August 2013

z/VM 6.2 Live Guest Relocation with Linux Middleware - Various Workloads

Table of Contents

August 2013

About this publication

This paper provides results for testing live guest relocation of z/VM® virtual systems running Linux® on IBM System z®.

This is the second of a series of papers and it focuses on workload considerations influencing the relocation behavior of z/VM virtual machines running Linux on System z.

For information about the first paper in this series, see z/VM 6.2 Live Guest Relocation with Linux Middleware.

Authors
Michael Johanssen
Dr. Juergen Doelle

Introduction

The test environment used to produce the results presented in this paper is the same as that used in *z/VM® 6.2 Live Guest Relocation with Linux® Middleware*.

z/VM 6.2 Live Guest Relocation with Linux Middleware provides an overview of z/VM concepts and of the principal test environment.

In this addendum, the analysis focuses on how the relocation characteristics change at the workload level, instead of on the IOCDS and z/VM configuration parameters. How the following different workloads are affected by the relocation process is investigated and how in turn the workload parametrization can affect the relocation process:

1. Java™ workload see (Java workload)
2. File system I/O workload see (File system I/O workload)
3. Database workload see (Transactional database workload)

In addition, the situation is analyzed whereby a mixture of workloads see (Mixed workload) is executed in parallel on the source z/VM system such that CPU resource constraints result on the source z/VM system, and how a live guest relocation of one workload can remedy the constraint situation.

Objectives

The objectives of this project are:

1. Evaluation of the impact of z/VM live guest relocation on workloads running under Linux for IBM System z® in the relocated z/VM virtual machine
2. Evaluation of the characteristics of such workloads on the live guest relocation process
3. Evaluation of the impact of the load situation in the source z/VM host system on the live guest relocation process
4. Evaluation of the impact of the guest size on the live guest relocation process
    - The resulting performance of the live guest relocation process is evaluated through parameters such as:
        - The relocation time and quiesce times
        - The amount of memory transferred during the relocation process

Table 1 lists the notational conventions applied in this paper, in accordance to IEC 60027 2 Amendment 2:

Table 1. Notational conventions

Symbol	Full name	Derivation
KiB	kibibyte	2 ** 10 byte = 1024 byte
MiB	mebibyte	2 ** 20 byte = 1048576 byte
GiB	gibibyte	2 ** 30 byte = 1073741824 byte
KiB/s	kibibyte per second	2 ** 10 byte / second
MiB/s	mebibyte per second	2 ** 20 byte / second
GiB/s	gibibyte per second	2 ** 30 byte / second

Summary

The performance behavior of various kinds of workloads was evaluated while the virtual system running the workload was relocated.

Variations of the workload parameters were inspected with respect to their influence on the live guest relocation process. The objective was to analyze the characteristics of the different workloads in regard to their relocation behavior, especially with respect to the critical quiesce times during which the virtual machine is no longer dispatched.

Another important aspect was how the workload level influences the relocation times. The aim was to provide an impression which workloads are expected to get relocated easily, which workloads might have a higher impact on the relocation process, and where it might make sense to consider additional optimization steps to mitigate that.

The Java workload was found to have the longest relocation times and highest effort required to transfer memory pages, due to the high memory access rates and a widespread memory access distribution which are typical for Java workloads. Objects are frequently created and released on the heap, and a garbage collection process periodically recovers the memory from the released objects.

The file system workload was analyzed in two flavors, with:

1. With page cache I/O
2. With direct I/O

With page cache I/O, data is temporarily cached by the Linux operating system before it is finally written to the I/O device.

With direct I/O, data is directly written to the disk devices, and bypasses the Linux page cache.

In both cases, the throughput rate of the written data was scaled. Using page cache I/O causes higher total relocation and quiesce times with increasing I/O throughput. However, both times are significantly lower than those observed for the Java workload, and with a value below two seconds comparatively small for a 4 GiB guest. The amount of pages to be transferred to the new guest in the page cache case scales with the throughput, while it is constant for the direct I/O case, which explains the shorter quiesce times.

The transactional database workload was found to be a candidate that is easy to relocate. The relocation times scale linearly with the database memory size, with the important quiesce times scaling with a much lower factor. The 4 GiB guest with a database memory of 1.6 GiB had a quiesce time below 1 second, and even a guest with a 22 GiB memory size has a quiesce time below 5 seconds.

At the user level the downtime of the servers during the quiesce time might cause timeouts which might cause the ongoing transaction to fail. In all our relocation tests, Linux and application recovery with the guests was transparent, and processing continued without reporting errors (except time shifts).

Finally it could be very clearly shown how the workloads in a totally overloaded system could be improved by relocating one guest to another z/VM node in the cluster. With the relocation the CPU steal times on the database guests was reduced from about 1.5 CPUs to 0.1 CPUs.

Hardware and software configuration

The z/VM installation is summarized in this section and is basically the same as that used in z/VM 6.2 Live Guest Relocation with Linux Middleware.

Figure 1 shows a symbolic representation of the hardware and software, including the Linux guest configurations that were used. For many tests, only one of the Linux guest configurations shown was actually used.

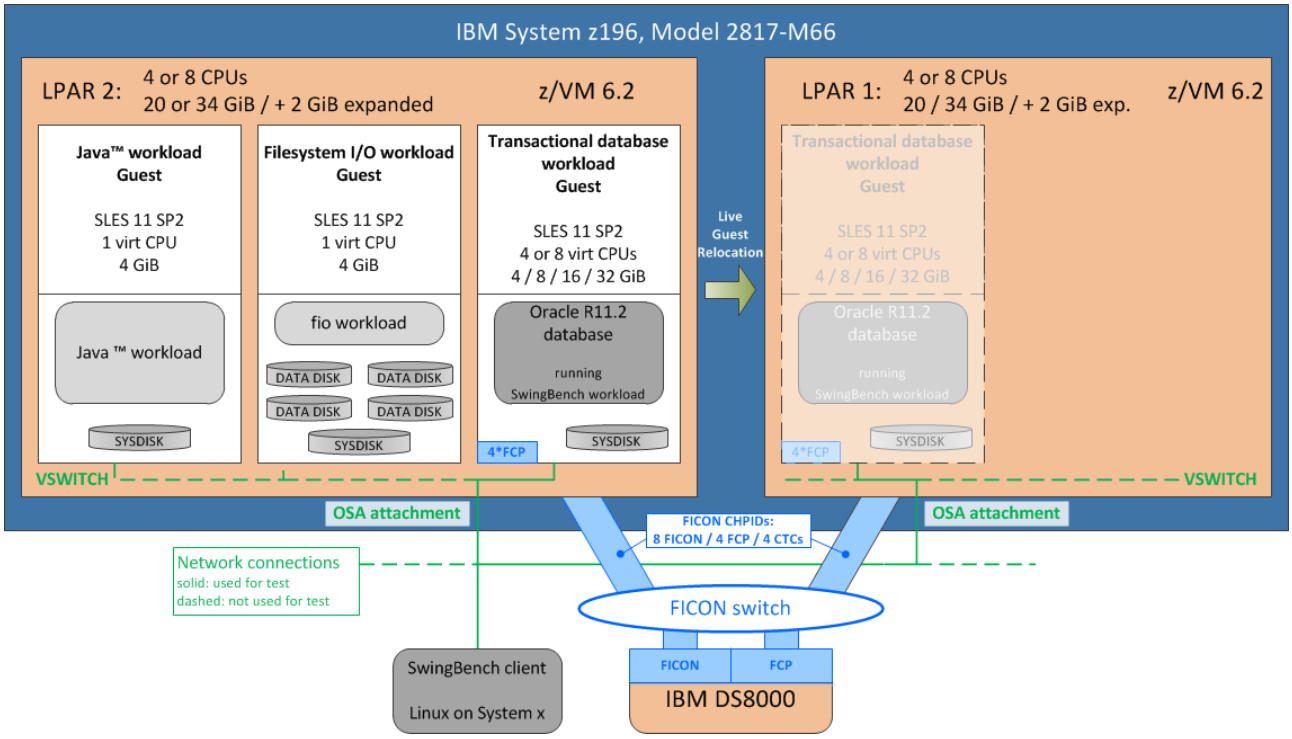

Figure 1. Hardware, software, and Linux guest configuration

Figure 1 also shows how the virtual systems were placed within the z/VM host systems, the network attachment for the guest systems and dedicated ECKD™ disks. SCSI disks are only defined for the transactional database workload guest; SCSI disks are not depicted in Figure 1.

Hardware

The host system was an IBM zEnterprise® 196 (z196), Model 2817-M66.

On that system two LPARs, LPAR1 and LPAR2, were identically configured with 4 shared CPs, 20 GiB central storage and 2 GiB expanded storage each. For some workload tests we used 8 shared CPs per LPAR, and central storage was increased to 34 GiB.

ECKD disk storage was provided on an IBM System Storage® DS8800, Model 951 with 8 host adapters connected with 4 FICON Express8 8 Gbit channel cards via a FICON® switch to the z196.

SCSI disk storage was provided on the same IBM System Storage DS8800, Model 951, with four host adapters connected with 4 FICON Express8 8 Gbit channel cards via a FICON switch to the z196.

System connectivity between the LPARs was provided through two FICON Express4 4Gbit channel cards connected through a FICON switch, using a total of four FICON channels.

The general IOCDS configuration is the same as that used in z/VM 6.2 Live Guest Relocation with Linux Middleware. However, only one specific ISFC logical link configuration was used for this part (see z/VM configuration).

Software

Table 2 lists software components and benchmarks that were used.

Table 2. Software elements

Software element	Version
z/VM	IBM z/VM 6.2 SLU1301 (Product number 5741-A07)
Linux for System z	SUSE Linux Enterprise Server 11 (390x), SLES 11, SP2
Fio	Fio 2.0.8 Committed 2012-05-29, checkout from Git repository: http://git.kernel.dk/?p=fio.git;a=commit;h=cf9a74c8bd63d9db5256f1362885c740e11a1fe5
Oracle Database	Oracle Database 11*g* Enterprise Edition Release 11.2.0.3.0 – 64-bit Production
Client software	
Linux for IBM System x®	SUSE Linux Enterprise Server 11 (x86_64), SLES 11, SP2
SwingBench	Version 2.4.0.845 (Stable). Updated 2011-12-08, Contained in the zip file: http://www.dominicgiles.com/swingbench/swingbench240845.zip

z/VM configuration

Within each of the two LPARs, a z/VM member was configured as part of a common SSI cluster.

This is the same as described in z/VM 6.2 Live Guest Relocation with Linux Middleware. Sufficient paging and spool resources were configured, for details, see section *z/VM configuration*.

An optimized ISFC logical link configuration was configured for connecting the two z/VM members, composed of sixteen straight FCTC connections on four FICON channels.

Note: With a straight FCTC connection the two connected LPARs share the connecting FICON channel because both the FCTC devices on both ends of the FCTC connection are configured in the IOCDS on that *same* FICON channel, for details, see z/VM 6.2 Live Guest Relocation with Linux Middleware, section *IOCDS configuration for FICON CTCs*.

Figure 2 shows a symbolic representation of the ISFC logical link configuration used.

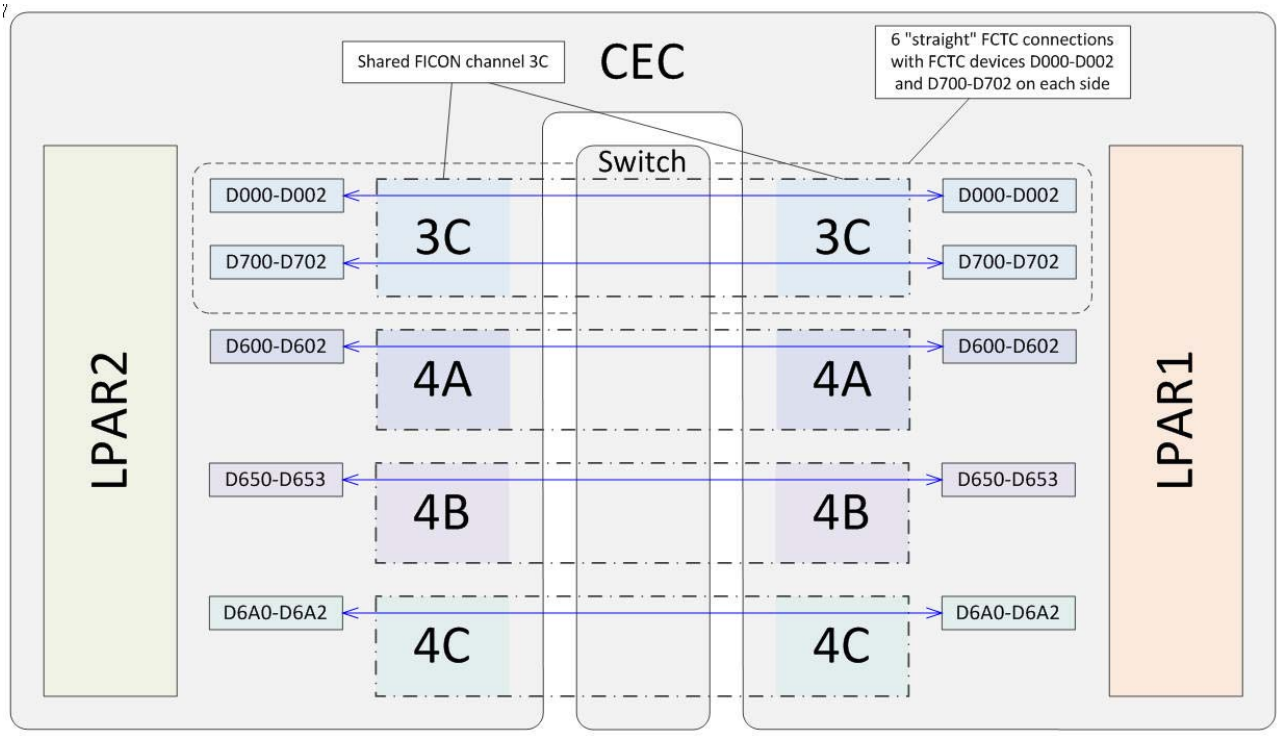

Figure 2. Symbolic representation of the ISFC logical link configuration

The ISFC logical link configuration shown in Figure 2 follows the guidelines provided in z/VM 6.2 Live Guest Relocation with Linux Middleware for an optimal ISFC logical link configuration. Sixteen FCTC connections are defined (exploiting the maximum imposed by z/VM):

- Six FCTC connections are on the FICON channel with CHPID 3C. The FCTC device numbers are separated into two ranges (D000-D002 and D700-D702) which cover both ends of the overall device number range used for the ISFC logical link configuration.
- Three FCTC connections are on the FICON channel with CHPID 4A, with FCTC device numbers D600-D602.
- Four FCTC connections are on the FICON channel with CHPID 4B, with FCTC device numbers D650-D653.
- Three FCTC connections are on the FICON channel with CHPID 4C, with FCTC device numbers D6A0-D6A2.

The use of the two sets of FCTC connections on the FICON channel with CHPID 3C minimizes the impact of z/VM dedicating certain devices for uni-directional use and ensure that under normal load conditions always fourteen FCTC connections are used for relocation data transfer in either direction, as follows:

- All FCTC connections on the FICON channel with CHPIDs 4A, 4B and 4C
- Four FCTC connections on the FICON channel with CHPID 3C, because z/VM designates two FCTC connections for unidirectional use:
- D000-D001 are designated tor the direction LPAR1 -> LPAR2
- D701-D702 are designated for the direction LPAR2 -> LPAR1
- D002 and D700 are used in both directions
- Overall, always fourteen FCTC connections are used in either direction

For further details on ISFC logical link configurations, see z/VM 6.2 Live Guest Relocation with Linux Middleware, section *ISFC logical link configuration*.

Monitor event and sample data collection was enabled for the processor, storage, user, I/O, network, ISFC and SSI domains.

Linux guest configuration

All Linux guests had the following in common:

- A dedicated disk on a FICON attached 3390 Model 27 DASD, formatted with two partitions: One partition for the root file system, and one for swapping
- A virtual NIC attached to a z/VM Virtual Switch (VSWITCH), connected through an OSA adapter to an internal network

Table 3 lists the specific Linux guest configuration variants.

Table 3. Linux guest configuration variants

Workload	# CPU	Guest Size	Additional I/O components
Java	1	4 GiB	-
File system	1	4 GiB	Four FICON attached 3390 Model 27 DASD For each DASD, eight HyperPAV alias devices were configured
Database	4 or 8	4, 8, 16 or 32 GiB	Four dedicated FCP adapters, used to connect to five 64 GiB SCSI disks. Multipathing was configured such that each SCSI disk was accessible through each FCP adapter.

Test approach

The live guest relocation test performed for this paper followed the same pattern used in z/VM 6.2 Live Guest Relocation with Linux Middleware, for details, see the section *Test approach*.

For each test, the following steps were performed:

1. IPL both z/VM members.
2. Configure the ISFC logical link, as described in z/VM configuration.
3. Start z/VM monitoring.
4. Start one or more Linux guest virtual machines running various workloads on LPAR2.
5. Let workloads stabilize for a certain time.
6. Relocate one Linux guest from LPAR2 to LPAR1.
7. Terminate workloads after some time.
8. Gather test data.

Test workloads

The tests are run using Java workloads, file system I/O workloads, transactional database workloads, and mixed workloads.

Java workload

For the Java workload the same internal Java application as in z/VM 6.2 Live Guest Relocation with Linux Middleware was used, for more information, see the section *Test workload*.

The Java workload is used as the base for comparisons. The Java workload parameters are listed in Table 4.

Table 4. Java workload parameters

Parameter	Value
Java heap size	3200 MiB
Java garbage collection algorithm	optthruput

File system I/O workload

For the file system I/O workload fio was used.

fio is an I/O tool intended to be used both for benchmark and stress/hardware verification, for more information, see: http://git.kernel.dk/?p=fio.git;a=blob_plain;f=README;hb=cf9a74c8bd63d9db5256f1362885c740e11a1fe5

It has support for various types of I/O engines (such as sync or libaio), I/O priorities, throughput, forked or threaded jobs, and much more. It can work on block devices as well as files. Fio displays all sorts of I/O performance information.

For our tests, a file system I/O workload was executed within a Linux on System z operating system instance running within the z/VM guest virtual machine that was relocated. The file system I/O workload was varied such that specific memory usage patterns result, using the following parameters:

- Use of page cache I/O versus direct I/O
  Page cache I/O causes high memory access rates because data is written through the Linux page cache. The use of page cache I/O is expected to cause a very different relocation behavior as opposed to not using the page cache.

- Use of asynchronous versus synchronous I/O
  Asynchronous I/O provides for more I/O operations being performed concurrently.
- I/O rate
  The I/O rate controls the maximum amount of data transferred by fio, effectively constraining the memory access rates of the fio processes.

  The I/O rate in terms of MiB/s controls the amount of data transferred by fio. This provides a means to control the effective memory change rate of the fio processes.

  The file system I/O workload was customized for producing specific loads on memory and I/O resources. The workload parameters are listed in Table 5.

Table 5. File system I/O workload parameters

Parameter	Name	Value
Number of fio jobs	numjobs	16
file size	filesize	1 GiB
block size	bs	32 KiB
type of I/O pattern	rw	randwrite
enable direct I/O	direct	0 (page cache I/O) 1 (direct I/O)
Preset throughput per job	rate	not limited 25000 KiB/s 12500 KiB/s 3125 KiB/s

The workload always consisted of 16 fio jobs, each doing random write operations on a file with a size of one GiB and a block size of 32 KiB on an ext3 filesystem.

Direct parameter
The direct parameter controls whether the page cache is used for fio I/O operations.

The page cache is an area of memory maintained by the Linux kernel for the purpose of caching data resulting from user process I/O operations. If the page cache is used, then Linux intermediately caches that data in the page cache. The page cache is flushed at a later point in time, based on internal decisions by the Linux system. The use of the page cache implies a higher memory usage, in the sense of that much more memory pages per second are dirtied. If direct I/O is used, then the data transfer by the fio user process goes directly from the application buffers to the data files, avoiding operating system caching.

Rate parameter
The rate parameter establishes a preset throughput for the fio processes.

In other words, the rate parameter sets the data rate at which the fio processes perform I/O operations. This preset throughput implicitly influences the memory write rate such that with a lower preset throughput less memory pages per second are dirtied.

Transactional database workload

SwingBench was used for the transactional database workload.

For more information, see:
http://www.dominicgiles.com/swingbench/swingbench240845.zip

SwingBench is a free load generator (and benchmarks) designed to stress test an Oracle Database (10*g*,11*g*). The code that ships with SwingBench includes four benchmarks, and these are OrderEntry, SalesHistory, CallingCircle, StressTest. For the tests, the OrderEntry benchmark was selected, which is a transactional workload.

The transactional database workload is driven by an external client (running on a x86 server under Linux) that communicates with the Oracle Database within the z/VM Linux guest using JDBC type IV connectors over the network.

The Oracle Database was configured using ASM as disk storage management tool, as follows:

- Two disk groups: one for data disks and one for log disks.
- Four 64 GiB data disks within the data disk group
- One 64 GiB log disk within the log disk group

The SwingBench oewizard tool was used to create the database elements required for the OrderEntry workbench, selecting a tablespace datafile target within the data disk group, and using a User Defined Scale of 30, resulting in a total size of disk space of about 96 GiB.

The transactional database workload was customized for producing specific loads on memory and I/O resources. The workload parameters are listed in Table 6.

Table 6. Transactional database workload parameters

Parameter	Name	Value
number of SwingBench users	-uc	varied between 32 and 600
virtual system size	vmsize	varied between 4GiB and 32GiB
Oracle Database memory target	memory_target	varied between 1616MiB and 22GiB

The `memory_target` parameter controls the amount of memory used by the Oracle Database by specifying the database system-wide usable memory.

Mixed workload

The relocation behavior of the transactional database workload was investigated when the source z/VM system was under load resulting from several Linux guests running different workloads.

For this type of tests, the LPARs for z/VM systems were configured with 34 GiB of real storage and four CPs.

Figure 3 shows the environment used for the mixed workloads.

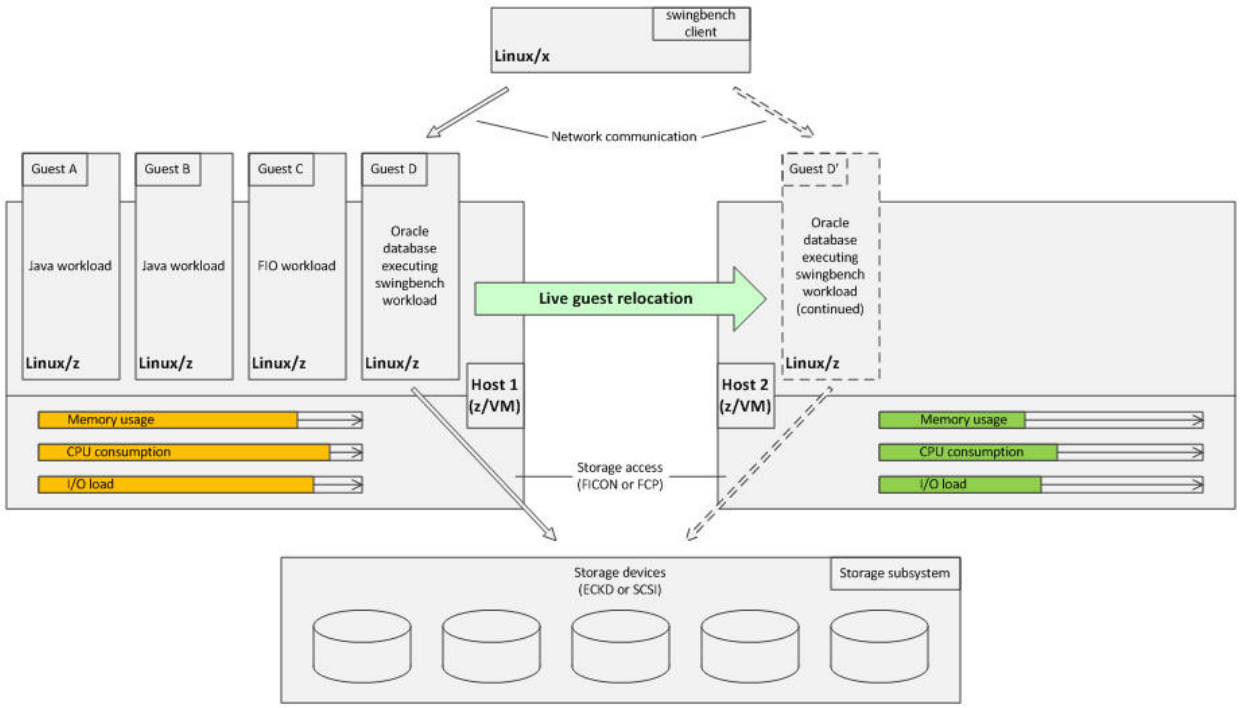

Figure 3. Mixed workload environment

To generate CPU pressure two guests are running the Java workload and one guest is running the page cache file system I/O workload on the source z/VM system (Host 1). In addition, one guest is running an Oracle Database executing the transactional database workload which is driven by the external SwingBench client.

Details about the virtual system configurations are provided in Table 7.

Note: The relative share is adjusted for the number of virtual CPUs, such that every virtual CPU receives the same relative share.

Table 7. System parameters used for the mixed workload scenario

Guest	VMSize	# virt. CPUs	rel. share	Workload
A	4 GiB	1	100	Java
B	4 GiB	1	100	Java
C	4 GiB	1	100	fio - page cache
D	16 GiB	4	400	Oracle Database, running SwingBench OE workload

As usual with our tests, after some time (about seven minutes) the z/VM guest with the Oracle Database running the SwingBench OrderEntry workload is migrated to the target z/VM host (Host 1). It is shown how the live guest migration resolves the CPU pressure situation on the source z/VM system.

Test metrics

The performance of live guest relocations was evaluated using the metrics outlined in this section.

Relocation time
The elapsed time of the complete relocation process.

Quiesce time
The elapsed time while the relocated system is quiesced; during this time, the guest virtual system is prevented from performing any work. The quiesce time is the most prominent metric when evaluating the impact of the guest relocation process, because the virtual system is stopped for that period.

Number of memory passes
The number of passes that z/VM performs when transferring the virtual memory of a guest virtual machine during a live guest relocation. Because the guest virtual machine continues its work while the live guest relocation is performed, that work may cause changes of memory content that was already transferred in a previous pass. Consequently, the modified memory content must be transferred in a subsequent memory pass.

Amount of memory transferred
During the relocation process overall, during various passes of the relocation process, and during the quiesce time. Depending on the update rates, this can be a multiple of the used resident memory.

Linux steal time
The percentage of time a virtual CPU waits for a real CPU while the hypervisor is doing other activities such as servicing another virtual processor

fio throughput
The average throughput (in MiB/s) achieved by all fio jobs together, averaged over the complete test duration, including the time needed for the relocation process.

Test results

The test results together with observations are analyzed in this section.

Reference workload - Java workload

In *z/VM 6.2 Live Guest Relocation with Linux Middleware* a Java-based workload was analyzed with respect to its relocation behavior when various ISFC logical link configurations were used.

For reference purposes, we selected the Java workload results obtained with the ISFC logical link configuration described in z/VM configuration that uses sixteen straight FCTC connections on four FICON channels. This ISFC logical link configuration is used for all tests described in this paper.

Figure 4 shows the relocation and the quiesce times for the Java workload.

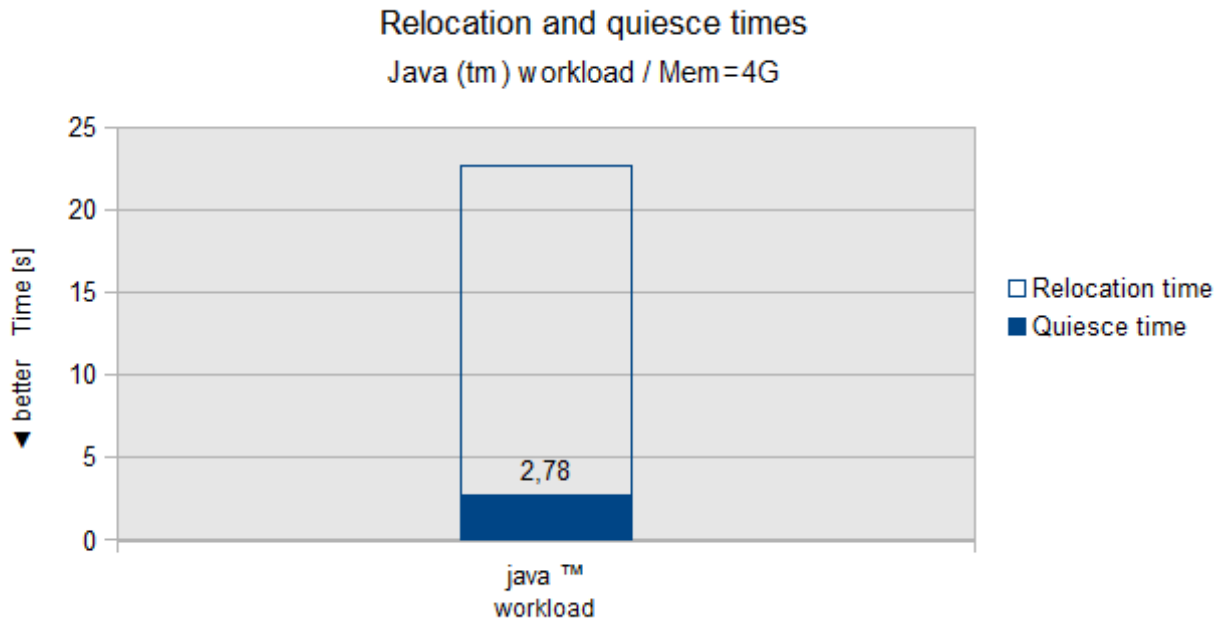

Figure 4. Relocation and quiesce times

Figure 4 shows that for the Java-based workload a relocation time of 22.6 seconds and a quiesce time 2.78 seconds were observed. This result is subsequently referenced for comparison.

File system I/O workload

Tests were run to investigate the influence of scaling the fio throughput and the influence of fio throughput on quiesce times.

File system I/O workload - scaling the fio throughput for page cache I/O

Figure 5 shows how the relocation and the quiesce times scale with respect to the preset fio throughput when page cache I/O is used.

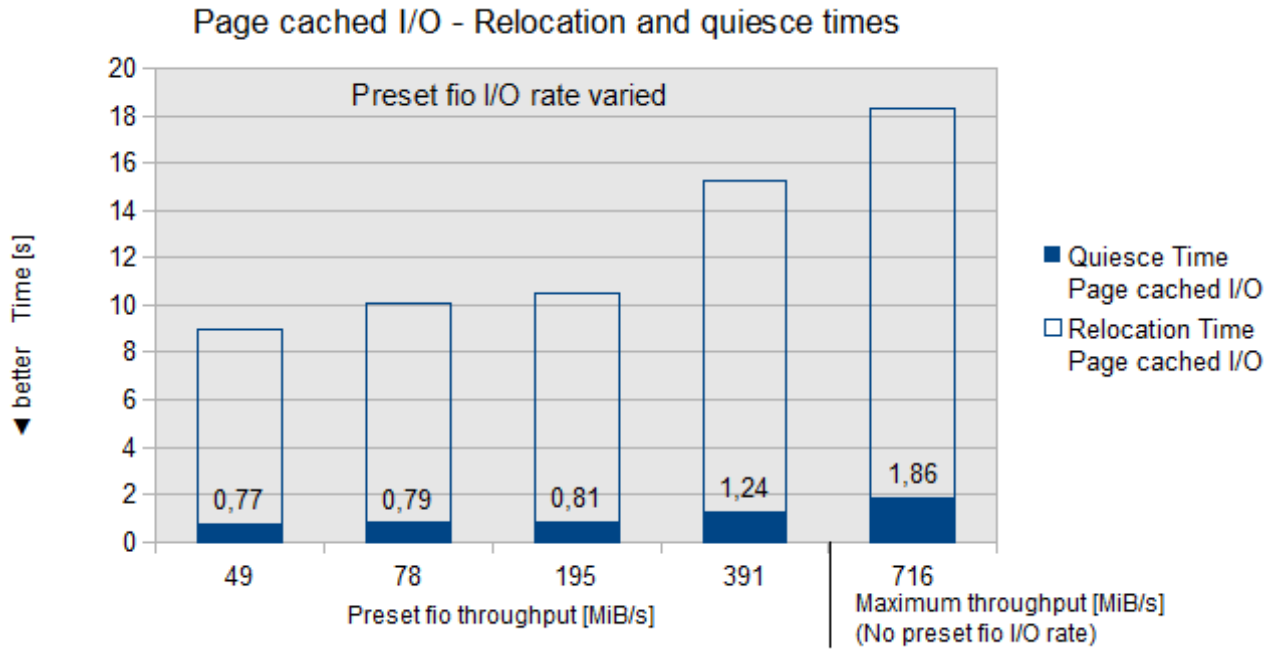

Figure 5. Paged cached I/O

Observation

Figure 5 shows that the relocation and quiesce times gradually rise as the preset fio throughput is increased.

The largest relocation and quiesce times result with the maximum throughput, that is, when no preset fio throughput is established.

Conclusion

The increase of the preset fio throughput causes an increasing number of memory write operations per second into the page cache.

These in turn increase the amount of time that z/VM needs for relocating the virtual system, because memory pages continue to written by the Linux guest at a higher rate while z/VM performs the live guest relocation.

Page cache file I/O is one factor that influences the relocation times. In the range from 49 to 195 MiB/sec the impact is moderate. With higher I/O rates, around 400 MiB/sec and more, the important quiesce time increased significantly but less than the throughput increases.

File system I/O workload - scaling the fio throughput for direct I/O

This figure shows how the relocation and the quiesce times scale with respect to the preset fio data rate when direct I/O is used.

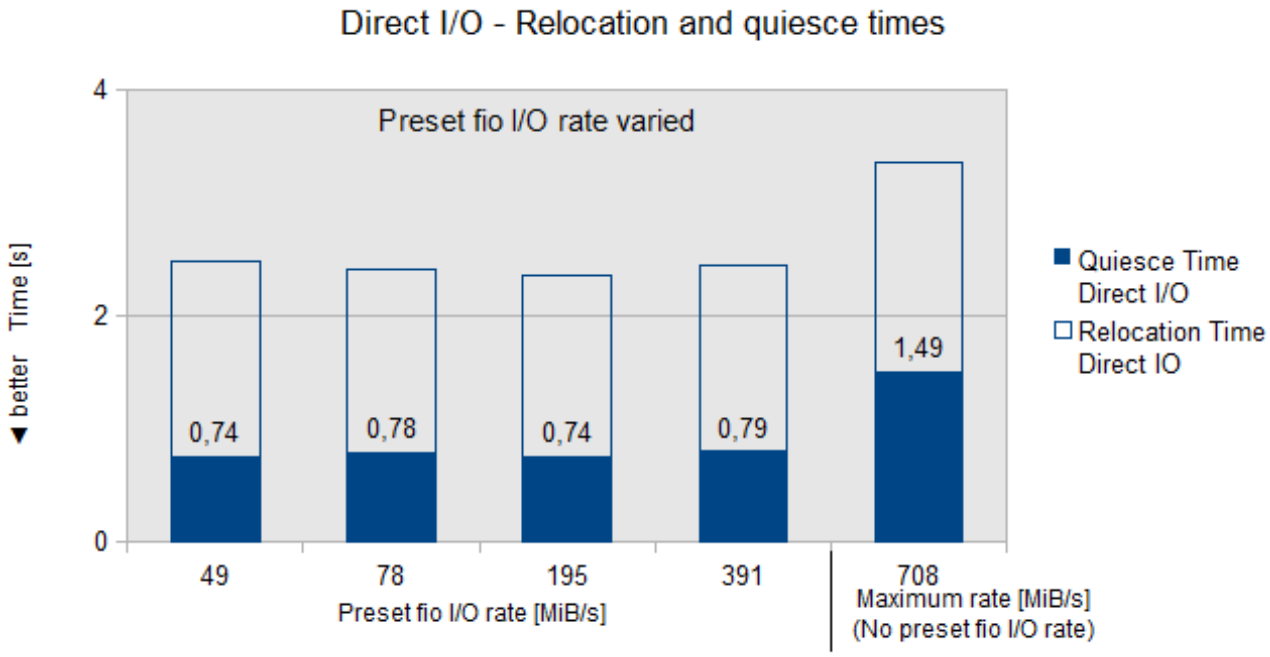

Figure 6. Direct I/O

Observation

For a direct I/O file system I/O workload, the relocation times are much lower than for the page cache file system I/O workload.

On the other hand, the quiesce times in the direct I/O fio case are in the same range as those in the page cache case, with the difference rising as higher preset fio throughputs are set.

For the direct I/O file system I/O workload, relocation as well as quiesce times stay almost constant while the throughput is preset; only with the maximum data rate (when no preset fio throughput is set) the quiesce time is about twice as high as that observed with a preset fio throughput.

A side observation is that the maximum data rate (when no preset fio throughput is set) with direct I/O (708 MiB/s) is only insignificantly smaller than with page cache I/O (716 MiB/s).

Conclusion

With direct I/O, memory consumption is much smaller than in the page cache case.

This is also apparent from the working set sizes reported by z/VM for the virtual machine (not shown in the diagram). With direct I/O, the resulting lower memory consumption (in terms of memory pages written to per second) results in much shorter relocation times.

The preset throughput does not have a significant impact on the relocation and quiesce times except for the case where no preset fio throughput is set (maximum rate), in which case the quiesce times are about twice as high as with preset throughput.

The fact that the maximum throughputs are almost identical for the page cache and for the direct I/O case results from data constantly being written in random order by the fio processes, using the total of 16GiB file space and the duration of the test (10 minutes).

Consider that writing with a data rate of about 700MiB/sec with no cache hits floods a 4GB cache in about 6 seconds. Latest at that point in time all that data lastly must be written to disk, regardless of whether it is temporarily cached in the page cache, or not. However, the slightly higher maximum data rate in the page cache case indicates that in this case the I/O operations might be a little bit more efficient. For example, I/O operations could be combined and regrouped before data is actually written, or even be avoided at all due to cache hits.

Another aspect is that the I/O subsystem on System z configured for the test system is able to process high throughputs in both cases, such that in the direct case no significant device and control unit wait times result.

File system I/O workload - influence of the fio throughput on quiesce time

Figure 7 shows how the quiesce time scales with respect to the preset fio throughput.

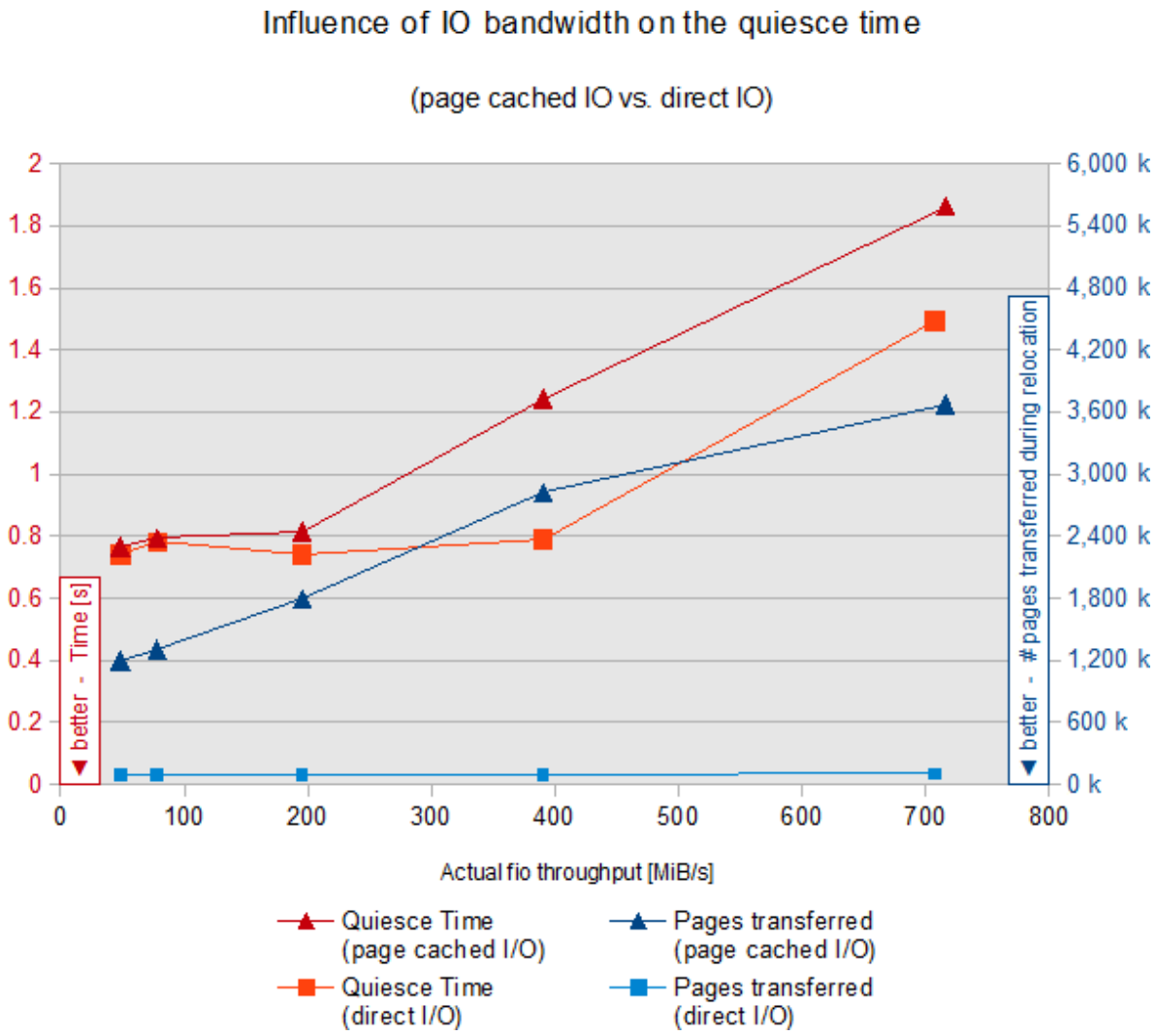

Figure 7. Influence of I/O bandwidth on the quiesce time

Observation

In case of the page cache I/O, the quiesce times (upper red graph) significantly rise as the actual fio throughput increases.

In the direct I/O case (lower red graph) the influence of the filesystem I/O workload on the quiesce time is lower for low fio throughputs, but rises sharply for the maximum throughput of 708 MiB/s.

This is different for the amount of pages transferred during the relocation process: While that number roughly scales with the actual fio throughput in the page cache I/O case (upper blue graph), in the direct I/O case there is no visible influence of the actual fio throughput on the amount of pages transferred (lower blue graph).

Conclusion

The chart shown in Figure 7 yields the following conclusions.

- Higher memory change rates during the relocation causes higher quiesce times, because the memory amounts to be transferred in intermediate and final passes increase.
- A reduced memory change rate during the relocation process can help improve the quiesce times.
- Given the comparatively small amounts of memory transferred in the final passes for the direct I/O file system workload it seems that the constant basic effort for the preparation of the memory transfer dominates the actual transfer times.

Transactional database workload

Tests were run to show the influence of Linux guest size and the Oracle Database on the relocation process as well as the influence the number of SwingBench users has on the relocation process.

Transactional database workload - scaling the Linux guest size and Oracle Database target memory

In this scenario, the influence of the Linux guest size together with that of the Oracle Database `memory_target` parameter on the relocation process is investigated.

Figure 8 shows the relocation times for various Linux guest sizes. The Oracle Database `memory_target` parameter was used to control the amount of memory available to the Oracle Database automatic memory management.

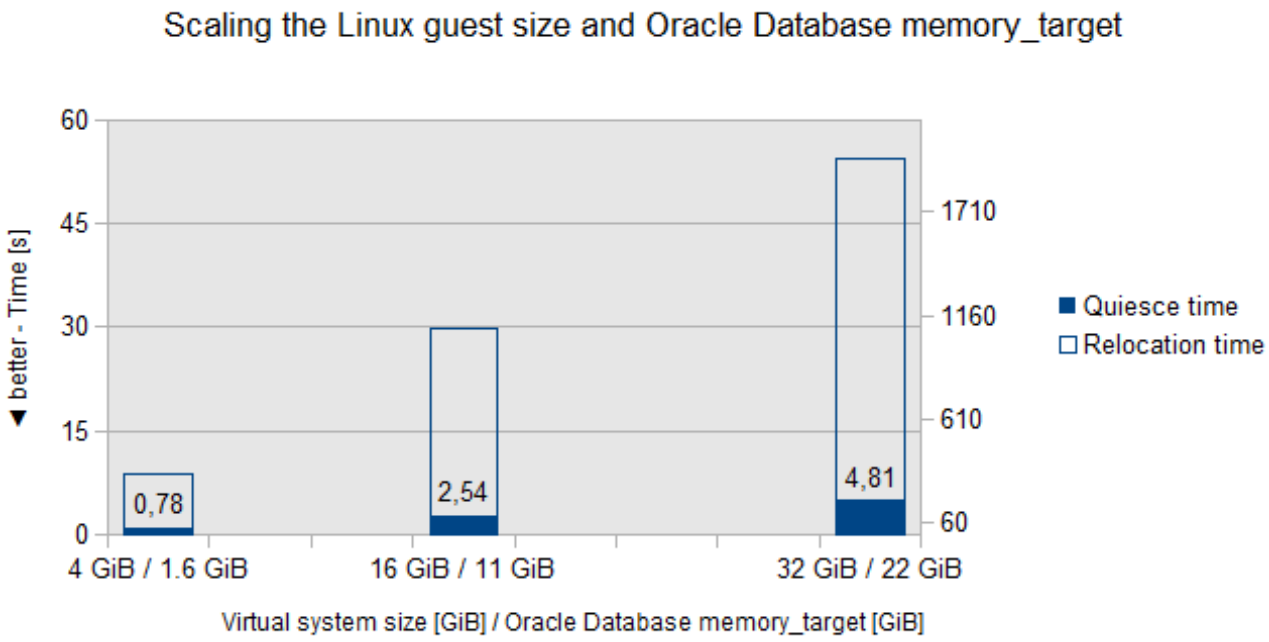

Figure 8. Scaling the Linux guest size and Oracle Database memory target

Observation

The relocation times scale almost linearly with the guest size, and the quiesce times also scale linearly, but with a much lesser scaling factor.

Figure 9 exposes the linear dependency between the `memory_target` parameter and the relocation times more clearly.

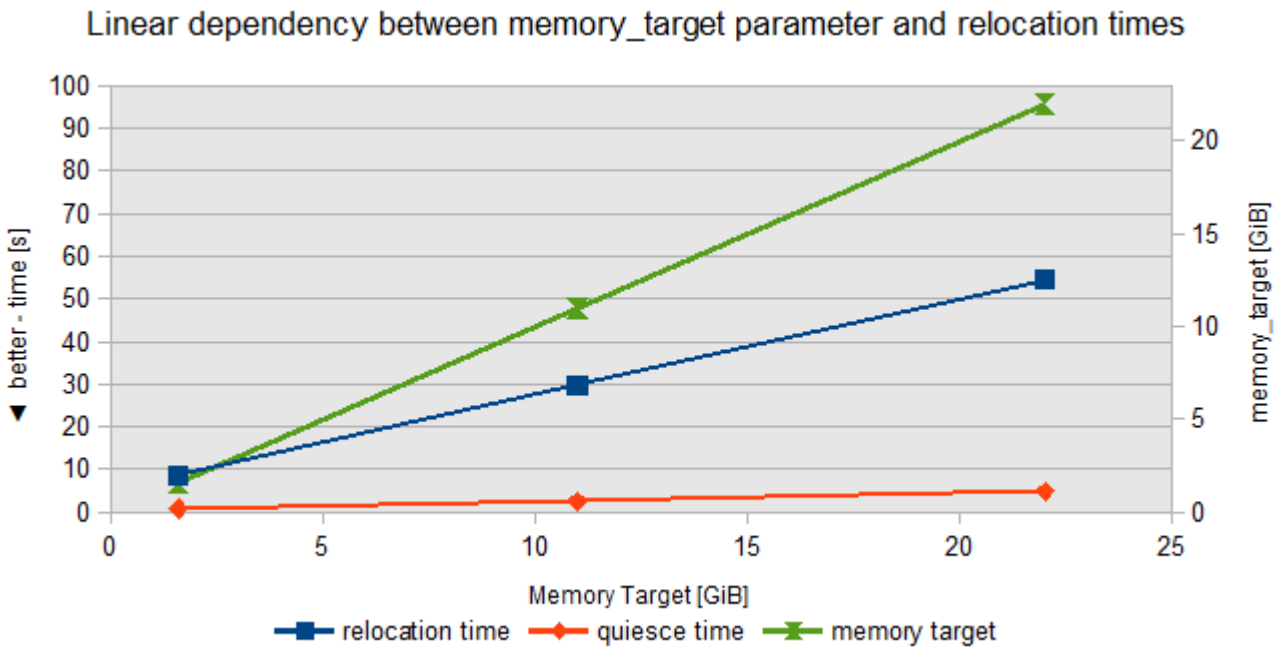

Figure 9. Linear dependency between memory_target parameter and relocation times

Conclusion

As usual, larger guest sizes with their memory actively used cause larger relocation and quiesce times. However, the quiesce times scale with a much lesser factor than the relocation times.

With respect to the used memory size this database workload exhibits comparatively short quiesce times. One reason for this is that most of the memory is read-only, and only a relatively small part of the memory gets modified over time.

Transactional database workload - scaling the number of SwingBench users

In this scenario, how the number of SwingBench users influences the performance of the relocation process is investigated.

To be able to support higher number of SwingBench users, the following system setup is used:

- z/VM host systems: 34 GiB main storage, 2 GiB expanded storage, eight CPs
- Linux guest virtual systems: 32 GiB virtual storage, eight virtual processors
- Oracle Database:
- Disable Oracle Database automatic memory support
- System global area (SGA) size: 13 GiB
- Process global area (PGA) aggregate target: 9 GiB

Figure 10 shows how the relocation and quiesce times as well as the number of SwingBench transactions per seconds scale as the number of SwingBench users is varied.

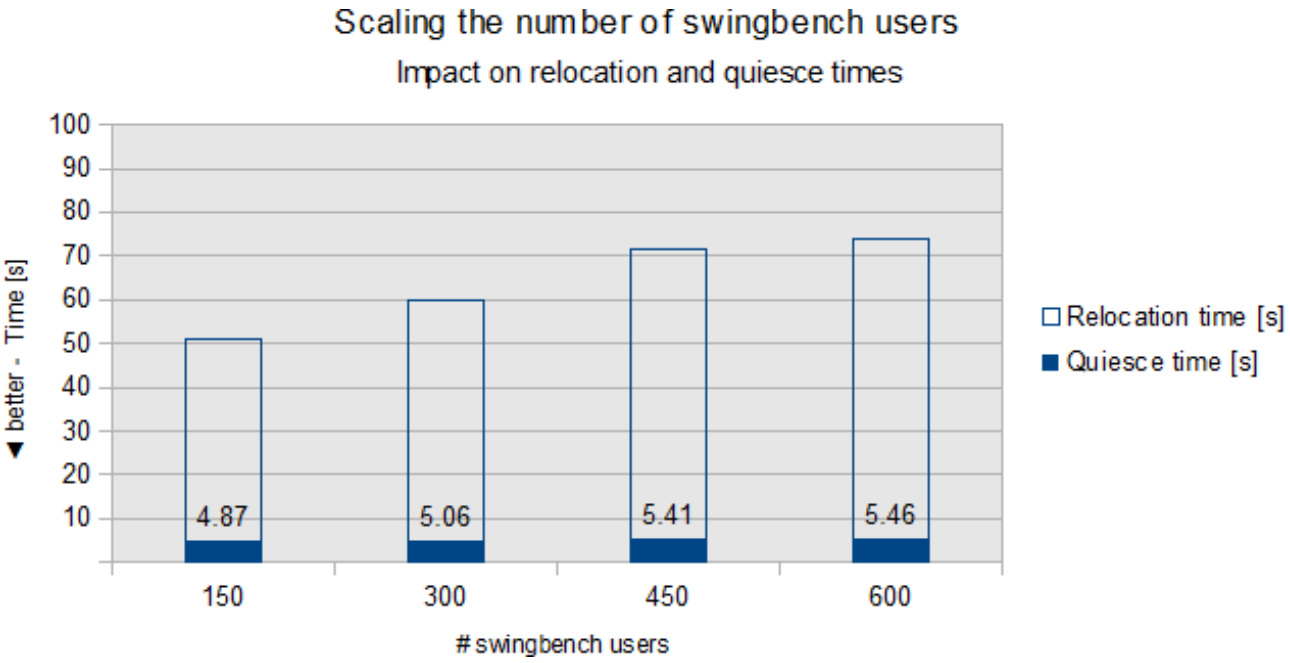

Figure 10. Scaling the number of SwingBench users

Various workloads - number of memory passes and amount of data transferred

To keep the quiesce time (the time during which a guest is stopped during the relocation process) as small as possible, the live guest relocation process performed by z/VM is organized into a number of passes.

During each pass, z/VM scans the memory of the z/VM guest for changed pages, and then transfers these changed pages from the source to the target z/VM system.

For the first pass the changed pages are all those pages that were so far written to by the programs (including the operating system) running within the z/VM guest.

During the intermediate passes (all but the last two ones), the guest virtual system on the source z/VM system continues its work, resulting in new memory changes. z/VM determines the pages that were changed during a particular pass, and transfers these during the subsequent pass.

During the last two passes (named the "penultimate pass" and the "final pass" in z/VM documentation) the z/VM guest is effectively stopped, such that no new memory changes can occur from programs being executed. However, there still may be pending I/O operations that need to be completed and the pages targeted by these I/O operations also need to be transferred. For that reason, two passes are performed while the guest is stopped.

Figure 11 shows how the number of pages transferred during various passes of the relocation process scales when different kinds of workloads run within the relocated z/VM guest.

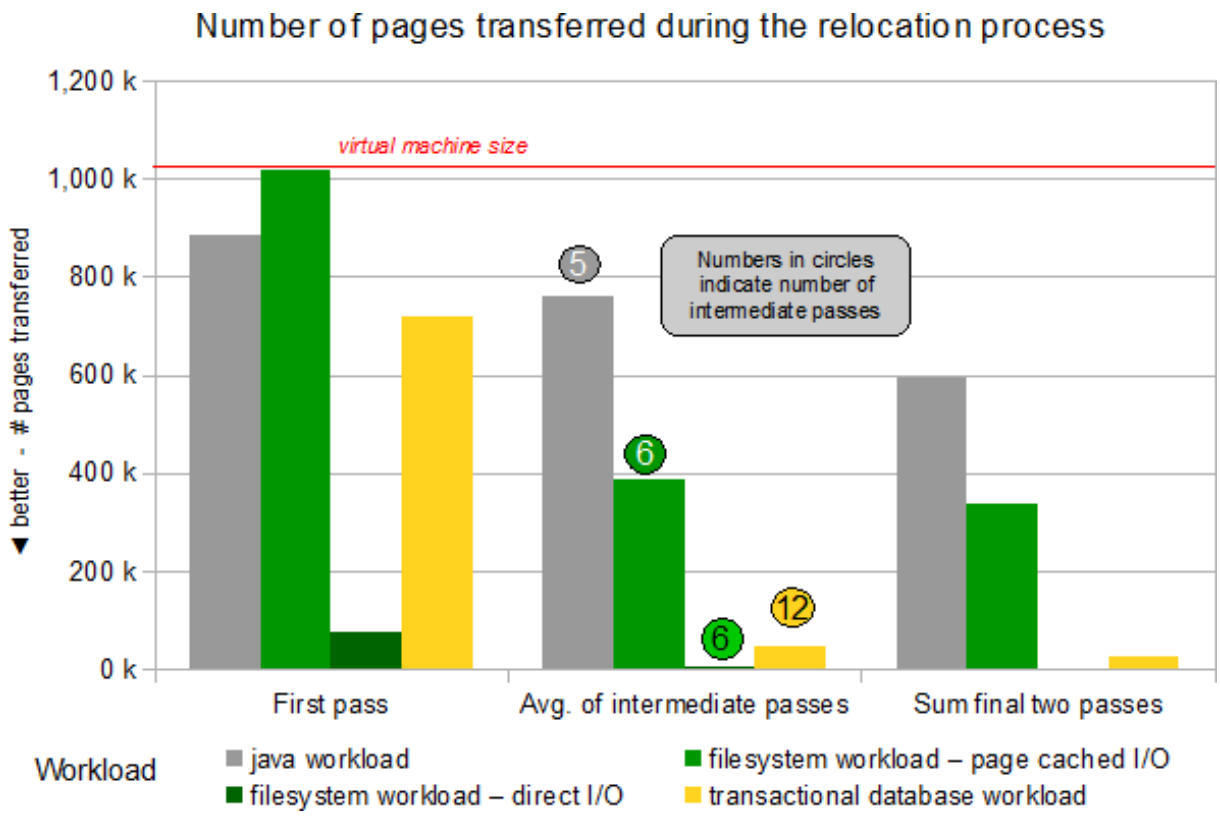

Figure 11. Number of pages transferred during the relocation process for various workloads

The results shown in Figure 11 correlate with the results from various workload that were previously discussed:

- The reference Reference workload - Java workload
- The maximum throughput page cache file system I/O workload in File system I/O workload - scaling the fio throughput for page cache I/O (rightmost in Figure 5)
- The maximum throughput direct I/O file system I/O workload inFile system I/O workload - scaling the fio throughput for direct I/O (rightmost in Figure 6)
- The transactional database workload in Transactional database workload - scaling the Linux guest size and Oracle Database target memory (leftmost in Figure 8)

In all these cases arranged in Figure 11, the virtual system size is 4 GiB (or 1024 k pages), and the size of the Oracle Database memory target is 1616MiB.

Observation

Figure 11 shows three values sets indicating the amounts of memory transferred during various passes of the relocation process scales for different workloads, as follows:

- The leftmost value set shows the amount of memory transferred during the first pass of the relocation process.
- The center value set shows the average amount of memory transferred during the intermediate passes, with the number of intermediate passes indicated within circles shown above the respective columns.
- The right value set shows the sum of the amount of memory transferred in the last two passes while the virtual system is quiesced.

For each workload, Figure 11 provides details about the number of pages transferred in various passes of the relocation process, as follows:

- The Java workload (leftmost position in value sets, grey color) exhibits an amount of memory of 887 k pages (or 3548 MiB of memory) transferred in the first pass. During the intermediate passes, the average number of pages transferred stays comparatively high at about 759 k pages, and the sum of pages transferred during the final two passes (594 k pages) is again the highest value observed for the set of observed workloads. The number of intermediate passes is the lowest for the set of observed workloads.
- The page cache file system I/O workload (second position in value sets, bright green color) exhibits the highest amount of 1019 k pages of memory transferred in the first pass of the relocation process - in other words, almost all of the virtual machine memory is transferred. During the intermediate passes, the average number of pages transferred is substantially lower (386 k pages). The sum of pages transferred during the final two passes (336 k pages) is still significant. The number of intermediate passes is just one pass more than that for the Java workload.
- The direct I/O file system I/O workload (third position in value sets, dark green color) exhibits the lowest amount of 74 k pages (296 MiB) of memory transferred in the first pass of the relocation process, and extraordinarily low numbers of pages transferred during the intermediate (4350 pages) and final (3185 pages) passes. However, the number of intermediate passes is the same as that observed for the page cache filesystem I/O workload.
- The transactional database workload (rightmost position in value sets, yellow color) exhibits a moderate amount of 721 k pages transferred in the first pass of the relocation process, which is about two thirds of the virtual memory. During the intermediate and final passes small amounts of pages (49 k pages and 24 k pages) are transferred. The number of intermediate passes is high, twice as much as that for the filesystem workloads.

Figure 12 shows that relocation and quiesce times for the workloads.

- The Java workload exhibits the highest times of 22.6 s and 2.8 s.
- The relocation time for the page cache file system I/O workload is with 18.3 s slightly shorter than that of the Java workload, and the quiesce time with 1.86 s is about 66 % of that of the Java workload.
- The direct I/O file system I/O workload exhibits the shortest relocation time of only 3.36 s, but the quiesce time of 1.49 s is only about 20% less than that of the page cache case.
- The transactional database workload has a relocation time of 8.65 s, and a quiesce time of 0.78 s.

Conclusion

The amount of pages to be transferred in the initial pass depends on the number of pages ever touched by the virtual system before the relocation process started.

In other words, this metric is influenced by the history of activities performed by the virtual system before the relocation process started.

- For the Java workload, the amount of memory of 887 k pages (3543 MiB) transferred in the first pass roughly relates to the configuration of the Java heap size (3200 MiB) and space allocated by the Linux operating system.
- For the page cache file system I/O workload, the high amount of 1019 k pages of memory transferred in the first pass of the relocation process indicates that apparently the virtual memory of that guest was fully utilized for the page cache at that point in time. Note: The amount of 1019 k pages (4076 MiB) covers almost all of the guest virtual storage size of 4096 MiB.
- The direct I/O file system I/O workload, the low amount of 74 k pages (296 MiB) of memory transferred in the first pass relates to the set of buffers used by fio for processing the direct I/O operations.
- For the transactional database workload, the amount of 721 k pages (2884 MIB) transferred during the first pass relates to the memory target configured for the Oracle Database server (1616 MiB) plus additional memory used by the database and by the operating system.

The **ratio** of the average amount of pages transferred in intermediate passes against the number of pages transferred in the first pass gives an indication of the memory write rate and/or memory access distribution of the workload during the relocation process:

- For the Java workload, this ratio is **86%**. This indicates a high memory write rate and/or memory access distribution. High memory write rates and a widespread memory access distribution are typical for Java workloads

where frequently objects are created and released on the heap, and a garbage collection process periodically recovers the memory from the released objects. As can also be seen from the comparatively high relocation and quiesce times shown in Figure 12, high memory write rates along with a widespread memory access pattern are detrimental to the relocation process.

- For the page cache I/O file system I/O workload, the ratio is **38%**. This indicates a moderate memory write rate and/or memory access distribution. Both can be attributed to the ongoing use of the page cache where data written by the fio user processes is temporarily stored before being flushed to respective I/O devices by the Linux guest operating system.
- For the direct I/O file system I/O workload, the ratio is **5.7%**. This indicates a low memory write rate and/or a close memory access distribution. This results from the fio processes working only on their I/O buffer and, once prepared, writing these buffers out to the disk storage before using them again for the next write operation. In other words, the memory access distribution in this case is rather limited to the set of I/O buffers. Within the set of observed workloads, this workload has the lowest memory requirements.
- For the transactional database workload, the ratio is **6.7%**. This indicates a low memory write rate and/or close memory access distribution. Again, it can be concluded that this is attributed to the database managing its data access through a set of buffers that are frequently reused, keeping the memory access distribution mostly within these buffers and retaining the data as long as possible.

The number of memory passes depends on the memory access behavior performed by the workload during the relocation process:

1. For the Java workload it was observed that eight memory passes were used (one first pass, five intermediate passes and two final passes).

z/VM documentation states that z/VM uses a default of eight memory passes if the aim of reducing the quiesce time by means of additional intermediate passes does not seem attainable, see
http://www.vm.ibm.com/perf/reports/zvm/zvmperf.pdf

- For the page cache file system I/O workloads, higher numbers of memory passes are typically observed . While not shown in Figure 11 , particularly in cases where a preset fio throughput was established (refer to Figure 5), the number of memory passes was between twelve and sixteen. Apparently in these cases the reduced memory modification rate per second lets z/VM perform additional memory passes, successfully minimizing the quiesce time.
- For the direct I/O file system I/O workload, nine memory passes were also observed. Apparently the amount of data transferred during the intermediate passes does not relate to the number of intermediate passes. In other words, nine passes are still seen in this case, but the amount of data transferred per pass is down by a factor of 100 as compared to the page cache fio case. Of course, the duration of each pass is much lesser than in the page cache file system workload case.
- For the transactional database workload, 15 memory passes were observed. Apparently, the transactional database workload nicely fits with the assumptions made by z/VM optimization approaches for keeping the quiesce time low by extending the number of memory passes.

Figure 12 presents the relocation times for the same set of workloads as those shown in Figure 11.

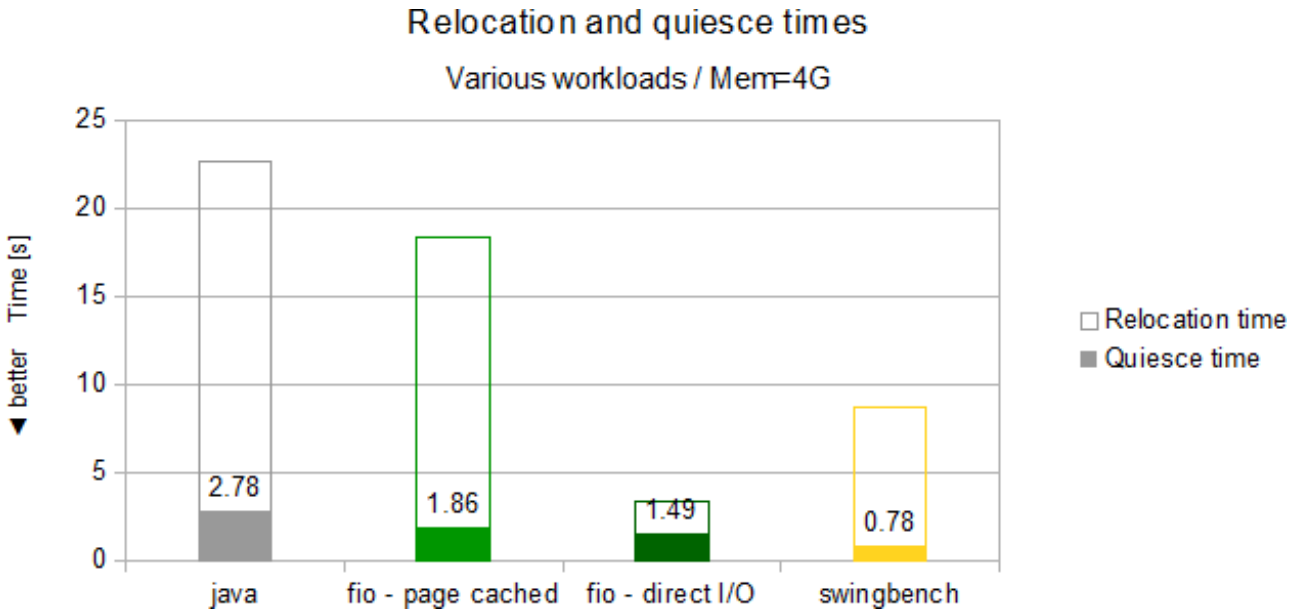

Figure 12. Relocation and quiesce times for various workloads

The amount of pages transferred in the final two passes (Figure 11), and the quiesce times (Figure 12) depend on both the size and the activity of the virtual system. However, the quiesce times also seem to have a base component:

- For the Java workload, 594 k pages were observed as being transferred during a quiesce time of 2.78 seconds. These are the highest numbers in our set of workloads, indicating a high degree of memory usage during the progress of the relocation process.
- For the page cache file system I/O workload, 336 k pages were observed as being transferred during a quiesce time of 1.86 seconds. As shown previously (see Figure 5), reducing the fio throughput, results in lower quiesce times, because in the page cache case a lower throughput implies a lower memory modification rate.
- For the direct I/O file system I/O workload, 3.1 k pages were observed as being transferred during a quiesce time of 1.49 s. Surprisingly, the quiesce time stays in the same region as that for the other workloads. Apparently, when only comparatively few pages are to be transferred, activities other than the actual data transfer become more significant for the quiesce time. Furthermore, particularly for a disk I/O benchmark, time is needed until all outstanding disk I/O requests are terminated, also the activation of the devices on the target z/VM.
- For the transactional database workload, 24 k pages were observed as being transferred during a quiesce time of 0.78 s. These are very small values for an active 4 GiB virtual system, and are attributed mostly to the large number memory passes and the locality of memory changes due to special buffers in the database. It can be concluded that the transactional database workload on the Oracle Database is well suited for being relocated while database activity continues.

Curing CPU constraints by offloading guests with high CPU load

In this scenario how the CPU pressure generated by several Linux guest systems working in parallel on the source host system can be alleviated by relocating the Linux guest with the highest CPU load (the transactional database workload) is investigated.

For this analysis, the mixed workload environment was used see (Mixed workload); particularly, each LPAR for the z/VM hosts was configured with 34 GiB of real storage and with four CPs (see Figure 1).

Mixed workload - analysis of the CPU usage of the relocated transactional database workload

The virtual system running the transactional database workload is configured with 4 virtual CPUs, and is assigned a relative share of 400% such that each virtual CPU receives a relative share of 100%.

Figure 13 shows the CPU usage generated by the transactional database workload before, during, and after the live guest relocation process.

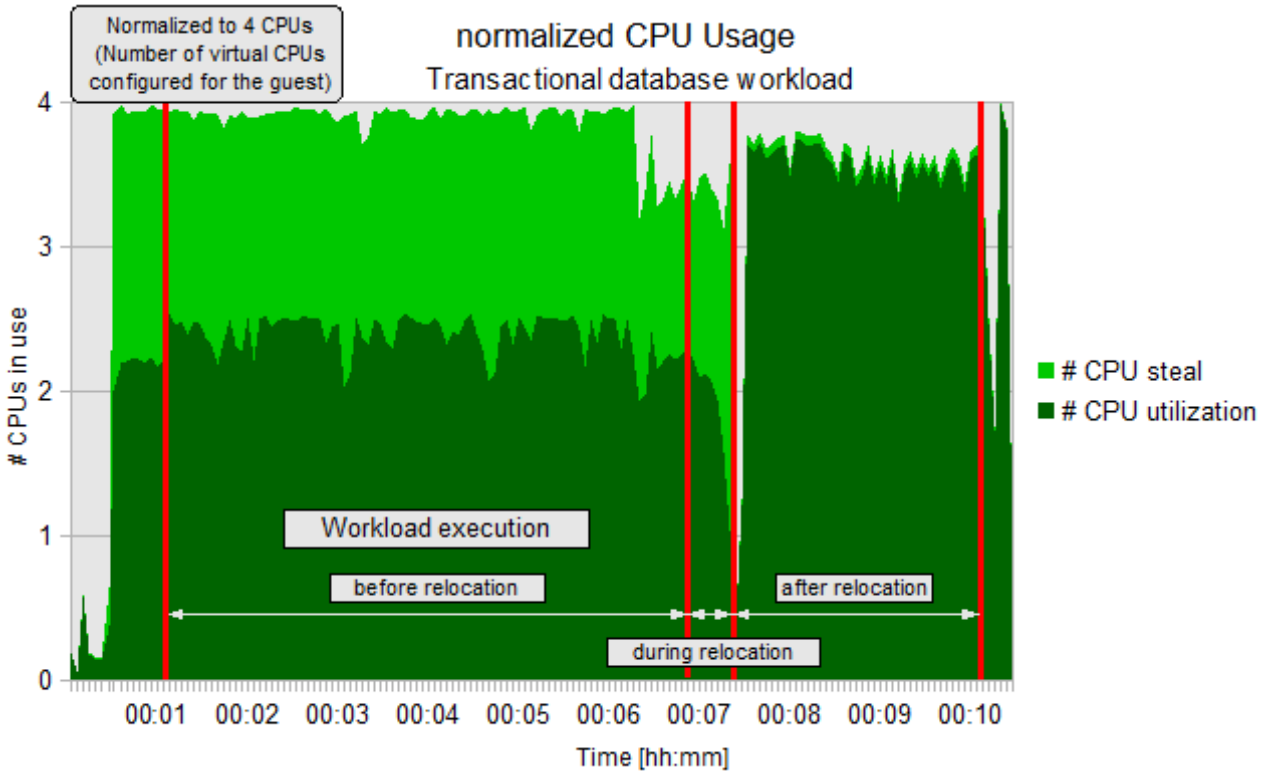

Figure 13. CPU usage generated by the transactional database workload

Observation

The utilization values (dark green lower area) in the previous figure show the active CPU usage, in units of CPUs based on Linux sar data.

That way, the utilization values show which fraction of the available four virtual CPUs was available and actively used for processing the transactional database workload. On the other hand, the steal values (bright green upper area) show how much CPU power was not available in spite of a CPU demand by the virtual system because z/VM had to perform other activities such as servicing another virtual processor or perform other hypervisor tasks. In other words, the CPU power reflected by the steal value was not available for processing the transactional workload. The remaining free space (grey area on top) reflects CPU capacity not requested from the guest.

It can be seen that after an initialization phase between 2 and 2.5 CPUs are consumed. As the relocation process starts after about seven minutes, during the initial phase of the relocation the utilization values first slowly and then progressively decrease. At the same time, the steal values increase correspondingly.

One additional point must be mentioned here: It is known that for the testcase shown in Figure 13 the quiesce time was 2.2 seconds. During the quiesce time, the virtual processors of the guest system are not dispatched by z/VM, such that during the quiesce time the number of CPUs covered by the steal value would have to show a value of 4 CPUs. However, the sampling rate for the Linux sar values is six seconds. Thus, the resolution for the values shown in Figure 13 is also six seconds. For that reason, the effective stoppage of the virtual processors during the quiesce time is not really visible from Figure 13.

After the relocation is completed, a sharp rise of the utilization values can be observed, followed by a phase of much larger utilization values between 3.3 and 3.7 CPUs. At the same time, the steal values now remain almost insignificant below 0.1 CPUs.

Conclusion

The left part of Figure 13 clearly reflects the CPU constrained situation for the Linux guest running the transactional workload, as visible through the comparatively high steal values. In this phase CPU demand is higher than the available processing capacity. This causes stolen" CPU capacity to appear to the guests, that is, the guests experience elapsed time intervals where their virtual CPUs appear stalled, because the real CPUs are needed for other guests or for work done by z/VM.

The center part of Figure 13 shows the impact of the relocation operation itself, while z/VM needs additional processing resources to perform the relocation process. At the end of the relocation phase the steal value would reach the maximum possible value of four CPUs for the duration of the quiesce phase. However, this is not visible from Figure 13 because during the quiesce time the guest is stopped, and thus Linux statistics data is not collected and thus not represented in Figure 13.

Once running on the otherwise unused target z/VM host, the relocated guest first needs a small amount of time to resume its workload processing, but then can make almost full use of its virtual processors, as reflected by the very low steal value.

Mixed workload - analysis of the CPU usage of the other workloads on the source z/VM system

Two virtual systems running the Java workload, and one virtual system running the file system I/O workload. Each virtual system is configured with one virtual CPU, and is assigned the default relative share of 100%.

Figure 14 shows the CPU usage generated by the other workloads running on the source z/VM system:

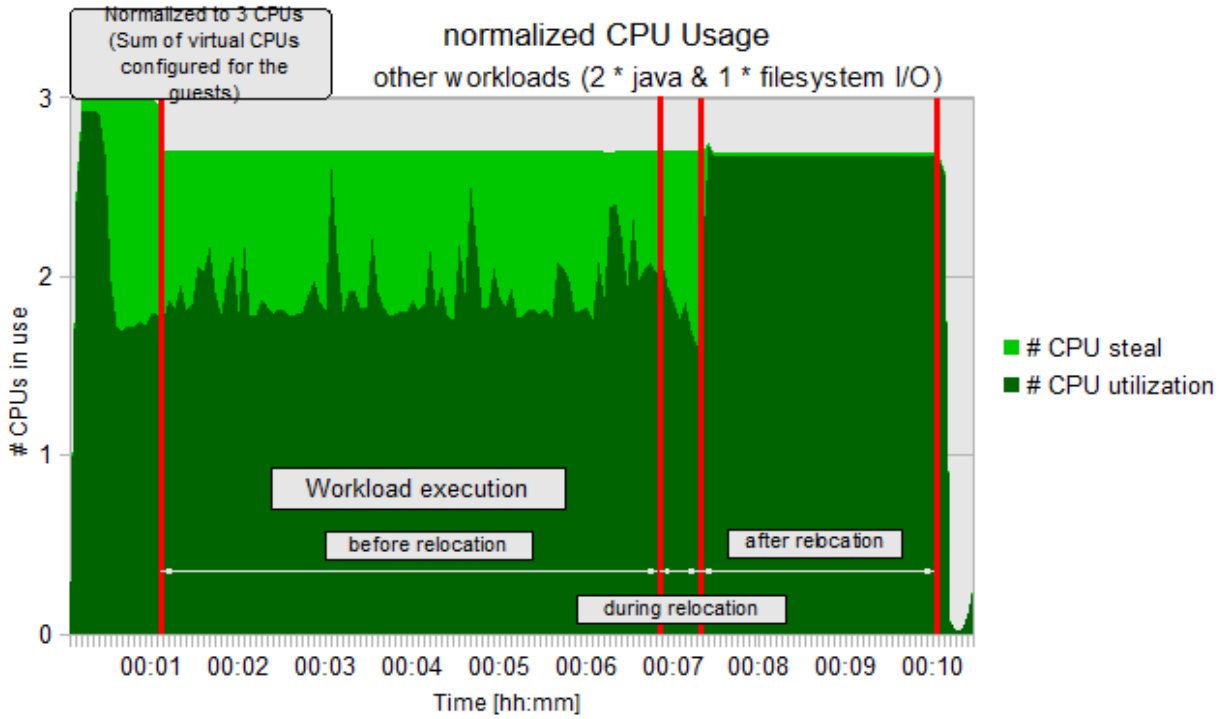

Figure 14. CPU usage generated by other workloads

Observation
The transactional database workload is depicted using the same scheme as previously used.

In Figure 14 the utilization values (dark green lower area) show the active CPU usage of the three other guest systems that comprise the mixed workload, in units of CPUs. CPU usage is based on Linux sar data from the three guest systems. Likewise, the steal values (bright green upper area) show how much CPU power was not available in spite of a CPU demands by the guests because it was needed elsewhere within the source z/VM system.

It can be seen that after an initialization phase around 1.8 CPUs (with spikes up to 2.5 CPUs) are consumed by the three other systems. Above the steal curve, an almost constant area reflecting unused CPU resource remains.

As the relocation process for the guest running the transactional database workload starts after about seven minutes, the steal value for the remaining guests exhibits a small increase at the expense of the utilization value can be observed. However, there is no sharp decline of the utilization value as the relocation process reaches its end (as was the case for the relocated transactional database workload).

As the relocation process is completed, the utilization value of the remaining guests sharply rises to a value of about 2.7 CPUs, while at the same time the steal values start remaining almost insignificant below 0.1 CPUs. As before the relocation, above the steal curve, an almost constant area reflecting CPU capacity not requested from these guests (idle times).

Conclusion
The left part of Figure 14 reflects the CPU constrained situation for the Linux guests running the other workloads, as visible through the comparatively high steal values.

Note: Each of the guests can at most consume the processing capacity of one CPU because only one virtual CPU is configured for each of them.

The center part of Figure 14 shows how a small impact of the relocation operation, as z/VM needs additional processing resources to perform the relocation of the guest running the transactional database workload.

Once the CPU intensive workload is relocated to the target z/VM host, the source z/VM host now disposes of sufficient CPU resources to support the CPU demand of the remaining guests running the other workloads, as is apparent from the negligible steal value during the after relocation phase in Figure 14.

The unused CPU power represented by the grey area at the top of the diagram results from the file system I/O workload waiting for I/O operations to complete.

References

A list of documents referenced in this white paper.

1. *z/VM 6.2 Live Guest Relocation with Linux Middleware*, Doelle, J; Johanssen, M., IBM Web publication. Document revision as of 2013-02-11.
   http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102250

2. *FIO 2.0.8 README*, Jens Axboe. Last modified on 2012-02-20. Accessible from the Git repository:
   http://git.kernel.dk/?p=fio.git;a=blob_plain;f=README;hb=cf9a74c8bd63d9db5256f1362885c740e11a1fe5

3. *SwingBench 2.2 Reference and User Guide*, Dominic Giles. Version 2.4.0.845(Stable) - Updated 2011-12-08. Contained within the zip file:
   http://www.dominicgiles.com/swingbench/swingbench240845.zip

4. IBM Web publication *z/VM Performance Report*, Chapter Live Guest Relocation, contained in pdf referenced below. Generated 2012-05-12.
   http://www.vm.ibm.com/perf/reports/zvm/zvmperf.pdf

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:

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.

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

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

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.

August 2013

IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.
Produced in the United States of America,
08/2013

ZSW03253-USEN-00