z/VM Large Memory – Linux on System z

Table of Contents

Objectives

This report analyzes the results observed with Linux® guests running a database server in a z/VM® environment using a relatively large amount of main memory (80 GB) and then also overcommitting that memory. To test the z/VM memory management behavior, the memory was overcommitted up to a factor of two and the impact of the following z/VM memory management features were tested:

- *VM Resource Manager Cooperative Memory Management (VMRM-CMM), initially available with z/VM 5.2 (also known as CMM Version 1)*
- *Collaborative Memory Management Assist (CMMA), initially available with z/VM 5.3 (also known as CMM Version 2)*

These features were expected to improve the overall system performance in cases where the overall z/VM system is constrained for real storage.

This paper shows the performance of the Linux database guests under an online transaction processing (OLTP) workload using asynchronous disk I/O. The number of guests was scaled to create the memory overcommitment on z/VM. The starting point for the scaling was a setup with multiple Linux guests that allocated all available memory and then the number of guests was increased until the memory overcommitment of a factor of two was reached.

Executive summary

Both z/VM 5.2 and z/VM 5.3 were able to handle the scenario with a relatively large amount of physical memory (80 GB) and the use of twice as much virtual memory than physically available (memory overcommitment). We recommend the usage of z/VM 5.3 and VMRM-CMM for this type of workload (transactional database workload using asynchronous disk I/O). In the memory overcommitted case, z/VM 5.3 shows significant improvements for performance compared to z/VM 5.2. The features VMRM-CMM and CMMA require that the most current fixes available for these features for Linux and z/VM are installed. Memory overcommitment is expected to cause performance degradation, where VMRM-CMM reduces that impact to 13%, which is a very small impact for a memory overcommitment of a factor of two.

Summary

This chapter summarizes the results observed with z/VM Linux guests running a database server on z/VM 5.2 and z/VM 5.3, and on z/VM 5.3, with and without the VMRM-CMM and CMMA features. A transactional OLTP workload using asynchronous disk I/O was used to drive the guests. The number of guests was scaled to create a memory overcommitment on z/VM. Memory overcommitment is expected to cause a performance degradation with a higher overcommitment having a bigger impact.

The following are our summary results:

- *As the guests were scaled from five to ten, z/VM 5.3 achieved much higher throughput than z/VM 5.2. At ten guests, and a planned memory overcommitment of 100%, throughput on z/VM 5.3 was about 90% higher than on z/VM 5.2.*
- *VMRM-CMM and CMMA were enabled on z/VM 5.3. It is important to note that several APAR fixes and bugzilla bug fixes needed to be applied to z/VM 5.3 and Linux. (See Software setup for details.) Both VMRM-CMM and CMMA improved the throughput results obtained from the ten guest runs with 100% memory overcommitment.*
  - *CMMA showed an 8% improvement in throughput over the results for ten guests on z/VM 5.3. There was a significant reduction in paging activity.*
  - *VMRM-CMM showed a significant improvement of 50% in throughput over the results for ten guests on z/VM 5.3.*
  - *With VMRM-CMM, the throughput result for the ten guest scenario, which uses twice as much virtual memory than is physically available, was only 13% lower than the results with five guests without memory overcommitment. This very good result is expected to be specific for this workload and can not be generalized without further tests for other workloads.*

Hardware equipment and software environment

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

- *Hardware used*
- *.. Network setup*
- *Storage server setup*
- *z/VM guest setup*
- *Client hardware*
- *Software setup*
- *The test environment*

Host hardware

One LPAR on a 16-way IBM System z9™, type 2094-S18, equipped with:

Table 1. z/VM LPAR configuration

System/Guest	Storage		Dedicated CPUs
	Central	Expanded	
z/VM LPAR	80 GB	4 GB	10

- *1 OSA Express 2 gigabit Ethernet card (OSA code level 0805) with two ports*
  - *Each of the five guests are sharing one Ethernet port*
- *8 FICON® express cards (each card had two ports)*
  - *Each storage server was connected via eight FICON paths*

Network setup

The z/VM LPAR was connected to the clients via two Fiber Gigabit Ethernet Interfaces.

Storage server setup

Two IBM System Storage™ DS8300 2421 Mod 932 servers were used

- *IBM 3390 disk models 3 and 9*
- *Physical DDMS with 15,000 RPMs*

The first storage server was used for:

- *Linux and database disks*
  - *Linux SLES10 GA system was installed on six 3390 mod-3s and the Linux SLES10 SP1 system was installed on three 3390 mod-9s*
- *The database was defined on five 3390 mod-9s. Each set of five database disks was defined on the same rank. For the ten guests, a total of ten ranks were used for the database disks.*
- *z/VM paging space*

The second storage server was used for the disks needed for the z/VM system and paging space.

z/VM guest setup

The z/VM recommendation for the paging setup is to not use more than 50% of the DASD paging space. Following this recommendation, the setup should have a minimum of 152 GB of DASD paging space ((160 GB guests memory - 84 GB total memory) * 2).

Total pages allocated for z/VM paging space was 37,256,192 pages.

Paging space defined was 1.66 times the number of pages defined for storage and expanded storage.

Paging space was distributed across 31 3390 mod-3s and 10 3390 mod-9s as shown in Table 2. This was done to get a high I/O bandwidth for the paging I/O.

Table 2. Page space distribution

Enterprise Storage Server® 1 (mod3s)		Enterprise Storage Server 2 (mod 9s)	
Rank	**Number of Disks**	**Rank**	**Number of Disks**
1	5	1	1
2	6	2	1
3	6	3	1
4	6	4	1
5	6	5	1
6	2	6	1
		7	1
		8	1
		9	1
		10	1

Table 3 shows the z/VM guest configuration we used for our tests.

Table 3. Test system configuration: database server

System/Guest	Memory	# of Virtual CPUs
Database server guest 1-10	16 GB	3

Client hardware

One through ten x330 PCs with two Intel® 1.26 GHz processors.

The workload driver ran on each client and only one client was attached to each guest database server.

Software setup

Table 4. Client software used

Product	Version/Level
Red Hat Enterprise Linux	RHEL 4 ES (clients)
SUSE Linux Enterprise Server	SLES10 GA (kernel level 2.6.16.21-0.8) SLES10 SP1 (kernel level 2.6.16.46-0.12)
z/VM	z/VM Version 5 Release 2.0, service level 0501 (64-bit) z/VM Version 5 Release 3.0, service level 0701 (64-bit)

Notes:

- *In order to enable CMMA and VMRM-CMM the following z/VM APAR fixes are required: VM64297, VM64253, VM63968, VM64225, VM64226, VM64228. SLES10 SP1 needs the patch for bugzilla bug 38500, which is available in kernel-default-2.6.16.53-0.16.s390x.rpm from the Novell support center.*
- *Use of CMMA requires SLES10 SP1 or higher.*
- *With SLES10 SP1, we also needed to set QIOASSIST OFF for each guest virtual machine. When a fix is available for z/VM APAR VM64287, QIOASSIST ON should function correctly.*

Test environment

Figure 1 shows the test environment we used for our tests.

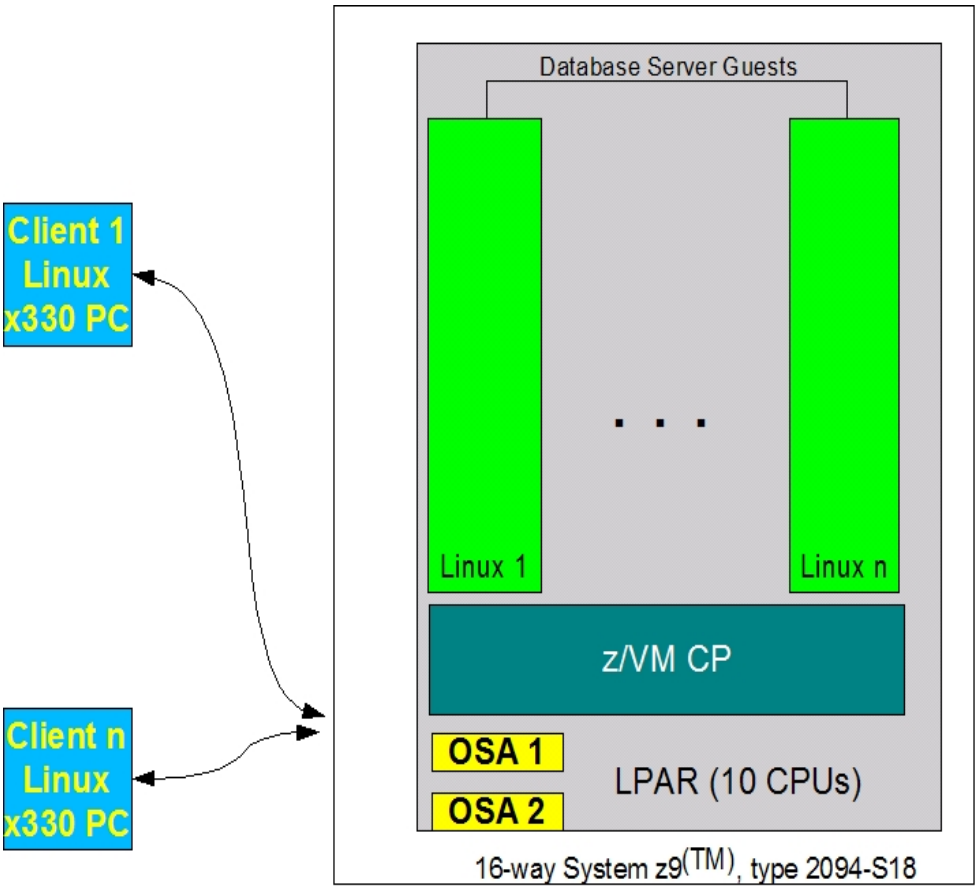


Figure 1: Environment overview

Setup

This chapter contains details on the setup we used for our tests.

Linux Setup

As described in section "z/VM guest setup" each Linux system has three CPUs and 16 GB memory. The swap space has a size of 2 GB and resides on a DASD device.

Database server

We used a popular database product for Linux on System z™ as a database server. The focus was on the memory management behavior of z/VM. We felt any database product would provide a good example of high memory usage. Therefore, we describe the database setup only in general terms. The database workload used was a typical OLTP transactional workload.

The database server was setup to use asynchronous disk I/O and didn't bypass the Linux page cache for file system operations.

The space for the test database was defined as follows:

- *Five physical volumes - 3390 Mod 9s*
- *One volume group created from the five physical volumes*
- *One logical volume defined from the volume group with five stripes and a stripe size of 32 K*
- *The total database size was 30 GB*

The database cache size was 8500 MB, the total memory size allocated for all the database buffers and caches was 9152 MB. The database caches were sized in such a way that the system did not swap.

z/VM memory management overview

The z/VM system maps the guests' virtual memory into the real memory storage of the System z machine. If there are not enough real memory frames to contain all the required active guests' virtual memory pages, the active guests' virtual pages are moved to XSTOR.

Once XSTOR becomes full, the guests' pages are moved from XSTOR to DASD paging space.

Figure 2 provides a picture view of the z/VM memory management overview.

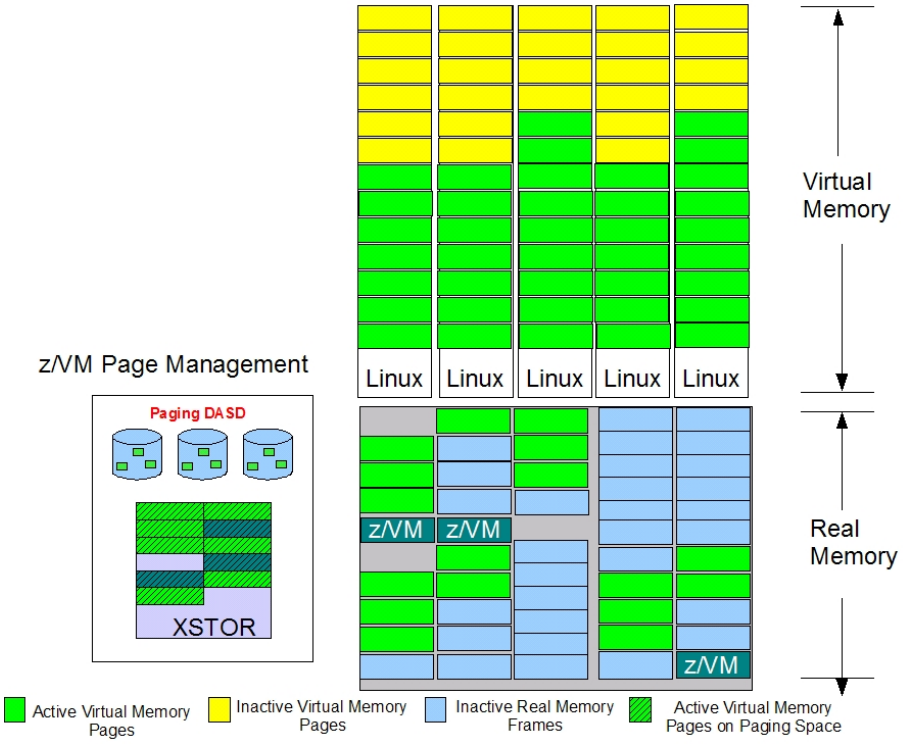


Figure 2. z/VM memory management overview

Collaborative Memory Management Assist (CMMA)

z/VM adds support for the hardware CMMA on the IBM System z9 Enterprise Class (z9™ EC) and the IBM System z9 Business Class (z9 BC) processors. This z/VM support, in conjunction with the software CMMA exploitation in guest operating systems such as Linux on System z, allows the z/VM Control Program (CP) host and its guests to communicate attributes for specific 4 KB blocks of guest memory. This exchange of information can allow both the z/VM host and its guests to optimize their use and management of memory in the following ways:

- *The CP knows when a Linux application releases storage and can select those pages for removal at a higher priority or reclaim the page frames without the overhead of paging-out their data content to expanded storage or disk.*
- *The CP recognizes clean disk cache pages, the contents of which Linux is able to reconstruct, allowing the CP to bypass paging-out the data contents when reclaiming the backing frames for these pages. If Linux or its application subsequently tries to refer to the discarded page, Linux is notified that the page has been discarded and can reread the contents from disk or otherwise reconstruct them.*

- *The guest further benefits from the hardware Host Page-Management Assist (HPMA), which was announced in the Hardware Announcement dated July 27, 2005. In conjunction with CMMA, HPMA allows the machine to supply fresh backing page frames for guest memory when the guest reuses a previously discarded page, eliminating the need for the z/VM hypervisor to intercept and resolve these host page faults.*

To set the guests to use CMMA, we enabled the CP memassist function as follows:

CP SET MEMASSIST ON ALL

The Linux support for CMMA has to be activated per IPL by using the boot option `cmma=on` (default is cmma=off).

We set cmma=on in the zipl.conf file of each guest.

VM Resource Manager Cooperative Memory Management (VMRM-CMM)

VMRM-CMM between a z/VM system and Linux guests assists in managing memory constraint in the system. Based on several variables obtained from the System and Storage domain CP monitor data, the VMRM detects when there is constraint and notifies specific Linux virtual guests when this occurs. The guests can then take the appropriate action to adjust their memory utilization in order to relieve the constraint on the system.

A user-defined VMRM configuration file is used to identify which guests should be notified. These guests must be prepared to handle the data being sent in the notification and take appropriate action, such as issuing a CP Diagnose instruction to release pages of storage.

In addition to the workload management functions provided by VMRM for the CPU and DASD I/O, the following enhancements have been made to the VMRM for Linux guests:

1. *A new NOTIFY statement with a MEMORY keyword is supported in the VMRM configuration file. Following the keyword is a user ID or a list of user IDs to be notified when virtual memory becomes constrained. (For the format of this statement, see z/VM 5.3 CP Commands and Utilities, SC24-6081, and z/VM 5.3 Performance, SC24-6109.)*
2. *System and Storage domains are monitored for data to be used for calculating memory constraint as well as how much memory to request that the guest machine release.*
3. *The VMRM issues a CP SMSG to notify the specified guests when memory is constrained as well as the amount required to release in order to relieve the constraint. (For the format of the CP SMSG*

buffer, see z/VM 5.3 CP Commands and Utilities, SC24-6081, and z/VM 5.3 Performance, SC24-6109.)

4. *A message is logged in the VMRM log file, which indicates which users were sent an SMSG and the text of the SMSG buffer. Also, if MSGUSER is specified on the VMRM ADMIN statement, the same message written to the log is written to the MSGUSER user ID's console as well.*

Once a system memory constraint has been detected, VMRM calculates how much memory each Linux guest should release to relieve the constraint. Using the SHRINK keyword on the SMSG command, a message indicating the amount of storage to release is sent to each logged on Linux guest in the notify list.

When system memory is no longer constrained, another SHRINK message with a smaller absolute value is issued. A smaller SHRINK request than the previous one effectively instructs the guest to reclaim some of the storage previously released.

To setup z/VM to use VMRM-CMM you need to create a VMRM configuration file (ours was VMRM CONFIG A1) on the VMRMSVM user ID that contains one of the following statements:

NOTIFY MEMORY LNX00080 LNX00081 LNX00082 LNX00083 LNX00084

or

NOTIFY MEMORY LNX0008*

We also added the following statement to get VMRMSVM messages:

ADMIN MSGUSER MAINT

This statement causes VMRMSVM messages to be sent to the MAINT user ID.

The "*" acts as a wild card. After creating the configuration file on the VMRMSVM A-disk we logged off VMRMSVM and issued an xautolog VMRMSVM from MAINT to start the VMRM server.

To stop the VMRMSVM server, log on to VMRMSVM and issue the command HMONITOR.

On the Linux servers we loaded the VMRM-CMM kernel extension using the following modprobe command:

modprobe cmm

Workload description

The workload we used for our tests was an OLTP benchmark involving a mix of five concurrent transactions of different types and complexity. The database was

comprised of nine types of tables with a wide range of record and population sizes. The workload results are measured in transactions per minute (tpm).

The workload simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is centered on the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. The benchmark portrays the activity of any industry that must manage, sell, or distribute a product or service.

Each database server was driven by a unique client machine executing the workload client driver.

There was no interaction between database servers.

CPU utilization charts explained

In the charts in this report, the CPU utilization values are always normalized in a way that 100% means one CPU is fully utilized.

The CPU utilization charts in the report look like Figure 3.

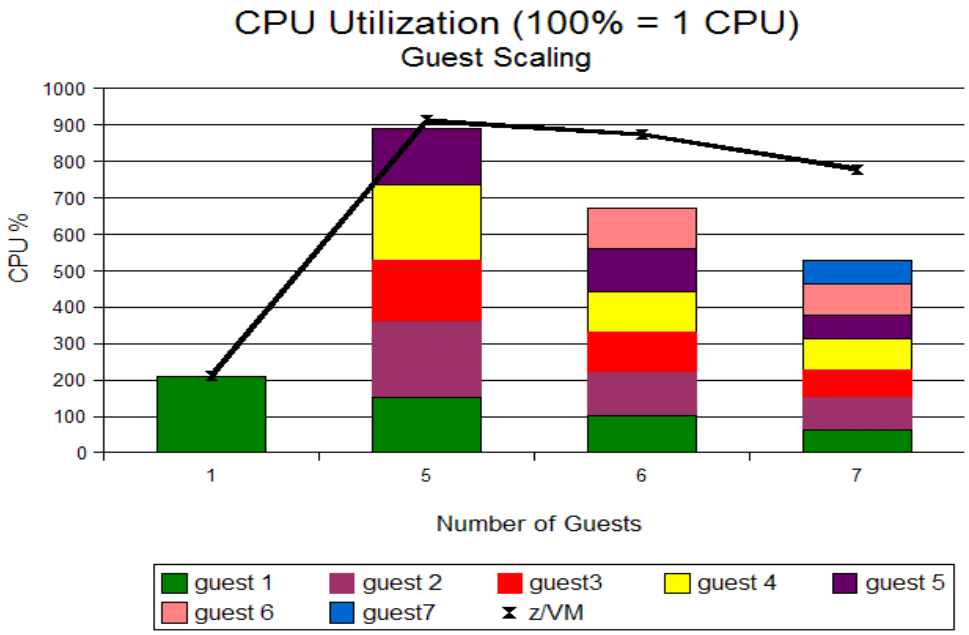


Figure 3. CPU utilization - guest scaling

The CPU utilization charts show the utilization for the z/VM LPAR by a line with triangle symbols. The details of how the z/VM guests use the CPUs (total CPU time accounted for the guests) are shown by stacked bars, which should sum up to the z/VM utilization without the CP related CPU part, so it is always a

little lower than the z/VM LPAR utilization. If the CPU utilization from a guest in the legend is not visible, that means it is too low to display.

Results

This chapter provides our detailed test results and conclusions. Results from the workload test runs are shown. All CPU utilization and paging results detailed in this chapter were obtained from the z/VM Performance Monitor reports. Completed transaction throughputs were reported by the workload driver.

Scale guests into memory overcommitment on z/VM 5.2

In these tests we started with one database server guest and then added four more database server guests for a total of five guests. These five guests used all of the available physical memory. The number of workload driver users was adjusted to reach maximum throughput and the users were distributed evenly among the guests. The guests were scaled from five to ten. We strove to maintain maximum throughput while scaling the guests from five to ten.

Total throughput

Figure 4 shows the normalized transactional throughput from all guests when scaling the number of guests on z/VM 5.2 into memory overcommitment.

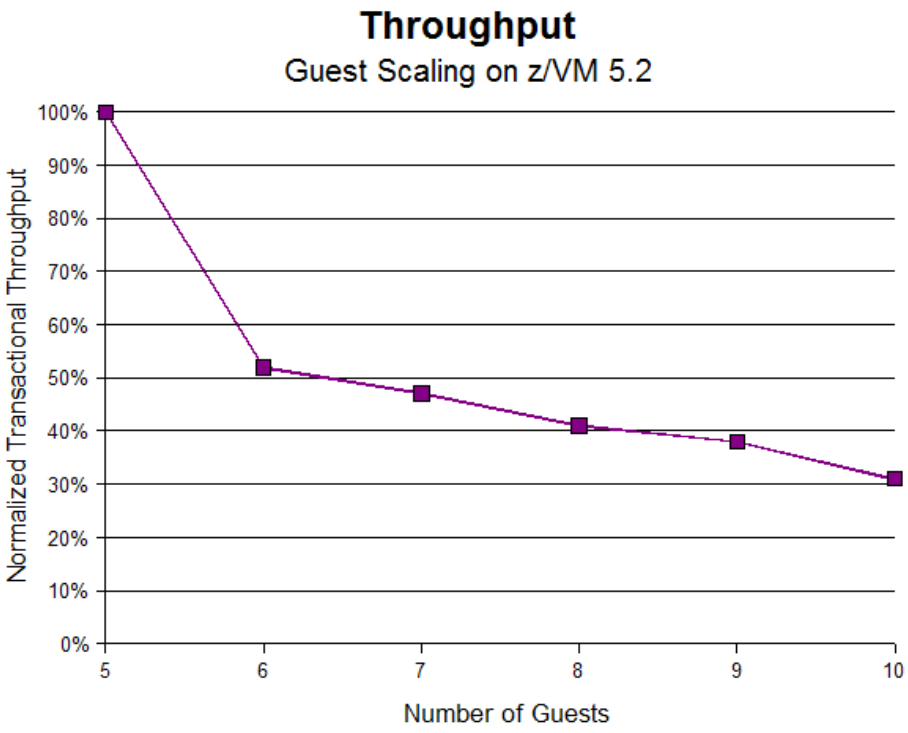


Figure 4. Normalized transactional throughput for scaling the number of guests on z/VM 5.2

Observations

Figure 4 shows the effect of memory overcommitment on the throughput. With five guests, all memory is in use. The addition of just one more guest caused a decrease in throughput of more than 50%. Adding further guests leads to a more moderate degradation.

CPU utilization

Figure 5 shows the CPU utilization of each guest and the total CPU utilization for the z/VM LPAR for the ten CPUs assigned to the LPAR.

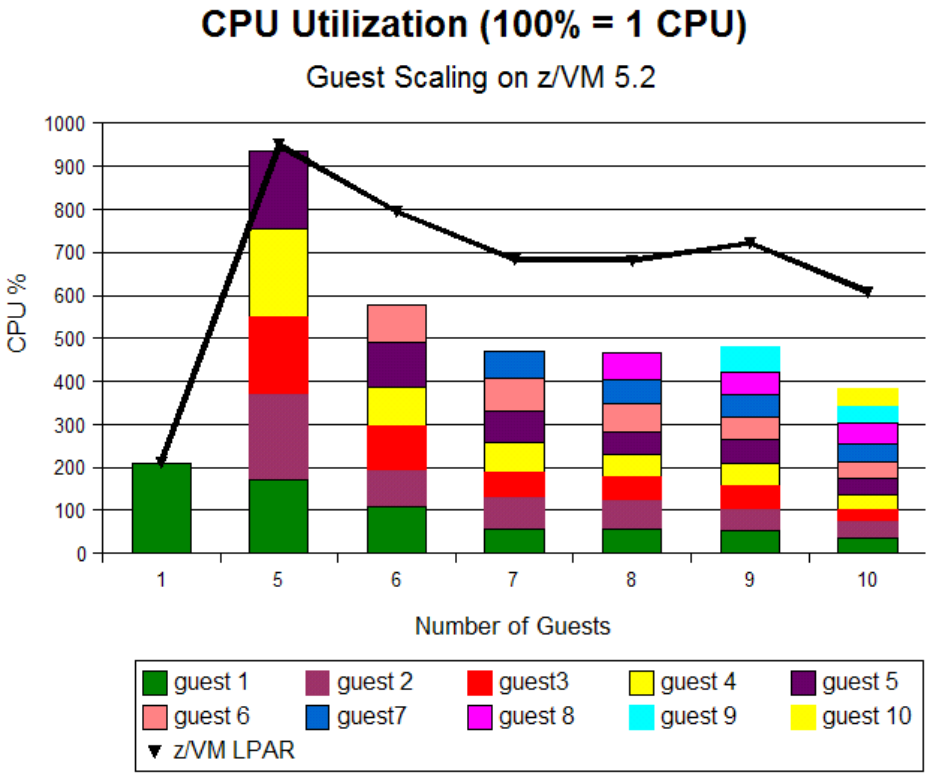


Figure 5. Total CPU utilization per number of guests

Observations

Figure 5 shows the effects of memory overcommitment on CPU utilization. At five guests we were able to achieve more than 900% utilization of the ten total CPUs and the z/VM overhead was small. As more guests were added, the total CPU utilization used by the guests decreased significantly and z/VM overhead increased.

Page location

Figure 6 shows the location of page frames associated with guests. The values are a sum of the location of the page frames for each individual guest.

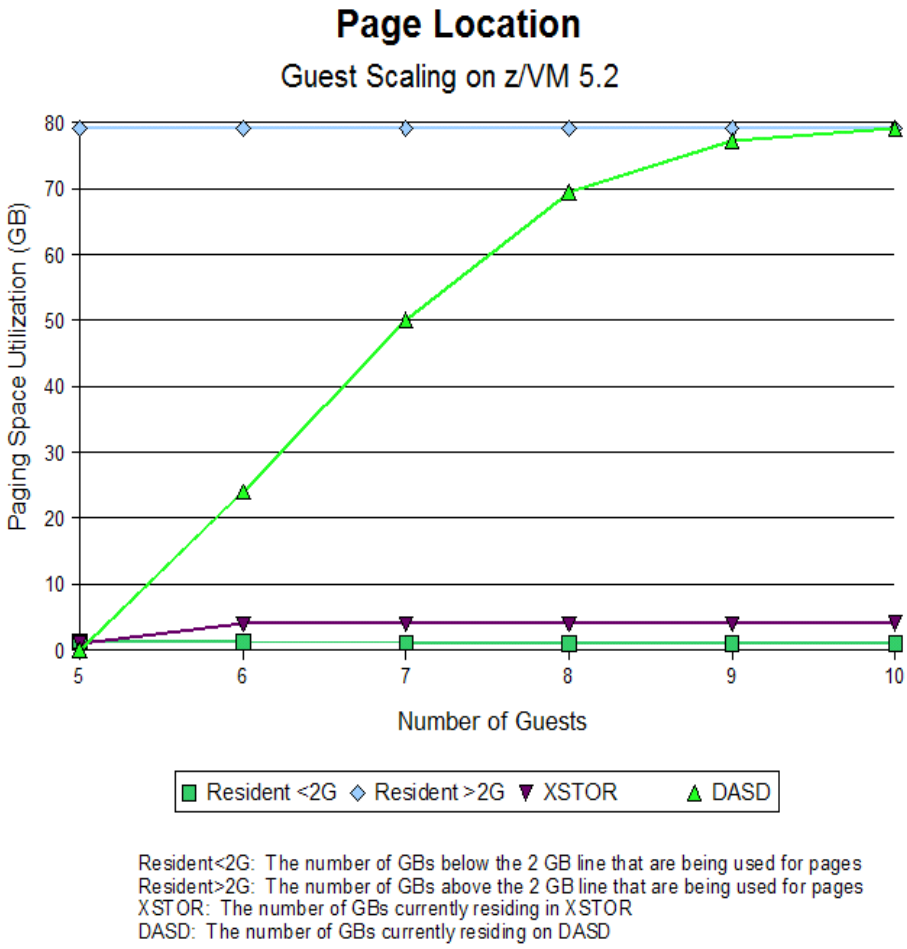


Figure 6. Page location - guest scaling on z/VM 5.2

Observations

Figure 6 shows the location of pages for the guests as the number of guests is scaled from five to ten. The most interesting curve is the utilization of the paging space on DASD. There is a steep increase as the number of guests is increased and at ten guests, there are as many pages on DASD as in memory.

Page movement

Figure 7 shows the relation of the memory overcommitment on the page movement rate to the various destinations.

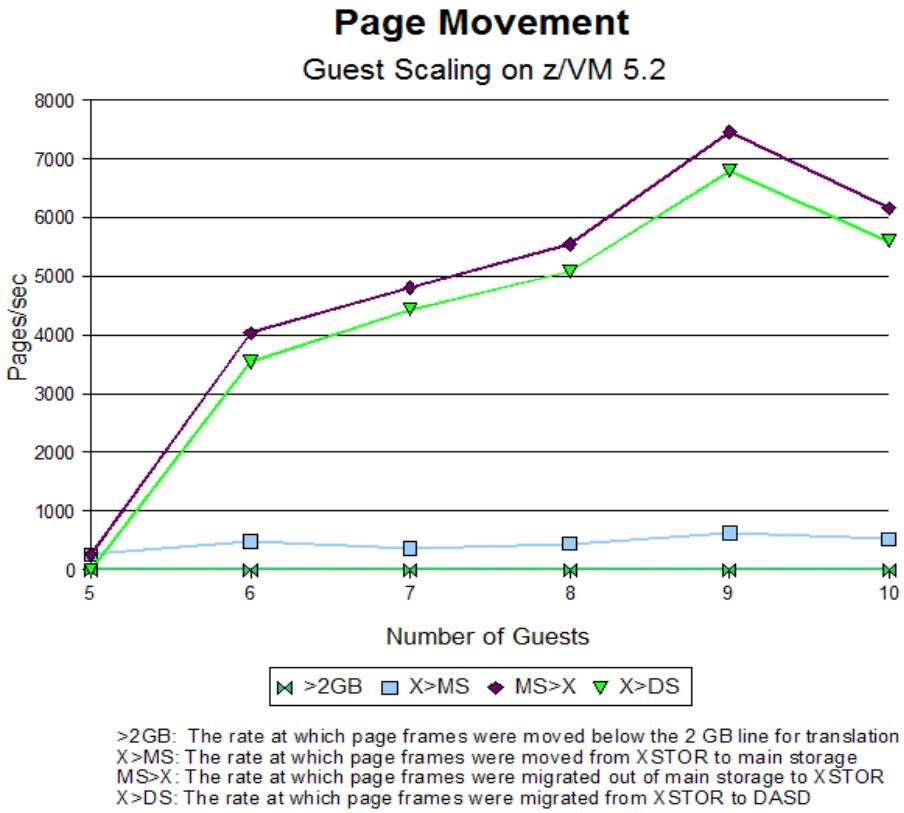


Figure 7. Page movement - guest scaling on z/VM 5.2

Observations

As guests are scaled from five to ten, the page movement rates from main storage to XSTOR and from XSTOR to DASD increase. The maximum page movement rate was observed at nine guests and at ten guests the movement rate had decreased. This decrease corresponds with the decrease in CPU utilization that was observed at ten guests. The highest traffic is seen in the direction from main storage to expanded storage, closely followed by the direction from expanded storage to DASD.

Conclusion

Six guests and a memory overcommitment of only 35% resulted in a throughput reduction of 48%. At six guests, after the initial affects of page movement to DASD, the addition of additional guests has a lower impact on throughput.

The total system memory of 80 GB (20,971,520 pages) is overcommitted as follow:

Table 5. Overcommitted system memory - z/VM 5.2

Number of Guests	Pages Not in Memory (= paged out)		Real Over commit (pages used: memory) - 1	Planned Over commit (virtual : physical) - 1	Through put % From Baseline
	XSTOR	DASD	%	%	%
5	267907	0	1	0	baseline
6	1034590	6280409	35	20	52
7	1032323	13128704	68	40	47
8	1039500	18219008	92	60	41
9	1038750	20236288	101	80	38
10	1045785	20754432	104	100	31

Interestingly, z/VM has more pages in use than it should have, based on the planned overcommitment. This happens because, in some cases, z/VM has pages in memory as well as in page space to avoid bouncing pages.

At six guests the throughput has decreased to 52% of the five guests' value. With the addition of more guests, throughput declines at a linear rate and at ten guests has decreased to 31%. As guests were added, z/VM overhead also increased and we were not able to achieve higher percentages of total CPU utilization.

Looking at the movement rates, it shows that, at most, the pages were moved to DASD via XSTOR, and that the movement rate back to main storage is very low, indicating that z/VM had moved the right pages.

Scale guests into memory overcommitment on z/VM 5.3

In these tests we performed the same tests as in Scale guests into memory overcommitment on z/VM 5.2, but using z/VM 5.3. We started with one database server guest and then added four more database server guests for a total of five guests. These five guests used all of the available physical memory. The number of workload driver users was adjusted to reach maximum throughput and the users were distributed evenly among the guests. The guests were scaled from five to ten. We strove to maintain maximum throughput while scaling the guests from five to ten.

Total throughput

Figure 8 shows the normalized transactional throughput for all guests when scaling the number of guests on z/VM 5.3 into the memory overcommitment.

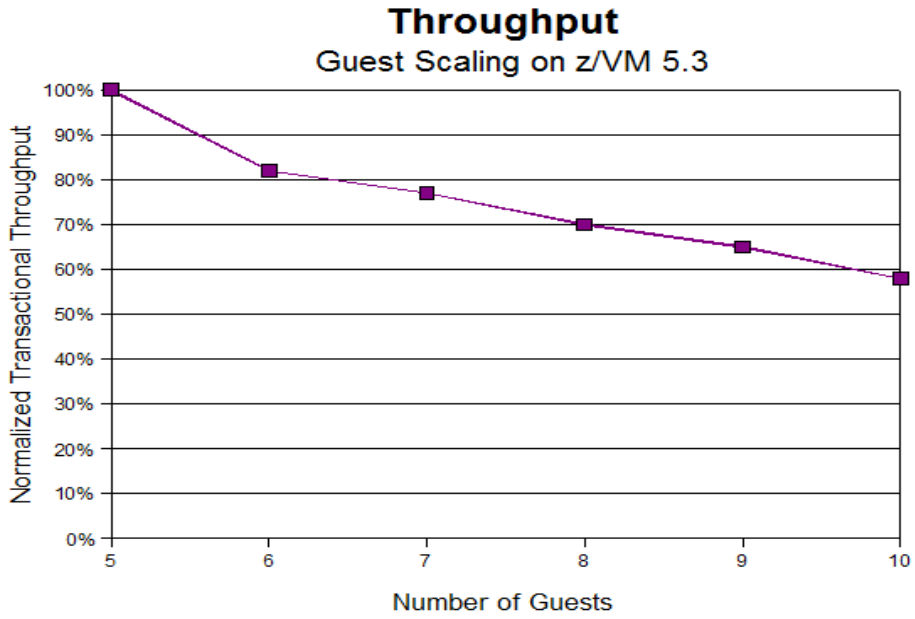


Figure 8. Normalized transactional throughput for scaling the number of guests on z/VM 5.3

Observations

Figure 8 shows the effect of memory overcommitment on throughput with a near linear decrease. At ten guests, throughput has deceased by slightly more than 40%.

CPU utilization

Figure 9 shows the CPU utilization of each guest and the total CPU utilization for the z/VM LPAR for the ten CPUs assigned.

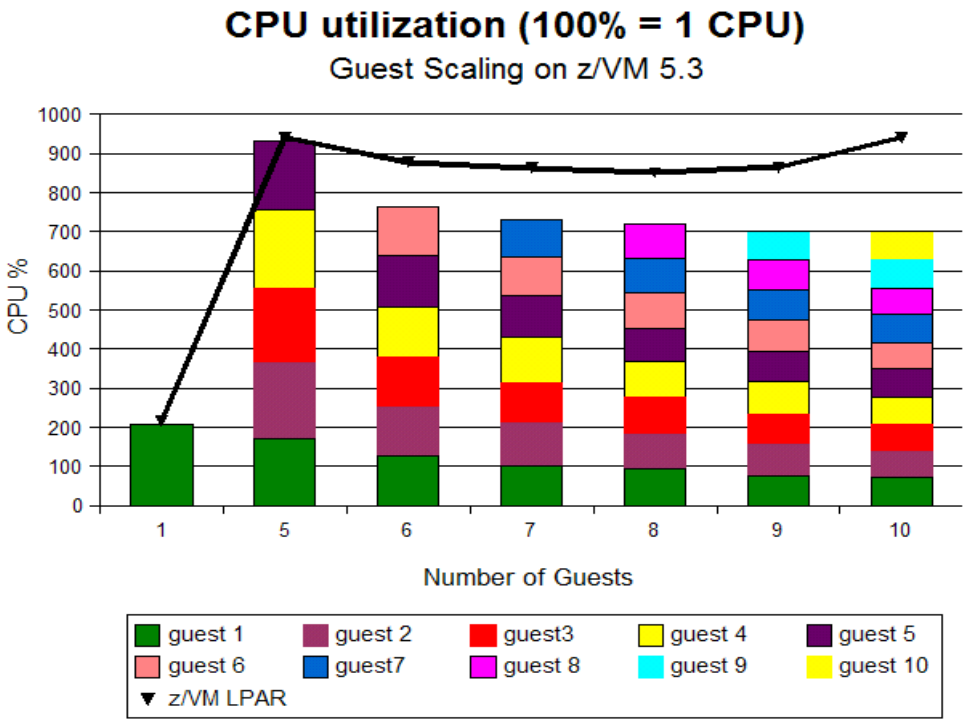


Figure 9. Total CPU utilization per number of guests

Observations

Figure 9 shows the effect of memory overcommitment on CPU utilization. Adding additional guests results in more z/VM overhead. According to the throughput behavior shown in Figure 8, as more guests are added, there is a near linear decrease in CPU utilization for the guests. The z/VM overhead starts to increase significantly at nine guests.

Page location

Figure 10 shows the location of page frames associated with guests. The values are a sum of the location of the page frames for each individual guest.

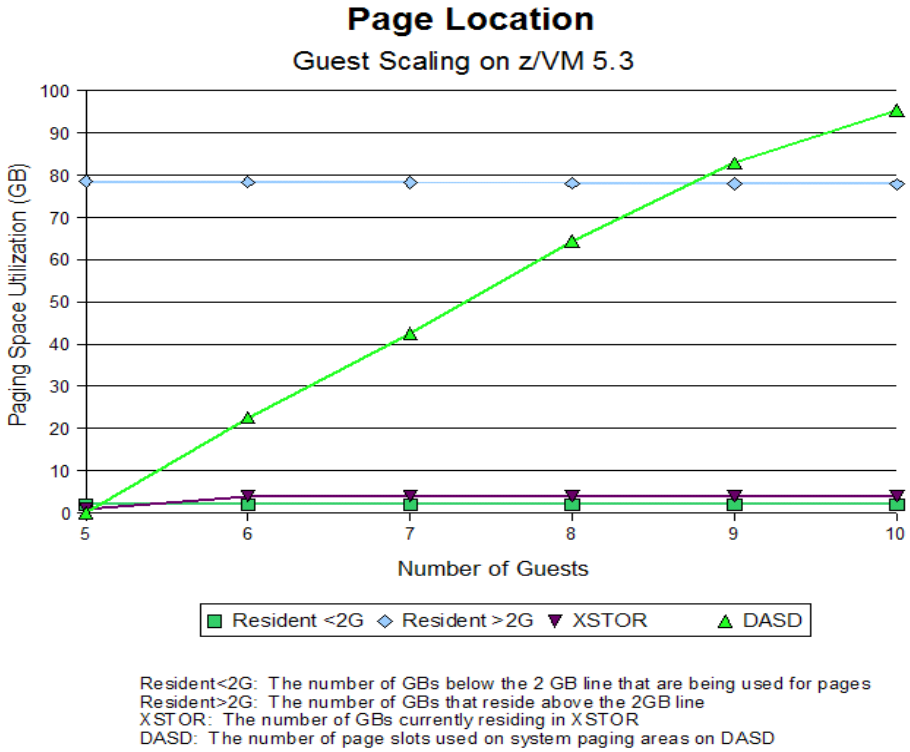


Figure 10. Page location - guest scaling on z/VM 5.3

Observations

Figure 10 shows the location of pages in the system as guests are increased from five to ten. The most interesting curve is the utilization of the paging space on DASD. Its increase is near linear at a constant slope. At ten guests there are more pages on DASD than in memory.

Page movement

Figure 11 shows the relation of the memory overcommitment on the page movement rate to the various destinations.

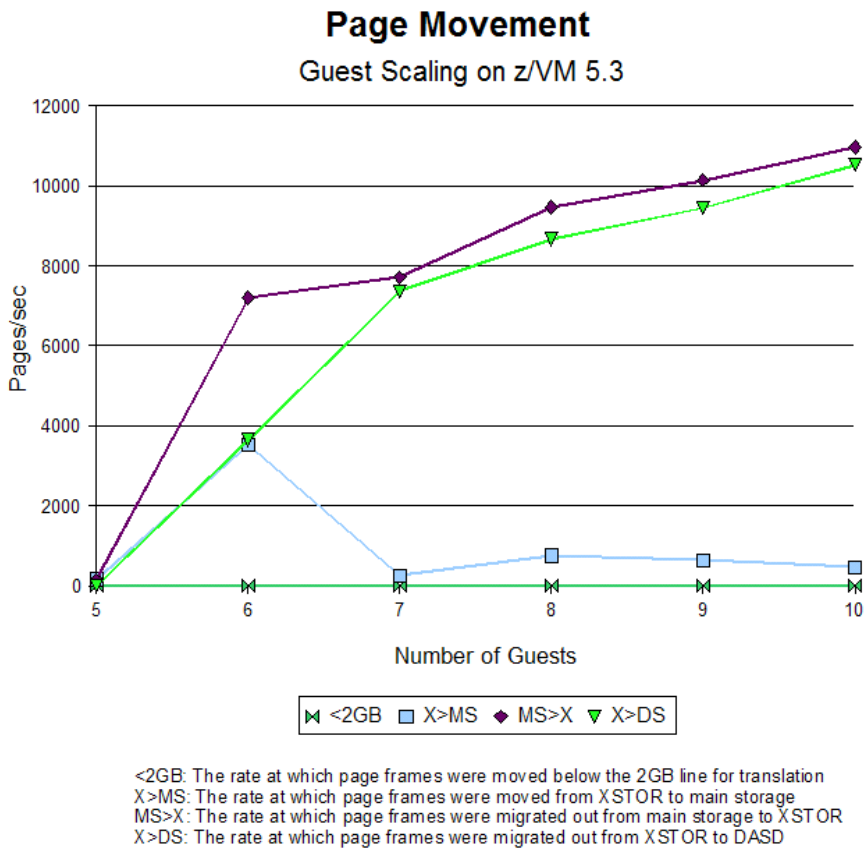


Figure 11. Page movement - guest scaling on z/VM 5.3

Observations

As guests are scaled from five to ten, the page movement rates from main storage to XSTOR and from XSTOR to DASD increase. The movement rates continue to increase all the way up to ten guests.

Conclusion

The total system memory of 80 GB (20,971,520 pages) is overcommitted as follows:

Table 6. Overcommitted system memory - z/VM 5.3

Number of Guests	Pages Not in Memory (= paged out)		Real Over commit (pages used: memory) - 1	Planned Over commit (virtual: physical) - 1	Through put % From Baseline
	XSTOR	DASD	%	%	%
5	263193	0	1	0	baseline
6	1031469	5905366	33	20	82
7	1030564	11152384	58	40	77
8	1033732	16862208	85	60	70
9	1034246	21728256	109	80	65
10	1034667	25014272	124	100	58

At six guests, the throughput has decreased to 82% of the five guests' value. With the addition of more guests, throughput declines at an almost linear rate and at ten guests, throughput has decreased to 58%. As guests are added, z/VM 5.3 overhead also increases. This time the real overcommitment is 24% higher than the planned overcommitment. It appears that the paging algorithm for z/VM 5.3 decides to move pages out to DASD more often, which seems to be an advantage for this workload.

Comparison of z/VM 5.3 versus z/VM 5.2

In this section we compare the most interesting parameters from z/VM 5.2 and z/VM 5.3 when scaling into the memory overcommitment.

Throughput

Figure 12 compares the throughput results for z/VM 5.3 with z/VM 5.2. The throughput is normalized to the z/VM 5.3 five guests' results.

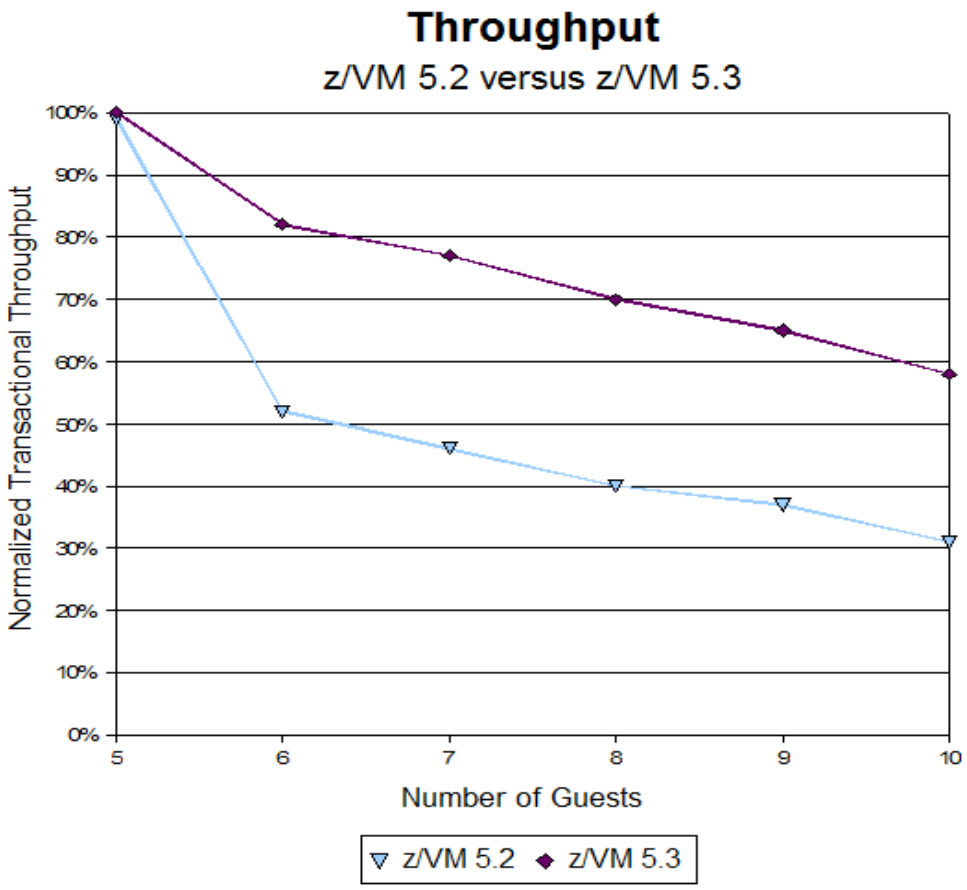


Figure 12. Normalized transactional throughput for scaling the number of guests on z/VM 5.3 versus z/VM 5.2

Observations

z/VM 5.3 maintains a higher throughput, which ranges from about 60% to about 90% higher than the z/VM 5.2 value. At ten guests, the throughput on z/VM 5.2 is degraded to a little bit above 30% from the throughput of the scenario without memory overcommitment with five guests. Here, z/VM 5.3 reaches a value that is nearly two times higher.

CPU utilization

Figure 13 compares the overall CPU utilization (LPAR load) observed with z/VM 5.3 versus z/VM 5.2.

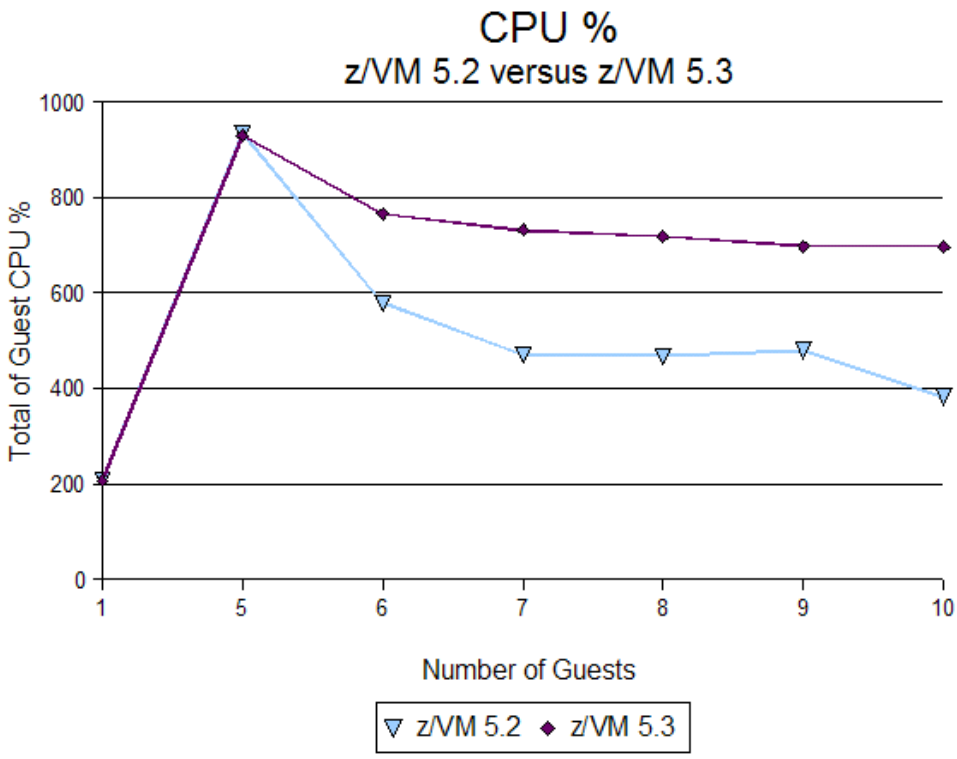


Figure 13. CPU % (100% = 1 CPU) - z/VM 5.2 versus z/VM 5.3

Observations

z/VM 5.3 experiences a higher CPU utilization rate as guests are scaled from five to ten (Figure 13) with higher throughput (Figure 12), but the throughput decreases faster than the CPU load, indicating that the effort for managing the lack of storage pages increases. This is better shown in Figure 14, which compares just the overhead observed with z/VM 5.3 versus z/VM 5.2. The overhead is calculated as the difference between the z/VM LPAR load and the guest CPU load.

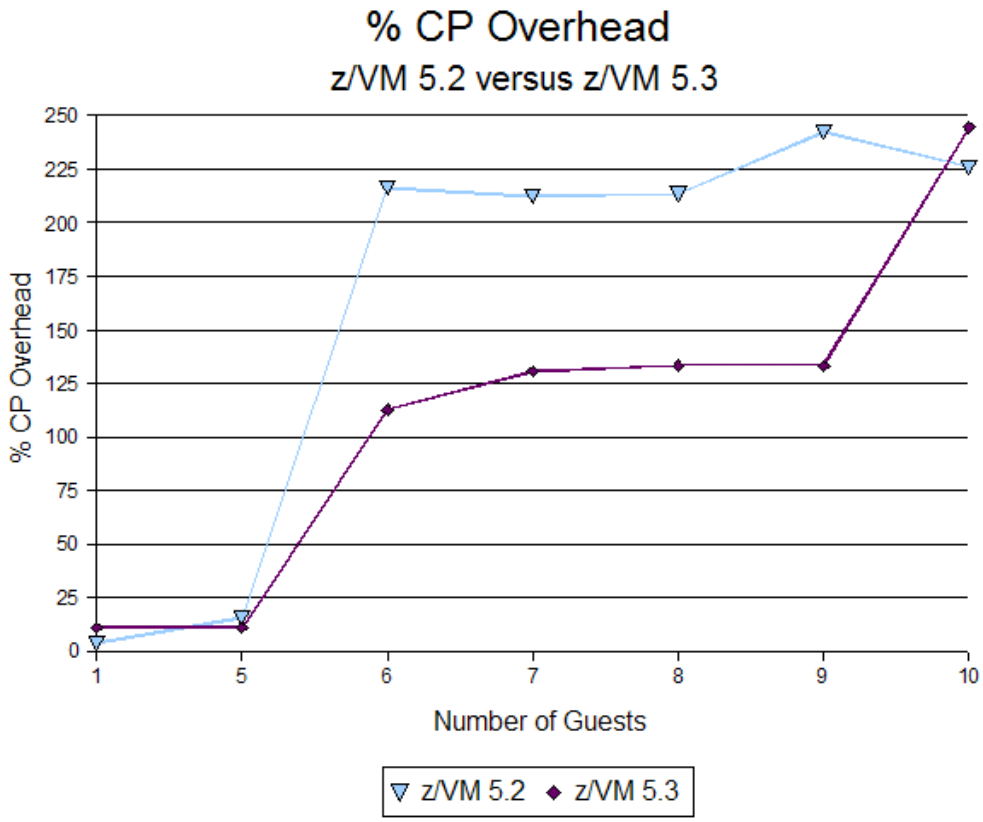


Figure 14. % CP overhead - z/VM 5.2 versus z/VM 5.3 (calculated as the difference between the z/VM LPAR load and the guest CPU load.)

Observations

In terms of CPU utilization spent for the CP, the overhead for z/VM 5.3 is much lower than z/VM 5.2 for six through nine guests.

DASD paging space

Figure 15 compares the total number of gigabytes of DASD space that are used to hold page frames on z/VM 5.3 versus z/VM 5.2.

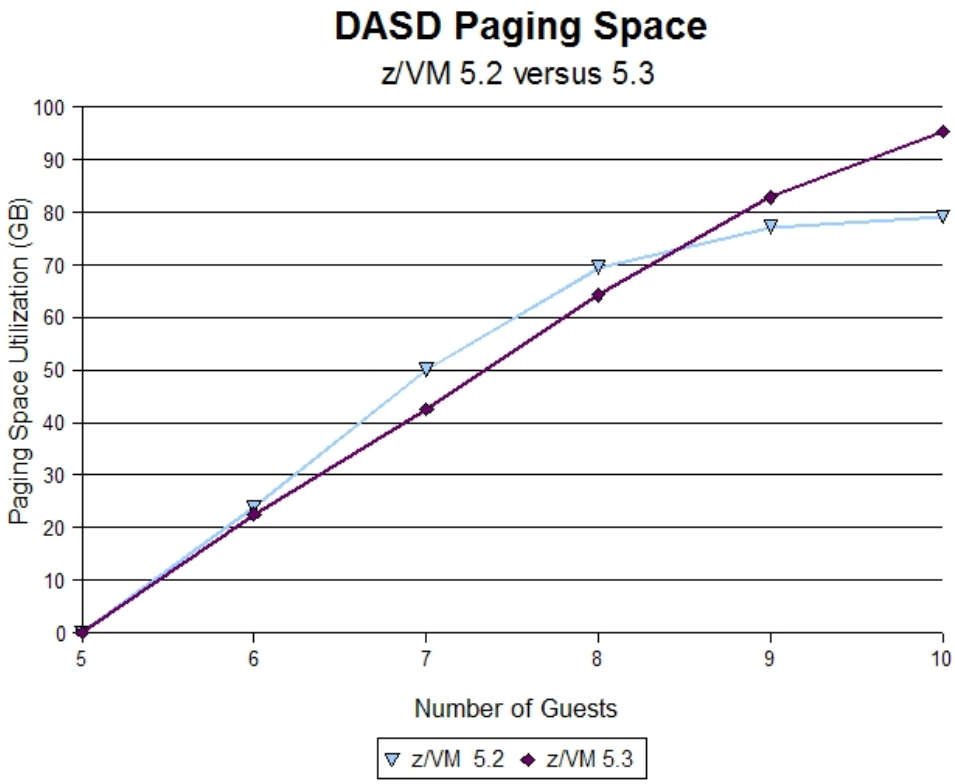


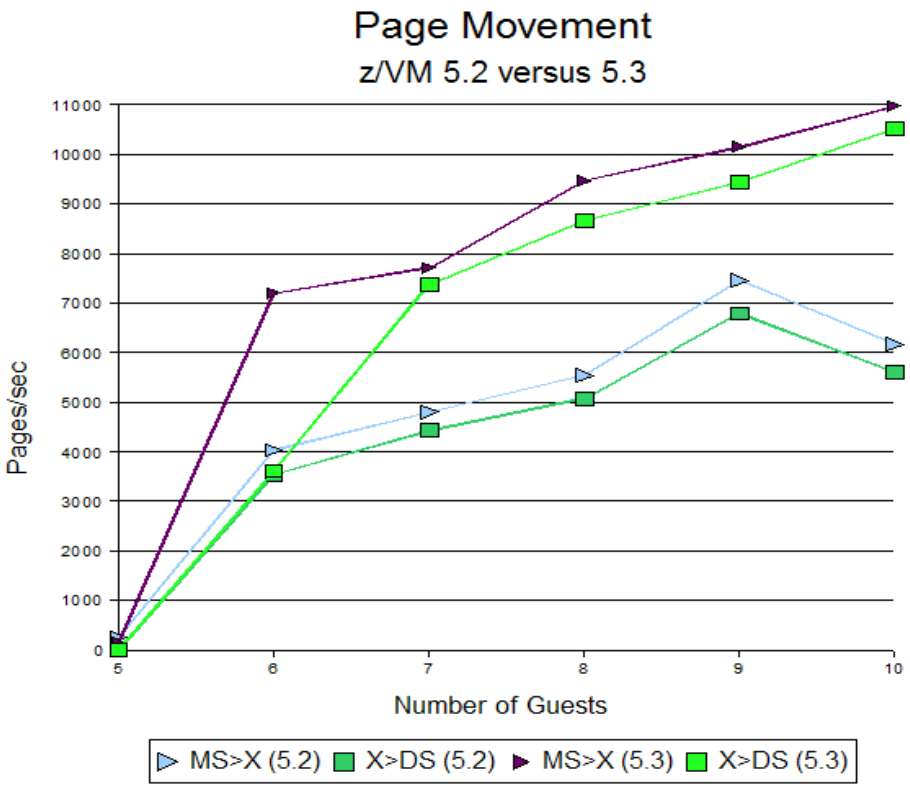
Figure 15. DASD paging space used by z/VM 5.2 versus z/VM 5.3

Observations

z/VM 5.2 shows a knee at eight guests, while the DASD paging space utilization for z/VM 5.3 continues to increase almost linearly with the number of guests.

Page movement

Figure 16 compares the movement between z/VM 5.2 versus z/VM 5.3.



MS>X (5.2 and 5.3): The rate at which page frames were migrated out from main storage to XSTOR
X>DS (5.2 and 5.3): The rate at which page frames were migrated out from XSTOR to DASD

Figure 16. Page movement - z/VM 5.2 versus z/VM 5.3

Observations

Figure 16 again demonstrates that, as memory overcommitment increases, z/VM 5.3 handles more page movements from XSTOR to DASD and main storage to XSTOR than z/VM 5.2. At nine guests, z/VM 5.2 has reached its highest page movement and slows down at ten guests.

Conclusion

Throughput degradation on z/VM 5.3 did not decline as much as z/VM 5.2 and we were able to achieve higher percentages of total CPU utilization. Overall paging performance on z/VM 5.3 is better than z/VM 5.2, in terms of higher throughput and less overhead.

Six guests and a memory overcommitment of 33% resulted in a throughput reduction of only 18%. Even at ten guests, the throughput reduction is only

42% , which shows that the memory overcommitment is being handled much more efficiently on z/VM 5.3 than z/VM 5.2. z/VM 5.3 is able to maintain a higher paging movement rate than z/VM 5.2.

Cooperative Memory Management and Collaborative Memory Management Assist Results

In this section the results obtained for ten guests for z/VM 5.2 and z/VM 5.3 are compared with the results obtained on z/VM 5.3 with VMRM-CMM and CMMA enabled. To achieve these results, we needed to upgrade the Linux system to SLES10 SP1 and apply the fixes listed in Software setup to enable VMRM-CMM and CMMA requests to Linux to perform memory management. We did these runs only at the ten guest scenario to show the impact at the point with the highest memory pressure.

Throughput

Figure 17 shows the throughput for ten guests normalized on the throughput for z/VM 5.2.

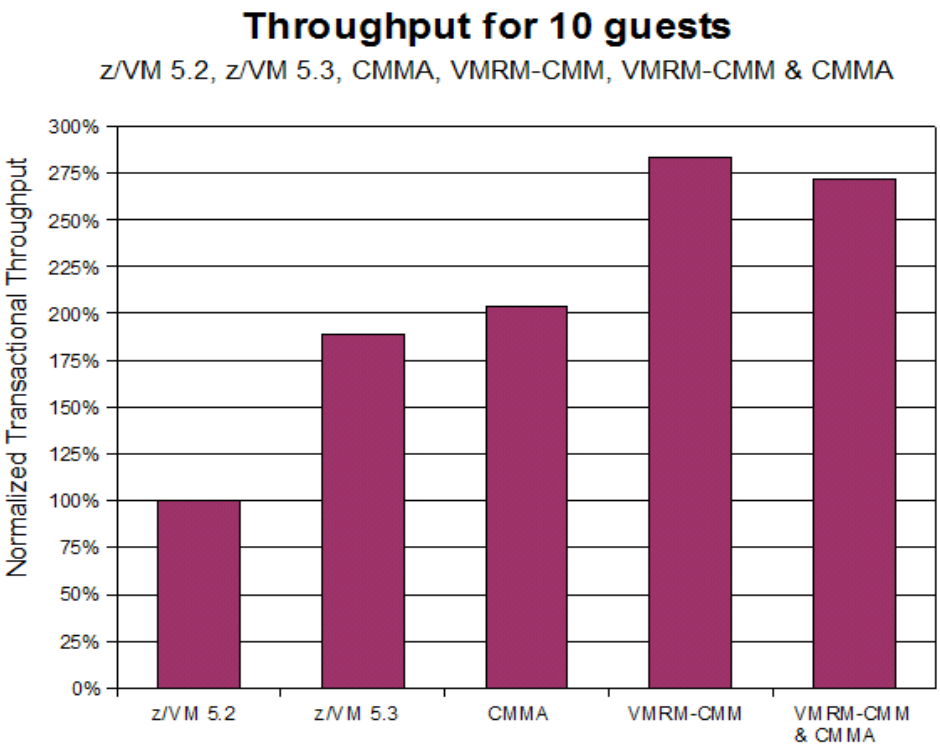


Figure 17. Throughput for 10 guests - z/VM 5.2, z/VM 5.3, CMMA, VMRM-CMM, and VMRM-CMM & CMMA

Observations

VMRM-CMM has almost three times the throughput observed for z/VM 5.2 and CMMA has more than double the throughput of z/VM 5.2. With respect to z/VM 5.3, CMMA has 8% more throughput and VMRM-CMM has 50% more throughput. When compared with the five guests baseline on z/VM 5.3, the ten VMRM-CMM enabled guests achieve 87% of the baseline throughput, while the CMMA enabled guests achieve 63% of the baseline throughput. With both VMRM-CMM and CMMA enabled, the throughput decreased from the run with only VMRM-CMM enabled. It appears that for this workload, CMMA has a small negative impact on throughput when used in combination with VMRM-CMM. The VMRM-CMM and CMMA-enabled combination is not compared further.

CPU utilization

Figure 18 compares the CPU utilization results for ten guests using z/VM 5.2, z/VM 5.3, CMMA, VMRM-CMM, and VMRM-CMM and CMMA combined.

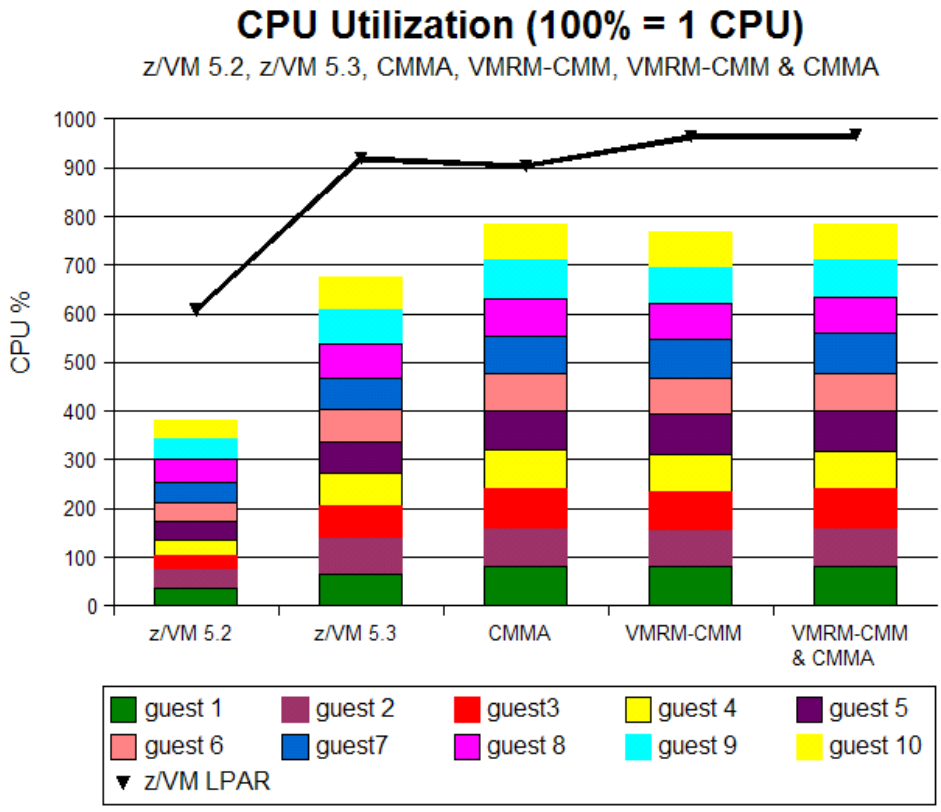


Figure 18. CPU utilization for 10 guests - z/VM 5.2, z/VM 5.3, CMMA, VMRM-CMM, and, VMRM-CMM & CMM

Observations

Enabling either CMMA or VMRM-CMM results in higher CPU utilization than z/VM 5.3 without either of them enabled. However, throughput is also much higher. Both VMRM-CMM and CMMA have lower z/VM overhead, with CMMA having the lowest overhead. The ten guests on VMRM-CMM reached 963% utilization, which means the system is nearly fully utilized. The combination of both VMRM-CMM and CMMA shows a guest CPU utilization similar to the CMMA only case. The overall CPU utilization for the LPAR is about the same as for the VMRM-CMM only case.

DASD paging space

Figure 19 compares the total number of gigabytes of DASD space that are used to hold guest pages on z/VM 5.3, z/VM 5.2, CMMA, VMRM-CMM, and VMRM-CMM and CMM combined.

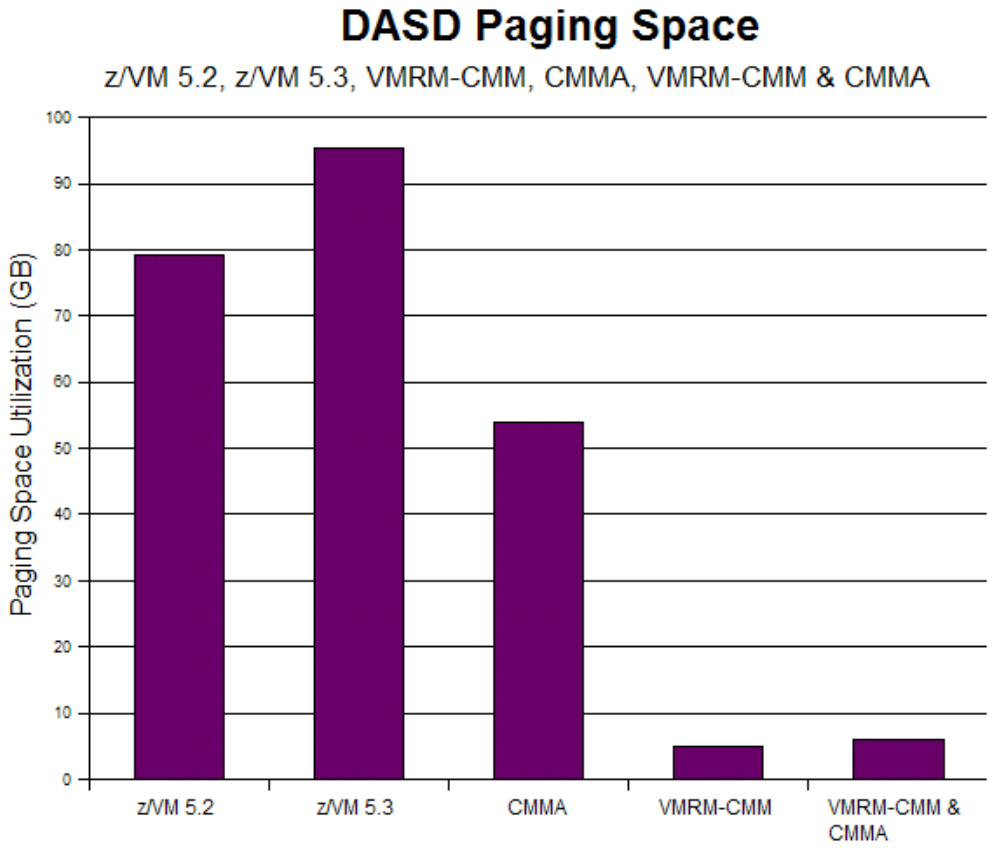


Figure 19. DASD paging space for 10 guests - z/VM 5.2, z/VM 5.3, CMMA, VMRM-CMM, VMRM-CMM & CMM

Observations

VMRM-CMM and CMMA significantly reduces the number of pages moved to DASD paging space. VMRM-CMM uses less than 5 GB of DASD paging space. For the VMRM-CMM and CMMA combination, the paging space is still around 5 GB, but slightly higher than with VMRM-CMM alone.

Page movement

Figure 20 compares page movement between z/VM 5.2, z/VM 5.3, CMMA, VMRM-CMM, and VMRM-CMM and CMMA combined.

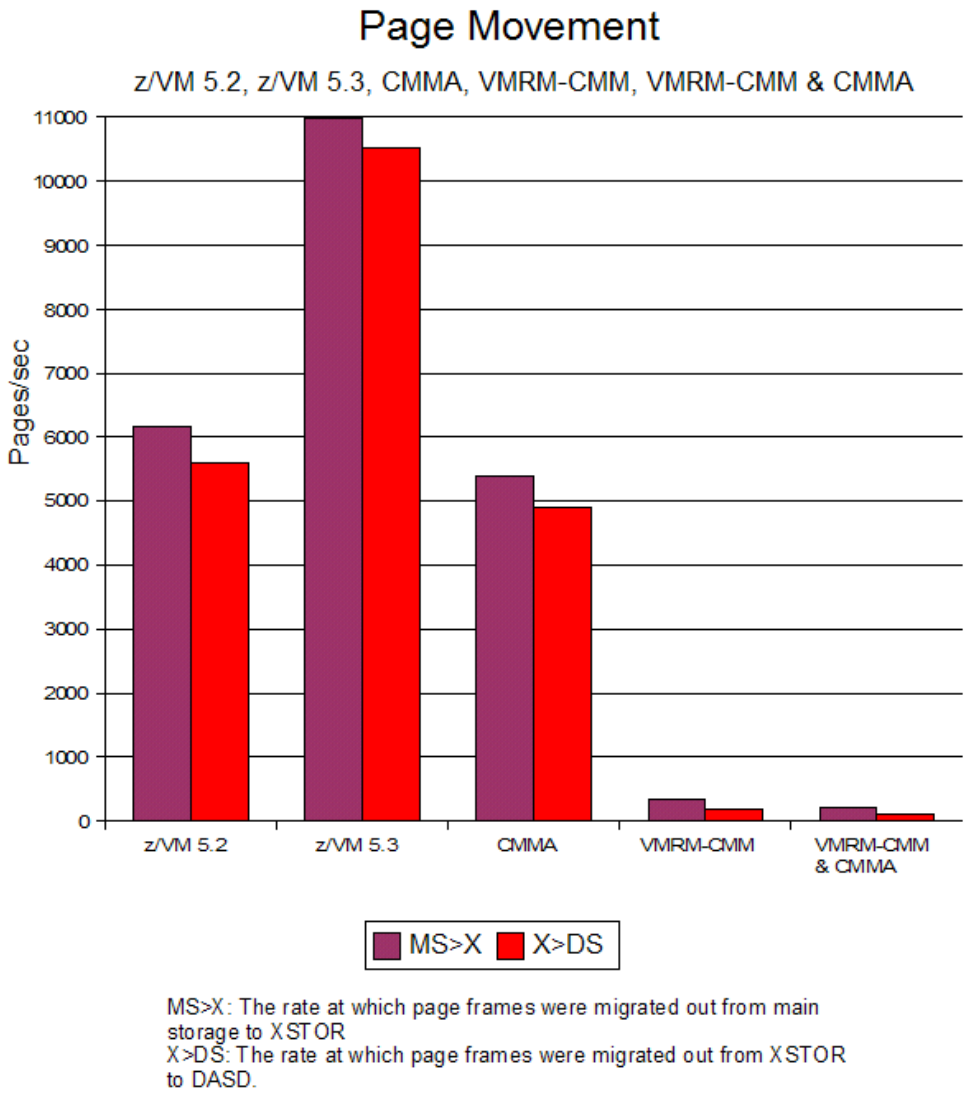


Figure 20. Page movement for 10 guests - z/VM 5.2, z/VM 5.3, CMMA, VMRM-CMM, and VMRM-CMM & CMMA

Observations

Compared to z/VM 5.3, CMMA shows more than a 50% decrease in the rate at which pages are moved and VMRM-CMM shows that page movement rate has decreased to less than 400 pages per second for main storage to XSTOR and less than 200 pages per second from XSTOR to DASD. For the VMRM-CMM and CMMA combination, the page movement rates are similar with a slight decrease, which corresponds with the observed decrease in throughput.

Conclusion

Using VMRM-CMM with z/VM 5.3 produced the best throughput and lowest z/VM paging activity at a planned memory overcommitment of 100%. As Table 7 shows, the real memory overcommitment for CMMA and VMRM-CMM is significantly lower.

The total system memory of 80 GB (20,971,520 pages) is overcommitted as follows:

Table 7. Overcommitted system memory - z/VM 5.3

10 Guests	Pages Not in Memory (= paged out)		Real Over commit (pages used: memory) - 1	Planned Over commit (virtual: physical) - 1	Through put % From Baseline
	XSTOR	DASD	%	%	%
z/VM 5.2	1045785	20754432	104	100	baseline
z/VM 5.3	1039536	26647552	132	100	189
CMMA	1032254	13497344	69	100	204
VMRM-CMM	986883	1215888	11	100	283
VMRM-CMM & CMMA	484979	1480396	9	100	273

While CMMA reduced the paging activity of z/VM, it did not result in a significantly higher increase in throughput. Enabling VMRM-CMM produced significantly better throughput performance and an impressive reduction of the page space utilization. This very good result for VMRM-CMM is expected to be specific for this workload and can not be generalized without further tests for other workloads.

Appendix A. Other Sources of Information

For information on Linux on System z see:
www.**ibm.com**/servers/eserver/zseries/os/linux/

For information on z/VM see:
www.vm.**ibm.com**

For information on z/VM CP and performance see:

z/VM 5.3 CP Commands and Utilities, SC24-6081

z/VM 5.3 Performance, SC24-6109

For information on z/VM Cooperative Memory Management see:
www.vm.**ibm.com**/sysman/vmrm/vmrmcmm.html

For information on IBM open source projects see:
www.**ibm.com**/developerworks/opensource/index.html

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

Produced in the United States of America
12/07

ZSW03029-USEN-00