Memory Sizing for WebSphere Applications on System z Linux

Steve Wehr
System z New Technology Center
Poughkeepsie

Getting the best TCO from a large virtualized server like System z Linux requires paying attention to the memory used by applications and z/VM Linux guests. This presentation gives a step-by-step approach to help you easily and accurately size the memory needed by large applications such as WebSphere, to help you achieve your TCO goal.

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

e-business logo
IBM*
IBM System z
IBM logo*
VM/ESA*
WebSphere
z/OS*
z/VM*
System z Linux

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Intel is a registered trademark of the Intel Corporation in the United States, other countries or both.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Penguin (Tux) compliments of Larry Ewing.
Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

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.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

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

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

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

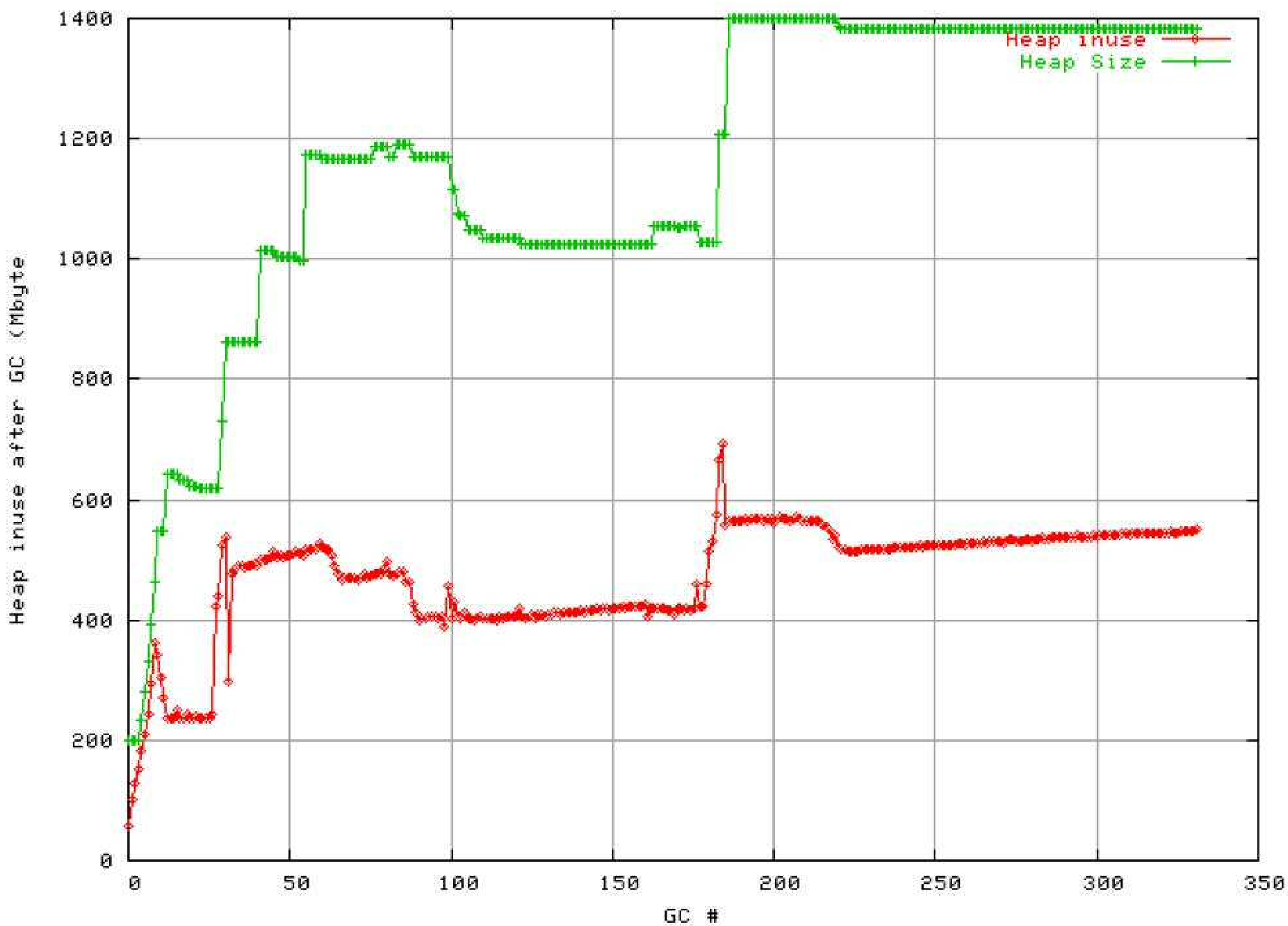
- How much memory do your current apps use?
- 32-bit vs. 64-bit
- Basics of z/VM and Linux memory management
- Creating “properly sized” Linux guests.
- Overcommitting memory
- Tools to monitor memory usage.

How much memory do your current Java apps use??

- A difficult question for many customers:
  - "We've got intel servers with 4GB of memory installed, and we use 2GB JVM heaps."
  - Memory is cheap and plentiful on these servers and there is often little need to conserve it.
- You need to measure memory usage in the JVM heap.
  1. Run "verbose GC trace" or use Tivoli Performance Viewer
  2. Observe the max memory used (not allocated) by the JVM.
  3. Make several observations, at peak periods.
- Max memory used should be about 55-75% of the max heap size.
- Even if using –Xms < –Xmx, the JVM doesn't (always) adjust the max heap size to maintain this optimal occupancy rate.

How much memory do your current apps use??

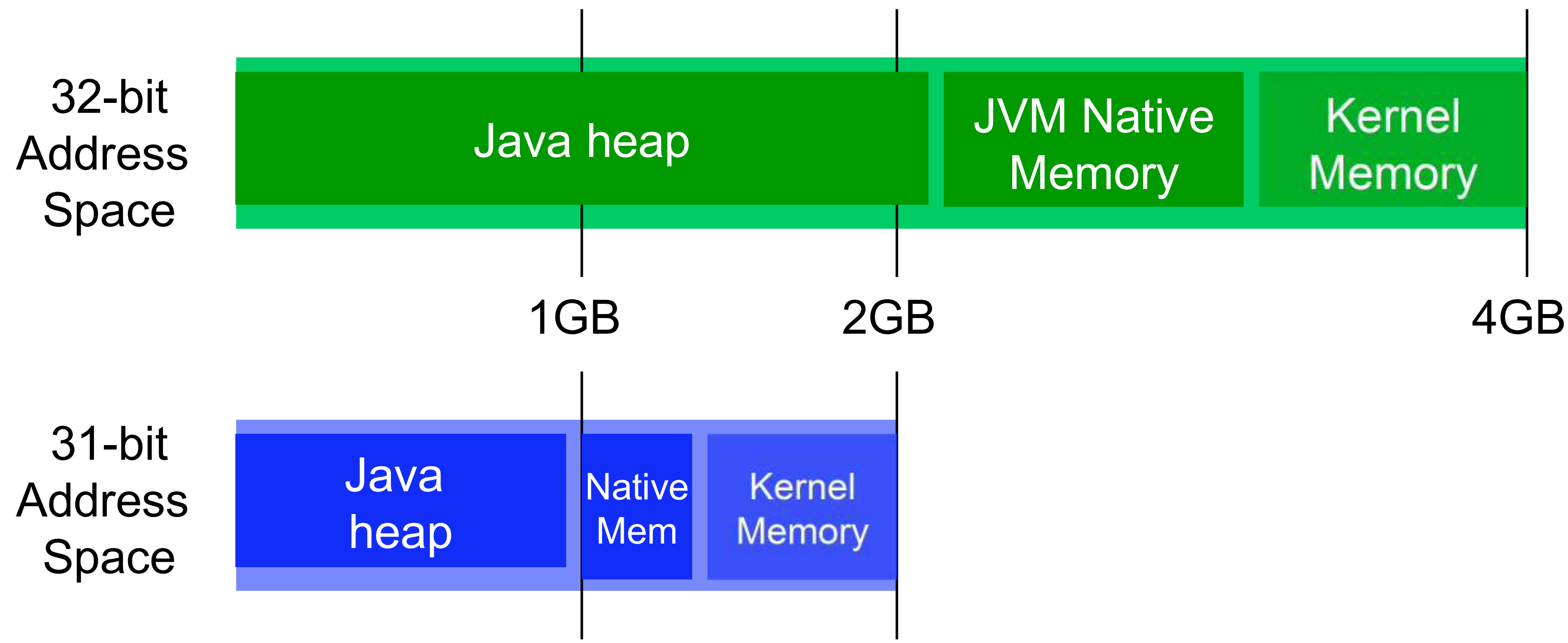
1. Take a Verbose GC trace, measure “Heap in use after GC”.
2. In this graph, the app is using about 550MB, max heap is 1400MB. So max memory is 40% of the heap size.
3. Heap size is probably* too large. Reduce to 846MB, to make 550MB 65% of the max heap.
4. Re stress-test the app
5. Reduce guest size.



Graph of “Heap in use after GC” from verboseGC trace.

Migrating from 32-bit WebSphere to System z

- System z hardware runs in 31-bit mode, not 32-bit.
- Address spaces are 2GB instead of 4GB.
- Therefore, Java Heaps are half the size.
- This is the only major issue when migrating Java applications to System z Linux.



Moving from 32-bit WebSphere to System z

- RHEL and SLES11 use a new kernel memory layout (flex_mmap) that increases user-space memory.
  - Heaps can now be about 1200MB.
  - Exact max heap size depends on the native memory used by your application.
  - SLES10 is limited to 850MB heaps, unless you use the mapped_base function, which will stretch that to about 1100MB.

Do you need larger heaps for your application?

- Run 64-bit Java -- Heap Sizes are unlimited.

But 64-bit Java has Drawbacks

- Memory Usage
  - Your heap can grow 20-60% larger. All your object addresses double in size.
  - But JVM 1.6 reduces this increase by compressing 64-bit references to 32-bit. So most heaps do not increase in size if using JVM 1.6.
  - **Native memory will increase 40-100%.** Native memory is the memory used by the java process over and above the heap. The “compressed references” function in JVM 1.6 does not reduce this.
- Performance
  - The larger memory footprint causes a slight performance degradation.
  - 2-5% in most apps using JVM 1.6.
- New Java Libraries
  - Your application will be using 64-bit-specific java libraries (new to your app) and should be tested more thoroughly.

Our Recommendation for System z Linux

- Use 31-bit WebSphere/Java when you can
  - Smaller memory footprint for the JVM.
  - A bit better performance.
- Use 64-bit WebSphere/Java only when…
  - You need larger heaps.
  - You want a standard WAS/Java version for all applications.
  - Your application uses 64-bit java features.

Basics of z/VM Memory Management

- z/VM has these main storage areas:
  - Central storage (real memory, Cstore)
  - Expanded storage (real memory, Xstore)
  - Page volumes (disk storage).
- In order to run a guest, z/VM must bring the pages it requires into central storage.
- z/VM paging uses two storage areas:
  - Xstore
  - Paging disks.
- We recommend real memory be allocated
  - 80% Cstore / 20% Xstore
  - But no more than about 4GB Xstore (except in special circumstances)
- We want z/VM to handle virtual memory paging, not Linux.
  z/VM has 40 years experience doing this.

Basics of Linux Memory Management

- The Linux OS was built to run on hardware that had slow I/O subsystems. Therefore it buffers all I/O to “Buffer” and “Cache” areas.
  - Linux will consume all available memory as buffer/cache.
  - This raises it’s memory footprint in z/VM.
  - There is no way to prevent the Linux kernel from using available memory as buffer/cache.
- So, we **define Linux guests to have the smallest memory size possible**, so that you can run many of them while buying the least real memory.
- Use z/VM VDISKs for swap disks. Allocate no more than 15% of the guest size to the swap file size. This small swap file ensures that Linux will not be able to heavily swap. Remember, we want z/VM to do the heavy lifting when it comes to paging virtual memory.

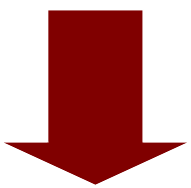
Sizing the memory needed for a Linux guest

1. What applications (and other agents) will run on this guest?
2. For each application, determine it's memory footprint.
   For WebSphere Applications:
   1. Size the JVM heap required for each app.
   2. Estimate: Add 300MB for native memory (31-bit) or 400MB (64-bit).
      Note that native memory can be much larger depending on the application. DMGR native memory can be larger depending on the applications being deployed and the cell configuration.
3. Add it all up
   1. Add the memory required for each application.
   2. Estimate: If using WAS ND, add 500MB for the node agent.
   3. Estimate: Add 150MB for the Linux kernel, and buffer/cache.
   4. Any other agents running on this guest? What memory do they need?
   5. = Total Linux guest virtual memory size.
4. Install your application and run it.
5. Use the following commands to monitor and adjust.

Good Memory Allocation for Linux

Remember Your Goal:

Give Linux as little memory as possible, without causing it to swap.

	Action	Why
	Increase the guest memory if Linux is swapping. Use the vmstat or sar commands to determine swap/paging rate.	Swapping uses CPU for virtual memory management that is more efficiently used by z/VM.
	Decrease the guest memory if there is too much free space. Use the free command to determine free memory.	Too much free space leads to a bloated memory footprint, and less memory for z/VM to use for other guests.

Good Memory Allocation for Linux
Increase the guest memory if Linux is swapping

- Use the vmstat or sar commands output to determine swap/paging rate.
- Use the “si” and “so” columns of the vmstat command output to determine if Linux is swapping. As long as these numbers stay in single digits or zero, you are good.

```
lnx1: /home/testuser>vmstat 15
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 309892  28760  32404 518864    4    2     9    87    2    1  5  3 89  3  0
 0  0 309892  26424  32452 520052    0    0    77   145 1129  193  2  1 97  0  0
 0  0 309892  18240  32636 520544    0    0    34   187 1303  587  7  3 88  3  0
 0  0 309892  31912  32556 509372    0    0    12   258 1139  232 12  2 86  1  0
 0  0 309892  32108  32592 512732    0    0   223    93 1066  189  1  1 96  2  0
```

- Increase the guest memory if the swap rate exceeds 100 pages/sec for more than 30 minutes.
- Ignore free command and others that display the “used” swap.

Good Memory Allocation for Linux
Decrease the guest memory if there is too much free space

- Run the Linux “free” command at times of peak load.
  (Remember to add buffer/cache to get an accurate count of free space).

```
lnx1: /home/testuser>free
                 total       used       free     shared    buffers     cached
Mem:           1027540    1005928      21612          0      90772     493756
-/+ buffers/cache:         421400     606140
Swap:          1052248     102400     949848
```

- Reduce the guest memory size if there is > 300MB free space.
  (This is just my opinion, you can be more or less aggressive)
- Ignore the swap file sizes.

Good Memory Allocation for Linux

Why it's important

Bad

z/VM
LPAR

8GB
Real

Linux guests are over-sized. z/VM Working Set Size (WSS) = 1.7GB. 4 x 1.7 = 6.8GB required.

Guests oversized at 2GB each.

Can fit: 4 Linux guests

Same work

Good

z/VM
LPAR

8GB
Real

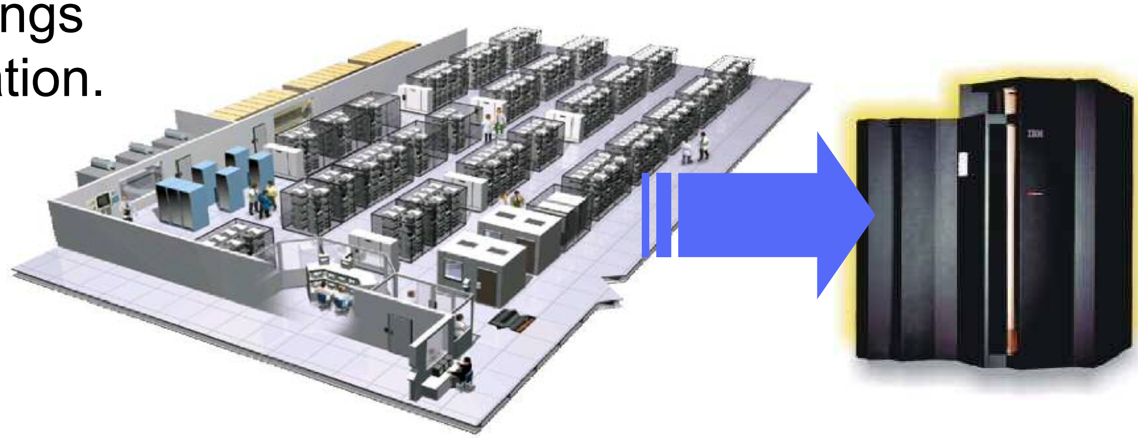
Linux guests are correctly sized. z/VM Working Set Size (WSS) = 850MB. 9 x 0.85 = 7.65GB required.

Guests right-sized at 1GB each.

Can fit: 9 Linux guests

“Overcommitting” Memory

- Now that each guest is sized correctly, start planning for adding many guests to z/VM.
- If each guest is defined at 1GB (virtual memory) and we have 50 of them, then do you need to buy 50GB of real memory for the LPAR?
- “Overcommitting” simply means that there is more virtual memory (guest memory) than real. But the key is “How much more”.
- The more memory you can overcommit, the less real memory you need to buy, and the greater the TCO for Linux on z.
- **Your Goal:** Reduce costs by overcommitting as much CPU and memory as possible, without impacting performance.
- This is one of the most difficult things to plan for in large-scale virtualization.
- You may be able to achieve:
  - Prod: 1.5 virtual : 1 real
  - Test: 1.5:1 – 3:1
  - Dev: 3:1 – 5:1

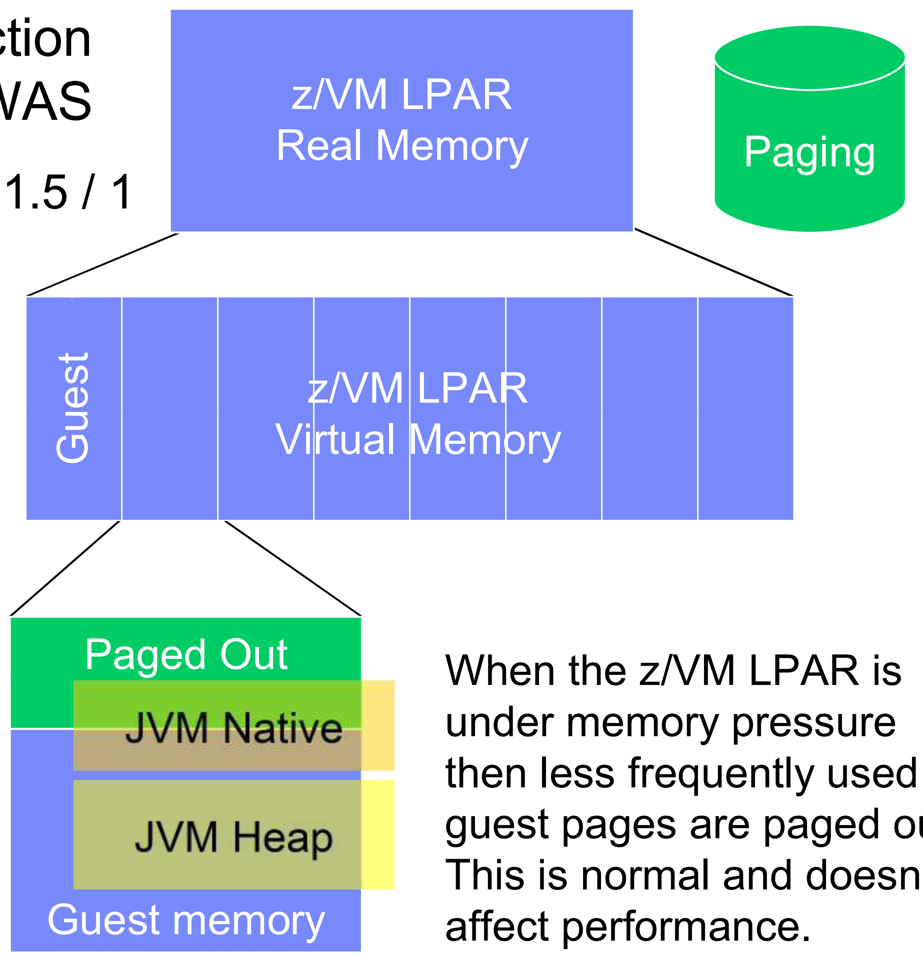


Overcommitting Memory

The total virtual memory of all the started guests is larger than the physical memory assigned to the z/VM LPAR.

Average Production LPAR running WAS

Overcommit ratio = 1.5 / 1



When the z/VM LPAR is under memory pressure then less frequently used guest pages are paged out. This is normal and doesn't affect performance.

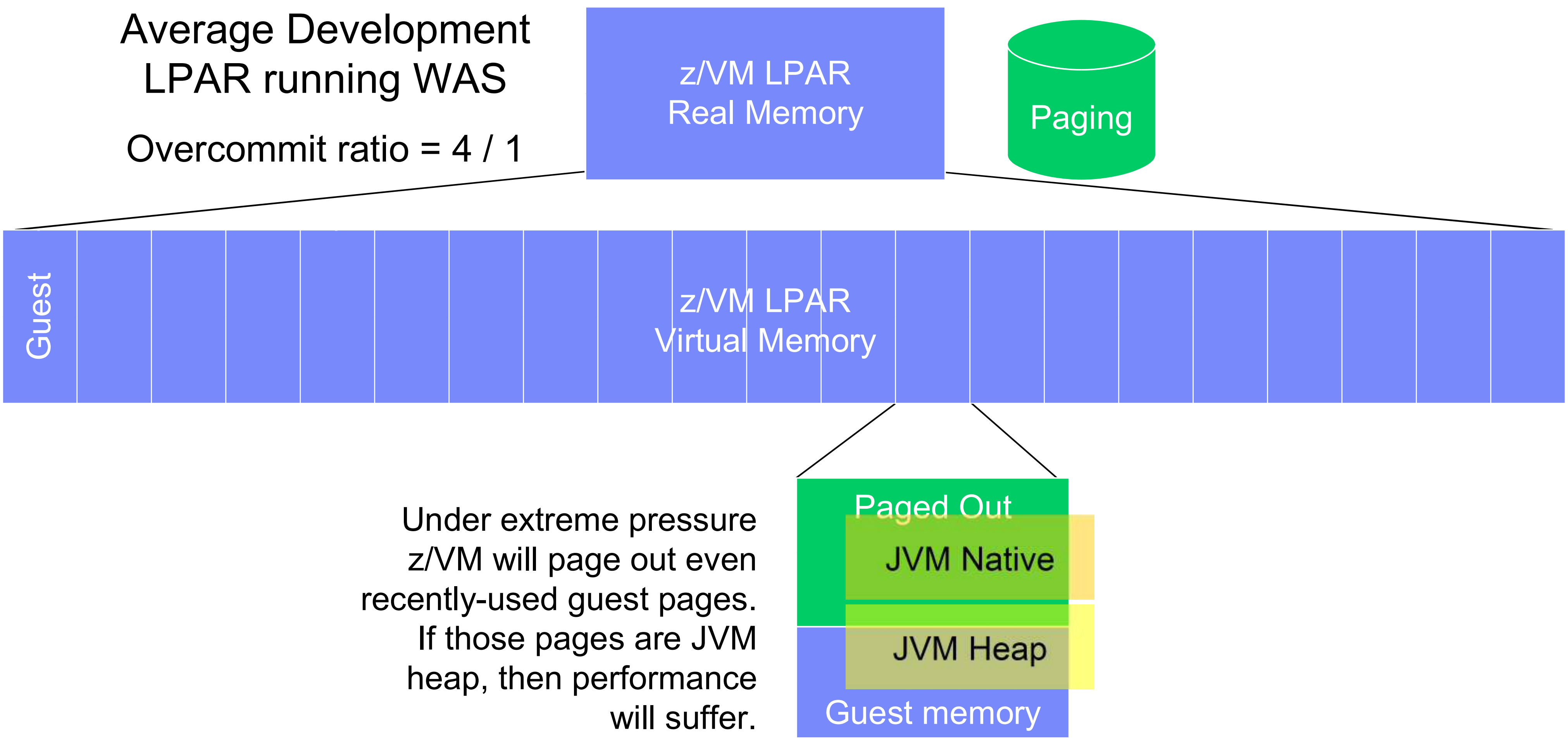
Performance not affected

Overcommitting Memory

The total virtual memory of all the started guests is larger than the physical memory assigned to the z/VM LPAR.

Average Development LPAR running WAS

Overcommit ratio = 4 / 1



Under extreme pressure z/VM will page out even recently-used guest pages. If those pages are JVM heap, then performance will suffer.

Performance Impacted

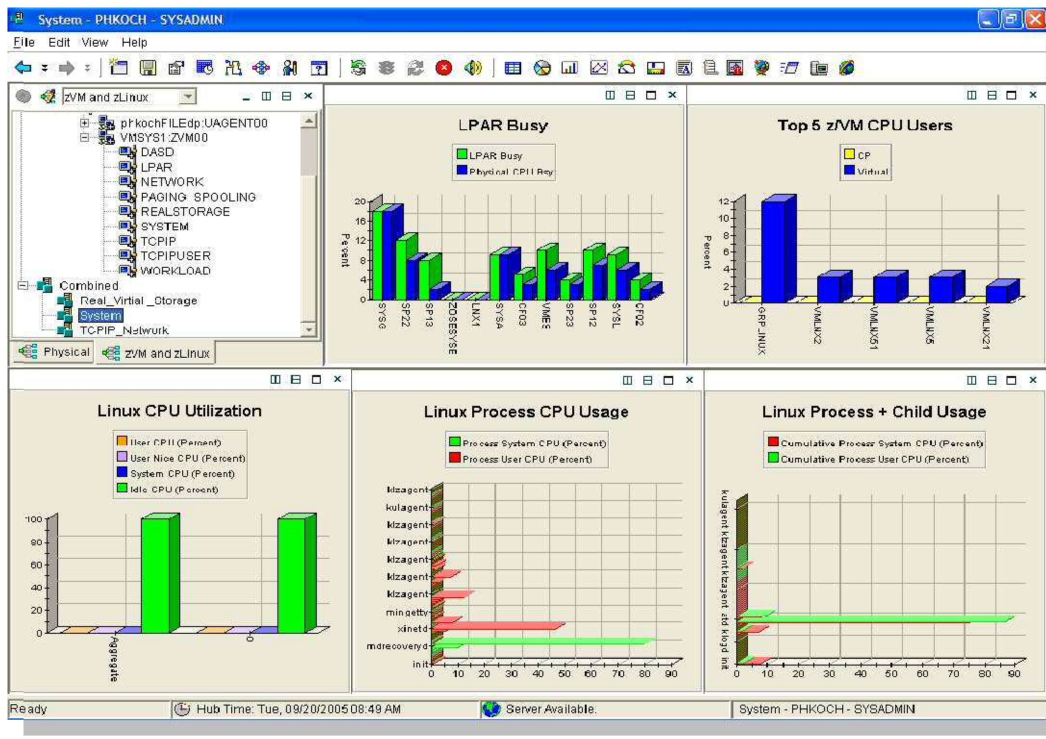
Monitoring / Debugging the Environment

You must be able to monitor

1. What resources the LPAR is getting.
2. What resources the guests are using
3. Performance of Linux on each guest
4. Performance of Java applications inside WebSphere

- Choose a z/VM monitor (1,2,3)
  - Omegamon XE displays LPAR, z/VM, and Linux perf data.
  - So do z/VM Performance Toolkit, or Velocity Software.
- Choose a WebSphere application monitor (4)
  - Tivoli Monitoring for WebSphere
  - Wily Introscope

Omegamon XE

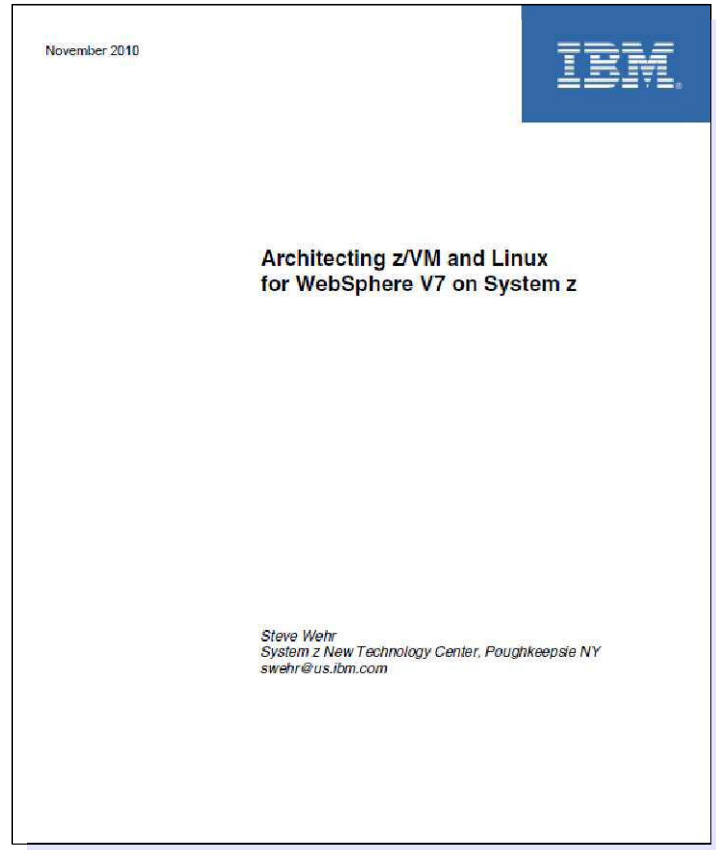


Good Memory Configuration is Crucial for Good Performance

- The Environment
  - Make sure **z/VM is not excessively paging.** Make sure you have enough real memory and that Linux guests are small.
  - Make sure **Linux is not swapping** (much). Make sure Linux virtual memory is large enough.
  - Make sure you have the CPU Power to drive the workload. Get a processor Sizing.
- WebSphere Tuning
  - Tune the JVM Heap size so that it is as small as possible, but still large enough so that the **JVM is not doing excessive GCs**.
  - Follow WAS doc for other small tuning tweaks for Linux.
- The Application
  - The application has the **largest impact on performance**.
  - **Inefficient code** that nobody noticed on dedicated hardware **will be noticed** on virtualized hardware.
  - Ask IBM for an application review for performance.

Getting more Information – Recommended Reading

- Linux Library website:
  - A multitude of helpful papers.
  - On the web at: www.ibm.com/servers/eserver/zseries/os/linux/library/
- System z Linux education
  - http://www.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=page&c=a0000631
- Step-by-step instructions for creating Linux virtual servers:
  - Virtualization Cookbook for SLES11 SP1. Redbook SG24-7931-00
  - Virtualization Cookbook for RHEL6. Redbook SG24-7932-00
- *Architecting z/VM and Linux for WebSphere.*
  A companion paper to this presentation.
  - Introduction to Memory configuration for z/VM, Linux, and WebSphere.
  - http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101803


November 2010

Architecting z/VM and Linux
for WebSphere V7 on System z

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

The End