WebSphere on Linux for zSeries
July 2004

How to Determine the Causes of Performance Problems with WebSphere Applications running on Linux for zSeries

Steve Wehr, zSeries New Technology Center

Table of Contents

Abstract

What do you do when you get a call from your customer, who says this:

> "We have a WebSphere application running on Linux for zSeries,
> and its performance is bad".

This document describes a process to follow to help diagnose performance problems, and suggestions for resolving them.

This document is intended to be used by IBM field personnel working with the customer to diagnose such a problem, or by the customer themselves.

Acknowledgements

My thanks to those who contributed technical information to this paper:

Jon von Wolfersdorf
Carlos Ordonez
Mike Everett
Mike Casile
Don Corbett
Ann Jackson
Fred Mora
Bill Reeder
Mary Hu
Scott Robertson

Chapter 1: Introduction

When you run WebSphere® on Linux® for the IBM @server® zSeries® you have a very large software stack: LPARs running VM/ESA®, VM running multiple Linux guests, each Linux running WebSphere, each WebSphere running your applications. Boy… that's a mouthful! With all these layers involved it can be easy to make mistakes in setup that affect performance.

When customers report such problems, you need a skilled team and a process they can follow to help diagnose problems.

Skills Needed

First you need to assemble a team that can address all aspects of the performance of the software stack. The following skills are usually required on such a team.

- *Overall Coordinator*
- *VM systems programmer. The person who set up all the Linux guests in VM.*
- *Linux administrator. The person who installed and configured Linux.*
- *WebSphere administrator.*
- *Lead Application programmer. The person who can answer questions about what the application does and how it does it.*
- *Network administrator.*

Optionally bring in specific skills for any other products that are part of the application flow. You may need help from IBM on some of these, since customers themselves may not have these skills:

- *DB2 Connect™ and DB2® performance*
- *WebSphere/Java™ application performance.*
- *WebSphere Commerce performance*
- *WebSphere Portal performance*

Process

It makes sense to start from the outside and work toward the application. About half of the performance problems are caused by the environment surrounding the application, and the other half by the application itself. So we want to start with the environment the application runs in, eliminating potential causes of performance problems, and working toward the application in the following manner.

1. *LPAR. Look at: Number of IFLs, Weight, caps, total real memory, Memory allocation between cstore and xstore.*
2. *VM. Look at: Communications configuration between Linux guests and other LPARs. Paging space, share settings.*

3. Linux. Look at: Virtual memory size, virtual CPUs, VM share and limits, swapping, swap file size, kernel tuning.
4. WebSphere. Look at: JVM heap size, connection pool sizes, use of caches.
5. DB2 or DB2 Connect.
6. WebSphere application performance characteristics.

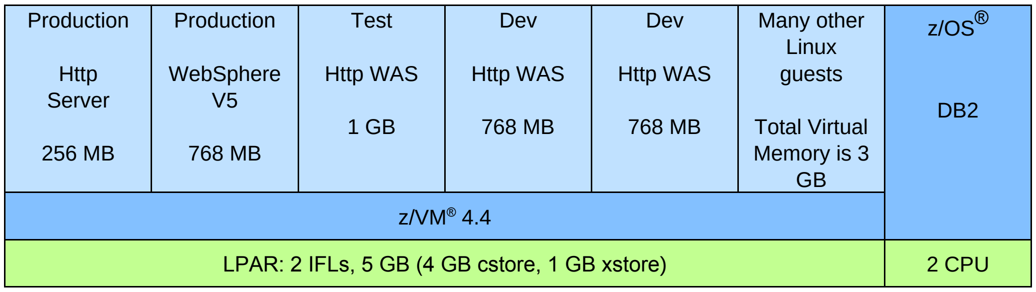
The following sections of this paper explain each of these in detail.

As you work through this process, looking at the above areas, you also want to have a repeatable testing process available so that you can measure how the application's performance responds to changes you have made. So we have included a chapter on page 15 about how to do that.

Finally, this document is not meant to replace the Solutions Assurance Review process. You are encouraged you to ask your IBM rep for a SAR for WebSphere on Linux for zSeries. The SAR covers a much wider range of topics than is covered in this paper. "Poor performance" really means that the customer expected better performance than they are seeing. This may caused by a performance tuning problem, or by the customer's expectations for performance being higher than the system can deliver. A good SAR and sizing (using SIZE390) will help eliminate such problems before they begin.

Typical System Configuration

This diagram depicts a typical configuration used by customers who are new to WebSphere on Linux. They typically have several Linux guests doing other work and are adding their first set of guests to handle the development, test and production for a new WebSphere application.



There are a set of guests running the production WebSphere application (HTTP, and WAS), a guest for testing the WAS application, and several guests for development. Then there are many other guests doing other production, test and development work.

Chapter 2: Is the Software Stack supported by WebSphere?

There are so many combinations of Linux distributions, Web Servers, JVMs, LDAP servers, etc., that WebSphere has to run on top of, that WebSphere cannot test with all of them. So they test most of these combinations and document which they have tested in the "WebSphere Support Matrix". Your particular configuration may not have been tested, and even though it may work, WebSphere will not accept PMRs for untested configurations. So please make sure you are running a supported configuration by checking the WebSphere support matrix. The URL is located on page 19.

Chapter 3: Investigate the LPAR Setup

Let's get started by looking at your configuration to see if we can find any settings that could cause performance problems. We first start with the LPAR settings. This is the LPAR that is running the Linux guests that are exhibiting poor performance. The LPAR defines the memory and CPU available to VM. Also the LPAR may be sharing CPUs with another LPAR.

Check This	Explanation
Does this LPAR share IFLs with another LPAR? • If so, what is the weight given to this LPAR? Is it capped?	You want to ensure that this LPAR is getting the CPU it needs to drive the Linux guests. Use VMPRF or some other tool to run an LPAR report to see if this LPAR's cap on CPU is being hit.
Has a Sizing been done to determine that there is sufficient CPU to drive this WebSphere application? If not, then request one through IBM's TechLine. (or contact SIZE390)	You should request a sizing to ensure that there is enough CPU for what the WebSphere application will consume. Request either a: • WebSphere sizing for Linux on zSeries. • Server Consolidation sizing.
How much real memory is assigned to this LPAR? • How does that memory compare to the total virtual memory requirements of all the Linux guests that are running? • Is approximately 75% of the memory assigned to Central Storage (cstore) and approximately 25% assigned to Expanded storage (xstore)?	Make sure there is sufficient real memory to back the virtual memory requirements of all the Linux guests. Use my spreadsheet to help calculate this. See "" on page for info on how to get the spreadsheet. Make sure cstore and xstore are allocated close to a 75/25 ratio.

Chapter 4: Investigate the VM Setup

VM is the hypervisor that will manage your virtual Linux servers. VM calls these virtual servers "guests." It is important that you have a skilled VM systems programmer on your team, because much of the successful setup of Linux servers is done in VM.

Check This	Explanation

Is your system CPU constrained? If so,
• What are the share settings of the Linux guests?
• Do the most important guests have the highest share?
• Do all production guests have higher shares than test or development guests?
 	
By default, all general users have a share setting of relative 100. We recommend that "infrastructure" guests, like a DB2 server, be given a higher share than a guest running WebSphere.
	In the z/VM directory entry for each Linux guest, you can set the users SHARE, which will prioritize CPU resources when CPU is constrained. Guests with a higher SHARE value will be given a priority when requesting CPU resources.
	
	We recommend using *relative* share values as opposed to *absolute* and we do not recommend mixing the two unless you have in-depth understanding of the process or are directed to so by an IBM consultant.
	
	We also recommend using a soft limit on the Linux guests. With soft limits, as opposed to hard limits, the guest can receive more than his share if CPU is not constrained.
	
	Here is an example:
	`Production user1:`
	`SET SHARE 400 REL LIMITSOFT`
	`Test user1:`
	`SET SHARE 200 REL LIMITSOFT`
	`Development user1:`
	`SET SHARE 100 REL LIMITSOFT`
	`Development user2:`
	`SET SHARE 100 REL LIMITSOFT`

Enable QUICKDSP for key service machines. Quick dispatch (QUICKDSP) can be set in the directory or via the CP SET command. We recommend setting QUICKDSP only for production guests and server virtual machines *that perform critical system functions* (i.e. VM TCP/IP or a Linux guest serving as a router). You should not set it for all your Linux guest machines.	QUICKDSP makes a virtual machine exempt from being held back in an eligible list during scheduling when system memory and/or paging resources are constrained. Virtual machines with QUICKDSP set on go directly to the dispatch queue and are identified as Q0 users. We prefer that you control the formation of eligible lists by tuning the CP SRM values and allowing a reasonable over commitment of memory and paging resources, rather than depending on QUICKDSP.
Are there network response issues? This can be affected by a mismatch of MTU sizes between Linux guests' network interfaces.	As much as possible, make sure that the MTU size is the same along the entire network path. A traceroute will identify the network path taken. MTU size negotiation and/or packet fragmentation will adversely affect network performance. Data streaming workloads benefit from large MTU sizes. To fully exploit HiperSockets™ LANs, large

	MTU sizes should be specified for HiperSockets Network interfaces. HiperSockets LANs support up to a 56k MTU.
Is the VM TCP/IP guest using significant CPU? Then consider implementing the VM Virtual Switch (VSWITCH) instead. We recommend using Virtual LAN (VLAN) or VSWITCH instead of VM TCP/IP. VSWITCH is the strategic solution for communications between z/VM guests and anything external to the zSeries machine.	With VSWITCH, the routing function is handled directly by VM's Control Program instead of the TCP/IP machine. This can help eliminate most of the CPU time that was used by the virtual machine router it replaces, resulting in a significant reduction in total system CPU time. • Decreases ranging from 19% to 33% have been observed when a TCP/IP VM router was replaced with Vswitch. • Decreases ranging from 46% to 70% were observed when a Linux router was replaced with Vswitch. However, the security of VSWITCH is not equal to a dedicate firewall or an external router, so when high security is required of the router function, consider using those instead of VSWITCH.
Is an eligible list (E3) forming while running production workloads? If so, it is recommended to adjust the CP SRM settings using the following values: • CP SET SRM STORBUF=300,200,200 • CP SET SRM LDUBUF=100,100,100 The CP INDicate command issued from a privileged user like MAINT will identify the formation of an eligible list. A Linux guest posted to an eligible list will appear to be hung.	The default CP SRM settings are based on a traditional CMS workload composed of large numbers of interactive CMS guests with relatively small working set sizes. When employing z/VM as a hypervisor supporting multiple Linux guests with larger working set sizes, it is often necessary to over commit real memory and paging resources to prevent the formation of eligible lists. A privileged user can set the System Resource Management values dynamically. To set the desired SRM values at system initialization, the CP SET commands can be placed in AUTOLOG1's PROFILE EXEC.
One of the common mistakes with new VM customers is to ignore paging space. The VM system as shipped contains enough page space to get the system installed and run some trivial work. However, you should add DASD page space before running a proof of concept, performance testing, or production. You may have enough real memory so that paging will not occur often. But if you don't and if z/VM pages frequently, then follow the directions on the right.	• Use whole volumes for VM paging, instead of fractional volumes. In other words, never mix paging I/O and non-paging I/O on the same pack. • Implement a one-to-one relationship between paging CHPIDs and paging volumes. • Spread the paging volumes over as many DASD control units as you can. • If the paging control units support NVS or DASDFW, turn them on (applies to RAID devices). • Provide at least twice as much DASD paging space (CP QUERY ALLOC PAGE) as the sum of the Linux guests' virtual storage sizes. • Having at least one paging volume per Linux guest is a great thing. If the Linux guest is using synchronous page faults, exactly one volume per Linux guest will be enough. If the guest is using asynchronous page faults, more than one per

	guest might be appropriate; one per active Linux application would be more like it.
Have you installed and configured a VM performance monitor on your system? z/VM 4.3 and older comes with the Real Time Monitor (VMRTM) and the Performance Reporting Facility (VMPRF) pre-installed. z/M 4.4 also comes with the new Performance Toolkit for VM™ pre-installed.	These performance monitors are fee based optional products, but should be considered mandatory for monitoring/tuning your z/VM system. They can be used to identify system constraints, bottlenecks and runaway guest machines as well as for capacity planning. The Performance Toolkit can monitor the performance of Linux guests as well as z/VM.

Chapter 5: Investigate the Linux Setup

The most important thing to check in the Linux setup is the amount of virtual memory you have assigned the Linux guest from VM. Make sure your guest has enough to do what it needs, but no more.

Check This	Explanation
Do you have a swap file defined for Linux? We recommend that you implement the swap file as a VM Virtual Disk (VDISK), sized at 15-20% of the size of the Linux guest's virtual memory.	You should always define a swap file. If Linux needs to swap and cannot it will degrade performance severely. VDISK consumes real memory, so if your system is memory-constrained you should consider using a minidisk (with Minidisk Cache turned off) or a dedicated DASD, instead of VDISK.
Is VM paging excessively? Make sure all your Linux guests have been given only the minimum virtual memory they need.	VM paging is caused by VM not having enough real memory to hold all the active Linux guests. Some guests must be paged out to make room for others. Assigning more virtual memory to a Linux guest than is needed will cause Linux to allocate that memory to buffers and cache, which VM must page in and out to run that guest. A "correctly sized" Linux guest has just enough virtual memory to run without swapping excessively.
Is Linux swapping excessively? Use the vmstat command and look at the "si" and "so" columns. Activity in these columns indicates swapping.	Your Linux virtual memory should be large enough to prevent "excessive" swapping, which uses up lots of CPU. But don't make your Linux memory so large that lots of memory is left unused. A correctly sized Linux guest has enough memory to prevent swapping, except for times of heaviest use (approx. 30 minutes per day), when a little swapping is OK. Use the memory estimating spreadsheet to help calculate the proper memory needed for a Linux guest running WebSphere. See "" on page for info on how to get the spreadsheet.

Are idle Linux guests consuming more CPU than you expect? By default Linux "wakes up" 100 times per second. These timer pops are part of the way that it determines if it has work to do. These timer pops have three system impacts when run as a guest of VM: 1. Processor time is consumed. The majority of the cycles are used by VM's control program in virtualizing the interrupt. 2. Parts of the interrupt reflection must run on VM's master processor. Because the Linux guest wakes up so often, VM considers it to always be active. Therefore, it will not be dropped from the dispatch list and the storage management routines will tend not to steal pages from these *idle* guests.	Do not define more virtual CPUs for a guest than are needed. The timer pops are on a per virtual processor basis (not per virtual machine). The correct number of virtual processors to define to a Linux guest can be determined by monitoring guest CPU utilization with a z/VM performance tool such as the Performance Toolkit. Never define more virtual processors to a guest machine than there are real processors. To start out: 1. Define the number of virtual processors for each guest to be equal to the number of real processors 2. Then reduce the number of virtual processors for each guest to be only what that guest can consume. VM monitoring data will tell you how much CPU each Linux guest consumes while running your application. A patch has been developed for the 2.4 kernel that avoids the timer pops. This is the so called "Jiffie" patch. Some zSeries distributions (such as SLES8) incorporate this function, but you must enable the feature. To enable it: echo '0' > /proc/sys/kernel/hz_timer But that file is reset at each boot, so to make the change permanent add the above statement to /etc/init.d/boot.proc, which is executed at each boot. Measurements show that this feature can lower the total processor time for an idle Linux guest by approximately 78%.
Does your WebSphere application do lots of DASD I/O (directly or via DB2) in a random access pattern? SUSE LINUX with kernels at 2.4 (This includes SLES7 and SLES8) and above use a default cache mode of 'read sequential mode'. Consider changing this setting to 'normal mode'.	While 'read sequential mode' is good for many applications we found that 'normal mode' was much better for workloads with a random DASD access pattern. This parameter can only be changed in SLES8, not in SLES7. Future releases of SLES will change the parameter. This parameter and how to set it are documented on the Linux developer works Web site under the "Tuning hints and tips", "DASD I/O Enterprise Storage Server" section, at **ibm.com** /developerworks/oss/linux390/perf/index.shtml Contact Don Corbett for more information.
Are there network response issues? This can be affected by an invalid name server.	Ensure that /etc/resolv.conf points to a valid and available DNS, and that the entries in /etc/hosts are also valid.
Are there DASD response issues? There are two Linux technologies that can help:	When software is using PAV, it can issue multiple

• Parallel Access Volume (PAV) • Logical Volume Manager (LVM) with Striping	channel programs to a volume, allowing simultaneous access to the logical volume by multiple users or processes. Reads can be satisfied simultaneously, as well as writes to different domains. Writes to the same domain still have to be serialized to maintain data integrity. Linux for zSeries can use PAV if all of the following apply: • Linux runs as a z/VM guest. • The volume resides on an IBM TotalStorage® Enterprise Storage Server® (ESS). • Linux uses the logical volume manager (LVM). LVM and Striping is setup entirely within Linux and available on all the 2.4 kernel distributions and newer. This topic goes beyond the scope of this paper, but you can find out more information from Carlos Ordonez (see last page).

Chapter 6: Investigate the WebSphere Setup

One of the most important, and simplest things to check in WebSphere is the size of the JVM Heap. The Heap is the amount of memory available in the JVM for the use of the application. There is one JVM heap per WebSphere application server.

Check This	**Explanation**
Are WebSphere and Java using lots of CPU but the application performance is slow? Then the JVM heap maximum size may be too small for this application. Use either: • Tivoli® Performance Viewer • Verbose garbage collection trace. to determine the amount of JVM memory the application uses while under load. Then adjust the WebSphere Application Server's JVM heap maximum to be 35% above this value.	If the JVM heap maximum is too small, the JVM will do excessive garbage collecting, which will stop the application and use all available CPU. Obviously very bad for performance. We prefer using the Verbose GC trace to determine how the application is using JVM memory. Turn it on by checking the box in the WAS Admin GUI, application server, JVM properties. The trace is written to the SystemOut.log of the application server, and is quite easy to read and understand. The point here is to leave a cushion above the normal high-water-mark of the JVM heap usage. A 35% cushion will give the JVM heap some room to grow if conditions push the application beyond the stress levels you used to test it.
Is your application waiting for a connection to the database? Use WebSphere's "Connection manager diagnostic function" or "Performance Advisors" to help diagnose problems in connection pool sizes, and recommend new values. Or you	"Maximum pool size" is the maximum number of physical connections to the database in the pool. This is the sum for all the data sources in this application server.

can use the Tivoli Performance Viewer to see the number of connections your application uses. You don't want to just arbitrarily increase this number, since each connection in the pool requires management overhead in WebSphere.	Coordinate this setting with the maximum number of connections your database is configured for. For example: If you have two application servers serving the same application (a cluster of 2) and each has a maximum pool setting of 10, then the database should be configured to handle at least 20 connections. Set the database connection pool size using the WAS Admin GUI: Application Server -> Database Connection Settings -> Maximum pool size.
Have you enabled these WebSphere caching mechanisms? • **Dynamic cache** – JSP, servlet and portlet caching with application server – **Edge Side Include** -- caches pages and page fragments at the HTTP server. – **Command Cache** – programmatic cache based on Command Pattern within application server • **Prepared Statement Cache** – caching of BMP based SQL statements within application server • **Fast Response Cache Accelerator (FRCA)** – static content cache on HTTP Server. • **Edge Cache** – static and dynamic cache at edge of network using Edge Components • **JNDI name cache** – WebSphere by default caches all objects looked up from WebSphere JNDI Namespaces. The cache can be programmatically controlled.	**Dynamic cache (Dynacache).** Servlet and JSP results can be cached in WebSphere. Your application must be coded to use this cache. Dynacache is primarily an in-memory cache, so after turning on the cache the JVM heap size will be larger to accommodate the cached objects. "Edge Side Include" will cache Servlet and JSP results at the plug-in that resides on the Web server machine. **Prepared Statement Cache** Specifies the number of free statements that are cached per connection. A prepared statement is a precompiled SQL statement that is stored in a prepared statement object. This object is used to efficiently execute the given SQL statement multiple times. The cache size can be tuned using WebSphere's performance advisors. **FRCA**. This cache is currently not supported in the IBM HTTP server for Linux. **WebSphere Edge Component's Caching Proxy** can cache static and dynamic web pages. When used in conjunction with WebSphere Application Server, the "Dynamic Caching plug-in" of the Caching Proxy has the ability to cache, serve, and invalidate dynamic content from JSPs and servlets in a WebSphere Application. **JNDI Name Cache.** The WebSphere Application Server JNDI implementation uses caching to reduce the number of remote lookup calls to the name server. For most cases, use the default cache setting.
Are static pages served from WebSphere? Instead use the HTTP server to serve static pages.	Static pages are best served via an HTTP server. There is much more overhead used by going into WebSphere to have it just return static HTML.

Chapter 7: Investigate the DB2 and DB2 Connect Setup

I've included DB2 and DB2 Connect in this paper since almost every WebSphere application uses DB2 either on Linux or on z/OS® to get some data, often the majority of its data. So it is key that WebSphere's connections to DB2 are fast and efficient.

Check This	**Explanation**
Have Database Snapshots been taken while the application is running in stress mode? The most likely cause of poor application performance is in the application's DB calls.	DB optimization is key to ensure a good-performing application. Take snapshots while you are stress testing the application. Look for lock contention problems, and table scans.
Are you using DB2 Connect V7 or earlier? There are significant performance improvements in DB2 Connect V8. If you are spending a significant amount of CPU time in DB2 Connect then consider migrating to DB2 Connect V8.	DB2 Connect V8 can connect to DB2 V7 databases. Review the DB2 V8 Migration Planning documentation for DB2 connectivity restrictions with DB2 clients and databases at different release levels. In addition review the WebSphere support matrix for restrictions between WebSphere release levels and DB2 V8.
Are you using DB2 on z/OS and getting large result sets from your queries? Then you should be using HiperSockets™ to communicate to z/OS	Large result sets can benefit from HiperSockets and the ability to use large MTU sizes.
Is the WebSphere application in a different building, or city, from the databases that it's using? If so move the application closer to the database.	If there are many network hops, or distance, between the application and it's database, this can result is significant degradation of performance. Traceroute or ping often does not show these delays due to the small amount of data they send.
Check the number of concurrent connections that DB2 is configured to accept. You may need to increase this number to match the size of the database connection pool in WebSphere.	DB2's maximum connections must be larger than the connection pool sizes for all the WebSphere application servers (and anything else) that will be connecting to this DB2.
Do you suspect that your DB2 settings may not be optimal for this WebSphere application?	Consider using the "DB2 Performance Expert" to help analyze the usage patterns of your applications and recommend buffer pool settings.

Chapter 8: Investigate the Application

Finally we get to the application. Often we find that the biggest impact on performance can be made by tuning the application, or changing its design. But we want to first eliminate any possible "environmental" causes in VM, Linux, or WebSphere. So since you are here we'll assume you have read the above sections and eliminated any possible problems in those areas. If the performance is still less than you expect, then it's time to start looking closely at the application.

Check This	Explanation
What are the transactions that the application performs? What steps does the application perform for each transaction? You can start with a high-level understanding of these, just enough to be able to describe the flow that happens on each transaction. This level of knowledge of the application is crucial for finding performance problems in the design of the application. There may also be problems with the coding of specific functions in the application. To find such problems you'll need much more detailed information on what the application does.	Often we will find that the application makes numerous database calls, or retrieves large pieces of data, or executes a complex flow through multiple machines. Any of these design issues can be the root cause of the application's poor performance. To find more detailed information on what the application does, you'll need to use either • a profiling tool such as Sitraka JProbe to profile a transaction. • a monitoring tool such as Tivoli or Wily Introscope to monitor the performance of the application during test. Such tools will tell you how long the application spends doing it's steps. You might be surprised to find that the application is waiting several seconds for a DB call to complete.
Does the application use SSL to encrypt Web pages? If these pages are a significant portion of the pages served by the application, then you should consider using zSeries hardware acceleration for this encryption. Without hardware acceleration you will see high CPU utilization in the HTTP server.	SSL handshakes and encryption use a lot of CPU. This work is done in the HTTP Server, so the extra CPU will appear in the Linux guest running the HTTP server. The IBM HTTP Server uses GSkit to interface with the zSeries PCI cards. As of this writing, this function works but is not yet supported by IBM.
Does the application not perform well, yet CPU utilization is low? Investigate the application's use of "synchronized" methods. These should be eliminated or minimized.	A method defined as "synchronized" cannot be executed while any other thread running a synchronized method is being executed in the JVM. These should be used sparingly.
Does the application repeatedly call methods that will return the same value? Examples are: • Doing expensive lookups with every request instead of caching the looked up value. (Examples are JNDI lookups, or LDAP calls.)	There are many opportunities in the application to cache values that are expensive to look up. Even for values that are cheap to lookup, caching can have a benefit if the lookup is done often.

A highly used method calls another method to lookup a value that changes infrequently. The call may cost little, but if it's executed thousands of times it's cost multiplies.	
Is the application using lots of JVM heap memory? One cause can be storing large amounts of session data.	Use session data sparingly. Session objects should be as small as possible.
Is the application persisting session data? Hardening session data can reduce performance, so use this only when really needed.	Persisting session data to DB2 is the most costly for performance. Consider just using Session Affinity instead. This will ensure that the session data is saved for the life of the session.
Does the application, or supporting code, do extensive logging or tracing? If so then turn off this feature, or minimize the logging.	Log or trace functions are often turned on during function testing and sometimes forgotten when stress testing begins.
Did the application create their own wait-loops or other functions that Java already has functions for? Replace these with existing Java functions if possible.	Such roll-your-own functions can burn a lot of CPU in the application, while Java already has efficient functions that do the same things.

We encourage all customers to use some kind of monitoring tool for your WebSphere applications. There are really two types of monitoring:

- *Health monitoring. Are all the components of your application up and active? Are user transactions being completed within the time expected? When something is going wrong in one of those components, can you pinpoint the problem?*
- *Performance monitoring. Are user transactions being completed within the time expected? Are there bottlenecks in your code? Is it performing as you would expect? Can you determine why?*

Performance monitoring should be started during load/stress test. Health monitoring should be started when the application goes into production. Tools to consider are WebSphere Studio Application Monitor, Tivoli Monitoring for Application Performance, and Wily Introscope, among others.

Chapter 9: Performance Testing

Very often the customer or IBM will want to do some sort of load/stress testing to determine that the application and its environment can handle the expected traffic of a live user load. This is often the time when performance problems are discovered, when the testing shows that the application is not reaching the customer's expected rates of throughput.

Performance testing must be a carefully planned, repeatable process in order to learn anything with certainty from the tests. To create such a repeatable environment, you will need:

- *A clear set of exit criteria. These are the goals that the customer wants to achieve during the performance test. Usually this is measured in terms of transactions per second, but can also include the maximum CPU to be utilized, number of concurrent users, or other goals.*
- *A test tool that can run a controlled, repeatable test. Such as Mercury LoadRunner, Segue Silk Performer, or IBM WebSphere Studio Workload Simulator. All these tools can record "scripts" that consist of a series of user interactions with the Web site. The tools can then replay these scripts simulating vast numbers of users, and record statistics about the test run.*

  The test scripts must accurately reflect how real users will interact with the web site. For example: if you are testing an e-commerce site, you must simulate users who are browsing the catalog, buying, and searching.
- *Controlled conditions for the Linux guests. Since the Linux guests you are testing may share CPU with other Linux guests, and the LPAR you are running in may be sharing CPU with another LPAR, it is crucial that you control how other LPARs and guests affect the guests you are testing.*
  - *If other LPARs are sharing CPU with the LPAR you are testing, then you'll probably have to test off-shift when the other LPAR has a steady load, or no load at all.*
  - *If other guests are sharing CPU with the guests you are testing then, if possible, shut them down during the testing. If this is not possible then you may have to test off-shift.*
- *Detailed records of each test you run. Record a baseline test run, with no changes. Then record each successive test run after making one configuration change only. Do not make multiple changes between test runs, because you'll never know which change had a beneficial effect.*
- *We also encourage the use of some kind of monitoring tool for your WebSphere applications. You need a monitoring tool to answer questions like: Are user transactions being completed within the time expected? Are there bottlenecks in your code? Is it performing as you would expect? Can you determine why not?*

  Tools to consider are WebSphere Studio Application Monitor, Tivoli Monitoring for Application Performance, and Wily Introscope, among others.

Once you have all this in place then begin running your load/stress tests. Each test should run for at least 15 minutes. It's tempting to run for just 5 minutes or so, but in such a short run, any small hiccup in performance can be magnified. It takes some time for WebSphere to populate caches, or to compile JSPs, or to create connection pools, or for the JVM to allocate memory. In a very short test these can affect the throughput. So run the test for at least 15 minutes to get a better sense of the real-world performance you can expect.

Once you have made a number of such tests runs and performance is approaching your goals, run the tests for longer periods. Your final load/stress tests should run for several hours to prove the stability of the application and it's environment.

Gather the following performance data during each run. If you are working with an IBM performance analysis team, then send them the data after each run.

- *Transactions (Web pages) per second, and response time, as reported by the test tool.*
- *The VM CPU utilization for each Linux guest. Use VM Monitor, VMPRF, Performance Toolkit, Velocity ESALPS, or some other tool to get this data. Set the monitoring interval to one minute. The report should include the period of time of the test, plus 5 minutes before and after. This report should also include DASD I/O activity.*
- *vmstat or sar data taken at 30 second intervals. On each Linux system involved in the test. (We don't recommend using the Linux "top" command during performance testing, because it uses too much CPU.)*
- *JVM Verbose garbage collection trace for each WebSphere application server.*

Record the following data for each test run:

- *What changed from the previous test run*
- *Transactions per second as reported by the test tool*

CPU utilization for each Linux guest involved in the test, as reported by VM.

The following table is an example of how to record the data from a performance test.

			CPU Utilization	
Date	**What Changed from the Current Baseline**	**Thruput**	**Where**	**%**
3/10/04 Run 1	Baseline	14.1 trans/sec	HTTP WAS DB2 Connect Total	8 47 9 64
3/11/04 Run 2	Changed JVM for WebSphere application: Original Heap Size: Initial 128 MB, Maximum 768 MB New Heap Size: Initial 256 MB, Maximum 256 MB	19.0 trans/sec	HTTP WAS DB2 Connect Total	9 63 12 84
3/12/04 Run 3	Changed JVM for WebSphere application: Original Heap Size: Initial 128 MB, Maximum 768 MB New Heap Size: Initial 256 MB, Maximum 256 MB	37.7 trans/sec	HTTP WAS DB2 Connect	15 130 21

	Added another IFL to VM to make a total of 3		Total	166

Chapter 10: I did all this and performance is still not what I expect. What now?

So, you've gone through all the suggestions above and still the performance of the WebSphere application doesn't meet expectations. What do you do now?

1. *Have your IBM rep contact Techline or TechXpress to get further help.*
2. *Techline and TechXpress may contact one of the following lab people if they cannot solve the problem, but your fastest response will be to go through Techline or TechXpress.*

Contact	Skills
zSeries New Technology Center Mgr: Kathy Grabarits	Linux for zSeries, and zSeries-specific skills in WebSphere on Linux and z/OS.
Linux Technical Support Mgr: Tim Hayford	Consulting and on-site support for Linux and WebSphere.
WebSphere SWAT Team Mgr: Katie Barrett	WebSphere on z/OS and Linux for zSeries
Advanced Technical Support Mgr: John Sutera	z/VM performance and communications
WebSphere Technical Support Mgr: Ken Ausich	WebSphere on all distributed platforms
Linux Scalability Center Team Lead: Chris Panetta	Performance measurements of Linux on zSeries, and WebSphere
zSeries Performance Team Lead: Carl Parris	Performance measurements of Linux on zSeries, WebSphere, and Java.
Linux Integration Center Team Lead: Jeff Richardson	Installing various Linux software and making them work together.
DB2 and DB2 Connect performance Team Lead: Steve Rees	DB2 Universal Database™ and DB2 Connect performance on all distributed platforms.
WebSphere Commerce Advanced Technical Support Mgr: Shawn O'Quinn	WebSphere Commerce consulting and services on all distributed platforms.
WebSphere Portal performance Team Lead: Mark Alkins	WebSphere Portal for multiplatforms performance.

Appendix 1: Further Reading

Source	Where to get it
"*How to – Architect VM and Linux for WebSphere V5*" is a paper that discusses the principles behind many of the recommendations in this paper, and is a good general introduction to running Linux under VM	Contact Steve Wehr
Memory Estimating Spreadsheet	Contact Steve Wehr
WebSphere support matrix	**ibm.com** /software/webservers/appserv/doc/latest/prereq.html
Performance Considerations for Linux Guests	vm.**ibm.com**/perf/tips/linuxper.html
"Best Practices for implementing Linux on zSeries"	Contact Jason Hortman. Also includes a great appendix listing Redbooks and papers relevant to Linux on zSeries.
White paper *"Development Best Practices for Performance and Scalability"* gives excellent tips on how to code Java applications for good performance under WebSphere.	Contact Harvey Gunther in Raleigh
Redbook *"Enabling High Availability e-business on zSeries".* Contains a good discussion about session affinity and Highly available Linux configurations, but it uses WebSphere on z/OS.	Search the Redbooks site for SG24-6850.
There are many good Redbooks on WebSphere and Linux. Too many to list, and more being added all the time.	Search the Redbooks site for "Linux and zSeries".
WebSphere's *"Tuning parameter hot list"* Contains some good tips on tuning WebSphere.	WebSphere InfoCenter. Look for the page entitled: "Tuning parameter hot list"
How Parallel Access Volumes (PAV) is supported under z/VM.	vm.**ibm.com**/techinfo/pav.html
How to use PAV with Logical Volume Manager (LVM).	software.**ibm.com**/developerworks/opensource/linux390/docu/lx24pav00.pdf
How to implement LVM	oss.software.**ibm.com**/developerworks/opensource/linux390/perf/tuning_how_lvm.shtml#begin
Solutions Assurance checklist for WAS on Linux for zSeries	w3.**ibm.com**/support/assure/assur30i.nsf/WebIndex/SA470
Solutions Assurance checklist for Linux for zSeries	w3.**ibm.com**/support/assure/assur30i.nsf/WebIndex/SA428

Appendix 2: Contact Information

Steve Wehr
zSeries New Technology Center, Poughkeepsie NY
swehr@us.ibm.com

Jon von Wolfersdorf
Advanced Technical Support, Endicott NY
wolff@us.ibm.com

Carlos Ordonez
zSeries New Technology Center, Poughkeepsie NY
carloso@us.ibm.com

Don Corbett
Linux Scalability Center, Poughkeepsie NY
dpcorbet@us.ibm.com

GM13-0635-00