen performance tests

To perform our z/VM and Xen tests, we created a customer-like environment for both our z/VM and Xen environments.

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*
- *A description of the workload used*

z/VM tests hardware and software

To perform our z/VM performance tests, we created a customer-like environment. We configured the hardware, software, and storage server.

Server hardware

z/VM Host

One LPAR on an 18-way IBM System z9® Enterprise Class (z9® EC), 1.7 GHz, model 2094-S18, equipped with:

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

Storage server setup

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

For the operating system and applications on 18 Linux guest systems:

- *36 ECKD™ mod9 spread over one rank/LCU*
- *8 FICON paths*

For the database data disks:

- *6 SCSI disks with 10 GB each, spread over 6 ranks*
- *6 FCP channels*
- *1 channel and disk per database guest*
- *From the same DS8000 that contained the ECKD disks*

Server software

Table 1. Server software used

Product	Version/Level
Apache HTTP Server	2.2.3
IBM DB2 Universal Database™ Enterprise Server	9.1 fixpack 3
SUSE Linux Enterprise Server	SLES 10, SP1, 64-bit
Trade	6.1
WebSphere Application Server	6.1 fixpack 11, 31-bit
z/VM	5.3.0 SLU 0701

Client hardware

IBM eServer™ xSeries® model x335, 2x2.4 GHz, Intel Xeon® 4 GB RAM, 73 GB (works as a Trade workload generator)

Client Software

Table 2. Client software used

Product	Version/Level
xSeries model x335	
SUSE Linux Enterprise Server	SLES 10 (Trade client)
WebSphere Studio Workload Simulator	engine level - iwl-0-03309L

Xen tests hardware and software

To perform our Xen performance tests, we created a customer-like environment. We configured the hardware, software, and storage server.

Server hardware

Xen Host

IBM System x3950, 4 Intel Xeon dual core, 3.5 GHz

- *8 physical CPUs*
- *24 GB memory*
- *2 Ethernet ports (onboard BROADCOM Corporation NetXtreme BCM5704)*

- *8x Emulex 4 GB FC HBA PCI-X Controller Dual Port*
- *Adaptec AAC-RAID (rev 02)*
- *6x IBM 146 GB 10 K 2.5in Hot-Swap SAS HDD, combined into four RAID5 arrays using the Adaptec AAC-RAID adapter.*

Storage server setup

For the operating system and applications on 15 Linux guests systems, 15 images (10 GB each) on the local SCSI RAID5 array from the x3950 were used.

The data files of the database system were on the disks from one RAID5 array from a DS4300 express storage server (consisting of 30 physical disks with 34 GB and 15 K RPMs):

- *5 * 2 FCP disks, 10 GB each*
- *5 FCP paths with 2 GB per second*
- *1 channel and disk per database guest*

Server software

Table 3. Server software used

Product	Version/Level
Apache2 Web Server	2.2.3
IBM DB2 Universal Database Enterprise Server	9.1 fixpack 3
SUSE Linux Enterprise Server	SLES 10, SP1, 64-bit
Trade	6.1
WebSphere Application Server	6.1 fixpack 11, 32–bit
Xen	3.1*
* Xen is part of the distribution. SP1 comes with Xen 3.0.2. We used the update to 3.1 that can be found at: http://www.novell.com/coolsolutions/tools/19338.html Xen 3.1 is required for support of Intel Virtualization Technology (VT (previously called Vanderpool)).	

Client hardware

- *LS41 Blade Dual-Core AMD Opteron Processor 8212 HE, SLES 10*

Client Software

Table 4. Client software used

Product	Version/Level
LS41 AMD Blade	
SUSE Linux Enterprise Server	SLES 10 (Trade client)
WebSphere Studio Workload Simulator	engine level - iwl-0–03309L

Workload description

We used the Trade performance benchmark for our z/VM virtualization performance tests. Trade 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 JavaBeans™ built to open J2EE APIs. Together these provide versatile and portable test cases designed to measure aspects of scalability and performance. Figure 1 shows an overview of the Trade J2EE components.

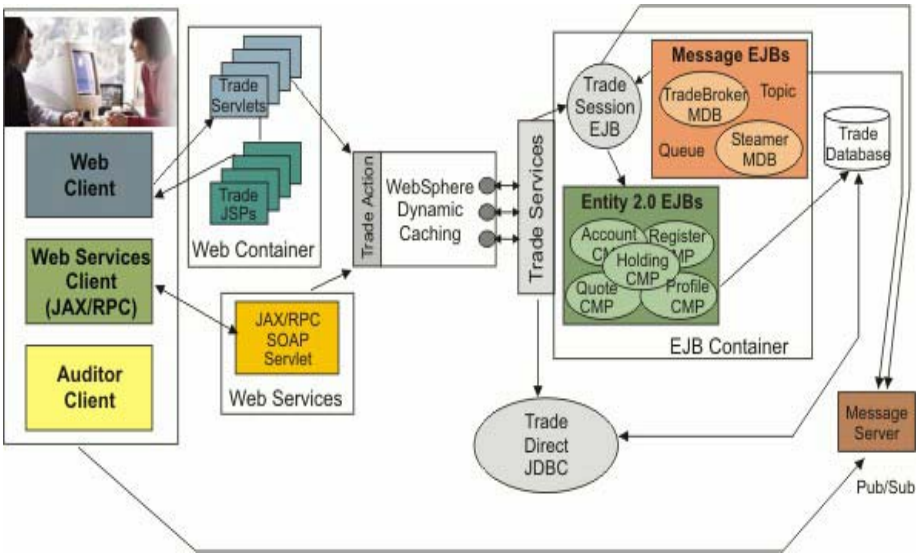

Figure 1. Trade J2EE components

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

The new Trade 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 1 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. Asynchronous_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.*
- *Web Services – 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.*
- *Distributed Map – 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.

z/VM and Xen system setup

This chapter details the modifications which we made to the system setup for our z/VM and Xen environments. It includes changes that were common to both environments as well as changes for the individual environments.

Common setup

This section describes the setup common on both the z/VM and Xen environments for our virtualization performance tests.

WebSphere environment

To emulate a customer-like configuration, one WebSphere Application Server environment consisted of an Apache HTTP Web Server, the WebSphere Application Server, and a DB2 UDB database server. This environment is called a triplet. In our tests, the number of triplets was scaled from 1, 2, 4, to 6.

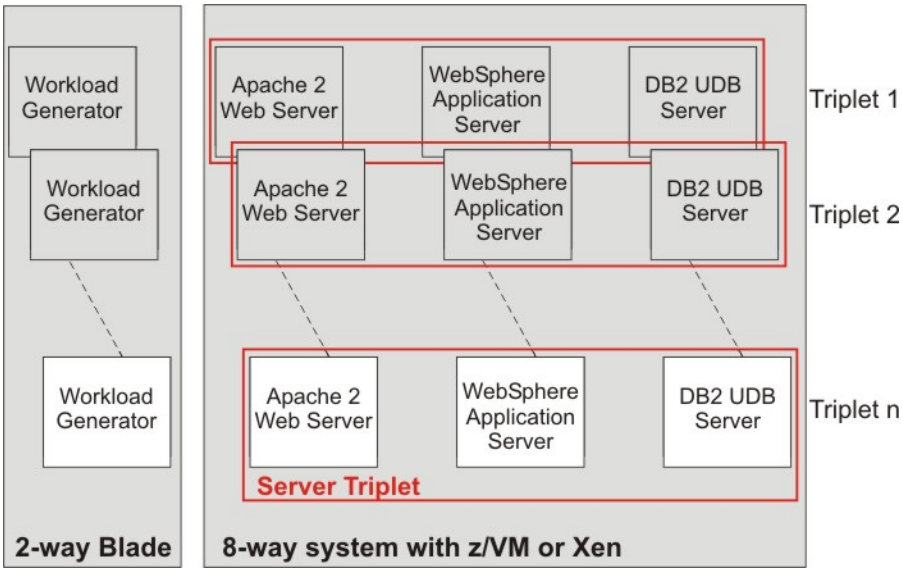

Figure 2. WebSphere environment – scaling server triplets and workload

WebSphere plugin for Apache Web Server

The 64-bit WebSphere Plugin 6.1 was installed and configured for use with the Apache Web Server in both the z/VM and Xen environments. Fixpack 11 provided 64-bit support. After the install, it was necessary to change the LoadModule specified in the httpd.conf file to point to the Apache 2.2 module. By default, it pointed to the Apache 2.0 module. Below is an excerpt from our httpd.conf file:

```
LoadModule was_ap22_module
/opt/IBM/WebSphere/Plugins/bin/64bits/mod_was_ap22_http.so
WebSpherePluginConfig
/opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
```

Java heap size

A Java heap size of 1 GB was used for both the z/VM and Xen WebSphere systems.

WebSphere and DB2 UDB tuning scripts

The scripts were used to tune both WebSphere and DB2 UDB to the Trade workload. Both of these scripts need only be run once on their respective WebSphere or DB2 UDB systems. Appendix A. z/VM and Xen setup examples contains copies of the WebSphere script (WebSphere sample tuning script) and DB2 UDB script (DB2 UDB sample tuning script).

SSL setup

Software encryption was enabled for the Apache Web Servers in both the z/VM and Xen environments. Below is a summary of the steps that we followed to enable this support:

- *Install required RPMs for openssl*
- *Generate a self-signed certificate using the openssl commands*
- *Update the Apache configuration file in /etc/sysconfig/apache2 to add ssl to the list of APACHE_MODULES and set APACHE_SERVER_FLAGS='SSL'*
- *Update the SSLCipherSuite directive in the /etc/apache2/vhosts.d/vhost-ssl.conf file to specify AES128-SHA for the cipher that the client is permitted to negotiate*

Additional steps were followed in the z/VM environment to enable hardware accelerated encryption. Refer to Configuring Linux and Apache HTTP Server to use hardware encryption detailed setup for more information.

WebSphere Studio Workload Simulator configuration

Parameter changes

Using the parameters specified on invocation of the workload generator engine, we made changes to the default WebSphere Studio Workload Simulator configuration, which was provided with the Trade distribution. The changes we made were:

- *The number of simulated clients was set to four. We determined that this number of clients enabled us to reach optimal throughput, and, at the same time, keep the system CP at around 90%.*
- *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.*

Below is a sample invocation of a WebSphere Studio Workload Simulator script for one triplet:

```
/var/iwl/bin/iwlengine -c 4 -e 0 -D on -r 10000 --enginename
triplet1 --max_clients \
300 --xml_interval 5 --timelimit 1200 -s
/etc/iwl/common/trade6_lnweb1.jxs
```

Script file changes

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

- *Hostnames lnweb[1 thru 6] were added to each of the six sample scripts.*
- *The "close_connections" command was added to increase the number of handshakes (WebSphere Studio Workload Simulator will reuse the connections otherwise). We used this to show the advantages of hardware encryption.*
- *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.

z/VM system setup

The following system changes were made for our z/VM test runs.

Guests setup

18 Linux guests were defined to the z/VM system. This gave us the ability to run a maximum of six triplets. This was possible due to the ability of z/VM to handle CP and memory overcommitment.

The z/VM guests were configured as follows:

Table 5. z/VM guests setup

Guest type	# CPUs	Memory [MB]	FCP Adapter	Physical Network Interface	Pure Virtual Network Interface	Crypto enabled
lnweb[n]	1	750	N/A	1	1	Yes
lnweb[n]	2	2048	N/A	N/A	1	No
lnweb[n]	1	1800	1	N/A	1	No

See z/VM directory templates for Linux guests in z/VM setup examples for more detail of how the guests were defined in the z/VM user directory.

Overcommitment

The z/VM LPAR is defined with a predetermined number of physical processors. z/VM can then define second level guests with a specific number of virtual processors. If the aggregate of virtual processors is greater than the number of physical processors defined to the LPAR, then the z/VM system is overcommitted in CPU. For example, assume each guest in a z/VM environment is defined with one virtual CPU. Also, assume that the z/VM LPAR has eight physical CPUs defined to it. Then the system will be CPU overcommitted once nine guests are brought up.

Changing the number of CPUs per guest will only lower or raise the virtual number of CPUs that the Linux guest will think it was defined to it. In other words, the physical aggregate will always be eight CPUs. The following table shows which scenarios were running with more virtual resources than physical resources available (overcommitment) and the grade of that overcommitment.

Table 6. Overcommitment – z/VM (based on an LPAR with eight physical CPUs and 22 GB memory)

Guests (# of Triplets)	# Virtual CPUs	Ratio of CPU Overcommitment	Memory [MB]	Ratio of Memory Overcommitment
6 (2)	8	1.0:1	9,196	N/A
9 (3)	12	1.5:1	13,794	N/A
12 (4)	16	2.0:1	18,392	N/A
15 (5)	20	2.5:1	22,990	1.1:1
18 (6)	24	3.0:1	27,588	1.4:1

The environment with 15 guests (five triplets) uses all the memory for the guests. This could be considered the start of an overcommitment because there is some space needed for the z/VM operating system itself. This scenario is actually running with a CPU overcommitment ratio of 2.5:1 (2.5 times the number of virtual CPs as there are physical). In any run with 18 guests (six triplets), the number of virtual CPUs and memory exceeds the physical resources significantly. Be aware that the z/VM system has a very fast paging device with the 2 GB of expanded storage.

z/VM shares

The set share command and the share directory statement allow you to control the priority that system resources are assigned to a guest. These resources can include processors, real storage, and I/O capability. An absolute or a relative share can be specified.

- *An absolute share – allocates to a virtual machine a percentage of all available system resources.*
- *A relative share – allocates to a virtual machine a portion of the total system resources minus any absolute shares. Also, a virtual machine with a relative share will receive access to system resources that are in proportion with respect to other virtual machines with relative shares. For example, if a virtual machine (VM1) has a relative share of 100 and a second virtual machine (VM2) has a relative share of 200, VM2 receives twice as much access to system resources as VM1.*

All measurements in this study that were run on z/VM did not set a specific share. Virtual Linux guests had a relative share value of 100, which is the default.

z/VM Quickdsp

The set quickdsp command and the quickdsp operand of the option directory statement allow you to designate virtual machines that will not wait in the eligible list when they have work to do.

All measurements in this study that were run on z/VM specified the quickdsp operand, for the guests, it was specified in the option directory statement for all z/VM virtual guest user directory definitions.

Network setup

The network setup for the z/VM Linux guests was as follows:

- *The guests with a Web Server had one network interface to the physical Gbit Ethernet for communications with the client system.*
- *A guest LAN with type HiperSockets™ was defined with a 16 KB frame size for guest-to-guest communications.*

Figure 3 gives an overview of the network configuration.

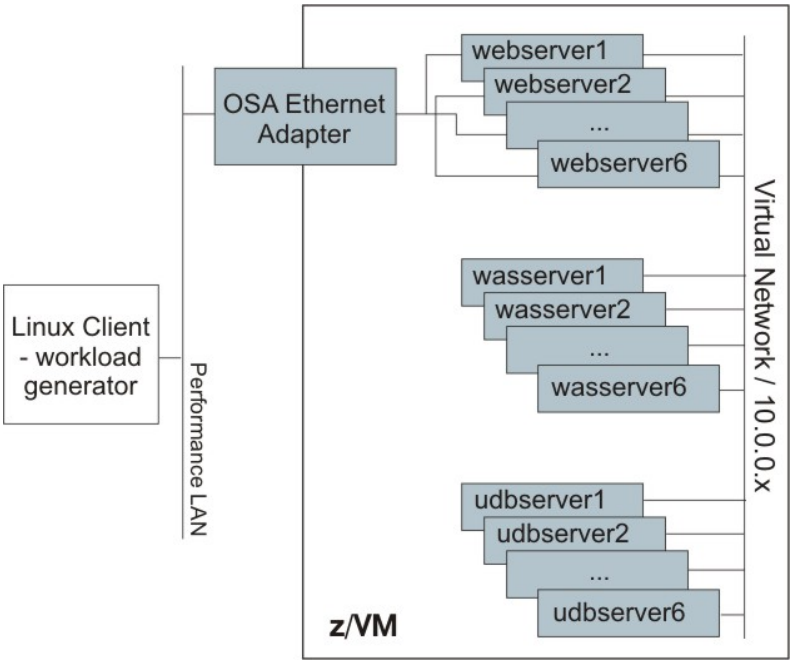

Figure 3. z/VM test environment

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/AppSrv01/bin #
cat startServer.sh
#!/bin/sh
echo 268435456 >/proc/self/mapped_base
WAS_USER_SCRIPT=/opt/IBM/WebSphere/AppServer/profiles/AppSrv0
1/bin/setupCmdLine.sh
export WAS_USER_SCRIPT
/opt/IBM/WebSphere/AppServer/bin/startServer.sh "$@"
```

Configuring Linux and Apache HTTP Server to use hardware encryption

Because hardware encryption SSL, which is available on System z, will yield better performance than software SSL, we needed to configure Linux and Apache HTTP Server to use hardware encryption. We used Chapter 6 in the Redbook Security on z/VM as a reference. This book can be found at:

http://www.redbooks.**ibm.com**/redbooks/pdfs/sg247471.pdf

This document addresses only SUSE SLES 10 SP1. The commands may change from release to release. As a summary, here are the steps we took to configure Linux and the Apache HTTP Server to use hardware encryption:

- *Install the RPMs for the z90crypt driver, libica, openCryptoki, and openssl-ibmca engine*
- *Load the z90crypt driver*
- *Check the z90crypt device status*
- *Prepare the ibmca engine*

- *Dynamically load the ibmca engine*
- *Create a self-signed certificate using the openssl commands*

For details on each of these summary points, see Configuring Linux and Apache HTTP Server hardware encryption detailed setup.

During this study, the cryptographic hardware was used for authentication when establishing new SSL sessions as well as for encryption of the HTTP requests from the client to the Web Server. The major part of the encryption work uses the CPACF feature for data encryption, therefore, does not require opening new SSL sessions. Traffic from the Web Server to the WebSphere Application Server was not encrypted, because the isolation of the guest LAN network would not allow external access, as long as there is no additional TCP/IP router. The Apache HTTP Server supports the Central Processor Assist for Cryptographic Function (CPACF) on System z. We used a cipher of AES128-SHA for all of the measurements with SSL to take advantage of the performance benefits of hardware accelerated encryption.

Xen system setup

The following system changes were made for our Xen test runs.

Hyper-Threading

Hyper-Threading technology provided by Intel allows two process threads to execute on the same processor core. These threads can share the memory cache on the processor. Because these additional processors provided by Hyper-Threading can only use unused units of the processor chip, we decided not to use hyper threaded processors. This avoids the disadvantages for the Xen system when guest runs are performed on one limited processor. We disabled it via a BIOS setting.

All the measurements done in this study disabled the Hyper-Threading support for the Xen virtual machines.

FCP setup

The data files of the Linux databases should reside on an external storage server, because it is typical on enterprise environments like System z. We decided to use fixed block disk (SCSI disks) connected via FCP. This requires that the database guests get access to the FCP adapters from the x3950. Because Xen does not

provide a virtualization layer for the FCP cards, it was necessary to attach them directly to the guest. The alternative would have been to mount the FCP disks to the domain 0 and attach them as emulated disks to the guests. The expectation was that passing the FCP adapter directly into the guest avoids one level of indirection which should perform better. To do this, we had to prevent the Xen hypervisor (Dom0) from getting access to that hardware. To give a guest direct (and exclusive) access to PCI cards, Xen provides a mechanism called PCI-Back-Hide. The basic requirement to use this is that the Domain 0 kernel contains the support for PCI backend and the Domain U kernel needs the support PCI frontend, determined by the config parameters CONFIG_XEN_PCIDEV_FRONTEND and CONFIG_XEN_PCIDEV_BACKEND.

The process should work the following way:

1. *The FCP card must be hidden during the boot process via the boot parameter:*
   `pciback.hide=(04:01.0)(04:01.1)(06:01.0)(06:01.1)(08:01.0)(08:01.1)`
   where the numbers 04:01.0 are the description of the PCI devices in terms of <bus>:<slot>.<func> as reported from lspci with the unhidden devices.
2. *The FCP card gets assigned to the guests via the profile statement:*
   `pci=['04:01.0']`

Our kernel only contained the support for the PCI front end. The support for PCIBACK was given via module, which leads to the following message in /var/log/boot.msg.

```
Unknown boot option
`pciback.hide=(04:01.0)(04:01.1)(06:01.0)(06:01.1)(08:01.0)\(08:01.1)': ignoring
```

Therefore, we used the late binding mechanism together with the definition of the load order in the file /etc/modprobe.conf by adding the following statements:

```
install lpfc/ sbin/modprobe pciback ;/sbin/modprobe --first-time -- ignore-install lpfc
options pciback hide=(04:01.0)(04:01.1)(06:01.0)(06:01.1)\(08:01.0)(08:01.1)
```

This makes sure that the PCIBACK module is loaded before the driver lpfc for the Emulex FCP card.

The success of this step is shown by the messages in/var/log/boot.msg:

```
<6>pciback 0000:04:01.0: seizing device
<6>pciback 0000:04:01.1: seizing device
<6>pciback 0000:06:01.0: seizing device
<6>pciback 0000:06:01.1: seizing device
<6>pciback 0000:08:01.0: seizing device
<6>pciback 0000:08:01.1: seizing device
```

After performing the above steps, the guests can access the FCP disk via the adapter assigned via the profile parameter after boot. However, this only works reliably after reboot. When a guest is shutdown and restarted several times, it can easily lose access to the FCP adapters. In that case, a reboot of the whole system is required.

Paravirtualized guests

It was very important to use paravirtualized guests, which means the guest systems use a special API provided by the hypervisor (Domain 0). For a non paravirtualized guest, the Xen system would need to emulate the complete behavior of physical hardware, however, this takes more effort and degrades the performance.

To provide the API, the Linux kernel with the Xen modifications is booted in Domain 0 (indicated by the suffix -xen. For example: vmlinuz-2.6.16.46-0.12-xen). The Xen profile parameter builder='linux', used in the Xen profile, (see Xen setup examples) specifies that the guests should run paravirtualized. In this case, the guest Linux kernel does require support for the para-virtualized Xen frontend device drivers, which is also provided by the vmlinuz-2.6.16.46–0.12–xen kernel and the initrd-2.6.16.46–0.12–xen (see http://wiki.xensource.com/xenwiki/XenSplitDrivers for more information).

Be aware that the special processor features for virtualization support, like VT from Intel, or AMD-V from AMD are usually required for full virtualized guests.

Guests setup

The Xen guests were sized according to the following table.

Table 7. Xen guests setup

Guest type	#CPUs	Memory [MB]	FCP Adapter	Physical Network Interface	Pure Virtual Network Interface
lnweb[n]	1	750	N/A	1	1
lnwas[n]	2	2048	N/A	N/A	1
lnudb[n]	1	1800	1	N/A	1

Additionally, each guest had one partition of the local RAID5 arrays defined as a virtual block device for the root file system. A 300 MB file was used as the swap device.

Our current Xen version, 3.1, was only able to handle guest memory static according to the definition in the profile, which means memory overcommitment was not possible. For that reason we limited the number of triplets to five because we did not want to adapt the scenario to the limitations of Xen.

Note: We had to specify an additional profile parameter to define the console.

extra = "TERM=xterm selinux=off xencons=tty"

Otherwise, the Xen guest had no console and did not boot.

Our sample profiles can be found in Appendix A.

Overcommitment for Xen

Table 8 shows which scenarios were running with more resources than the physical resources available (overcommitment) and the grade of that overcommitment. On this release of Xen, memory overcommitment was not possible because of the failing on demand memory feature.

Table 8. Overcommitment – Xen

Guests (triplets)	# Virtual CPU's	CPU Overcommitment	Memory Overcommitment
6 (2)	8	1.0:1	N/A
9 (3)	12	1.5:1	N/A
12 (4)	16	2.0:1	N/A
15 (5)	20	2.5:1	N/A

Network Setup

The network setup for the Xen system was done in the following way:

- *The guests with a Web Server had one network interface to the physical Gbit Ethernet network card for the communication with the client system.*
- *All guests had one pure virtual network interface (dummy device) for the guest-to-guest communication.*

Figure 4 gives an overview of the network configuration.

Creating the guest networks

To create the two networks we needed two Xen bridges in Domain 0 (Xen host), bound to the corresponding host and guest devices. The Xen bridges were created in the script /etc/xen/scripts/network-bridge/network-custom and we specified that the script should be used in /etc/xen/xend-config.sxp with the statement:

```
(network-script 'network-custom')
```

The script network-custom contains the following statements to create the two bridges:

```
# First arg is the operation name
OP=$1
shift
script=/etc/xen/scripts/network-bridge

# mapping to generic network-bridge - args are the configuration
$script ${OP} vifnum=0 bridge=xenbr0 netdev=eth2
$script ${OP} vifnum=1 bridge=xenbr1 netdev=dummy0
```

The first bridge connects the physical network interface with the bridge xenbr0. The second bridge `xenbr1` connects to the dummy interface. The vif statement in the guest profile connects the guest with the bridges.

```
vif=[ 'mac=00:16:3e:00:00:10, bridge=xenbr0',
'mac=00:16:3e:00:00:11, bridge=xenbr1' ]
```

In all measurements, the interface to xenbr0 (external network) was only configured on the Web server.

Creating the dummy device

The dummy device was created in the Domain 0 (Xen host) by adding the following statements to /etc/modprobe.com:

```
alias dummy0 dummy
install dummy0 /sbin/modprobe -o dummy0 --ignore-install dummy
```

Also, the following network script must also be created in /etc/sysconfig/network/ifcfg-dummy0 to configure the device:

```
BOOTPROTO='static'
BROADCAST='' ETHTOOL_OPTIONS=''
IPADDR='10.3.0.202'
MTU=''
NAME='Dummy network interface vor bridged virtual network'
NETMASK='255.255.0.0'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
UNIQUE='mY_N.QMtraf_3M34'
USERCONTROL='no'
```

This creates the dummy0 network device (see also http://virt.kernelnewbies.org/XenWifiNetwork)

Checksumming on the external interfaces

An essential performance improvement was to toggle the checksumming off on the external interfaces from the corresponding guests, enabling it later had no further impact on the performance.

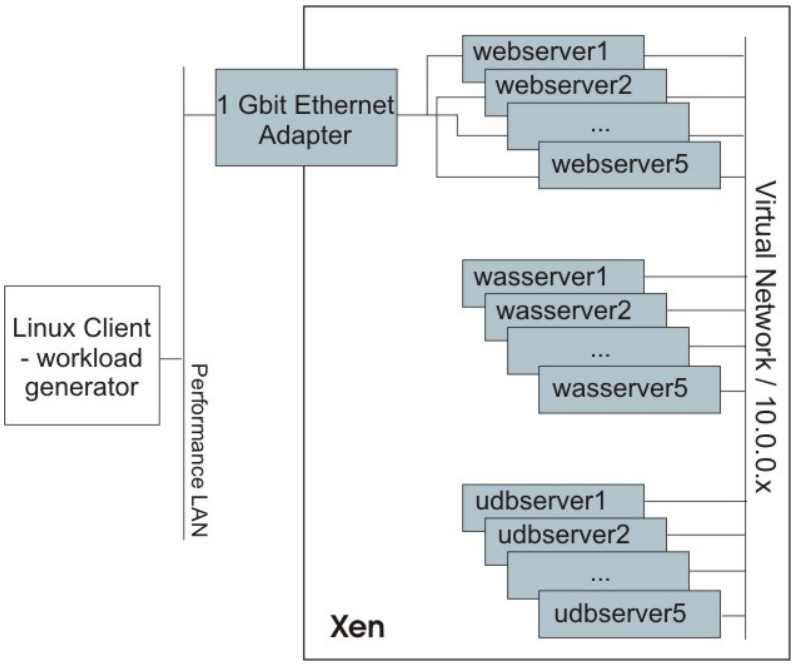

Figure 4. Xen test environment

Gathering statistics data

In both the z/VM and Xen environments, performance data was collected in one minute intervals for ten minutes during the steady state period.

On z/VM, the z/VM Performance Toolkit was used to determine the CPU load.

On Xen, xentop was used with the following parameters:

```
xentop -b -d 60 -i 10
```

The output was redirected to a file. In both cases, the CPU utilization is reported that 100% means all eight CPUs are fully utilized CPU.

Collecting the output

This chapter details the steps we took to gather the output for our z/VM and Xen 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*
   a. *FTP the trade_tune.jacl file to the WebSphere Application Server system*
   b. *Start the WebSphere Application Server*
   c. *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: 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.
2. *Run the Trade DB2 tuning script*
   a. *Backup the DB2 UDB system by issuing the following commands:*
   db2 connect to tradedb
   db2 get db cfg for tradedb > <some file name>
   db2 terminate
   b. *Run the Trade tune database script as db2inst1*
   Note: You only need to run the WebSphere and DB2 tuning scripts once during your testing (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 Apache or IBM HTTP server*
4. *Request the Trade URL*
5. *Via the Trade dialogs, reset the Trade DB*
6. *Via the Trade dialogs, repopulate the Trade DB with 500 users*

7. *Run runstats on the DB2 UDB system. To do this, enter the following commands while under your DB2 userid:*
   a. *db2 connect to tradedb*
   b. *db2 reorgchk update statistics*
   c. *db2 terminate*
   d. *db2stop force*
   e. *db2start*
8. *Shutdown the DB2 UDB, WebSphere Application Server, and the WebServer Linux guests*
9. *Reboot z/VM*
10. *Start up the Linux guests*
11. *Start WebSphere, Web Server, and UDB*
12. *Run the WebSphere Studio Workload Simulator script*
    Have it run for 20 minutes. The first five minutes will be used as a "warmup" period.
13. *Start the z/VM Performance Monitor after the warmup period*
14. *Issue the following command at the beginning and end of each run. The run will last 10 minutes after the warmup period.*
    run netstat -s (once at the beginning of the run and once at the end)
    collect sar data for the measurement period of 10 minutes
15. *Stop the VM performance monitor after 10 minutes*
16. *Stop all servers, DB2, WebSphere, and Web Server at the end of the 20 minutes*
17. *Gather measurement data and WebSphere Studio Workload Simulator results*

When running the workload from the Xen system, use the following modification:

- *After step 10, toggle checksumming off then on, on the external network interfaces.*
- *Run the xentop monitor for the 10 minutes during steady state. This is issued from the Xen Domain 0 partition.*

The key items that we used for this paper came from the WebSphere Studio Workload Simulator output. They were Page element throughput, Transactions throughput, and average page element response time. Processor utilization was

then taken from either the z/VM Performance Toolkit data or Xentop depending on which platform was run.

Cumulative statistics were printed every five minutes and at the end of the WebSphere Studio Workload Simulator execution time interval. This interval was set to 1200 seconds.

Results for the virtualization tests

After performing our z/VM and Xen virtualization performance tests, we charted our test results, interpreted the results, and created recommendations.

We performed the following tests and comparisons:

- *z/VM and Xen Comparison for resource overcommitment*
  - *CPU overcommitment*
  - *Memory overcommitment and z/VM memory management*
- *Using encrypted communication*
- *Comparison with prior software releases*

z/VM and Xen comparison for resource overcommitment

To compare the virtualization features of z/VM and Xen we took a look at the results of the transaction throughput, the response times, and the CPU load when scaling the guests (triplets), consisting of a Web Server, a WebSphere Application Server, and a DB2 UDB database server.

Scaling the number of triplets means scaling the amount of virtual CPUs and memory. With three triplets we run into CPU overcommitment, with five triplets memory overcommitment starts at the z/VM environment (for more details see Table 6 and Table 8). It was not possible to compare the memory overcommitment behavior, because the memory handling up to Xen 3.1 is static, which means overcommitment is not possible. Therefore, we left out the 18 guest (six triplets) results for Xen. Because the memory overcommitment is only slight here, we did an additional test to demonstrate the behavior of z/VM for memory overcommitment under heavier conditions. All tests done here are with unencrypted communications.

CPU overcommitment

Throughput

z/VM and Xen transaction throughput for scaling the CPU overcommitment are compared in this section.

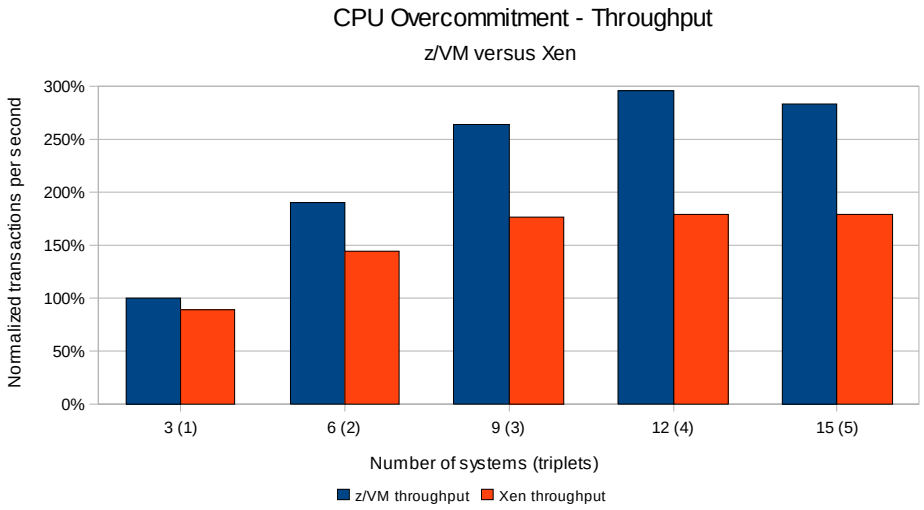

Figure 5. CPU Overcommitment – Throughput z/VM versus Xen

Observations

The throughput rate for z/VM is significantly higher than that of the Xen environment. Maximum throughput for both environments is reached with 12 guests (four triplets). The results show that z/VM throughput scales much better and the change in the delta of throughput between the two environments is much steeper on the z/VM side. Once nine guests (three triplets) is reached, the throughput for Xen seems to flatten out. The z/VM throughput increases up to 12 guests (four triplets), where z/VM reaches 66% higher throughput.

Conclusion

From a throughput perspective, z/VM scales better than Xen. z/VM throughput after nine guests (three triplets) continues to scale, while on Xen, the throughput rate flattens out after nine guests. With nine guests (three triplets) both environments are CPU overcommitted. The throughput on z/VM, though the

system is CPU overcommitted, does not start to degrade until we reach 15 guests (five triplets). Xen never runs with memory overcommitment.

CPU utilization

z/VM and Xen CPU utilization for scaling the CPU overcommitment are compared in this section.

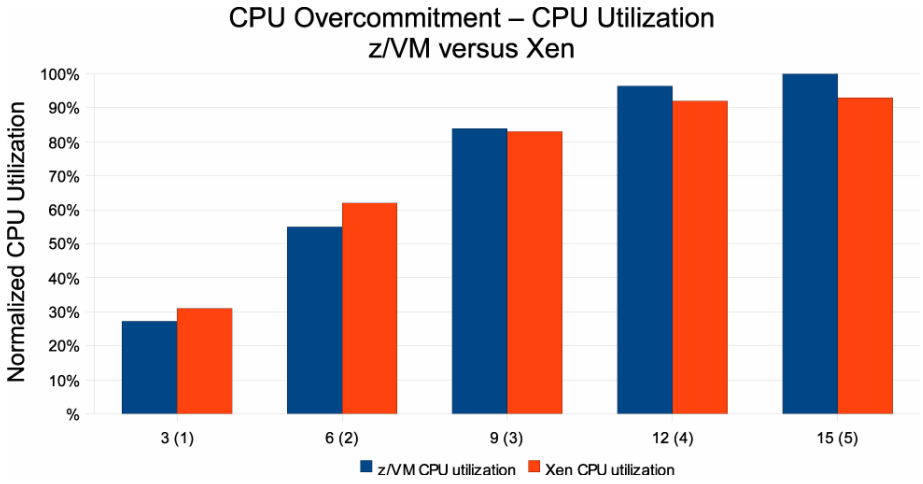

Figure 6. CPU Overcommitment – CPU Utilization – z/VM versus Xen

Observations

The z/VM system was fully utilized at 15 guests (five triplets). The Xen system starts with a higher CPU utilization, but it did not reach full utilization. At nine guests (three triplets) the two environments are running at equivalent utilization. However, the throughput on z/VM is 50% higher. The maximum throughput for both environments is reached with 12 guests (four triplets), where the throughput on z/VM is 66% higher.

Conclusion

The Xen test cases start with lower throughput, but higher CPU utilization. When scaling the guests, the level on resource overcommitment increases. With a CPU overcommitment ratio of 1.5:1, the throughput flattens out. This is true despite available CPU cycles. Scaling further on the Xen system increases only the CPU load but not the throughput. The static handling of memory in Xen 3.1

limited the amount of memory that could be allocated to the guests. Since scaling the guests also means increasing the memory footprint, a memory limit was reached and we were unable to start additional guests. Scaling on z/VM is limited by the total CPU utilization. We were able to scale the CPU overcommitment on z/VM up to a ratio of 3:1 at 18 guests (six triplets) with only a small degradation in throughput. We did not scale further due to the lack in resources.

There are several factors that may be contributory to the degradation in throughput in both environments. With 15 guests (five triplets) on z/VM, the CPU is fully utilized and, therefore, would cause jobs to wait and create management overhead, thereby lowering throughput. On Xen we see that with 15 guests (five triplets) CPU utilization has increased, but not to full utilization and with no increase in throughput. We have job queuing here and probably serialization effects due to the degree of overcommitment (2.5:1) at this point. We see from these values that z/VM is more efficient in managing resource overcommitment. Again, on Xen, we are not able to overcommit memory.

Response time

z/VM and Xen response times for scaling the CPU overcommitment are compared in this section.

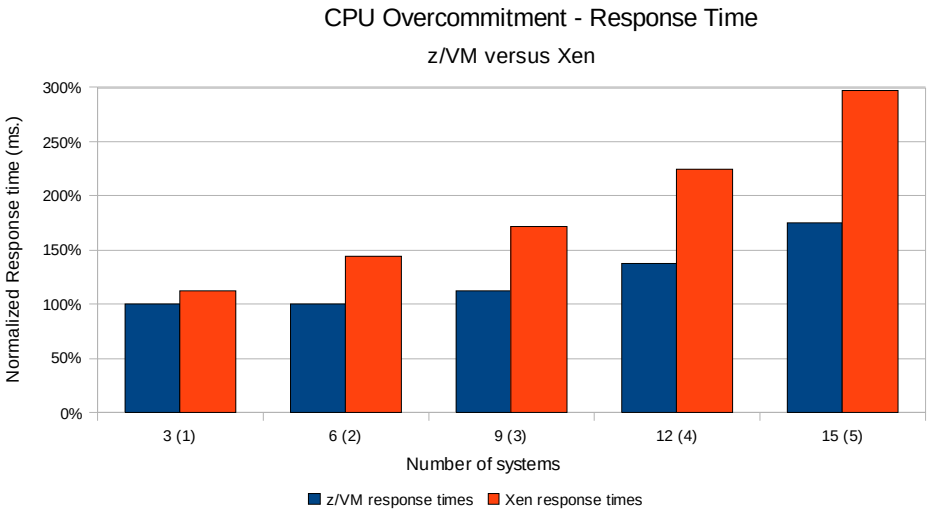

Figure 7. CPU Overcommitment – Response times – z/VM versus Xen

Observations

We see the response times on z/VM stay flat until CPU overcommitment starts with nine guests (three triplets). On Xen, the response times are increasing with the first scaling step. We also see the increase change as we increase triplets to be greater in the Xen environment.

Conclusion

This is another indicator that illustrates how z/VM manages overcommitment of resources more efficient than Xen. z/VM gets sensitive when CPU overcommitment starts, while Xen shows increasing response times from the first scaling step. Response times do increase, but they do not increase as drastically in z/VM as the do in Xen. At the same time, throughput in the z/VM environment continues to increase at a faster rate than in Xen. Finally, the response times on z/VM are nearly two times faster than on Xen.

Memory overcommitment and z/VM memory management

The Linux guests for each triplet were defined as follows.

- *The WebSphere images were defined with 2048 MB of memory because of their heap size.*
- *The DB2 UDB images were defined with 1800 MB to account for bufferpool storage.*
- *The Web Server was defined with 750 MB.*

All tests were run with five triplets active, which results in a total of 22990 MB storage defined for the guests. In each test, 2 GB was defined to expanded storage in z/VM. Central storage was varied.

Table 9 shows the test case results used to create Figure 8.

Table 9. Scaling z/VM memory with 15 guests (five triplets) defined with 22,990 MB total

z/VM Central Storage (MB)	Resident Memory Pages (MB)	Memory in XSTORE (MB)	Memory on Paging DASDs (MB)	Ratio of Memory Over-commitment
20480	11318	0	0	1.1:1
11264	10894	774	16	2.1:1
10240	9878	1176	0	2.3:1
9216	8862	1930	338	2.5:1
81921	7846	2034	1471	2.8:1
7168	6831	2057	2525	3.2:1
6144	5839	2146	3461	3.8:1

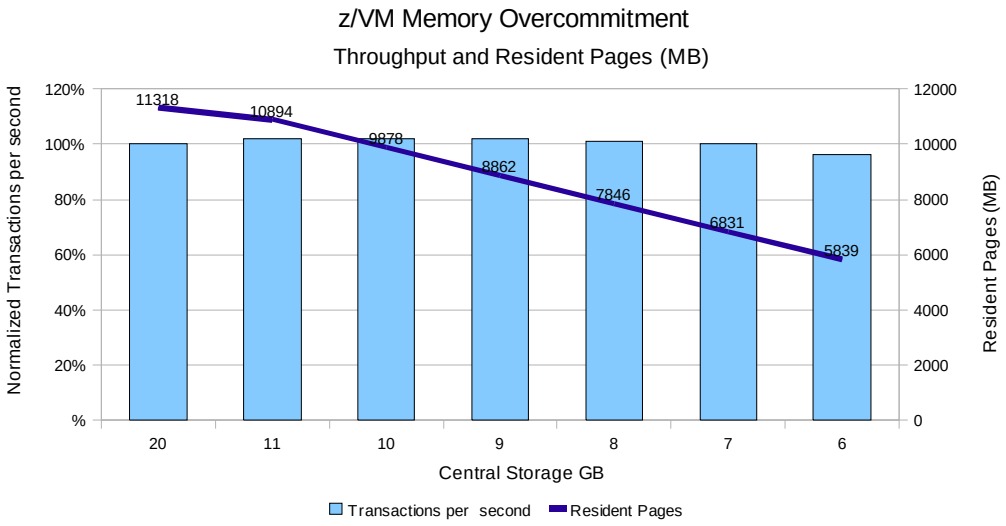

Figure 8. z/VM Memory Overcommitment – Throughput and Resident Pages (MB)

Observations

The amount of resident pages (pages allocated in main memory) for the guests, at 20 GB central storage, was only about 11 GB. Reducing the central storage to 11 GB improves the transaction throughput slightly. At 7 GB, central storage throughput is at the same level as with 20 GB. The throughput degradation starts at 6 GB central storage.

Be aware that the memory which could not be held in the central storage, has been moved to expanded storage and the paging devices.

Conclusion

z/VM manages resources very efficiently so the throughput does not degrade in the overcommitted scenarios. In fact, as seen in Figure 8, a decrease of central storage from 11 GB down to 7 GB translated to only an overall throughput degradation of 1.5%. This shows that z/VM allows for some efficiencies in memory management. First, z/VM only backs memory of guests that has actually been used. That is, if a page of memory for a guest machine is unused, z/VM does not keep a copy of it anywhere. Further, z/VM has the ability to page out pages that are not currently in use and then page them back into real memory when needed. The largest drop in throughput (4.2%) occurred when the system central storage was lowered from 7 GB to 6 GB. Yet the overall decrease in throughput, with central storage dropping from 11 GB to 6 GB, was only 5.8%. When we decrease down to 6 GB, we had an increased amount of paging to DASD, causing degradation in throughput.

As storage is lowered, the z/VM system does a good job of scaling the amount of resident pages from the Linux guests to allow the work flow to proceed without much degradation in throughput.

It was not possible to compare the memory overcommitment behavior of Xen since the memory handling up to Xen version 3.1 is static, which means overcommitment is not possible.

Using encrypted communication

z/VM and Xen encrypted transaction throughput, utilization and response time are compared in this section. Because the Central Processor Assist for Cryptographic Function (CPACF) is given to you on a System z at no extra cost, we decided to compare the System z hardware accelerated encryption with the software encryption provided by Xen. We have also included the software encryption throughput for z/VM. Be aware that the CPUs on Xen are twice as fast as the CPUs on z9, which is expected to be an advantage for the Xen system.

The following three figures compare the throughput, CPU utilization and response time for an encrypted (both hardware and software for z/VM) workload for the two environments.

Throughput

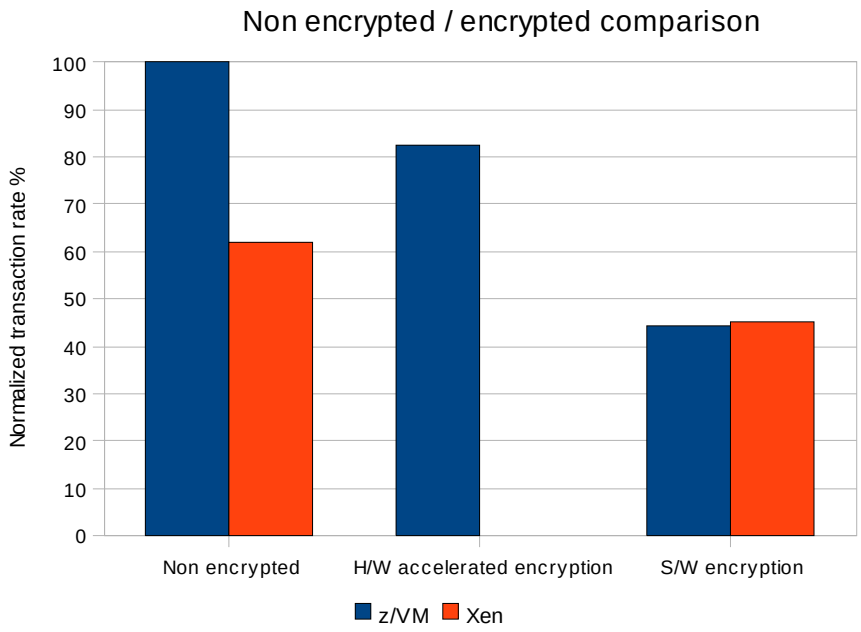

Figure 9. Non encrypted versus encrypted comparison – z/VM versus Xen

Observations

The throughput from an encrypted workload for z/VM using hardware accelerated encryption is nearly double that of the Xen encrypted throughput.

Conclusion

Using encrypted communication is a very CPU intensive workload, which is shown when using software encryption. A close to 2x increase in throughput shows clearly that off loading the cryptographic functions to hardware in the z/VM environment is a great advantage over the Xen encrypted environment, even the Xen environment uses two times faster CPUs. When using the hardware accelerated encryption, the impact of encryption is reduced to an acceptable 18% degradation.

CPU utilization

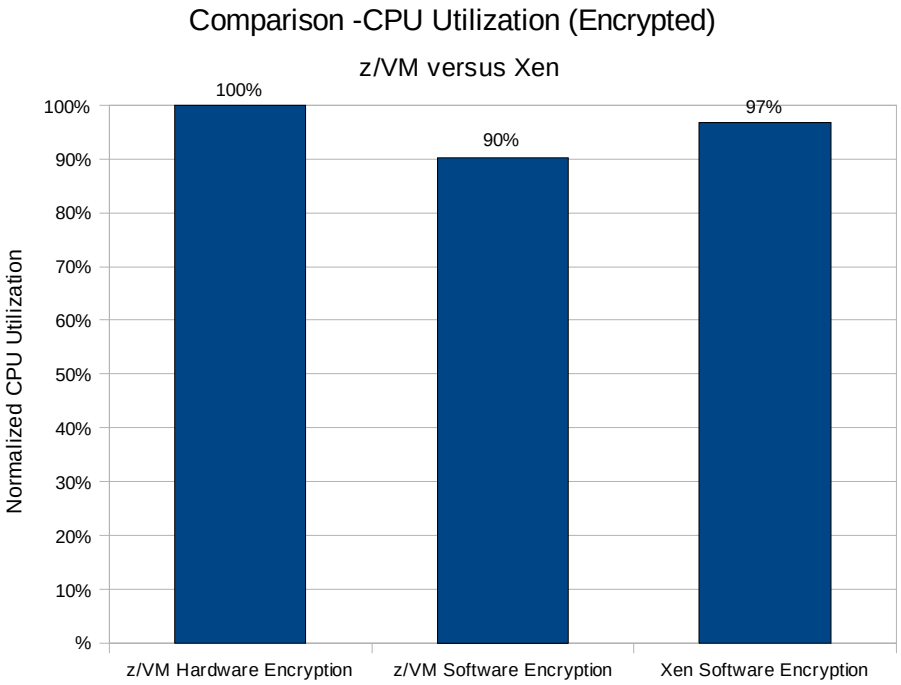

Figure 10. Comparison – CPU Utilization (encrypted) z/VM versus Xen

Observations

The z/VM hardware accelerated encryption workload fully utilizes the CPUs. The software encryption on Xen uses nearly the full CPUs for half of the throughput.

Conclusion

The values charted are for a maximum throughput case. z/VM utilization for hardware accelerated encryption is 100%. This shows that z/VM can fully utilize each percent of CPU and the throughput is nearly twice that of Xen. This is despite Xen running on xSeries with CPUs that are twice as fast as the tested System z.

Response time

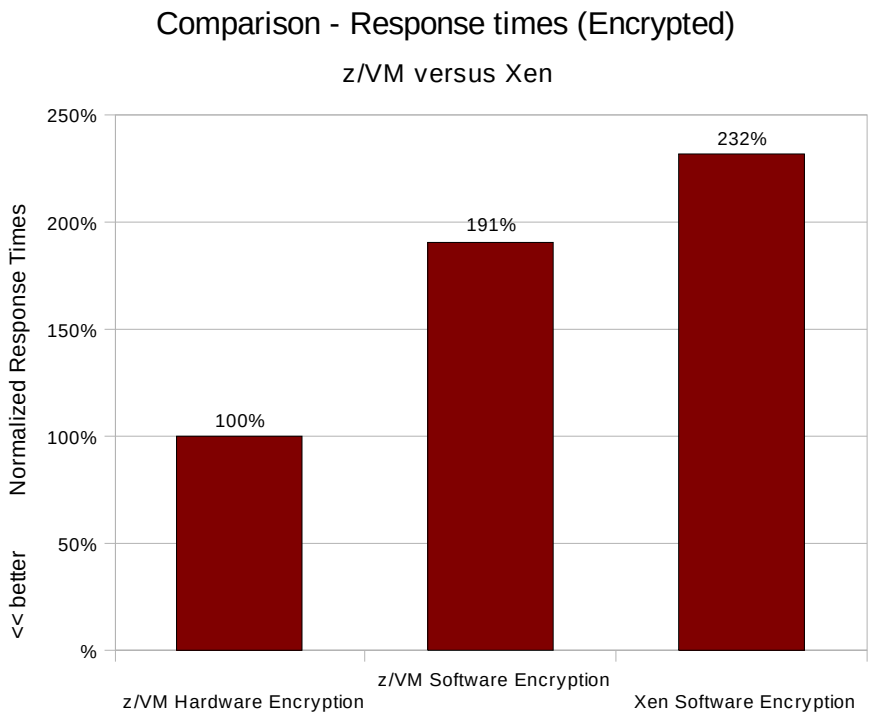

Figure 11. Comparison response times (encrypted) – z/VM versus Xen

Observations

The response time for Xen is more than double that on z/VM for hardware accelerated encryption. Even for the software encryption, z/VM has shorter response times.

Conclusion

Again, offloading the encrypted work to the cryptographic hardware on a System z results in more than half the response time of the Xen system at twice as higher throughput, even though CPUs on the Xen system are twice as fast as on the z/VM system. Keep in mind that, with this configuration, we are also resource overcommitted and z/VM is more efficient at managing resources, which contributes to its shorter response time.

Comparison with prior software releases

Comparisons of prior software releases are shown in this section.

In a prior study (more details at http://download.boulder.**ibm.com**/ibmdl/pub/software/dw/linux390/perf/ZSW03020-USEN-00.pdf) we did the same test with older software levels but the same workload driver and on the same System z9 hardware. This enables us to show the impact of the software upgrades.

Table 10. Software levels used for testing

z/VM	Web Server	Distribution	WebSphere Application Server	Java	DB2
5.2	IBM HTTP Server 6.0.2	SLES 9, SP3	6.0.2 fixpack 1	1.4	8.2 fixpack 4
5.3	Apache 2.2.3	SLES 10, SP1	6.1 fixpack 11	1.5	9.1 fixpack 3

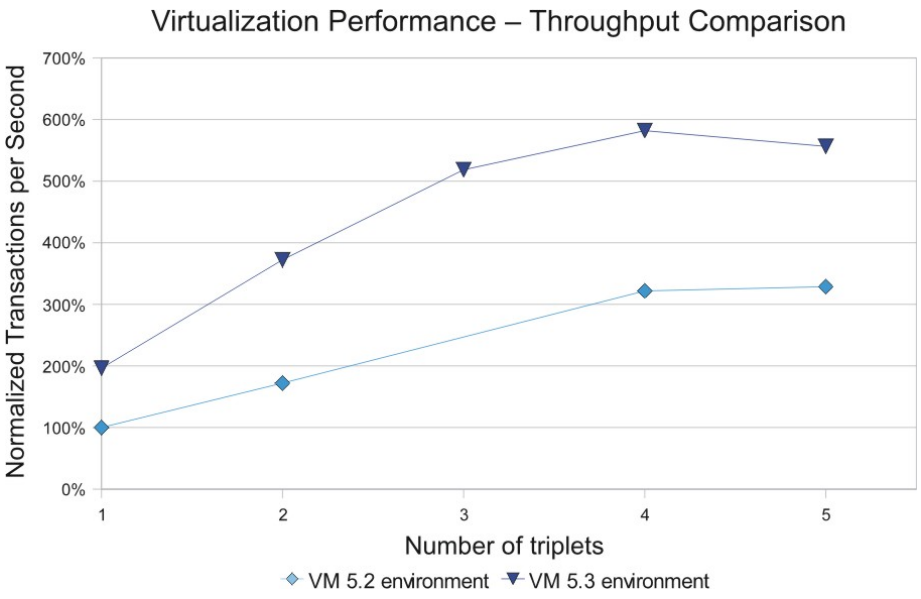

Figure 12. Virtualization Performance – Throughput Comparison

Observations

The newer software stack used in the z/VM 5.3 environment provides a significant higher throughput, around a factor 2 higher, starting from the first scaling step. Additionally we had verified in the z/VM 5.3 environment that the difference in throughput between using IHS or Apache is below 2%.

Conclusion

We highly recommend the upgrade to z/VM 5.3, WebSphere Application Server 6.1 with Java 1.5, and DB2 9.1, for it provided an impressing performance improvement, around a factor 2 for all scaling steps.

Appendix A. z/VM and Xen setup examples

The following section contains details on our setup and configuration scripts for our z/VM and Xen test runs.

z/VM setup examples

The follow section contains details on our setup and configuration scripts for our z/VM test runs.

Configuring Linux and Apache HTTP Server to use hardware encryption detailed setup

The following items are required in order to enable hardware encryption on a Linux image running under System z:

1. *z90crypt device driver*
2. *OpenSSL*
3. *libica*
4. *openCryptoki*
5. *ibmca engine (OpenSSL engine)*

Prior to installing openCryptoki on SLES 10 SP1, the rpm package for xcrytolinz must be installed. This contains the IBM Common Cryptographic Architecture (CCA) host libraries and tools. This package is not delivered with the SLES 10 SP1 distribution. It can be downloaded from the Web site, http://www-03.**ibm.com**/security/cryptocards/pcixcc/ordersoftware.shtml. Reference the section titled "Obtaining CCA software for System z servers running Linux".

Install the RPMs

Below is a list of the RPMs that needed to be installed on SLES 10 SP1.

```
-rw------- 1 root root 367925 Oct 16 14:06 xcryptolinzGA-3.28-rc08.s390x.rpm
-rw------- 1 root root  65366 Oct 16 08:32 libica-1.3.7-0.17.s390x.rpm
-rw------- 1 root root  41243 Oct 16 08:33 libica-32bit-1.3.7-0.17.s390x.rpm
-rw------- 1 root root  49105 Oct 16 08:53 openCryptoki-2.2.2-24.14.s390.rpm
-rw------- 1 root root 124303 Oct 16 08:54 openCryptoki-32bit-2.2.2-
24.14.s390.rpm
-rw------- 1 root root 207913 Oct 16 08:36 openCryptoki-64bit-2.2.2-
24.14.s390x.rpm
-rw------- 1 root root  15082 Oct 16 13:57 openssl-ibmca-1.0.0-7.11.s390x.rpm
-rw------- 1 root root  13313 Oct 16 13:57 openssl-ibmca-32bit-1.0.0-
7.11.s390x.rpm
```

Load the z90crypt driver

Issue the command `rcz90crypt start`. This loads the z90crypt module. The domain and poll parameters are read from /etc/sysconfig/z90crypt. For clear key crypto with shared Adjunct Processors (APs), it is not necessary to specify a domain. The default domain of -1 can be used. In our environment, we set domain=-1 and poll=0 as shown below:

```
# This variable selects the crypto domain to be used,
# required if an LPAR owns several crypto domains.
# The value of -1 is used for autodetect.
#
Z90CRYPT_DOMAIN=-1

## Path:        Kernel/z90Crypt
## Description: Turn poll thread on/off
## Type:        list(0,1)
## Default:     1
#
# When running with polling thread one CPU without
# outstanding workload is constantly polling the
# crypthographic requests. The polling thread will
# sleep when no cryptographic requests are currently
# processed. This mode utilize the cryptographic card
# as much as possible at the cost of blocking one CPU.
```

```
# Without polling thread the cryptographic cards are
# polled at a much lower rate resulting in higher
# latency and reduced throughput for cryptographic
# requests but without a notable CPU load.
#
Z90CRYPT_POLL=0
```

Alternatively, you can use the modprobe or insmod command to load the z90crypt driver.

```
modprobe z90crypt domain=<domain> poll_thread=<#>
```
or
```
insmod z90crypt domain=<domain> poll_thread=<#>
```

The <domain> is a number in the range of 0 to 15 or -1 for autodetect. The <#> is either 0 (disabled) or 1 (enabled). Use the `chkconfig z90crypt` on command to ensure that the z90crypt device driver is automatically loaded when the Linux system is booted.

To disable the device driver, issue one of the following commands:

```
rcz90crypt stop
```
or
```
modprobe -r z90crypt
```

Check the status of the z90crypt device

Issue the command: `cat /proc/driver/z90crypt`

Your results should be similar to the following:

```
lnweb1:/home/gene/rpms # cat /proc/driver/z90crypt

zcrypt version: 2.1.0
Cryptographic domain: 5
Total device count: 1
PCICA count: 0
PCICC count: 0
PCIXCC MCL2 count: 0
PCIXCC MCL3 count: 0
CEX2C count: 1
CEX2A count: 0
requestq count: 0
pendingq count: 0
```

```
Total open handles: 1

Online devices: 1=PCICA 2=PCICC 3=PCIXCC(MCL2) 4=PCIXCC(MCL3) 5=CEX2C
6=CEX2A
0000000000000000 0000500000000000 0000000000000000 0000000000000000

Waiting work element counts
0000000000000000 0000000000000000 0000000000000000 0000000000000000

Per-device successfully completed request counts
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000007 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Prepare the ibmca engine

Apache uses the OpenSSL interface for cryptographic operations. OpenSSL then passes the encryption request to the IBM engine ibmca, which invokes the services of the underlying library libica. The request is then passed to the CPACF.

Append the contents of /usr/share/doc/packages/openssl-ibmca/openssl.conf.sample to end of /etc/ssl/openssl.cnf and move the line openssl_conf = openssl_def to the top of the configuration file.

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#
openssl_conf = openssl_def

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                    = .
RANDFILE                = $ENV::HOME/.rnd
```

Dynamically load the ibcma engine

```
lnweb1:/usr/lib64/engines # openssl engine -c
dynamic) Dynamic engine loading support
ibmca) Ibmca hardware engine support
RSA, DSA, DH, RAND, DES-ECB, DES-CBC, DES-EDE3, DES-EDE3-CBC, AES-128-ECB,
AES-128-CBC, AES-192-ECB, AES-192-CBC, AES-256-ECB, AES-256-CBC, SHA1,
SHA256]
```

Generate a self-signed certificate

Use the openssl commands to generate a self-signed certificate.

```
lnweb1:/etc/ssl #cd /etc/ssl

lnweb1:/etc/ssl # openssl req -config openssl.cnf -new -nodes -keyout
  apachenodes.key -out apache.csr

lnweb1:/etc/ssl # openssl rsa -in apachenodes.key -des3 -out
  apache.key
```

```
lnweb1:/etc/ssl # openssl x509 -in apache.csr -out apache.crt -req -
signkey
  apache.key -days 999

lnweb1:/etc/ssl # cp apache.crt /etc/apache2/ssl.crt/server.crt

lnweb1:/etc/ssl # cp apache.key /etc/apache2/ssl.key/server.key
```

Update Apache configuration

Use the a2enmod and a2enflag commands to modify the Apache configuration file to enable SSL.

```
lnweb1:/etc/sysconfig # a2enmod ssl
```

This updates the /etc/sysconfig/apache2 file with the following:

```
APACHE_MODULES="actions alias auth_basic authn_file authz_host
authz_groupfile
  authz_default authz_user authn_dbm
autoindex cgi dir env expires include log_config mime negotiation
setenvif ssl suexec
userdir php5"

lnweb1:/etc/sysconfig # a2enflag SSL
```

This updates the /etc/sysconfig/apache2 file with the following:

```
APACHE_SERVER_FLAGS="SSL"
```

Copy vhost-ssl.template to vhost-ssl.conf

Copy the sample ssl configuration file and update the SSL Cipher Suite directive to specify the AES128-SHA cipher. On System z, encryption with the AES128-SHA cipher causes the CPACF instruction set to be utilized.

```
lnweb1:/etc/apache2/vhosts.d # cp vhost-ssl.template vhost-ssl.conf

<VirtualHost _default_:443>

  #  General setup for the virtual host
  DocumentRoot "/srv/www/htdocs"
  #ServerName lnweb1.pbm.ihost.com:443
  #ServerName www.example.com:443
  #ServerAdmin webmaster@example.com
```

```
  ErrorLog /var/log/apache2/error_log
  TransferLog /var/log/apache2/access_log

  #   SSL Engine Switch:
  #   Enable/Disable SSL for this virtual host.
  SSLEngine on

  #   SSL Cipher Suite:
  #   List the ciphers that the client is permitted to negotiate.
  #   See the mod_ssl documentation for a complete list.
  # SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
  SSLCipherSuite AES128-SHA
```

Update /etc/apache2/gobal-ssl.conf

Add the SSLCryptoDevice directive to the global ssl configuration file and specify the ibmca engine.

```
<IfDefine SSL>
<IfDefine !NOSSL>
<IfModule mod_ssl.c>
        SSLCryptoDevice ibmca
        #
```

Verify that port 443 is defined to the WebSphere Application Server as a Virtual Host alias.
Start the Apache web server

```
rcapache2 start
```

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 SSL testing. Lines that we changed are in **bold** for emphasis only. Note that this is just a portion of this script.

```
//
/*trade6 version 1.2*/
init_section
{
        string hostname = "lnweb1:443";
        int botClient = 0;
        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;
HttpResponse r;
close_connection(); //force connection close if open to force SSL
handshake
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,start,"",
"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,start,"?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");
```

```
weight 4:
 start_transaction("update_account");
 // Performan an account, followed by an account update
 startpage(6);
thinktime(1000);

getpage_ssl(""+hostname+"","/trade/app",1,close,0,start,"?action=account",
"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)",
```

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.

```
#-------------------------------------------------------------------
# WebSphere Tuning Script for Trade
#-------------------------------------------------------------------
#
# 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 tuneTrade.jacl <scope> <id>
#      scope      - 'cluster' or 'server'
#      id         - name of target object within scope (ie. servername)
#
# Examples:
#   wsadmin -f tuneTrade.jacl server server1
#
```

```
#   wsadmin -f tuneTrade.jacl cluster TradeCluster
#
#
#-----------------------------------------------------------------


$AdminConfig setValidationLevel NONE

set buildDate "v1.10 - 09292005"

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

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

# ORB properties (false)
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

# JVM properties
# Note: OS specific changes to the heap size settings are located
#  in the next section -> "Base OS Specific JVM settings". Changes
#  to the generic JVM arguments are handled below.
set minHeap        1024
set maxHeap        1024
set verboseGC      "true"
set genericArgs    ""

# OS Specific JVM options
set IBMJDKoptions       ""
```

```
set SUNJDKoptions
 "-XX:+UseTLAB -Xnoclassgc -XX:MaxPermSize=64m -Xmn256m
  -XX:SurvivorRatio=16"
set HPJDKoptions
 "-Xmn512m -XX:PermSize=80m -XX:+ForceMmapReserved
 -XX:SurvivorRatio=16 -Xoptgc -XX:+UseParallelGC
    Have it run for 20 minutes. The first five minutes will be used as a
"warmup" period.
   -
Djava.nio.channels.spi.SelectorProvider=sun.nio.ch.DevPollSelectorProvider
  -XX:-ExtraPollBeforeRead -XX:+UseSpinning"
set ISeriesJDKoptions   "-Djava.compiler=jitc"

# 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]

#-------------------------------------------------------------------
# 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 $HPJDKoptions]
} elseif {$env(os.name) == "z/OS"} {
   set genericArgs [concat $genericArgs $IBMJDKoptions]
   set minHeap        768
   set maxHeap        768
} elseif {$env(os.name) == "OS/400"} {
   set genericArgs [concat $genericArgs $ISeriesJDKoptions]
   set maxHeap        0
}

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

proc printUsageAndExit {} {
   puts " "
   puts "Usage: wsadmin -f tuneTrade.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
   }

}

#-------------------------------------------
# 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 "   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 " 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 ""
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 orb config

   puts "Accessing ORB configuration..."
   set orb [$AdminConfig list ObjectRequestBroker $server]
   puts "ORB noLocalCopies (old/new): $AdminConfig showAttribute $orb
     noLocalCopies]/$noLocalCopies"
   set attrs [subst {{noLocalCopies $noLocalCopies}}]
   puts "Updating ORB..."
   puts " "
   $AdminConfig modify $orb $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
$webPool
     maximumSize]/$maxWebPool"
   puts "ThreadPool MinSize (old/new):[$AdminConfig showAttribute
$webPool
     minimumSize]/$minWebPool"
```

```
   set attrs [subst {{maximumSize $maxWebPool} {minimumSize
$minWebPool}}]
   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]

   if {[llength $HTTPInbound] != 0} {
      set oI [lsearch -glob $HTTPInbound "*HTTP_2*"]
      set http2 [lindex $HTTPInbound $oI]
      puts "KeepAlive Enabled (old/new):[$AdminConfig showAttribute
$http2
        keepAlive]/$keepAliveEnabled"
      puts "Max Persistent Requests (old/new):[$AdminConfig showAttribute
$http2
        maximumPersistentRequests]/$maxPersistentRequests"
      set attrs [subst {{keepAlive $keepAliveEnabled}
{maximumPersistentRequests
        $maxPersistentRequests}}]
      puts "Updating HTTP KeepAlives..."
      puts " "
      $AdminConfig modify $http2 $attrs
   }

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

```
{verboseModeGarbageCollection $verboseGC} {genericJvmArguments
"$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 $log fileName] Rollover Type (old/new):
[$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 $traceService
     startupTraceSpecification]/$traceSpec"
puts "Rollover File Size (old/new):[$AdminConfig showAttribute $traceLog
     rolloverSize]/$traceRolloverSize"
   puts "Max Backup Files (old/new):[$AdminConfig showAttribute $traceLog
     maxNumberOfBackupFiles]/$maxFiles"
   set attrs  [subst {{startupTraceSpecification $traceSpec}}]
   set attrs2 [subst { {rolloverSize $traceRolloverSize}
{maxNumberOfBackupFiles $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,
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.

```
Zdb2 -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"
```

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   750M 750M 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 MODIFY
*                     LINUX IPL DASD
 DEDICATE 6400 6400
 DEDICATE 6401 6401
1.DEDICATE 6402 6402
*                     Gigabit Devices - Admin LAN
 DEDICATE 1823 1823
 DEDICATE 1824 1824
 DEDICATE 1825 1825
*                     Gigabit Devices - Performance LAN
 DEDICATE 1304 1304
 DEDICATE 1305 1305
 DEDICATE 1306 1306
 DEDICATE 1307 1307
*                     Guest LAN setup
 SPECIAL 1000 HIPER 3 SYSTEM XENLAN
*                     "A" disk
 MDISK 191  3390  541 50 V62M01 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
*                     LINUX IPL DASD
 DEDICATE 6400 6403
 DEDICATE 6401 6404
 DEDICATE 6402 6405
*                     Gigabit Devices- Admin LAN
 DEDICATE 1823 1826
 DEDICATE 1824 1827
 DEDICATE 1825 1828
*                     Gigabit Devices - Performance LAN
 DEDICATE 1304 1308
 DEDICATE 1305 1309
 DEDICATE 1306 130A
 DEDICATE 1307 130B
*                     Guest LAN setup
 SPECIAL 1000 HIPER 3 SYSTEM XENLAN
*                     "A" disk
 MDISK 191  3390 591 50 V62M01 MR  ALL WRITE MULTI
*-------------------------------------------------------
USER LNUDB1  999999   1800M 1800M 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
```

```
*                     LINUX IPL DASD
 DEDICATE 6400 6406
 DEDICATE 6401 6407
 DEDICATE 6402 6408
*                     SCSI device
 DEDICATE 0800 0800
*                     Gigabit Devices- Admin LAN
 DEDICATE 1823 1829
 DEDICATE 1824 182A
 DEDICATE 1825 182B
*                     Gigabit Devices - Performance LAN
 DEDICATE 1304 130C
 DEDICATE 1305 130D
 DEDICATE 1306 130E
 DEDICATE 1307 130F
*                     Gigabit Devices
 SPECIAL 1000 HIPER 3 SYSTEM XENLAN
*                     "A" disk
 MDISK 191  3390 641 50 V62M01 MR  ALL WRITE MULTI
```

Xen setup examples

The follow section contains details on our setup and configuration scripts for our z/VM test runs.

Xen Guest profiles

Profile for the WebSphere Application server guest

```
kernel = "/etc/xen/vmlinuz-xen"
ramdisk = "/etc/xen/initrd-xen"
builder='linux'
memory = 2048
name = "lnwas1"
vcpus = 2
vif = [ 'mac=00:16:3e:00:00:10, bridge=xenbr0', 'mac=00:16:3e:00:00:21,
bridge=xenbr1' ]
disk = [ 'phy:sdd5,xvda1,w', 'phy:/dev/hda,hdc:cdrom,r' ]
root = "/dev/xvda1"
extra = "TERM=xterm selinux=off xencons=tty"
vfb=[ "type=vnc,vncunused=1" ]
```

Profile for the Web Server guest

```
kernel = "/etc/xen/vmlinuz-xen"
ramdisk = "/etc/xen/initrd-xen"
builder='linux'
memory = 750
name = "lnweb1"
vcpus = 1
vif = [ 'mac=00:16:3e:00:00:10, bridge=xenbr0', 'mac=00:16:3e:00:00:11,
bridge=xenbr1' ]
disk = [ 'phy:sdc5,xvda1,w', 'phy:/dev/hda,hdc:cdrom,r' ]
root = "/dev/xvda1"
extra = "TERM=xterm selinux=off xencons=tty"
vfb=[ "type=vnc,vncunused=1" ]
```

Profile for the database server guest

```
kernel = "/etc/xen/vmlinuz-xen"
ramdisk = "/etc/xen/initrd-xen"
builder='linux'
memory = 1800
name = "lnudb1"
vcpus = 1
vif = [ 'mac=00:16:3e:00:00:10, bridge=xenbr0', 'mac=00:16:3e:00:00:31,
bridge=xenbr1' ]
disk = [ 'phy:sde5,xvda1,w', 'phy:/dev/hda,hdc:cdrom,r' ]
root = "/dev/xvda1"
extra = "TERM=xterm selinux=off xencons=tty"
vfb=[ "type=vnc,vncunused=1" ]
pci= [ '04:01.0' ]
```

Grub menu.lst

The following entry boots Xen with the Domain 0.

```
title XEN (/dev/sdb1)
 root (hd1,0)
 kernel /boot/xen.gz
 module /boot/vmlinuz-2.6.16.46-0.12-xen root=/dev/disk/sdb1 vga=normal
splash=0
 showopts pciback.hide=(04:01.0)(04:01.1)(06:01.0)(06:01.1)(08:01.0)(08:01.1)
 module /boot/initrd-2.6.16.46-0.12-xen
```

Appendix B. Other sources of information for the z/VM and Xen performance tests

Additional resources to provide information on the products, hardware, and software discussed in this paper can be found in various books and at various Web sites.

For information on DB2 UDB see:

- http://www.**ibm.com**/software/data/db2/udb/support/manualsv9.html

For information on Linux on System z see:

- www.**ibm.com**/systems/z/os/linux/

For information on WebSphere Application Server see:

- www.**ibm.com**/software/webservers/appserv/was/

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

For information on hardware encryption, see the following IBM Redbooks®:

- *Security on z/VM at* http://www.redbooks.**ibm.com**/redbooks/pdfs/sg247471.pdf
- *Using Cryptographic Adapters for Web Servers with Linux on IBM System z9 and zSeries® at* http://www.redbooks.**ibm.com**/redpapers/pdfs/redp4131.pdf

Copyright IBM Corporation 2009
IBM Corporation
New Orchard Rd.
Armonk, NY 10504
Printed in the United States of America, 01/2009
All Rights Reserved

IBM, IBM eServer, the IBM logo, DB2, DB2 Universal Database, DS8000, ECKD, FICON, HiperSockets, Redbooks, System Storage, System z, System z9, WebSphere, xSeries, z9, z/VM and zSeries are trademarks or registered trademarks of International Business Machines Corporation of the United States, other countries or both.

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.

InfiniBand and InfiniBand Trade Association are registered trademarks of the InfiniBand Trade Association.

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

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

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.

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

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

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

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

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

[1.] This paper is intended to provide information regarding performance of environments using WebSphere Application Server 6.1, DB2 9.1, z/VM and Xen. 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.

ZSW03051-USEN-00