December 2012

Linux on System z
Oracle Database on Linux on System z - Disk I/O Connectivity Study

Dr. Juergen Doelle
Linux end-to-end Performance Project Leader,
Linux on System z.

Table of Contents

Introduction

IBM System z® provides two different disk types, ECKD™ and FCP disks, with many additional features, some of which are prized features. The various configurations that are possible within Linux® increase the amount of configuration variants available for applications such as a database on Linux on System z.

This study aims to produce guidelines for disk configuration and tuning hints with regard to ECKD and FCP disk devices, and specifically for the standalone Oracle Database using a transactional workload (OLTP), which is a very typical database workload.

The Oracle Database was used as single instance database server, installed together with the Oracle grid architecture and the Automatic Storage Manager. The Oracle Automatic Storage Management (ASM) was used to manage the disks. The setup was installed in a LPAR from IBM zEnterprise® 196 (z196).

When we started this study the available Oracle release was 10*g*. But very shortly after this study was started, Oracle release 11*g* became available for Linux on System z. This provided a good reason to switch to the new version as well as offering the possibility to examine what advantages, the new release has over the previous release.

The objective of this paper is to help you decide which disk type is appropriate and to describe how to optimize disk throughput rates.

Note: Terms used in this paper:

- DASD (Direct Access Storage Device) and ECKD (Extended Count Key Data) device are used as synonyms for fiber channel (FICON®) attached disk devices using the channel subsystem and channel programs for the ECKD layout.
- FCP or SCSI disks are used as synonyms for fiber channel attached disk devices using the Fiber Channel Protocol. They are integrated into Linux as SCSI disks via the QDIO (Queued Direct I/O) device driver.
- Memory and storage sizes used are based on 1024 bytes. To avoid confusion with values based on 1000 bytes, the notations are used according to IEC 60027-2 Amendment 2:

Table 1. Memory sizes

Symbol	Bytes
KiB	$10241 = 0$
MiB	$10242 = 1.048.576$
GiB	$10243 = 1.073.741.824$

- Oracle 10*g*: Oracle Database 10*g* Release 2.
- Oracle 11*g*: Oracle Database 11*g* Release 2.

Objectives

A database is a very intelligent mechanism that speeds up access to data, especially by using caches in an optimized way. To ensure that we see the impact of our changes in the disk I/O layer on the throughput, it was necessary for the workload to run in a disk I/O dependent mode, with lower cache hit ratios. When most of the data came from the caches, it was clear that changes in the underlying disk I/O subsystem might have only a moderate to no impact on the transactional throughput rates. Therefore, the requirement was that the total disk I/O throughput (read + write) was at least 100 MiB/sec. This was reached by reducing the SGA (decreasing the cache hit ratios) and scaling the amount of user.

A known issue with ECKD devices is the 'subchannel busy' issue. Due to the ECKD channel protocol requirements a disk (subchannel) is considered as busy, which means blocked, when one channel program (I/O request) is active. In database environments this leads to the effect that parallel executed I/O requests (which is very typical for these environments) to the same disk get queued. This can become a severe bottle neck. It becomes even more relevant the larger the single disk is, and the higher the amount of parallel running I/O processes is. The solution here is to use PAV or HyperPAV devices. For this reason we used mod27 DASD devices instead of mod9 devices to ensure that this effect really impacts our environment.

Another interesting setup topic is whether it matters if database logs and data are mixed in the same disk devices (same ASM disk group). And finally we tried to identify performance relevant setup parameters for ECKD and FCP devices.

We tried to make the FCP setup as close as possible to the ECKD setup.

Executive summary

For the database setup we recommend that you place the logs into separate disk groups to reduce the wait time for log events!

The comparison of Oracle 11*g* with Oracle 10*g* showed a very impressive improvement of approximately a factor of four for a transactional workload, together with more effective caching behavior. The CPU effort per transaction was also lower, which highly recommends an update to Oracle Release 11*g* at least for transactional workloads.

For DASD devices and database workload we highly recommend the use of HyperPAV. It provides a 40% performance improvement and is very convenient to manage.

For FCP devices the recommended setup for our workload is a multipath setup with the policy multibus and the **rr_min_io** parameter 100. It provides the highest transactional and disk throughput at the lowest CPU cost.

Comparing FCP and ECKD disk devices

Taking into account the aspects summarized in the executive summary above, some clear recommendations can be made as to which device configuration is most suitable:

- When maximum performance should be reached FCP devices are good candidates, but it requires additional CPU capacity to drive the workload
- When administration effort and low CPU requirements have a high priority, ECKD devices with HyperPAV are good candidates

There are certainly other criteria such as the existing environment and skills which might favor one or the other disk type.

Summary

To ensure that variations in the disk setup do impact the database transactional throughput, we verified that the system performance depended as much as possible on disk I/O performance.

This was achieved by reducing database cache as much as possible to minimize the effect of caching hiding differences in the I/O bandwidth. A SGA size of 4 GB was found as appropriate for our workload.

The first basic setup tests with Oracle 10*g* showed that placing the logs into a separate disk group with two additional disks heavily reduces the wait time for log events!

When the number of users was scaled the transactional throughput scaled much lower than the amount of users, which confirms that the workload is waiting for disk I/O. The CPU load and the transactional throughput scaled at the same rate, indicating that there is no CPU overhead related to the workload in the used range. The disk transfer rates of 200 MiB/sec could be considered as relatively high for a database workload. This shows clearly that the test system is suitable for testing variations in storage connectivity setup, especially in combination with the high role of the wait event class USER I/O. It also shows that the used storage server provides a relatively high disk I/O bandwidth and is not a bottleneck.

The comparison of Oracle 11*g* versus Oracle 10*g* showed a very impressive improvement of approximately a factor of four for a transactional workload. The reduction in the disk I/O read rate indicates that the caching is improved. The database now finds more data in the caches and needs to read less data from disk. The increase in write rates roughly follows the throughput increase. This confirms that our transactional throughput has really improved that much. Additionally, the CPU effort per transaction was also lower, which highly recommends the update to Oracle 11*g*, at least for transactional workloads.

The usage of HyperPAV devices improved the transactional and the disk I/O throughput by 40%, even for the dramatically reduced disk I/O rate coming up with the switch to Oracle 11*g*. The expectation is that the impact of HyperPAV devices increases with increasing disk I/O requirements. The amount of required HyperPAV devices was for this workload with 20 devices relatively small. Another important finding is that too many HyperPAV devices lead to a degradation, even if it is moderate. Here monitoring the response times of the storage server (for example with the DASD statistics) can help to decide what is a good number. From the administrative perspective the big advantage of HyperPAV with respect to PAV is the simple handling, it just needed to be enabled in Linux, the Linux kernel then handles multipathing and workload balancing.

When using FCP devices the workload was able to saturate the 4 IFLs, leading to effect that the throughput was limited by the CPU capacity. Adding two further IFLs provides a higher throughput but a CPUs utilization still up to 90%, which might still limit the transactional throughput. Adding even more CPUs provides only a slight improvement for this workload. With FCP disks the multipath daemon is required to implement multipathing. A setup which uses the multibus policy and switches between the paths after each 100 requests (**rr_min_io** parameter) provided the highest disk I/O bandwidth in our scenario. This value for **rr_min_io** might be specific to this type of workload. In terms of disk I/O pattern this means a throughput up to 200 MB/sec with small request sizes (8 KB Oracle block size), which is remarkable, but not too high. The expectation is that higher throughput rates, especially with larger block sizes, require smaller even **rr_min_io** values, which means sending less requests to the same path and switching earlier to another path.

Hardware and software configuration

This section provides details about the hardware and software used in our testing.

Server configuration

System	Operating System	CPU	Memory [MiB]	Network	Architecture
1x LPAR	SUSE Linux Enterprise Server 11 SP1	4, 6	16 GiB	1Gbit ETH	z196
1 x workload generator	Red Hat Enterprise Linux 4.3	4	2 GiB	1 x 1Gbit ETH	IBM System x®

Server hardware
System z

One z/LPAR on IBM zEnterprise z196 model 2817-M32 (2 books) 5.2 GHz, equipped with:
- Up to 6 physical CPU IFLs (cores) 16 GB Central Storage
- 2 x OSA express 4 cards
- 8 Ficon Express8 S LX (shortwave) supporting ECKD for DASD device access
- 4 Ficon Express8 S LX (shortwave) supporting FCP for SCSI device access

Storage server setup
The storage server was a DS8300 2107-932. For all system z systems and applications on up to 16 Linux host systems:

DASD
- Each test uses 10 ECKD devices type mod 27s spread over 2 Logical Control Units (LCUs), one from each internal process complex
- Each LCU had devices available in 5 ranks.
- For normal set up the devices were spread across the ranks
- 4 dedicated Host adapters to the system under test

FCP
- Each test uses 10 FCP devices, size 20 GiB spread over 2 LSS's, one from each internal processor complex
- Each LSS had one extent pool striped over 3 ranks each (storage pool striping)
- 4 dedicated Host adapters to the system under test

The disk devices were selected from both internal processor complexes on the DS8300, to make use of the full cache capacity from both complexes.

Both connection types use a switch between System z and the DS8700.

Server software

Table 2. Server software used for cpuplugd daemon tests

Product	Version and release
SUSE Linux Enterprise Server	SLES 11 SP1
Oracle Database 10*g* Release 2	10.2.0.5.0
Oracle Database 11*g* Release 2	11.2.0.2.0

Client configuration

Client hardware

One IBM xSeries® x335 Type 8676 2-way was used.

Client software

Red Hat Enterprise Linux AS release 4 (Nahant Update 3)

Linux and database setup

This section describes HyperPAV, the multipath setup, the tools used for monitoring, and the database setup.

HyperPAV

Parallel Access Volumes (PAV) is the concept of using multiple devices or aliases to address a single ECKD disk device.

If there is no aliasing of disk devices then only one I/O transfer can be in progress to a device at a time, regardless of the actual capability of the storage server to handle concurrent access to devices. Parallel access volume exists for Linux on System z in the form of PAV and HiperPAV. PAV and HiperPAV are optional features that are available on the DS8000® series.

HyperPAV has been supported on Linux on System z since Red Hat Enterprise Linux 6.0, SUSE Linux Enterprise 10 SP4, SUSE Linux Enterprise 11, Red Hat Enterprise Linux 5.9. This study uses HyperPAV only because it is much easier to administer and provides the greater flexibility.

The Base and HyperPAV devices are defined on the storage server and on the IOCDS on System z, where the DASD devices are defined with unit addresses. A LCU is a logical set of DASD devices on a DS8000 series disk storage unit and it can have up to 256 possible DASD device addresses from 00 to ff. After the base devices are defined any remaining device numbers in an LCU can be used as an alias by the system to access any of the base addresses in the same LCU.

While the usage of PAV devices requires the usage of the multipath daemon and its configuration, the advantage of HyperPAV is that it only needs to be enabled on Linux, everything else is handled by the kernel, and is transparent to the user.

HyperPAV devices on a SUSE Linux Enterprise distribution can be activated by using yast or the tool dasd_config, for example, dasd_configure 0.0.c062 1

This also creates the required udev rules.

Note:

- Ensure that the devices are not excluded from the device list via cio_ignore in zipl.conf
- There is at no time a fixed connection between a HyperPAV device and a Base device!

Multipath setup

The multipath daemon is relevant for multipathing with FCP disks or when PAV devices are used instead of HyperPAV.

Multipathing with FCP disks is required for failover in case the path to the disks fails or is in maintenance, or to balance the load over multiple paths. Each path defined to a FCP disk appears in the system as independent disk device, but it is still the same disk. These devices should not be used, because they do not provide failover and bear the risk of destroying the file system structure or files when multiple processes use the disks via different paths. Instead a tool is required which manages the different paths and ensures consistency by providing one unique device to the system. This is done by the multipath daemon.

Multipath devices

The multipath daemon provides the resulting disk device either as **/dev/dm-[n]** device or if an alias is defined in the **multipath.conf**, via `/dev/mapper/<alias name>.` The latter is highly recommended to give the multipath devices a meaningful name, for example, `FCP-<FCP-LUN>`.

path_grouping_policy

Another important option is the *path_grouping_policy*, it can be:

- *failover* Only one path is used, when that fails, the next path which works is used
- *multibus* All paths are used in a balanced mode
  - multibus Introduces the **rr_min_io** parameter which defines the number of requests after which the path is changed

wwid

The multipath devices are indexed in the multipaths section via the World Wide Identifier (*wwid*). To identify the wwid's of a given setup:

1. Enable all FCP disks devices and paths
2. Run the **multipath** command, it reports the wwid and the disk type, for example,

```
multipath
create: 36005076309ffc5ee000000000000e427 undef IBM,2107900
```

3. Create a suitable `multipath.conf` file in `/etc`
4. Call **multipath -F** to flush the existing map In case of a logical volume it reports

   ```
   Jul 31 16:46:53 | 36005076309ffc5ee000000000000e027_part1:
   map in use
   ```

   Then disable the volume group using `vgchange -an <volumegroup>` and repeat the command
5. Call the **`multipath`** command again to activate your multipath.conf file

Important pitfall

The multipath.conf file is structured in multiple sections, the relevant sections are:

- defaults
- devices
- multipaths

This also introduces a hierarchy, whereby definitions in the defaults sections are overruled by the devices and multipaths section, and the multipaths section overrules the devices section. This is important because there are a lot of defaults defined in the devices section. These definitions overwrite what is defined in the defaults section, which might affect both parameters the *path_grouping_policy* and **rr_min_io**.

multipath.conf setup

The safest way is to specify all definitions in the multipaths section, but these definitions are for each multipath device, and may require many changes. The alternative is to use the **lsscsi** command to show the used disk devices, for example:

```
lsscsi
[0:0:18:1076314336]disk  IBM      2107900          .190  /dev/sdc

[0:0:18:1076314337]disk  IBM      2107900          .190  /dev/sdd
```

Here the disks are of type `IBM` and `2107900`.

As a next step, call **multipath -t** to show the internally used defaults, which are for our disks:

```
device {
vendor "IBM"
product "2107900"
path_checker tur
checker tur
rr_min_io 1000
}
```

This overwrites a definition of **rr_min_io** defined in the defaults section. For this reason we introduced a devices section in our `multipath.conf` file to overwrite defaults.

The final test to check if it works correctly is **multipath -l**, the output looks like this:

- For a multibus setup:

```
FCPe527 (36005076309ffc5ee000000000000e527) dm-2 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=0 status=active
  |- 0:0:18:1076314341 sdb 8:16  active undef running
  |- 1:0:1:1076314341  sdf 8:80  active undef running
  |- 2:0:5:1076314341  sdj 8:144 active undef running
  `- 3:0:3:1076314341  sdn 8:208 active undef running
```

- For a failover setup:

```
FCPe527 (36005076309ffc5ee000000000000e527) dm-2 IBM,2107900
size=10G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=0 status=active
| `- 0:0:18:1076314341 sdb 8:16  active undef running
|-+- policy='round-robin 0' prio=0 status=enabled
| `- 1:0:1:1076314341  sdf 8:80  active undef running
|-+- policy='round-robin 0' prio=0 status=enabled
| `- 2:0:5:1076314341  sdj 8:144 active undef running
`-+- policy='round-robin 0' prio=0 status=enabled
  `- 3:0:3:1076314341  sdn 8:208 active undef running
```

The dmsetup table command can be used to check the **rr_min_io** parameter. For example for a **rr_min_io** value of 100 and multibus:

```
dmsetup table
FCPe527: 0 20971520 multipath 1 queue_if_no_path 0 1 1 round-robin
0 4 1 8:16 100 8:80 100 8:144 100 8:208 100
```

The values 8:16 100 are showing the `<disk major>:<disk minor> <`**rr_min_io** `value>`

Here a sample of a `multipath.conf` file (resides in /etc)

```
defaults {

        user_friendly_names     no
#       path_grouping_policy    multibus
        rr_min_io       1
}

blacklist {
        devnode "*"
}

blacklist_exceptions {
        devnode "^sdp+[0-9]*"
        devnode "^sdo+[0-9]*"
        devnode "^sdn+[0-9]*"
        devnode "^sdm+[0-9]*"
        devnode "^sdl+[0-9]*"
        devnode "^sdk+[0-9]*"
        devnode "^sdj+[0-9]*"
        devnode "^sdi+[0-9]*"
        devnode "^sdh+[0-9]*"
        devnode "^sdg+[0-9]*"
        devnode "^sdf+[0-9]*"
        devnode "^sde+[0-9]*"
        devnode "^sdd+[0-9]*"
        devnode "^sdc+[0-9]*"
        devnode "^sdb+[0-9]*"
        devnode "^sda+[0-9]*"
        device {
                vendor "IBM"
                product "S/390.*"
        }
}

devices {
        device {
                vendor "IBM"
                product "2107900"
                path_grouping_policy multibus
```

```
                path_checker tur
                checker tur
                rr_min_io 100
        }
}

multipaths {
        multipath {
                wwid    "36005076309ffc5ee000000000000e527"
                alias   "FCPe527"
        }
        multipath {
                wwid    "36005076309ffc5ee000000000000e427"
                alias   "FCPe427"
        }
        multipath {
                wwid    "36005076309ffc5ee000000000000e127"
                alias   "FCPe127"
        }
        multipath {
                wwid    "36005076309ffc5ee000000000000e027"
                alias   "FCPe027"
        }
}
```

Monitoring tools

The study used the Oracle AWR reports as a starting point for performance analysis.

If contention or excessive latencies take place, they show up in the Top 5 wait events as a high percentage of the run time, and they can be analyzed further with detailed information such as the relevant wait events or the cache information.

SADC and SAR were the primary tools used to evaluate the Linux performance. iostat data is used to complete the disk statistics data. The dasd statistics as reported in /proc/dasd/devices are evaluated specifically for the tests with DASD devices,

Database setup: the Oracle Automatic Storage Manager

The Automatic Storage Manager (ASM) is the Oracle storage management program. It is part of the Oracle 11*g* package and must be installed as part of the Oracle grid architecture.

From the user space view ASM is similar to a raw device, because the data "files" are only accessible via the application tools, they cannot be accessed via the normal file system access. ASM manages the full disk storage, including striping the storage over several disks.

The ASM disks and disk groups were managed by the ASM Configuration Assistant (asmca) for both disk types DASD and FCP. It is also possible to perform ASM tasks by using **`asmcmd`** at the command line in the same Oracle HOME, or by using **`sqlplus`** to control ASM.

To prepare the disks for ASM, the disks were partitioned and added to ASM disk groups. DASD devices are low-level formatted first of all.

Configuring DASD for ASM

DASD Disk devices are formatted the usual way.

For example:

- **`chccwdev -e 0772`** to bring device 0772 online
- **`lsdasd`** to find out the Linux name of the device `/dev/dasdy`
- **`dasdfmt -f /dev/dasdy -b 4096 -p`** to low level format, where `dasdy` is the Linux device name)
- **`fdasd -a /dev/dasdy -b 4096 -p`** to create one partition

Setting the disk ownership and access permission (DASD and FCP disks)

For a SUSE Linux Enterprise 11, the ownership and access permission for the disks used by ASM can be set via udev rules in /etc/udev/rules.d/98-oracle.permissions.rules. The following statements assign ownership to oracle:dba and read/write access to the disks at boot process:

```
KERNEL=="dasdd1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasde1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdf1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdg1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdh1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdi1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdj1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdk1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdl1", OWNER="oracle", GROUP="dba" MODE="0660"
KERNEL=="dasdm1", OWNER="oracle", GROUP="dba" MODE="0660"
```

For the changes to take effect immediately, enter **`/etc/init.d/boot.udev restart`** or reboot the system.

ASM start

In order that ASM starts up automatically at boot we modified the file /etc/oratab to show the location of the **$CRS_HOME** which is the home of +ASM and set it to Y as shown.

```
+ASM:/product11g/base/product/11.2.0/crs:Y
```

Sometimes it is necessary to start ASM manually. Make sure that the **cssd** is running and then enter the following command:

```
$ORACLE_HOME /bin/srvctl start asm -n server_name
```

srvctl can fail with this error:

```
JVMDG080: Cannot find class com/ibm/jvm/Trace
JVMXM012: Error occurred in final diagnostics initialization
          Could not create the Java virtual machine.
```

The problem is avoided with this workaround:

```
oracle@oracle1:/product/or1/oracle/bin> JAVA_COMPILER=NONE
oracle@oracle1:/product/or1/oracle/bin> export JAVA_COMPILER
```

Results

The results of the tests we performed are documented in this section together with a summary that compares FCP and ECKD disk devices.

DASD devices

Oracle 10*g*: scaling the SGA size

To identify the impact of various disk setups on the transactional workload the system must run in a mode whereby the workload throughput depends on the disk throughput.

Normalized transactional throughput

It is clear that if the database manager finds all data in the cache, the performance of the disk subsystem plays a minor role. Therefore this first test was to identify the impact of SGA size and disk dependency with regard to cache hit ratios and wait times. The aim of this test is to identify a setup which still works without errors, but has the highest disk dependency. This setup is related to the lowest throughput.

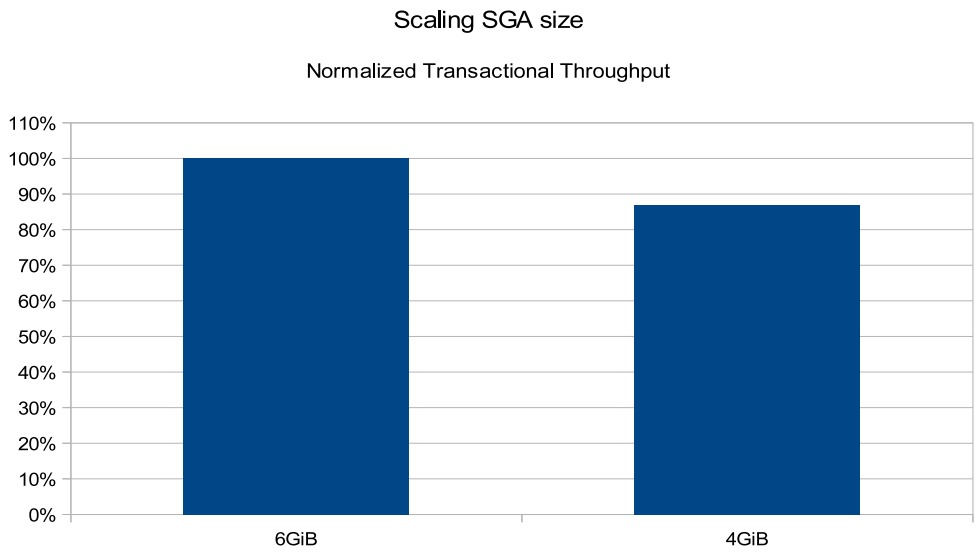

Figure 1 shows the transactional throughput when the SGA size is decreased from 6 GiB to 4 GiB.

Figure 1. Normalized transactional throughput when scaling the SGA size

Observation
The transactional throughput degrades by 13%, when decreasing the SGA size from 6 GiB to 4GiB.

Conclusion
The degradation is expected because the smaller SGA reduces the caches, which should decrease the cache hit ratio and increase the amount of disk I/O required to bring the data into the SGA. The smaller SGA should make the transaction throughput more dependent on the disk I/O.

Buffer hit ratio and wait times for the class User I/O
Figure 2 shows the impact of decreasing the SGA size from 6 GiB to 4 GiB on the buffer hit ratio and the amount of wait time spent from class User I/O. To be comparable, the wait times in seconds are normalized with the time covered from the AWR report.

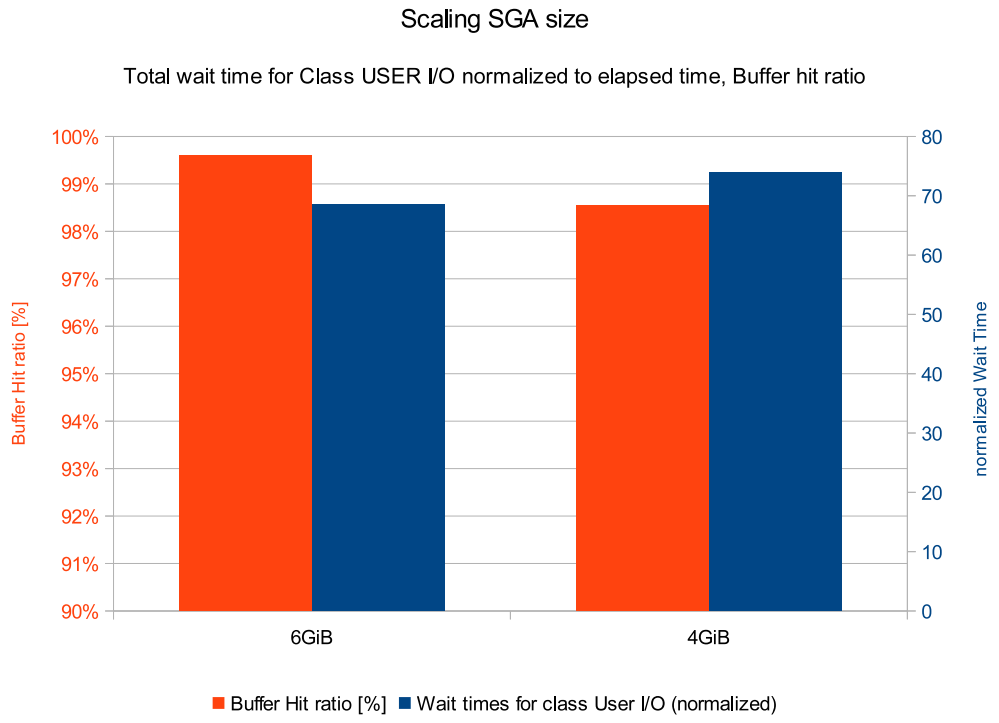

Figure 2. Buffer hit ratio and wait times for class User I/O when scaling the SGA size

Observation

When the SGA decreases from 6 GiB to 4 GiB, the buffer hit ratio decreases from 99.6% to 98.5% and the wait time increases by 8%.

Conclusion

We decided to use the 4 GiB SGA size for our further tests, because it has a higher dependency on the disk throughput and changes in the disk I/O bandwidth are expected to impact the throughput in a notifiable manner.

Oracle 10*g*: disk setup for logs and data devices

Another important setup aspect is the dependency of the transactional workload on the logging mechanism.

The speed of the updates depends on the speed the log is written. The aim was to minimize the impact of this parameter by using dedicated disks in a dedicated ASM disk group only for the database logs. From the I/O subsystem point of view the disk I/O pattern for normal data consists of small requests, randomly distributed over the disk space, and is a mix of reads and writes. From the performance point of view this is an 'ugly' pattern, which is highly dependent on disk head positioning times. On the other hand the log writing has a very 'nice' disk I/O pattern, comprising of sequential I/O and pure writes. These are executed

asynchronously on the storage server in very large blocks. Mixing log files and data files on the same disks destroys this nice pattern. In addition, the operating systems prefers read requests against write requests in the default setup, which is bad for log writing.

Figure 3 shows the impact of separating the logs into two additional disks in a separate ASM disk group on the log wait events log file sync and log file parallel write.

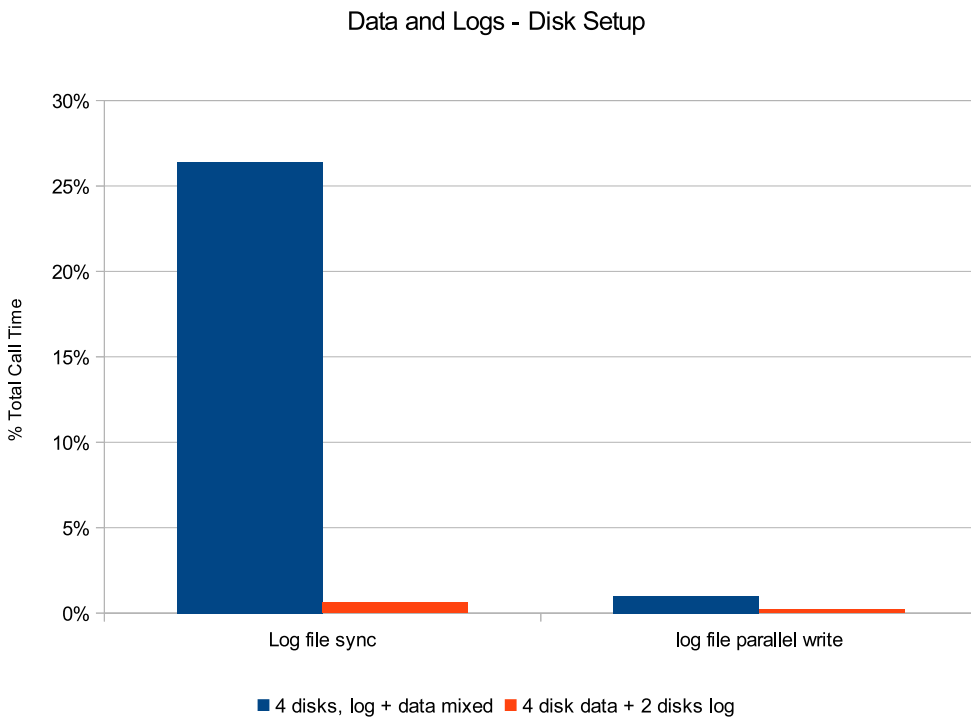

Figure 3. Log-related wait events when mixing data and log files in the same ASM disk group versus separate disk groups for logs and data

Observations

The percentage of time spent for log file sync is reduced by 98% and for log file parallel write is reduced by 80%.

Conclusion

Placing the logs to a separate disk group with two additional disks severely heavily reduces the wait time for log events!

Oracle 10*g*: user scaling

To see how the system reacts to changing workloads the amount of users is scaled from 80 to 120 in steps of 20 users.

Normalized transactional throughput and CPU load when scaling the amount of workload generating users

The result for throughput and CPU load is shown in Figure 4.

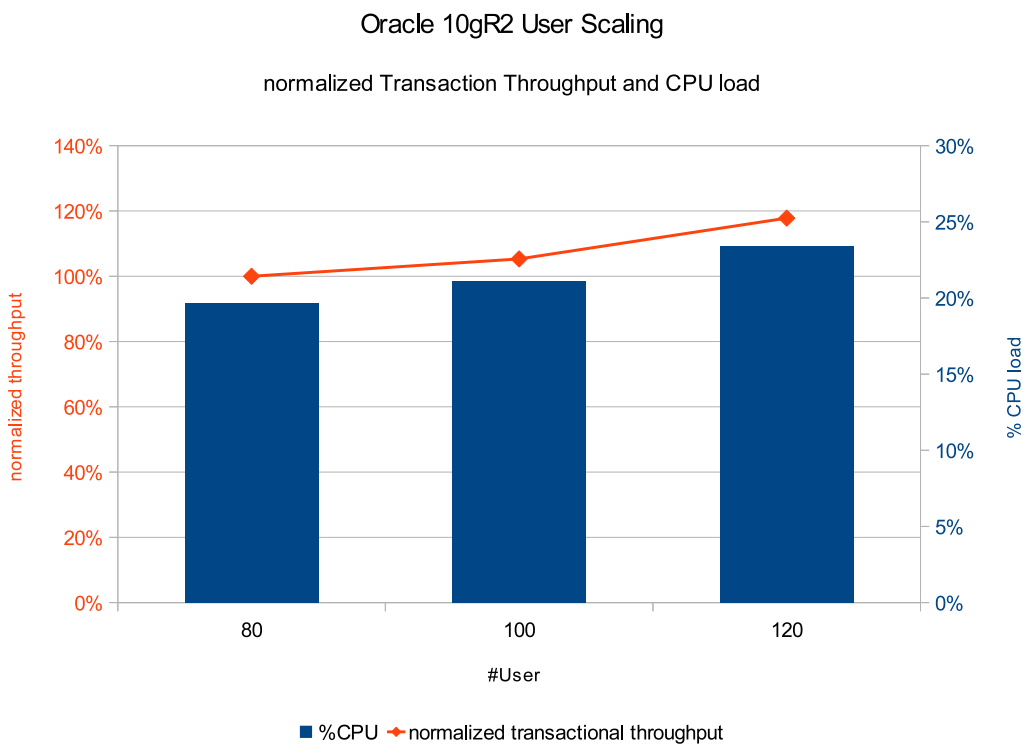

Figure 4. Normalized transactional throughput and CPU load when scaling the amount of workload generating users

Observation

The transactional throughput and the CPU load scale in a very similar fashion. But while the amount of users is finally increased by 50%, the transactional throughput only increases by 18%.

Conclusion

The fact that the CPU load and transactional throughput scale at the same rate, indicate that there is no overhead related to the workload in the range. The fact that both scale much lower than the amount of users, shows that the workload is waiting for disk I/O.

Disk I/O throughput when scaling the amount of workload generating users

Figure 5 shows the disk throughput rate when the amount of users is scaled.

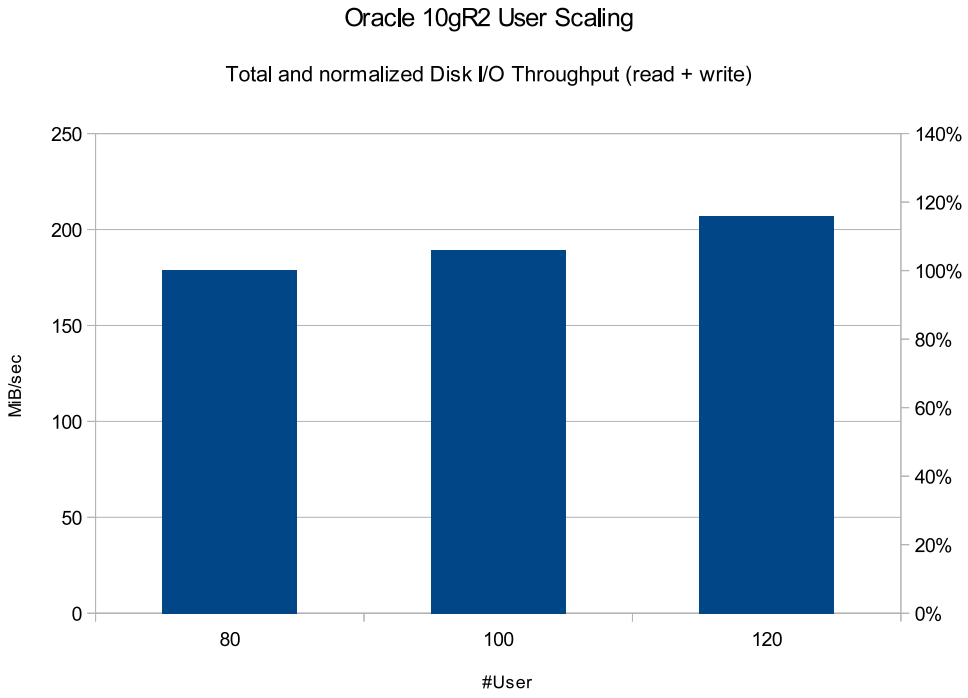

Figure 5. Disk I/O throughput when scaling the amount of workload generating users. The right axis shows the percentage relative to 80 users.

Observation

The disk I/O throughput rate scales in the same manner as the transaction rates. The disk transfer rates exceed 200 MiB/sec. The dominant wait event class is USER I/O which covers nearly 100% of the wait times, it increases when the amount of users is scaled from 80 to 120 by 50%.

Conclusion

The disk transfer rates of 200 MiB/sec could be considered as high for a database workload. This shows clearly that the test system is suitable for testing variations in storage connectivity setup, especially in combination with the high role of the wait event class USER I/O. It also shows that the used storage server and the attachment provide a relatively high disk I/O bandwidth and are no bottlenecks.

Oracle 11g versus Oracle 10g

During the test phase the new Oracle 11*g* version became available for Linux on System z. This offered the possibility to compare Oracle 10*g* with Oracle 11*g*.

User scaling and transactional throughput

Figure 6 shows the user scaling results with Oracle 10*g* as shown in Figure 4 with the results from Oracle 11*g* for the transactional throughput.

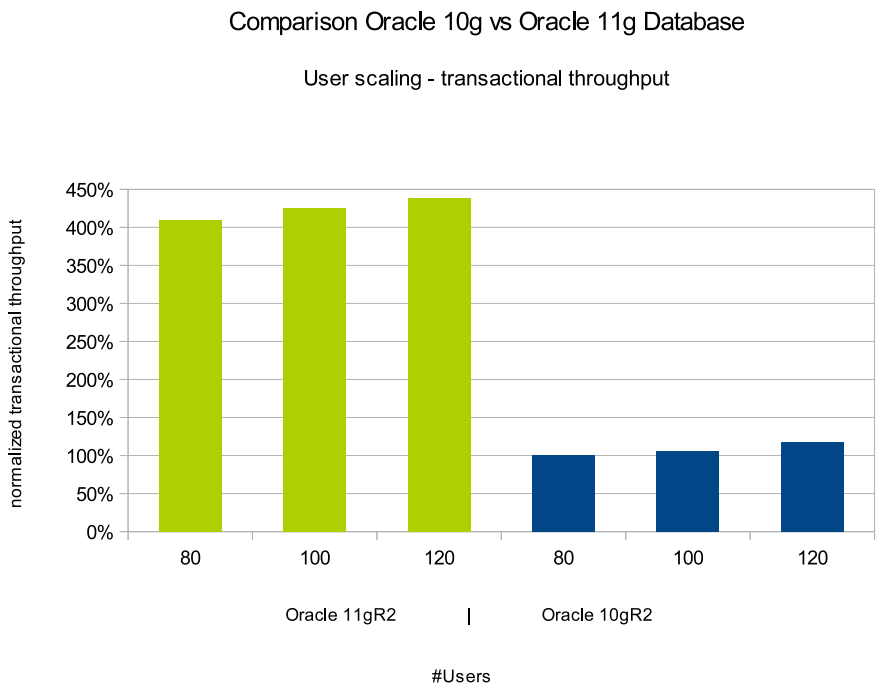

Figure 6. Oracle 11*g* versus Oracle 10*g*, user scaling and transactional throughput

Observation

The transactional throughput is increased by more than a factor of 4.

Conclusion

This is a very impressive improvement for the new version, and leads to the question, what causes this improvement?

User scaling and disk throughput

To get a better understanding what has changed in the behavior with Oracle 11*g*, Figure 7 shows the disk I/O behavior for version 10*g* and 11*g*.

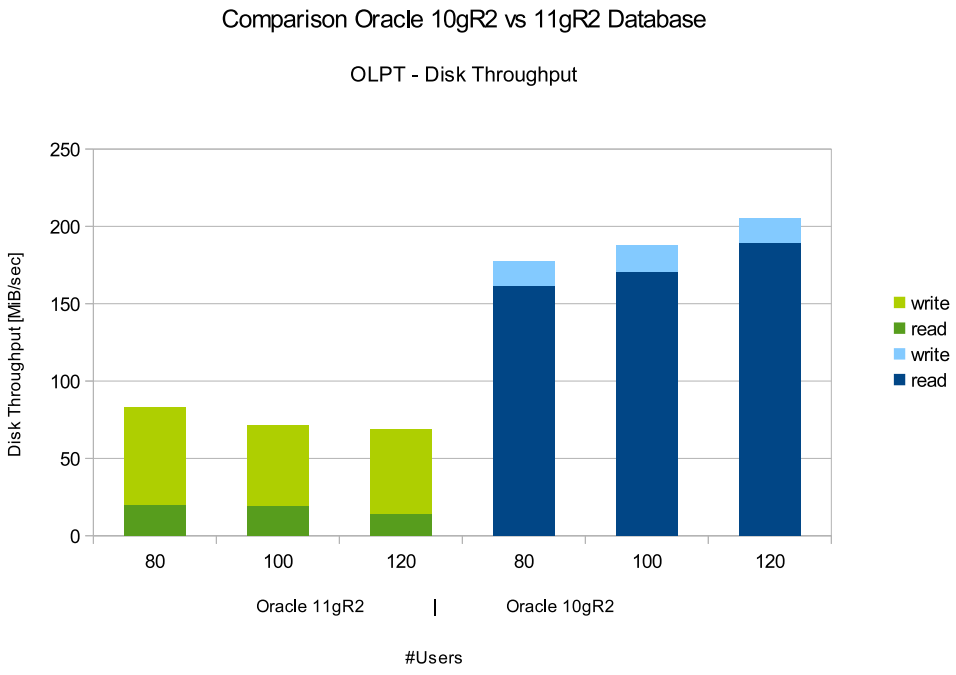

Figure 7. Oracle 11*g* versus Oracle 10*g*, user scaling and disk throughput

Observation
The disk read rates has dramatically decreased, around -90%, while the write rates have increased between a factor of 3 and 4.

Conclusion
The lower read rate is a clear indicator, that the caching is improved in terms of efficiency. The database now finds much more data in the caches and needs to read less data from disk. The increase in write rates follows roughly the throughput increase. This confirms that our transactional throughput has really improved that much.

User scaling and CPU cost per transactional throughput
Having more of the required data in the caches reduces the waits and means that the CPUs can drive a greater workload. This is of cause related to a higher CPU load, therefore the CPU cost to drive the workload it is much more interesting. This is measured in units of how many transactions are driven by one CPU. A higher value indicates that the cost is lower, that means less CPU is required for the same amount of workload. The normalized values are shown in Figure 8.

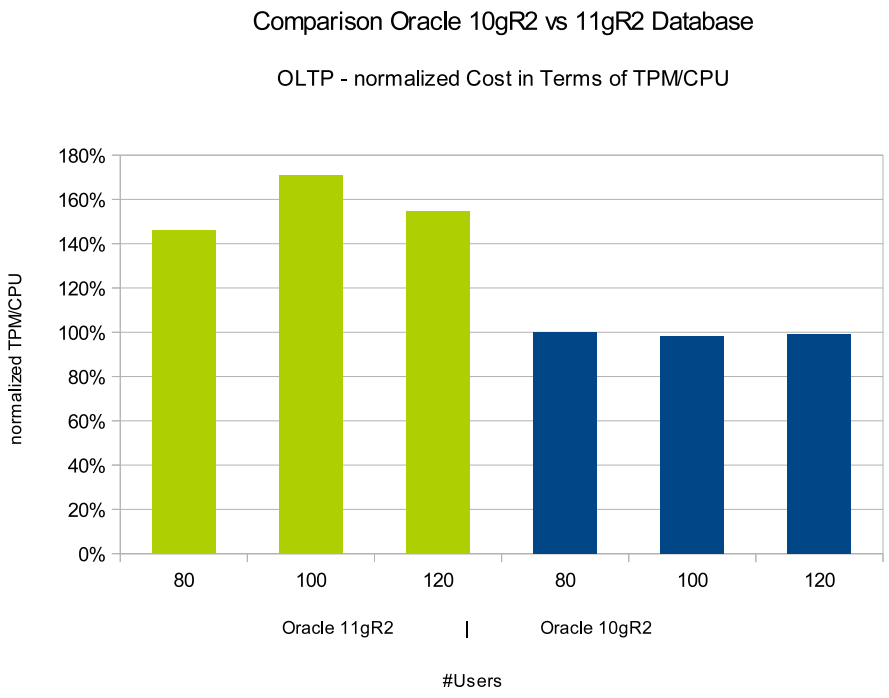

Figure 8. Oracle 11*g* versus Oracle 10*g*, user scaling and CPU cost per transactional throughput

Observation

The amount of transactions driven per CPU is increased between 45% and 70%. The system CPU is very similar, and the major contribution comes from the user space CPU.

Conclusion

The results show that we have not only a dramatic improvement in transactional throughput, the CPU effort per transaction is also lower, which highly recommends the update to Oracle 11*g*, at least for transactional workloads.

Oracle 11g: using hyperPAV devices

Impact on transactional throughput and disk I/O throughput

When running databases with high I/O requirements on DASD devices, especially large disks, a significant degradation can appear by the subchannel busy contention. When one request is sent to a disk (=subchannel), the logical device subchannel is busy and does not accept further requests. In the past the use of PAV alias devices was the recommended way to deal with that issue. But it does have some disadvantages, as it requires a certain amount of PAV devices per disk, which can heavily increase the amount of disk devices in the system, and it requires the setup of the multipath daemon. This adds as a minimum code to the path length

and management overhead. A much more convenient solution is the use of HyperPAV. This uses a pool of alias devices for all DASDs from one LCU (LSS). The devices need only to be configured on the storage server and enabled in Linux. Everything else including load balancing is handled by the kernel. An additional advantage is that the same HyperPAV alias devices can also be used from other LPARs and from other systems.

The important question is, how many alias devices are needed and what is the impact on transactional throughput. Figure 9 shows transactional and disk throughput, when scaling the amount of HyperPAV devices for our workload. We used two LCUs (one from each internal server of the DS8000), the chart specifies the sum of the HyperPAV devices from both LCUs.

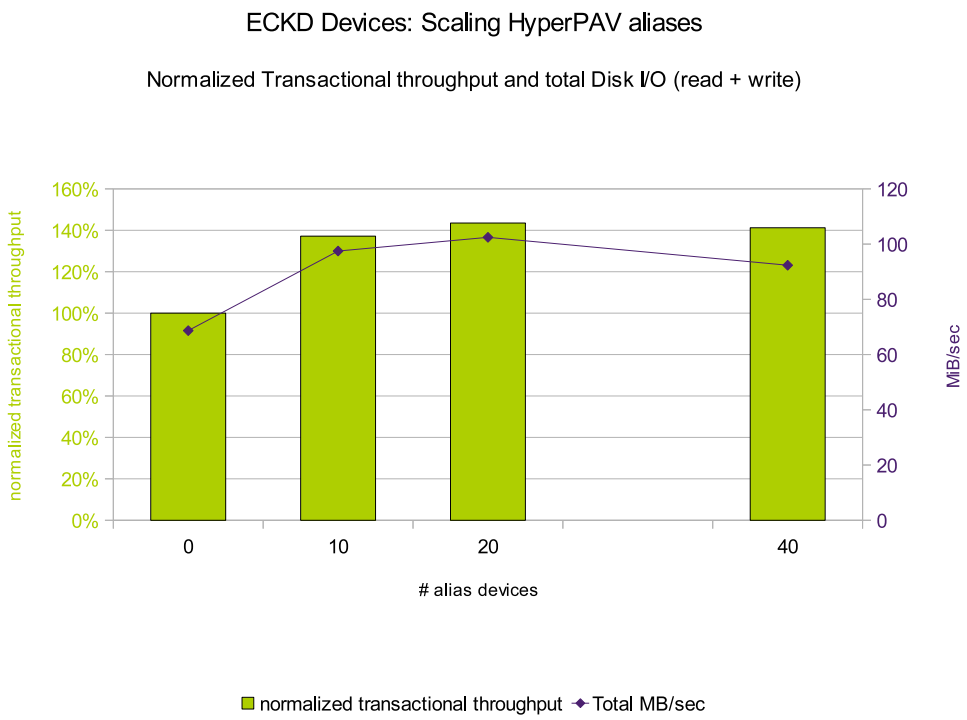

Figure 9. Scaling the amount of HyperPAV devices and impact on transactional throughput and disk I/O throughput

Observation

When increasing the amount of HyperPAV aliases the disk I/O throughput increases heavily in the first step. But with further scaling steps it increases at a much lower rate and then reduces slightly. The transactional throughput follows the disk throughput, but the reduction at the end of the scaling is only very slight. The transactional throughput increases with 20 HyperPAV aliases by more than 40%.

Conclusion

The usage of HyperPav devices can improve the transactional and the disk I/O throughput significantly. Be aware that with the switch to Oracle 11*g* the disk I/O rate was dramatically reduced. The expectation is that the impact increases with increasing disk I/O requirements. It is also interesting that the amount of required HyperPAV devices is for this workload with 20 devices relatively small. Too many HyerpPAV devices might limit the positive impact. The big advantage of HyperPAV with respect to PAV is the really simple handling, it just needed to be enabled in Linux, all the multipathing and workload balancing is done by the Linux kernel.

Impact on the total execution times of the DASD requests

To show what happens when HyperPAV devices are added, the following Figure 10 shows the impact of the amount of HyperPAV devices on the run times of the DASD I/O requests, as reported from the dasd statistics in /proc/dasd/statistics. It depicts the percentage of I/O requests from all requests of a run with an I/O time within a certain period.

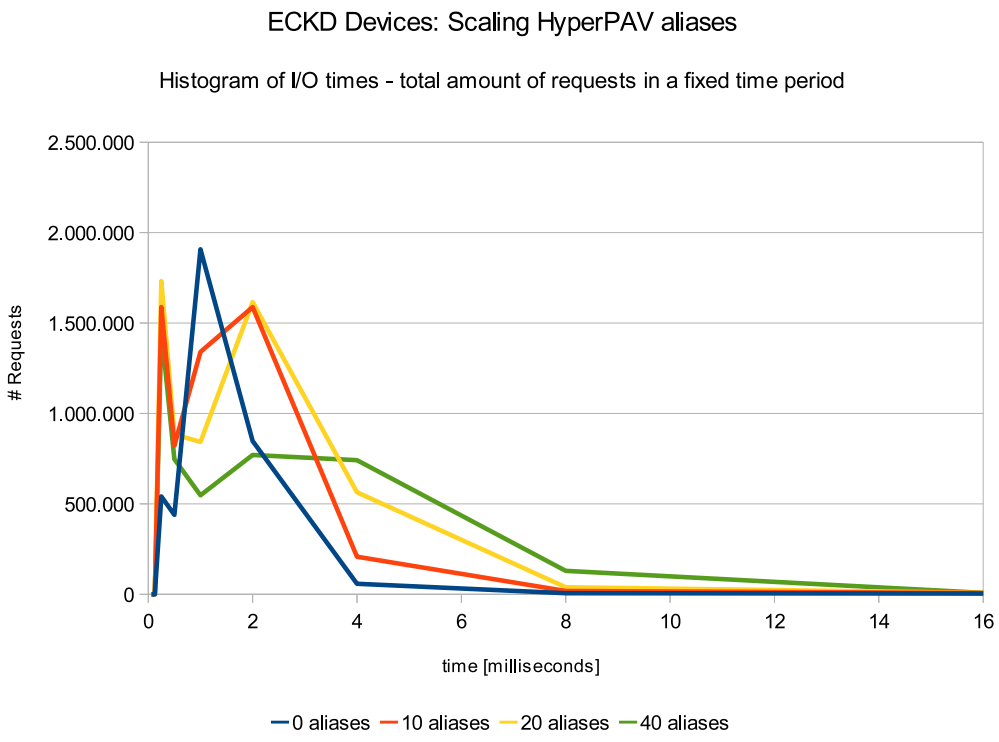

Figure 10. Scaling the amount of HyperPAV devices and impact on the total execution times of the DASD requests (0 aliases means the setup without HyperPAV)

Observation

Without HyperPAV there is a peak around 1-2 milliseconds. With 10 alias devices a peak with very short running requests already appears at below 0.5 milliseconds. This peak does not change with more alias devices, but the peak at 1-2 milliseconds reduces the height and disperse to longer time values. The overall number of requests increases, according to the throughput.

Conclusion

Without HyperPAV, most DASD requests have four other requests in the device driver queue before themselves, which would provide a delay of up to 2 ms (4 x 0.5ms) for each request. This contention is already significantly reduced with 10 alias devices already. But the tendency to have longer running requests with more alias devices indicates that the pressure on the storage server causes it to react with longer services times for the additional requests.

Impact on duration times of the DASD requests in the SAN network

To analyze the behavior of the SAN and the storage server Figure 11 shows the histogram of the times from the start subchannel command (SSCH) until the execution of the request is signaled via interrupt.

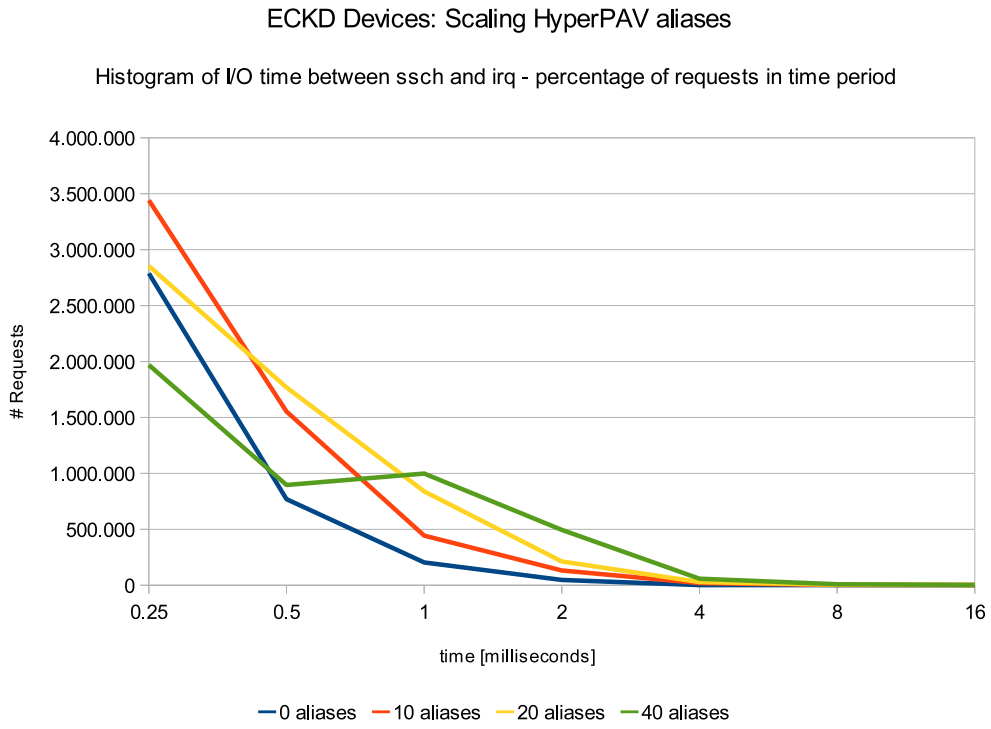

Figure 11. Scaling the amount of HyperPAV devices and impact on duration times of the DASD requests in the SAN network (0 aliases means the setup without HyperPAV)

Observation

From no HyperPAV to 10 HyperPAV devices the amount of requests just increases, with 20 alias devices the shape starts changing, it goes down at the short times end and increases with longer time values, which continues with 40 alias devices.

Conclusion

The time from issuing the start subchannel command until the interrupt signals the availability of the responses, consists of the time spent in the SAN and the response times from the storage server. Typically the time spent in the SAN is very short, that means the major contribution comes from the storage server. Therefore this chart shows that with the increasing amount of requests the storage server answers with longer response times for some of the requests. This is a typical indication of pressure on the storage server side, even when the times with 4 ms are still very short. Comparing these runtimes with the total times shown in figure 14 shows that with 40 HyperPAV devices there are a large amount of requests with total times of (from enter to the exit of the DASD layer) between 2 and 8 minutes. This indicates that in average each request now has to wait for at least one request to finish, because of the longer storage server response times.

The overall conclusion is that HyperPAV is a very good feature for increasing the I/O bandwidth with DASD devices. 10 to 20 alias devices per LCU was a good number for this workload. But another important finding is that too many devices lead then again to a degradation, even if it is moderate. Here monitoring the response times of the storage server (for example with the DASD statistics) can help to decide what is a good number. It should also be mentioned that the administration effort to use HyperPAV in Linux is minimal, which makes it a very convenient tool.

FCP devices

Oracle 11*g*: multipath daemon - options

The FCP disks devices are another important disk feature. They are integrated into the Linux on System z stack as SCSI devices.

Normalized transactional throughput

They support having many requests open against one disk, but do not provide the advantage DASD devices have with regard to the automated management of multipathing and load balancing between the paths. Setting up multiple paths to one FCP disk can be done easily, but it requires the management of these paths to ensure consistency and to provide a single device to the application. This is provided by a user space tool, the multipath daemon. The multipath daemon can be used with multiple configurations, from a single failover setup to a workload balanced switching between the paths called multibus policy. The latter has as an additional option, the parameter **rr_min_io** (see Multipath setup), which defines the criteria to switch between the paths based on the amount of requests. The default is to switch to another path after 1000 requests.

Figure 12 shows the performance when the multipath daemon is used in failover mode, and in multibus mode. In the multibus mode the parameter **rr_min_io** is scaled between 10 and 1000. Due to the observation that in the initial setup with 4 CPUs runs fully utilized, additional CPUs are added.

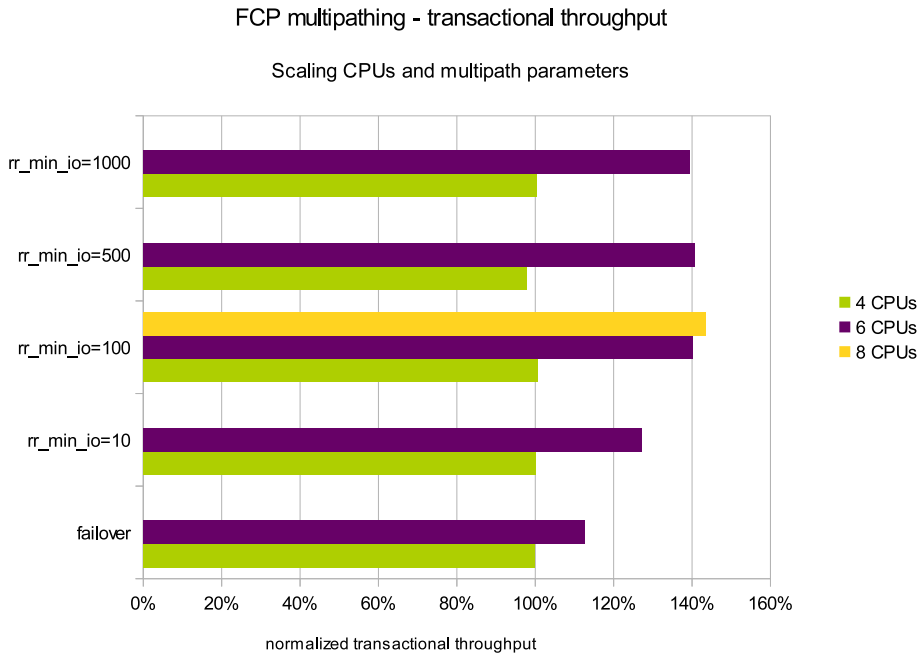

Figure 12. Normalized transactional throughput, when scaling the multipath options failover and multibus including multibus parameter rr_min_io

Observation

With 4 CPUs the system shows no difference in performance. This is because the CPUs are fully utilized. With 6 CPUs the maximum value for **rr_min_io** is between 100 and 500. The failover mode provides the lowest throughput. For testing purpose a run with 8 CPUs with a **rr_min_io** value of 100 was done. The improvement was very slight.

Conclusion

With 4 CPUs the throughput was limited by the missing CPU capacity. With 6 CPUs the utilization is still up to 90%, which limits the transactional throughput only slightly, adding even more CPUs provides only a slight improvement.

Total disk throughput (read + write)

FCP disks are driven by the Linux CPUs, they do not take advantage of the System z SAP (system assist processors). The fact that the best rr_min_io setting here is 100 to 500 is probably specific to this type of workload. In terms of disk I/O pattern this means a throughput up to 200MB/sec with small request sizes (8 KB Oracle block size), which is remarkable, but not too high. The expectation is that higher throughput rates, especially with larger block sizes, require smaller **rr_min_io** values, which means sending less requests to the same path and switching earlier to another path.

Figure 13 shows the disk I/O throughput data reached (reads and writes are summed up).

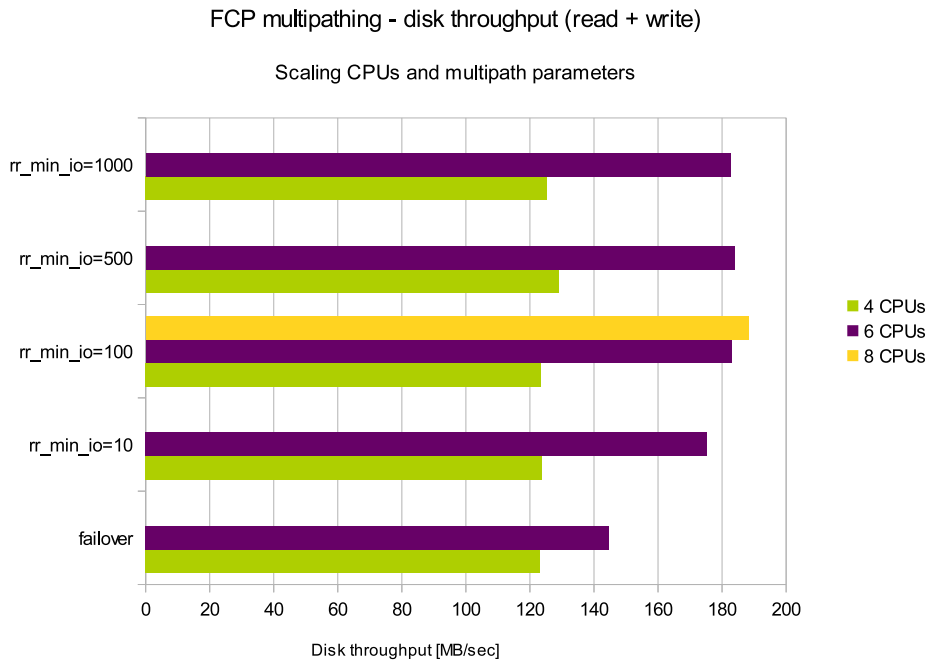

Figure 13. Total disk throughput (read + write), when scaling the multipath options failover and multibus including multibus parameter rr_min_io

Observation

The more CPUs available the higher the disk throughput, the maximum value is 188 MB/sec. With 6 CPUs the throughput in failover mode is 21% lower than with multibus and **rr_min_io**=100. The variation within the multibus setups is much less, the disk throughput with **rr_min_io**=10 is only 4% lower than with **rr_min_io**=100. The remaining values are identical.

Conclusion

It needs sufficient CPU to drive the FCP disk throughput, and much higher throughput can be reached when the amount of CPU is not limited. Switching between the paths with a multibus setup and after each 100 requests (or later) provided the highest disk I/O bandwidth in our scenario.

CPU cost per transactional throughput

The next interesting question is with regard to the CPU cost required to drive a certain workload. Is failover cheaper, or does frequently switching the path cost additional CPUs. This is shown in Figure 14.

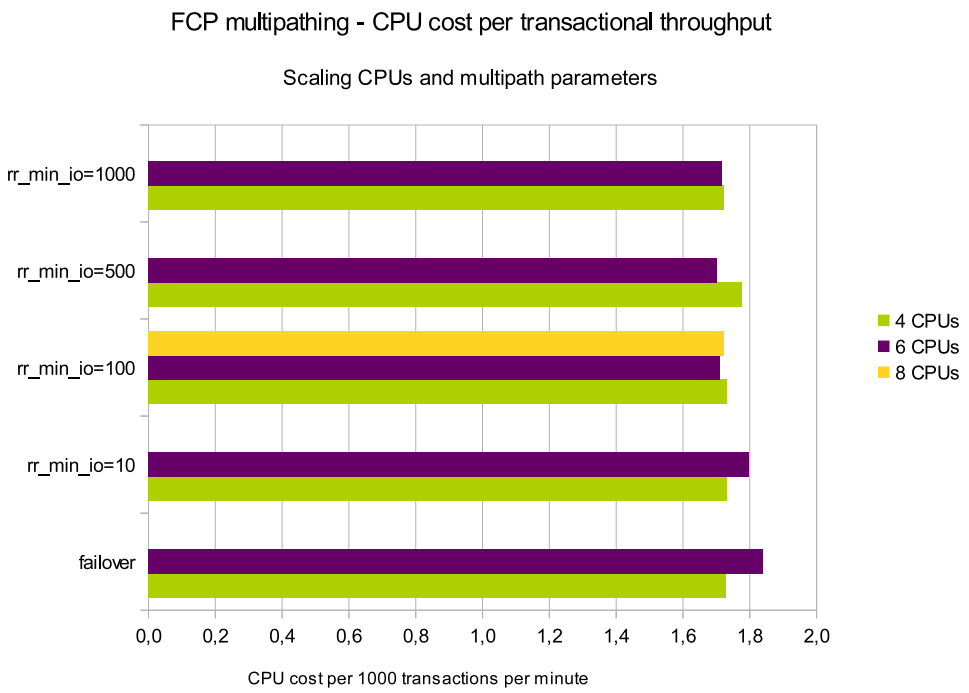

Figure 14. CPU cost per transactional throughput, when scaling the multipath options failover and multibus including multibus parameter rr_min_io

Observation
The CPU cost per throughput is fairly constant for all scenarios, independent of the amount of CPUs. In the setup with 6 CPUs the cost for failover and **rr_min_io** value of 10 is at 7% slightly higher than for the multibus configurations with the larger **rr_min_io** values.

Conclusion
Overall, the recommended setup for our workload is a multipath bus setup with the policy multibus and the **rr_min_io** parameter 100. It provides the highest transactional and disk throughput at the lowest CPU cost.

Comparing FCP and ECKD disk devices

In this final section we compare our results with FCP and ECKD devices to see if there are criteria which recommend one or the other disk connectivity type, based on the performance aspect.

Comparing normalized transactional throughput
The first criteria is the transactional database throughput reached. For this test we used the setup from each connectivity type which performs the best:

- For ECKD the HyperPAV setup with 20 aliases
- For FCP the multipath setup with policy multibus and **rr_min_io** =100

The result is shown in Figure 15.

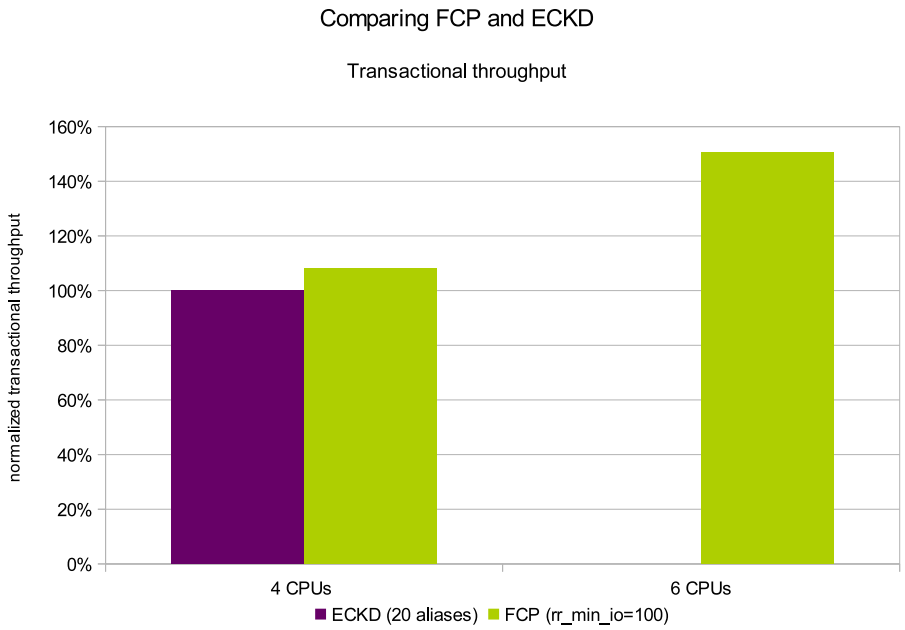

Figure 15. Comparing normalized transactional throughput when using FCP or ECKD disk devices

Observation

With 4 CPUs the throughput using FCP or ECKD disks is not very different. With 6 CPUs the throughput with FCP devices is increased by 50%. We did not do this test with the ECKD disk, because the setup with 4 CPUs had a utilization of only 80%.

Conclusion

With FCP devices higher rates of throughput can be achieved, but this requires more CPU capacity.

Comparing CPU cost per transactional throughput

The other interesting criterion is CPU cost. Figure 16 shows the CPU required to drive 1000 transactions.

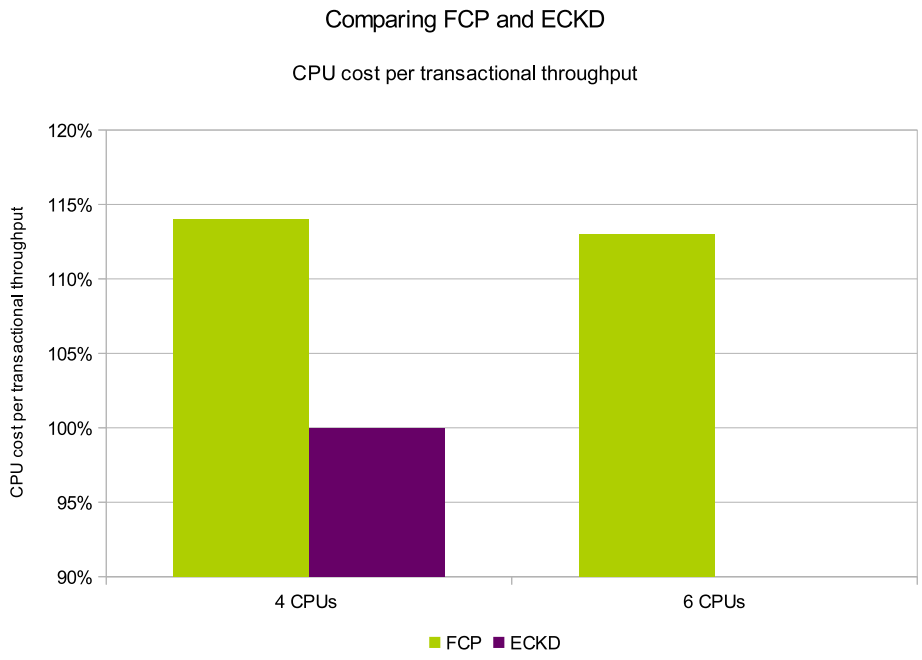

Figure 16. Comparing CPU cost per transactional throughput when using FCP or ECKD disk devices

Observation

With ECKD the lowest amount of CPU is required to drive a certain amount of workload. Using FCP requires around 14% more CPU. Be aware that the Y-axis starts at 90%.

Conclusion

ECKD devices required less CPU to drive the same amount of workload than FCP. There are two reasons for that:

- ECKD devices take advantage of the usage of SAP processors (which are not accounted for)
- For ECKD disks switching between the paths is part of the DASD driver and can be done very efficiently. For FCP devices a user space tool, the multipath daemon, is required which has a much longer path length.

As final criterion the administration effort should be considered. The handling of DASD and HyperPAV is much easier and less error-prone than the complex setup of FCP devices itself and the corresponding multipath setup.

Summary: comparing FCP and ECKD disk devices

Taking into account the aspects summarized in the executive summary above, some clear recommendations can be made as to which device configuration is most suitable:

- When maximum performance should be reached FCP devices are good candidates, but it requires additional CPU capacity to drive the workload
- When administration effort and low CPU requirements have a high priority, ECKD devices with HyperPAV are good candidates

There are certainly other criteria such as the existing environment and skills which might favor one or the other disk type.

References

This section provides information about where you can find information about topics referenced in this white paper.

- Device Drivers, Features, and Commands: http://public.dhe.ibm.com/software/dw/linux390/docu/l3n2dd14.pdf
- Setting up the zfcp device driver
- Hyper PAV and Large Volume Support for Linux on System z: ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/gse2012/nuernberg/VM02_Hyperpav.pdf
- How to Improve Performance with PAV: http://public.dhe.ibm.com/software/dw/linux390/docu/lk35hp01.pdf
- ECKD versus SCSI: http://public.dhe.ibm.com/software/dw/linux390/perf/ECKD_versus_SCSI.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

IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.
Produced in the United States of America,
12/2012

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

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

ZSW3241-USEN-00