August 2011

Linux on System z
Methods to pause a z/VM guest: Optimize the resource utilization of idling servers

Authors:

Linux end-to-end Performance team:

Dr. Juergen Doelle, David Sadler

Before using this information and the product it supports, read the information in Notices.

Table of Contents

About this publication

This paper provides results for testing the process of suspending and resuming a Linux® instance on IBM System z® processors.

Introduction

This paper analyzes the behavior of five Linux z/VM® guests running a WebSphere® workload and five Linux guests running a relational database (RDB) workload when they are paused and restarted. The Linux suspend and resume function is compared to the z/VM CP STOP and BEGIN commands.

When running in a virtualized environment, the requirements on the behavior of software might change significantly due to the sharing of hardware resources. For example, does it matter what a server on a dedicated box is doing when it has no user requests to process? The system is there, and if it uses some more CPU cycles or not might not be important. If the server runs virtualized, especially in an overcommitted environment where more virtual resources are defined than are physically available, the expectation is that unused systems do not allocate physical resources. But the observed results can be very different. Many servers are polling so often for new work that they never appear idle from the Hypervisor's point of view. Depending on the number of these guests, this idle activity might sum up to a significant quantity of CPU capacity. Also, the effect of these guests on the allocation of memory pages is important because pages from active guests should stay in real storage and should not be moved out to the paging space.

Therefore z/VM requires a certain period of real inactivity until a guest becomes dormant. Only then are their memory pages preferred for migration to the paging devices, and the real memory pages become available for other guests. This means that when a guest is idling from the user perspective, but not from the Hypervisor perspective, because the guest is looking so frequently for work, the guest's pages are still competing with other genuinely active guests for real storage pages. This contention might limit the level of possible memory overcommitment.

For example, in case the usage of the guests is organized in shifts, it is important that the guests from one shift become inactive (dormant for z/VM) when the work is done and their physical resources can become available for the guests from another shift. On the other hand, it is very uncomfortable and time consuming to shutdown and close everything at the end of the work and restart it on the next day. The solution for this issue, discussed in this paper, is to freeze the processing of the guest from the Hypervisor view, and then continue processing when required. There are two ways to freeze the guest processing: with the Linux suspend and resume mechanism, or with the z/VM CP STOP and BEGIN mechanism.

The Linux suspend and resume support provides a controlled stop of a running Linux on IBM System z instance, and allows operations to continue later. When the Linux instance is suspended, a memory image is written to a swap partition and all interfaces are shut down. The resume process uses this data to continue the Linux instance from where it left off when it was suspended. A suspended Linux instance does not require memory or processor cycles. This can provide better performance, resource utilization, and power savings.

Summary

This paper describes how to pause a running Linux on IBM System z instance and later resume operations. An analysis is done of the amount of time needed for suspend and resume, and the system resources required.

There are two methods available:

- The Linux suspend and resume mechanism, which hibernates the Linux guest and restarts it
- The z/VM CP STOP and BEGIN commands, which stop and restart the virtual CPUs

The Linux suspend mechanism requires SUSE Linux Enterprise Server 11 (SLES 11) or Red Hat Enterprise Linux 6. The CP STOP and BEGIN mechanism is a general z/VM feature.

For smaller guests (768 MB), depending on the amount of memory used, it takes 10 - 30 seconds to suspend the guest and approximately 20 seconds to resume the guest. With larger guests the resume time increases, but slower than the guest size increases. This means that in a slightly different environment (with a 5.2 GB guest running six WebSphere Application Servers), it requires 30 seconds until the first Web site could be delivered when resumed. For comparison, only starting the same six WebSphere Application Servers in that guest serially took approximately 80 seconds. The CP STOP and BEGIN commands take effect immediately.

The test paused a set of guests, previously warmed up to ensure that the memory pages are really used, and moved the workload to a set of standby guests, which where idling before. After a while, the paused guests are restarted and the workload is directed again to these systems. The z/VM is sized so that activating the idling guests should create a memory shortage and force paging. The expected behavior was that the paused guests become dormant and their pages are moved to the paging space.

Both mechanisms put the guests in the dormant state, but only with the Linux suspend and resume method is it permanent. With the CP STOP and BEGIN commands, the guests become scheduled (active) more or less frequently depending on the type of servers running (WebSphere ND, relational database). The reason is that this mechanism only stops the CPUs but leaves the virtual interfaces such as network or FICON® adapters active. In this case, when an external event causes an interrupt, z/VM reacts on behalf of that guest.

Both mechanisms operate that in the case of memory pressure the pages from the paused guests are preferred to be moved to the paging space, which makes both mechanisms suitable features to run with higher levels of memory overcommitment. The z/VM CP STOP and BEGIN mechanism is available in all z/VM versions and takes effect immediately, but it keeps everything in main memory. In case the guest was logged off or there is an IPL of the LPAR, the memory content for the guest is lost. The Linux suspend and resume mechanism is a controlled shutdown of the interfaces, and it writes the memory image to disk. This feature makes the state of the guest robust as a really stopped system, but provides a very fast restore.

Pausing production systems is *not recommended*, but this technique is very suitable for test or development systems.

Test environment

The test environment used for the process of suspending and resuming a Linux instance on IBM System z consisted of IBM system z hardware and various software running on Linux.

Hardware

Table 1 lists the hardware used to test suspending and resuming a Linux instance.

Table 1. Hardware used for Linux suspend and resume testing

System	Operating System	Number of systems	Number of processors	Memory in GB	Network	Architecture
z/VM LPAR	z/VM 5.4	1	10	10 GB + 2 GB	One 1Gb OSA	z10™ LPAR
Linux server	SUSE SLES 11	6 or 11	1 or 2	768 MB (WAS) 650 MB (RDB)	Shared OSA	z/VM guest

The workload generating clients use two 4-way systems running Linux:

- One system for creating the DayTrader workload
- One system for creating the SwingBench workload

Software

Table 2 lists the software used to test suspending and resuming a Linux instance.

Table 2. Software used for Linux suspend and resume testing

Product	Version and release	Comments
DayTrader	2.0 ee5, r343	Use with normal database
WebSphere ND	7.1 (31-bit)	• Latest fix pack • The 31-bit version is used because a system of less than 2 GB does not require 64-bit
DB2® for Linux	9.7	Latest fix pack
z/VM	5.4 RSU 0902	Order number UM97540
Linux	SUSE SLES 11 SP1	Latest update SP1
SwingBench	2.3.0.422	Use with read only database

Environment

Figure 1 illustrates the environment used to test the suspend and resume of a Linux instance.

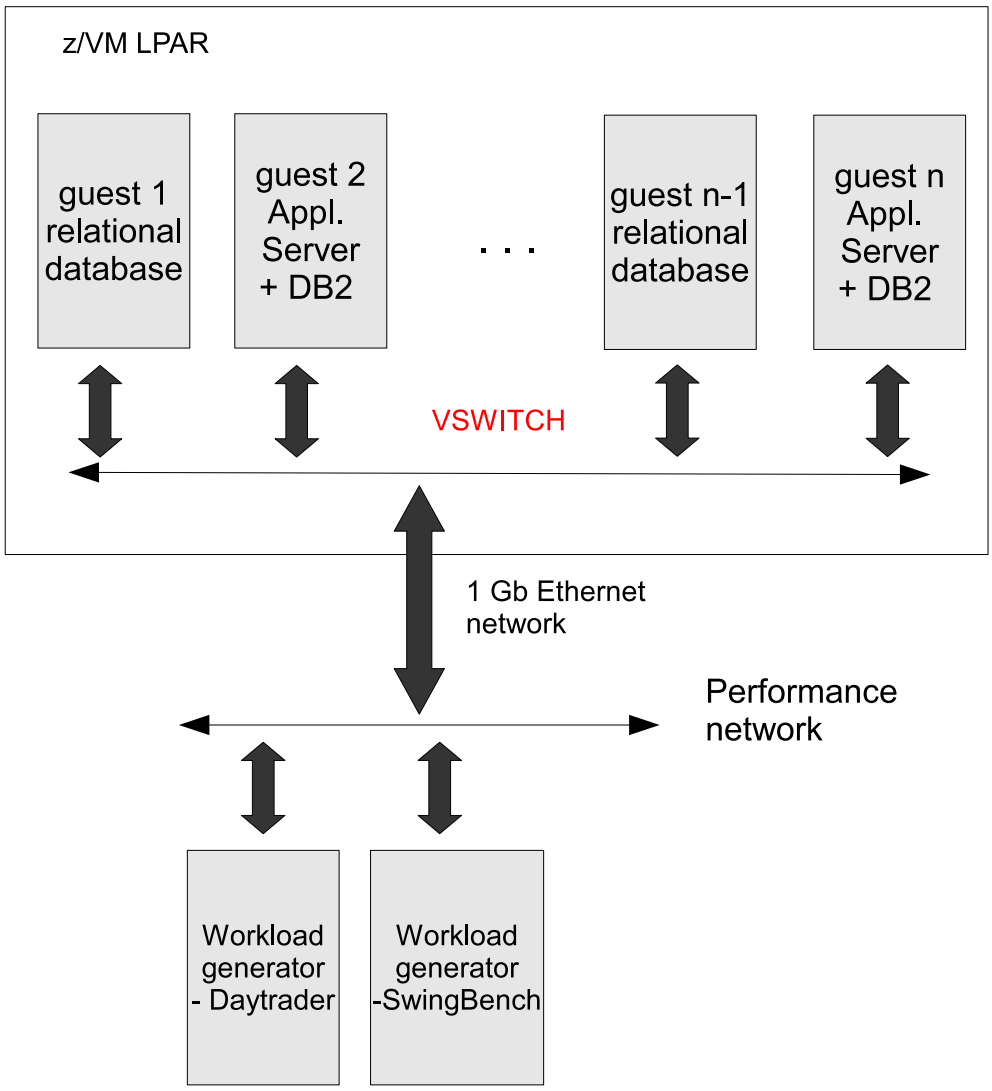

Figure 1. Environment for the Linux suspend and resume versus CP STOP and BEGIN test, using a VSWITCH

Network

The network used to perform Linux suspend and resume testing consisted of an external interface using a one Gb Ethernet card, and internal connectivity provided by a VSWITCH.

System setup

This section describes the steps necessary to set up the Linux systems for suspend and resume testing. This section also provides information about the swap partition needed to suspend and resume a Linux instance.

These parameters are specified in the 'parameters' option of the /etc/zipl.conf configuration file. All the following commands and changes are run by the root user.

Setting up Linux for suspend and resume

This section describes the Linux kernel parameters used for setting up suspend and resume support.

The kernel parameter used to configure support for suspend and resume is:

resume=<device_node> [no_console_suspend] [noresume]

Where:

resume=<device_node>
: Specifies the standard device node of the swap partition with the data that is required for resuming the Linux instance.

[no_console_suspend]
: Prevents the Linux consoles from being suspended early in the suspend process. Without this parameter, you cannot see the kernel messages that are issued by the suspend process.

[noresume]
: Restarts the Linux kernel without resuming a previously suspended Linux instance. Use this parameter to circumvent the resume process, for example, if the data written by the previous suspend process is damaged.

As an example, to use a partition named /dev/dasda2 as the swap partition, and prevent the Linux consoles from being suspended early in the suspend process, use this Linux kernel parameter:

```
resume=/dev/dasda2 no_console_suspend
```

This is a sample `zipl.conf` file:

```
[defaultboot]
defaultmenu = menu
[SLES11_SP1]
    image = /boot/image-2.6.32.12-0.7-default
    target = /boot/zipl
    ramdisk = /boot/initrd-2.6.32.12-0.7-default,0x2000000
    parameters = "root=/dev/disk/by-path/ccw-0.0.7335-part1 mem=32768M
TERM=dumb init=/linuxrc resume=/dev/disk/by-path/ccw-0.0.791d-part1
no_console_suspend"
[Failsafe]
    image = /boot/image-2.6.32.12-0.7-default
    target = /boot/zipl
    ramdisk = /boot/initrd-2.6.32.12-0.7-default,0x2000000
    parameters = "root=/dev/disk/by-path/ccw-0.0.7335-part1 TERM=dumb x11failsafe
noresume"


:menu
    default = 1
    prompt = 0
```

```
target = /boot/zipl
timeout = 10
```

Setting up a swap partition

During the suspend process, Linux writes data to a swap partition. This data is required later to resume Linux. Set up a swap partition that is at least as large as the available LPAR memory, or the memory of the z/VM guest virtual machine.

Do not use this swap partition for any other operating system that might run in the LPAR or z/VM guest virtual machine while the Linux instance is suspended. You cannot suspend a Linux instance while most of the memory and most of the swap space are in use.

If there is insufficient remaining swap space to hold the data for resuming the Linux instance, suspending the Linux instance fails. To assure sufficient swap space you might have to configure two swap partitions: one partition for regular swapping, and another for suspending the Linux instance. Configure the swap partition for suspending the Linux instance with a lower priority than the regular swap partition.

Use the `pri=` parameter to specify the swap partitions in the `/etc/fstab` file with different priorities. See the **swapon** man page for details.

The following example shows two swap partitions with different priorities:

- `/dev/dasdb1 swap swap pri=-1 0 0`
- `/dev/dasdc1 swap swap pri=-2 0 0`

Suspending a Linux instance

Issue the following command to suspend a Linux instance:

```
echo disk > /sys/power/state
```

On the Linux console you might see progress messages until the console itself is suspended. Most of these messages require log level 7 or higher to be displayed. You cannot see the progress messages if you suspend the Linux instance from an ssh session.

To change the log level of console messages, issue this command:

```
klogd  -c n
```

where *n* is the log level value. The default is 7.

Issue this command to change the log level to 7:

```
/sbin/klogd -c 7
```

Resuming a suspended Linux instance

Restart Linux to resume a suspended Linux instance. Use the same Linux kernel, initial RAM disk, and kernel parameters that you used to first boot the suspended Linux instance. You must reestablish any terminal session for Hypervisor Console (HVC) terminal devices and for terminals provided by the iucvtty program. You also must reestablish all ssh sessions that have timed out while the Linux instance was suspended. If resuming the Linux instance fails, restart Linux again with the noresume kernel parameter. The restart process then ignores the data that was written to the swap partition and starts Linux without resuming the suspended instance.

More information about the use of suspend and resume can be found in the *Device Drivers, Features and Commands* manual found at this location:

http://public.dhe.ibm.com/software/dw/linux390/docu/lk36dd08.pdf

z/VM CP STOP and BEGIN commands

The Linux guest can issue the CP command CP STOP ALL to place the Linux guest in a stopped state. All virtual CPUs are stopped and execution is halted. The state of attached devices might remain undefined.

With the CP STOP command, you can place the processors into a soft stop state or a hard stop state. This test used the hard stop option.

CPU ALL

Specifies the processor or processors that you want to place in hard stop. You can specify a single processor address, a list of processor addresses delimited by blanks, a non-wrapping pair of addresses separated by a hyphen (-) with no blanks, or the keyword ALL. If you omit this operand, all of your virtual processors are placed into a soft stop state.

Important: If you enter the CP STOP command without operands, all of your virtual processors are placed into soft stop.

To restart the Linux guest, the guest issues the command CP BEGIN. This will reactivate the stopped virtual CPUs and execution will begin again.

The CP BEGIN command can be issued from a SECUSER user that has been set for the Linux guest or from the CP administration guests MAINT or OPERATOR. A class G user can issue these command. Any z/VM guest can issue CP STOP and CP BEGIN for itself.

More information about the use of the BEGIN and STOP commands can be found in the *CP Commands* manual found at this location:

http://publib.boulder.ibm.com/infocenter/zvm/v6r1/index.jsp?topic=/com.ibm.zvm.v610.hcpb7/toc.htm

Linux guest setup

In this test, 11 Linux guests were used. Five guest were set up as a relational database server (RDB) and driven by the SwingBench workload. Six guests were setup for WebSphere; one of the WebSphere guests acted as the Network Deployment Manager node and the other five guests were set up as WebSphere Application Server (WAS) nodes. Each WebSphere Application Server node had it's own local DB2 instance that contained the DayTrader database.

Workload description

The use of the applications DayTrader and SwingBench is described in detail, as well as the methodology for suspend and resume testing.

DayTrader

DayTrader is a benchmark application that simulates an online stock trading system.

DayTrader was originally developed by IBM with the name "Trade Performance Benchmark Sample". In 2005, DayTrader was donated by IBM to the Apache Geronimo community. A typical user of DayTrader performs these tasks:

- Login the DayTrader user interface
- View the user's stock portfolio
- Find current stock prices
- Buy or sell shares of stock
- Logout from the DayTrader user interface

With the aid of a Web-based load driver such as Rational Performance Tester or Apache JMeter, the real-world workload provided by DayTrader can be used to measure and compare the performance of Java Platform, Enterprise Edition (Java EE) application servers offered by a variety of vendors. In addition to the full workload, DayTrader also contains a set of primitives used for functional and performance testing of various Java EE components and common design patterns

DayTrader requires a read/write data base, and was used to provide a transactional workload on WebSphere and DB2.

The test scenarios used DayTrader to generate a load on the WebSphere guests.

For more information about DayTrader, see:

https://cwiki.apache.org/GMOxDOC20/daytrader.html

SwingBench

SwingBench is a Java™-based application that generates a workload used to run stress tests on a relational database. SwingBench comes with several predefined benchmarks, but also has the capability for the user to define their own customized benchmarks.

SwingBench has these components:

- The load generator
- The coordinator
- The cluster overview

The test scenarios described in this paper used SwingBench to generate a workload on the relational database, and record statistics for later analysis. In the test scenarios, SwingBench was used for read-only transactions.

For more information about SwingBench, see:

http://dominicgiles.com/swingbench.html

Linux Suspend and resume test methodology

The test is run as follows:

1. A set of three guests running WAS and DB2, and three RDB guests run at constant load without z/VM paging to warm up the environment before the first guests are suspended.
    - One of each kind is never suspended or stopped to create a base load.
    - Two of each kind are selected as system of interest to be suspended.
    - Two guests from each kind are kept inactive as standby systems.
2. Suspend the systems of interest and start the standby servers and workload to cause memory pressure. The increasing load on the standby guests increases their need for memory. The suspended guests should become dormant and their pages should be moved to the paging space to release physical memory for the increasing need from the standby guests. The total amount of LPAR memory was adjusted to accomplish this.

3. Reactivate the systems of interest and start the workload to get the pages moved back into main memory.

Figure 2 illustrates the processing for Test case 2 for WAS DB2 guests only. For more information about this test, see Test case 2: Pausing z/VM guests with memory pressure.

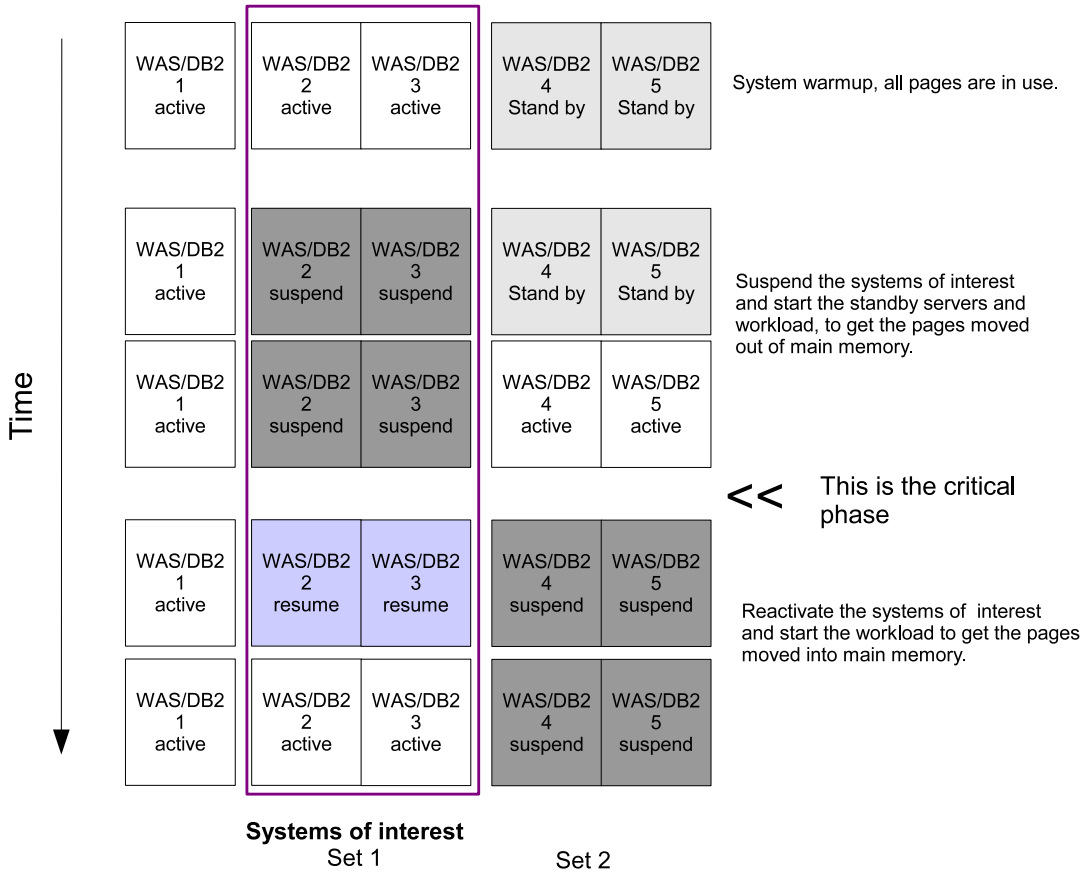

Figure 2. Test case 2: System and workload processing

Scenarios for suspend and resume testing

These test cases were run in order, each having different objectives, and building on the results obtained from the previous test.

For these tests, two methods are used to pause and restart the guests:

- The Linux suspend and resume method
- The z/VM method to stop and start virtual CPUs using the CP STOP and BEGIN commands

Test case 1: Base test

The objective of this test is to determine the amount of time needed to pause and restart one guest.

Test case 2: Pausing z/VM guests with memory pressure

The objective of this test is to study the suspend and resume of multiple warmed up but idling guests, which means all servers are started but no workload is issued against them. After the guests (called *systems of interest*) are suspended, workload on other idling guests are started to create memory pressure. The pages of the idling guests should move to the paging devices. When they are resumed, a recovery might happen on the servers, but a new workload should run without errors. Figure 2 illustrates this test case. This test should show that the memory pages of the paused guests are really eligible for other guests.

Additional test case: Time to resume a larger Linux guest

This test measured the time to resume a larger z/VM guest having varying memory sizes and numbers of JVMs.

Results

The results of various tests suspending and resuming a Linux instance on IBM System z are presented in this section.

Important: Terminology: In the following sections, a term of *n* + *m* refers to

- *n* number of WebSphere Application Server (WAS) DB2 guests, and
- *m* number of relational database (RDB) guests

Test case 1: Elapsed times to pause and restart a guest

This test records the amount of time needed to pause and restart a guest.

Note: The following time numbers are dependent on the guest memory size. In this scenario, the guests have 650 - 768 MB of memory.

Pause times

The *pause time* for a guest is the amount of time that elapses from when the suspend is started until the processing of the guest ends.

The Linux suspend measuring method is:

- Issue on Linux a CP QUERY TIME command to get the z/VM time (requires module vmcp to be loaded).
- Then suspend the guest by issuing the following command:

  echo disk >/sys/power/state # this causes the guest to switch to suspend

- Monitor messages on the z/VM console.

For the CP STOP mechanism, no time is gathered at the operating system level, because the command stops the CPUs immediately.

Table 3 lists the time in seconds until the guest (both WebSphere and RDB) is halted for both the Linux suspend and CP STOP command methods.

Table 3. Time in seconds until the guest is halted

Linux: suspend		z/VM: STOP command	
RDB guest	WebSphere guest	RDB guest	WebSphere guest
8	27	Immediately	Immediately

Restart times

The *restart time* for a guest is the amount of time elapsed from when the resume is started until the processing of the guest continues, for example when the WebSphere Application Server is again able to provide a Web page. This time is measured only for the WebSphere guest.

The restart time measuring method is:

- Suspend or stop the guest.
- Issue a **wget** command for a Web page provided by the application server.
- Resume or start the guest.

The processing time of the **wget** command is observed using the Linux **time** command.

Table 4 lists the time in seconds for the WebSphere guest to start up, measured from the **wget** command, Linux resume, and CP BEGIN command.

Table 4. Time in seconds for WebSphere guest to start up

wget command with an active guest	Linux: resume	z/VM: BEGIN command
0.6	21	3

The time used with the active guest is the normal execution time of that command. The increase for the other scenarios is the time needed to reactivate the guests until the WebSphere Application server is able to respond. Even though the stop and begin method is faster, it is recommended that the suspend and resume method be used because it provides a very controlled method for stopping the Linux operating system, compared to just freezing it when the CP STOP command is used.

Test case 2: Pausing z/VM guests with memory pressure

This test is to study the suspend and resume behavior of idling guests, which means that the servers are warmed up, but the workload is stopped before they are suspended.

The process is broken down into three phases:

1. Warmup
2. Systems of interest suspended
3. Systems of interest resumed

After the guests (called *Systems of interest*) are suspended, workload on other idling guests are started to create memory pressure. The paused guests should become dormant and their pages should be moved to the paging devices. The CPU load on the guests is monitored to demonstrate

the procedure of the test. The important result is the scheduling state of the guests and the location of the memory pages during the various test phases.

After the guests (System of interests) are suspended, workload on other idling guests are started to create memory pressure. The pages of the idling guests should be moved to the paging devices. When they are resumed, a recovery might happen on the servers, but a new workload should run without errors. Figure 2 illustrates this test case.

Guest usage

Table 5 shows the names of the guest systems used. They are divided into Types according to their function.

Table 5. Guest systems used

Type	System running	System name	System name
Systems of interest	RDB	LNX00102	LNX00104
	WebSphere	LNX00105	LNX00107
Standby	RDB	LNX00106	LNX00108
	WebSphere	LNX00109	LNX00111
Base load	RDB	LNX00100	
	WebSphere	LNX00103	
	Deployment manager	LNX00101	

CPU load analysis

The CPU load is analyzed for the system types: Systems of interest, standby systems, and base load systems. Idling guests are used.

For more information about the guest systems and their types, see Guest usage.

Figure 3 shows the real CPU utilization for the Systems of interest, measured in Integrated Facility for Linux (IFL), to demonstrate the workflow of the test.

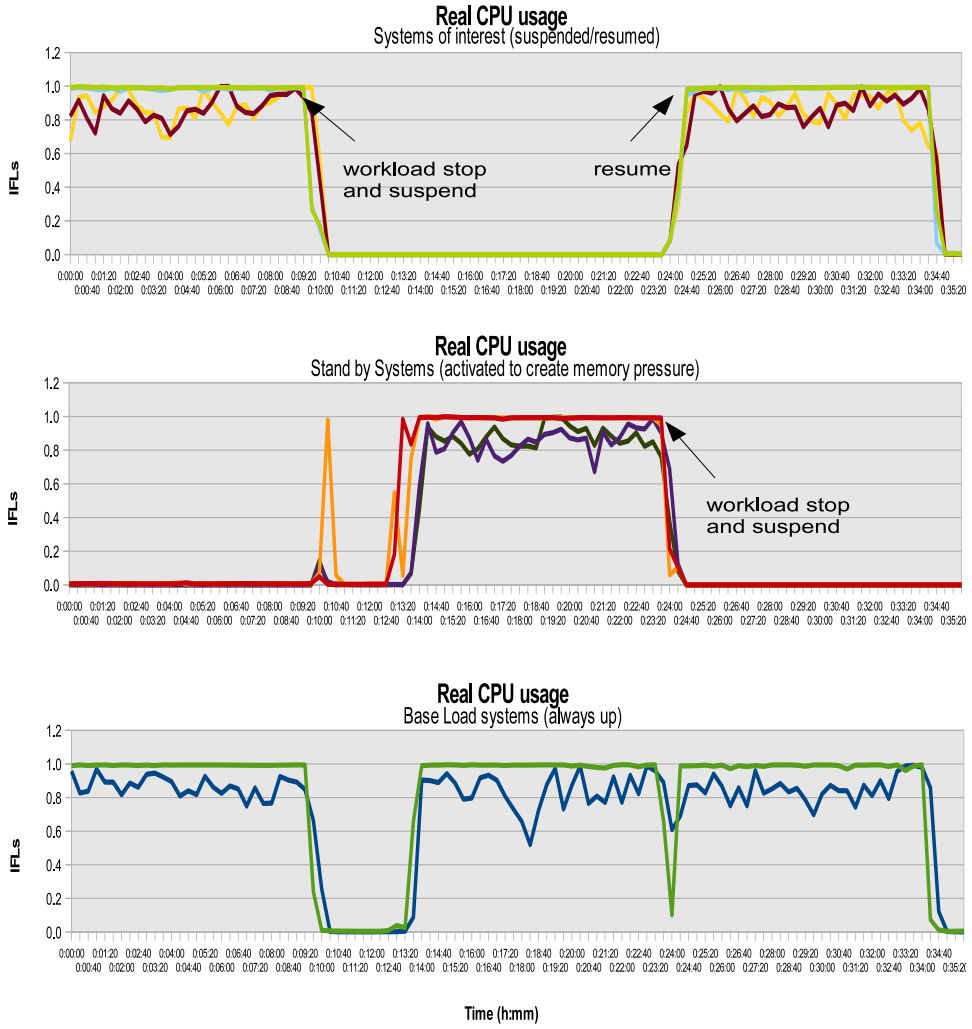

Figure 3. Real CPU utilization

Warmup phase:

- The Systems of interest guests are under load to get warmed up (each system has one virtual CPU that is fully utilized). The memory pages of these guests should reside in central storage.
- The standby systems are up, the servers are not started, but the node agent from the WebSphere cluster is started. The expectation is that these guests have only a very small amount of memory pages allocated.
- The base load systems are under workload. The memory pages of these guests should reside also in central storage. The z/VM is sized such that there is space for the base load and the Systems of interest.

End of warmup phase:

- Workload assigned to the Systems of interest is stopped and the systems are suspended.
- The CPU peak at the standby systems appear only on the WebSphere guests, and is related to the cluster recovery, because the two WebSphere Application Servers from the Systems of interest no longer interact.
- A workload pause is introduced to let the z/VM stabilize any page migration activity. This means especially that the Systems of interest should change to the dormant state.

Systems of interest suspended:

- The servers of the standby guests are started and workload is issued against them to create memory pressure. In that phase the pages from the System of interest should be migrated to the paging space and the quantity of pages for the standby guests in central storage should increase.

Systems of interest resumed:

- Now the workload against the standby systems is stopped and the guests are suspended. They should become dormant.
- The Systems of interest guests are resumed and workload is assigned to them, which means their memory pages should be migrated back from paging space to central storage, forcing the pages of the dormant standby system to become migrated to the paging space.
- The base load systems continue processing under workload. The memory pages of these guests should still reside in central storage.

Scheduling status of the guests (Systems of interest)

The scheduling status of the guests for the Systems of interest is analyzed using the z/VM Performance Toolkit, for both Linux suspend and resume and during z/VM CP STOP and BEGIN command processing.

For more information about the guest systems and their types, see Guest usage.

Linux suspend and resume

Figure 4 shows the scheduling status of the guests (Systems of interest) during Linux suspend and resume.

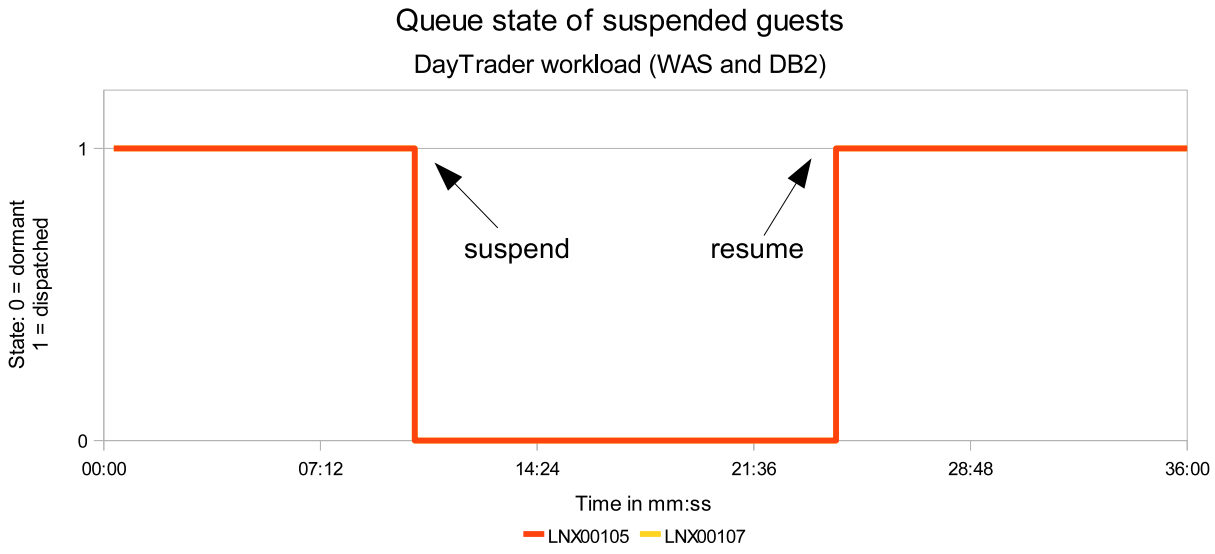

Figure 4. Scheduling status of Systems of interest during Linux suspend and resume

Observation

When the guest was suspended, the state changed immediately to dormant. This state is kept constant until the system is resumed. This behavior is the same for all suspended guests, regardless of the type of server.

Conclusion

The suspend method is a controlled shutdown of the Linux operating system, and includes all its interfaces. This absolutely disables all activity, and causes the guest to become reliably dormant.

z/VM CP STOP and BEGIN commands

Figure 5 shows the scheduling status of the guests (Systems of interest) during CP STOP and BEGIN, for the DayTrader workload.

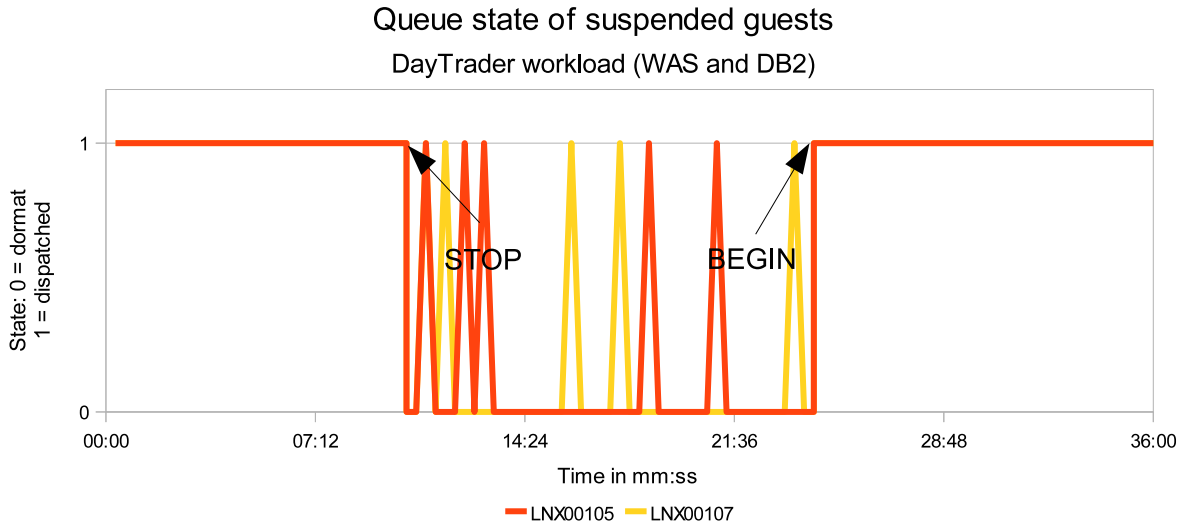

Figure 5. Scheduling status of Systems of interest during CP STOP and BEGIN

Observation

When using the z/VM CP STOP and BEGIN method to pause the guest, the result is slightly different than with the Linux suspend and resume method. The guest becomes mostly dormant, but the WebSphere and DB2 guest is more or less often scheduled for a short interval. This happens also sometimes with the RDB guest, but very seldom.

Conclusion

It seems that there are cases where the CP may do work on behalf of the virtual processor of the guest, even when the virtual CPUs are stopped. From the z/VM viewpoint, all interfaces from the guest (network or FICON channels) are still active and can generate interrupts when triggered from outside. One example is network requests sent to the guest, where VM does some work with the virtual network interface. The fact that this happens heavily for the WebSphere Application Server indicates that cluster mechanisms might be responsible.

Location of the guest memory pages (real storage or XSTOR)

The guest memory is analyzed to see how much is in real storage and how much has been migrated to XSTOR, at various phases of the test. At no point in time during the test are there any pages on DASD.

For more information about the guest systems and their types, see Guest usage.

In the figures that follow, these abbreviations are used:

W	A guest with WebSphere Application Server and DB2
O	A guest with an RDB database, maximum number of pages
SOI	Systems of interest

For example, SOIW means a Systems of interest guest with WebSphere Application Server and DB2.

The maximum number of pages for this test is as follows:

- Each RDB guest's virtual machine storage size is 650 MB or 166,400 pages.
- Each WebSphere Application server guest's virtual machine storage size is 768 MB or 196,608 pages.

Using Linux Suspend and Resume Method

This test should show where the pages of the guest reside when the idling guests are paused using the Linux suspend and resume method, and memory pressure is created by bringing the standby guests under load.

Middle of warmup phase

Figure 6 shows the location of the memory pages for the various Linux guests in the middle of the warmup phase.

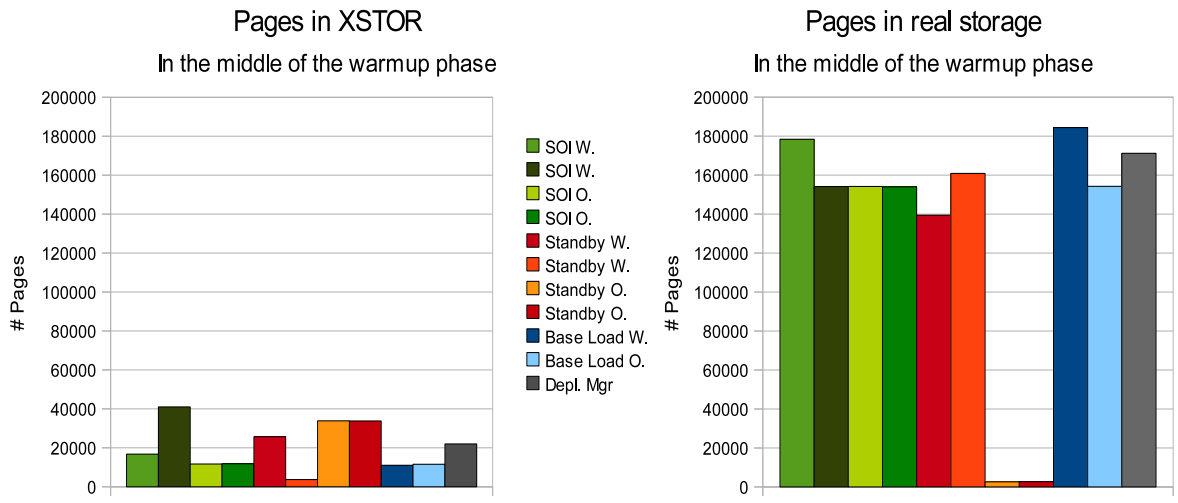

Figure 6. Location of memory pages for various guests during the warmup phase

Observation

All guest have some pages in expanded storage. The Standby systems with the WebSphere Application Server have an unexpectedly large number of pages already in real storage, while the idling RDB system has only a very small footprint in real memory.

Conclusion

There is a certain, but slight, level of memory pressure in the system (in average 1500 pages per second between main storage and expanded storage). On the Standby systems the servers

themselves are not started, but the node agent on the WebSphere system was up. This is a special kind of WebSphere Application Server required for the cluster. The large memory footprint of this idling system demonstrates exactly the situation that pausing the guest was anticipated to resolve.

Middle of suspend phase

Figure 7 shows the location of the memory pages for the various Linux guests in the middle of the suspend phase.

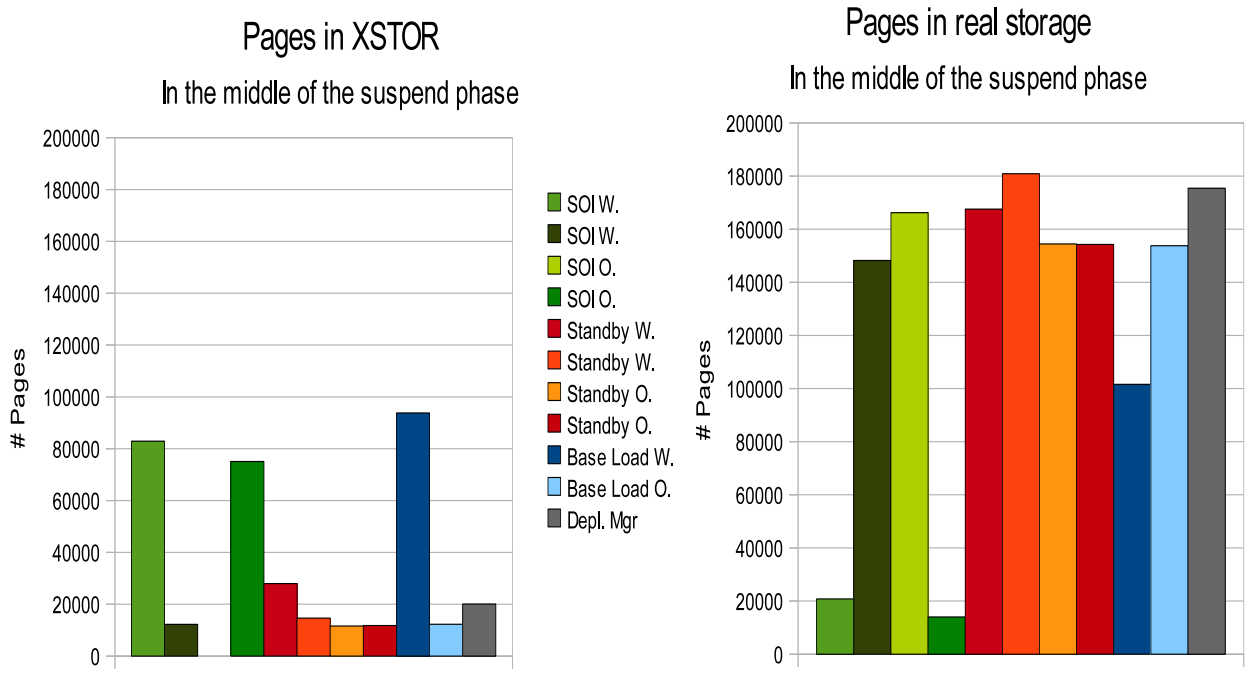

Figure 7. Location of memory pages for various guests during the suspend phase

Observations

The quantity of pages from the now active Standby systems is on the same level as the Systems of interest in the warmup phase. Most of the pages of two suspended guests (one WebSphere and one RDB guest) have been moved to XSTOR. Many pages of one of the Base load systems have been moved to XSTOR, and the number of pages in real storage has been reduced.

Conclusions

The amount of memory pages of the active Standby systems are now on the level as expected. To provide space for these pages, it was sufficient for z/VM to move only two of the suspended systems to XSTOR. An unexpected outcome is that one of the base load systems was also partially moved to the XSTOR. When totaling the pages in XSTOR and real storage, it seems that this guest has now pages that are in both places. There is a clear preference to take pages away from the suspended guests.

After resume

Figure 8 shows the location of the memory pages for the various Linux guests at the end phase after resume.

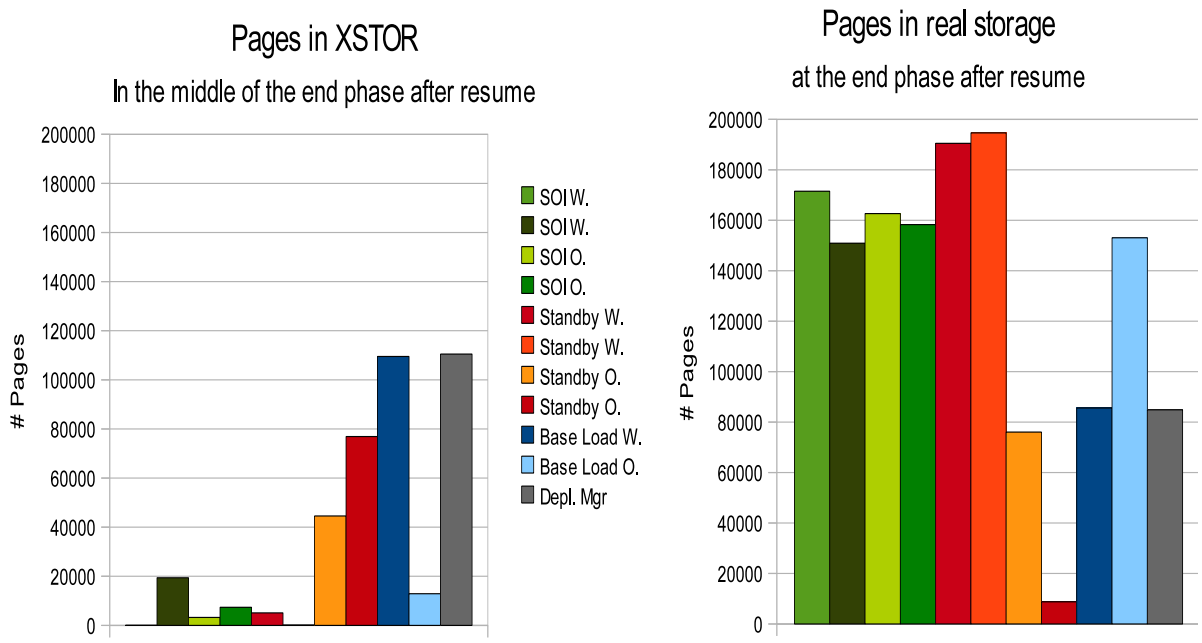

Figure 8. Location of memory pages for various guests at the end phase after resume

Observations

Most of the pages from the Systems of interest are brought back to real storage. From two of the four now suspended standby guests, many pages were moved to XSTOR. The pages from one of the base load systems are even more migrated to XSTOR, although this system was active and running the whole time. This is observed consistently in several tests, but was not expected.

Conclusions

After resuming the suspended guests, their pages come back to main storage. The major part of the memory pages were taken from two of the suspended standby guests, which have been moved from real storage to XSTOR. The memory pressure was possibly not high enough to force the pages from all suspended guests to be migrated to XSTOR.

Using z/VM STOP and BEGIN method

This test should show where the pages of the guest reside when the idling guests are paused using the z/VM CP STOP and BEGIN commands, and memory pressure is created by bringing the standby guests under load.

For more information about the guest systems and their types, see Guest usage.

Middle of warmup phase

Figure 9 shows the location of the memory pages for the various Linux guests in the middle of the warmup phase.

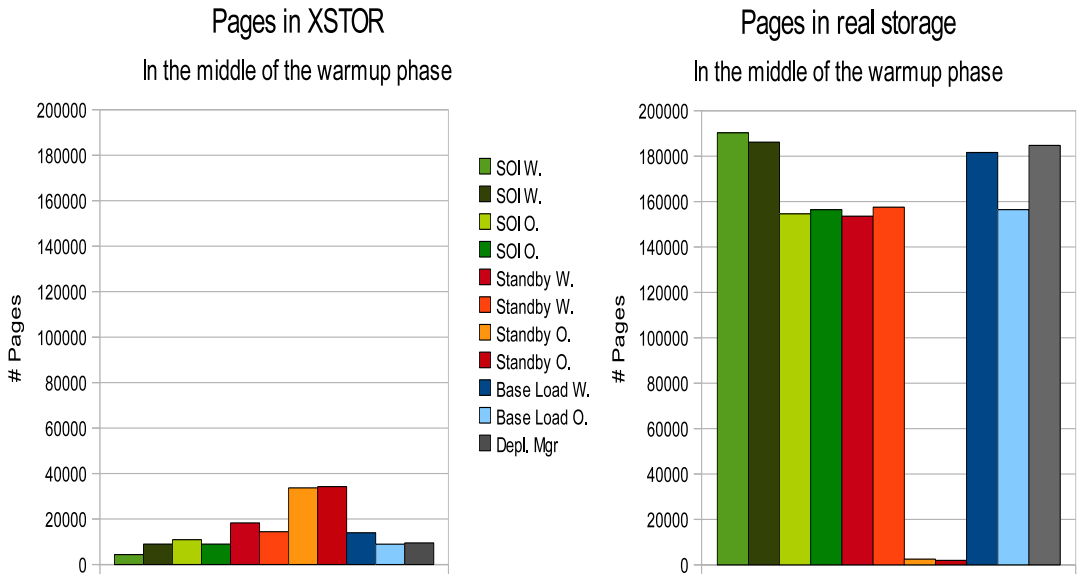

Figure 9. STOP and BEGIN of idling guests, location of memory pages during the warmup phase

Observations

The distribution of the pages is very similar to the case that uses the Linux suspend and resume mechanism. The idling Standby WebSphere guests have an unexpectedly high number of pages in real storage. However, the Systems of interest WebSphere guests have more pages in real storage than at the beginning of the suspend and resume test.

Conclusions

So far this should be the same starting situation as in the Linux suspend and resume case. The large number of pages in real storage from Standby WebSphere guests is due to the active node agents. The slightly different distribution of the System of interest WebSphere guests is not really relevant, but it shows very clearly that the paging behavior is not completely deterministic.

Middle of suspend phase

Figure 10 shows the location of the memory pages for the various Linux guests in the middle of the suspend phase.

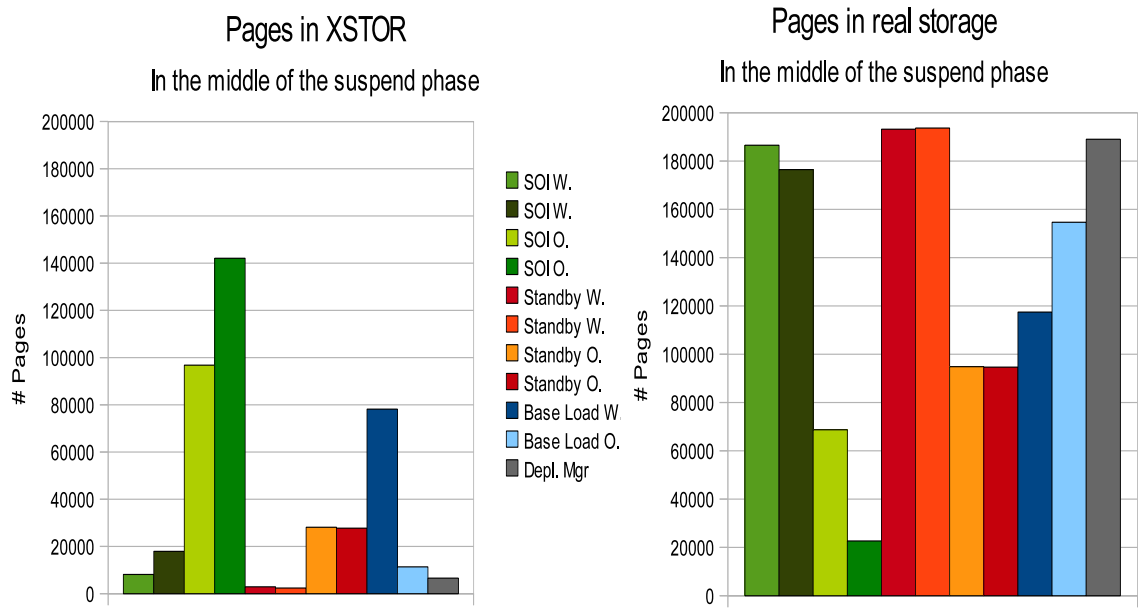

Figure 10. STOP and BEGIN of idling guests, location of memory pages during the suspend phase

Observations

Once again, the distribution of the pages is very similar to the case that used the Linux suspend and resume mechanism, but the distribution is not identical. This time, the majority of pages are taken from only one of the suspended WebSphere guests and one of the RDB guests. And again, one of the base load system has a significant number of pages in XSTOR and in real storage.

Conclusion

There is a clear preference to take pages away from the suspended guests.

After resume

Figure 11 shows the location of the memory pages for the various Linux guests at the end phase after resume.

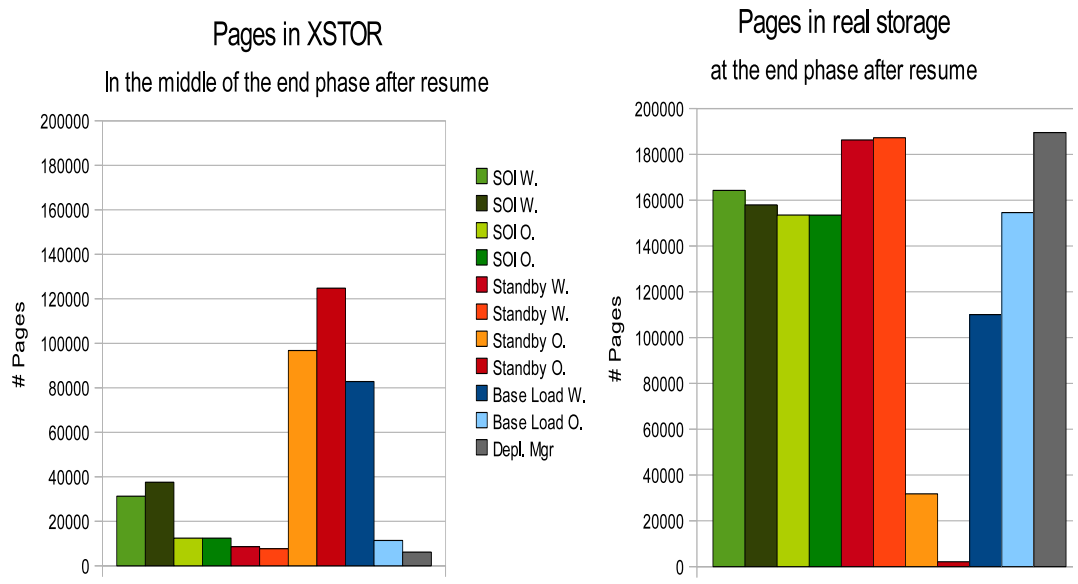

Figure 11. STOP and BEGIN of idling guests, location of memory pages at the end phase after resume

Observation

Pages from the now suspended Standby guests are moved out to XSTOR, and the pages from the resumed Systems of interest are moved back into real storage.

Conclusions

The process where the pages of the dormant guest are moved to XSTOR, and the pages of the active guests are moved into real storage works in a similar manner for both pausing mechanisms. It seems also that z/VM tries to minimize the movement of pages, because in no case are all pages from one guest located only in one storage location. And there is always a slight non-deterministic characteristic, which makes it hard to predict exactly what will happen. This is certainly caused by the complexity of the algorithms used.

Additional test case: Resume time of larger guests

In the tests above, the z/VM guests with a size less than 1 GB were relatively small. Because there is interest in resume times of larger z/VM guests, a slightly different environment with larger guests was created and analyzed.

This test ran Linux as a z/VM guest with 10 virtual CPUs in an LPAR with 10 physical CPUs. WebSphere was installed into a DCSS. The test used multiple JVMs with a 1 GB heap inside the WAS guest. A workload was run against each JVM before suspend to ensure that the memory is really used. With multiple smaller JVMs, it is much easier to ensure that the memory of the guest is really used, than compared to using one large Java heap. This is important because the suspend mechanism does not includes unused pages in the memory image. Therefore, large Java heaps partially utilized might create much smaller memory images. The use of smaller memory images will shorten the resume time, providing better results when compared to the results of a system using all the memory.

The guest size was scaled, and the number of JVMs scaled accordingly. The limiting factor for the guest size was that the whole memory image needed to fit on one swap disk, which was a mod9 ECKD™ DASD in this case. Larger guests would require a larger disk, because the memory image cannot be split over multiple swap devices.

The resume time of the suspended guest was measured by taking the time from the restart of the suspended guest until the completion of a successful http get from an installed application as described in Test case 1: Elapsed times to pause and restart a guest.

For comparison, the time to start 6 JVMs serially with a script was measured. This measurement includes only the pure startup time for running the startServer command for servers 1 - 6, while the resume times includes a Linux restart. Note that the start of the first server takes significantly longer than the start of the subsequent five servers.

Figure 12 shows the result when scaling the guest size from 2 GB to 5.2 GB. The guest size and the number of JVMs is shown on the x-axis.

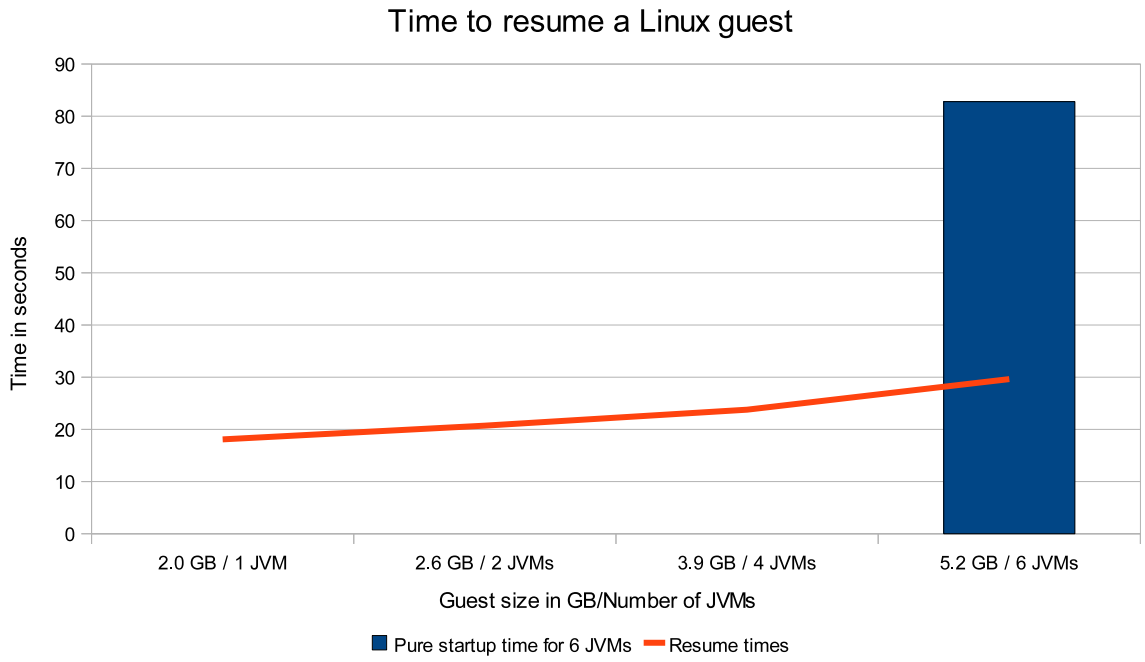

Figure 12. Resume times when scaling the guest size from 2 GB to 5.2 GB

Observations

The resume times are increasing nearly linearly with the guest size, but at a much slower rate. This means when doubling the guest size from 2.6 GB to 5.2 GB, the resume time increases by only 50%. The start time for the six WebSphere Application Servers is more than twice as long as the resume times.

Conclusions

The resume times are much shorter than the startup time of the WebSphere Applications Server. Therefore, the Linux suspend and resume mechanism is a convenient method for a controlled shutdown of the guests with a very fast resume to the state in which it was left. During the suspend phase, the physical memory pages are available for other, genuinely active guests.

Notices

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

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

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

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

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

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

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

August 2011

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

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

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

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

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

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

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

IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.
Produced in the United States of America,
08/2011

ZSW03192-USEN-00