2009 System z Expo

October 5 – 9, 2009 – Orlando, FL

Session Title:
Performance Experience with Databases on Linux for IBM System z

Session ID: zLP02

Speaker Name: Dr. Juergen Doelle

Authorized IBM Training

Trademarks

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

DB2*
DB2 Connect
DB2 Universal Database
e-business logo
IBM*
IBM eServer
IBM logo*
Informix®

System z
Tivoli*
WebSphere*
z/VM*
zSeries*
z/OS*

ECKD
Enterprise Storage Server®
FICON
FICON Express
HiperSocket
OSA
OSA Express

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

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

Notes:

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

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

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

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

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

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

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

Agenda

- **Objectives**
- **Avoid unnecessary disk I/O!**
  - Buffer Pool
  - Read Ahead
- **If you can't avoid it - make it fast!**
  - Disk Setup
  - Log Files
  - I/O Schedulers
  - Direct I/O and Asynchronous I/O
  - Hyperpav
  - Storage Pool Striping
- **z9/z10 comparison**
- **A little tuning story**
- **Summary**

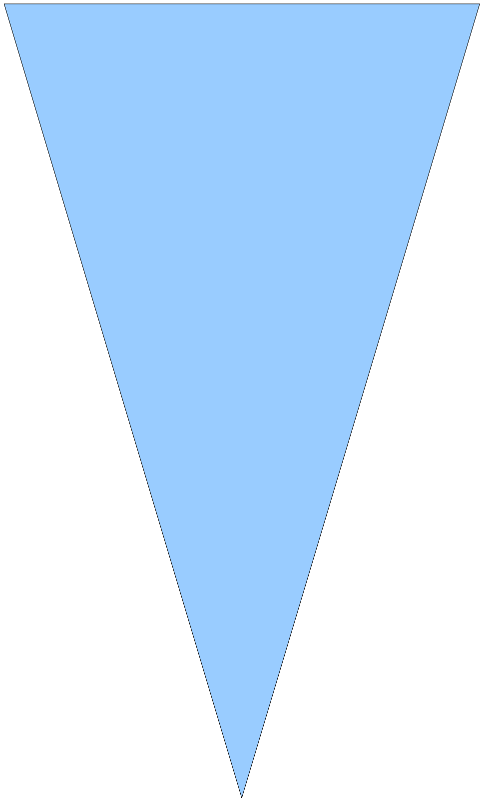
Objectives

- **Demonstrate how database application performance can benefit from the advantages provided by IBM System z**
  - What needs to be done to get the best performance
  - How to improve the disk I/O performance
  - How databases on Linux on System z scale
- **We did no high end benchmarking!**
  - Customer-like environments are used.
- **Most results are not limited to one database product. Tests have been made with**
  - Oracle 9i and 10g
  - DB2 8.1, 8.2 and 9
  - Informix 9.4.0

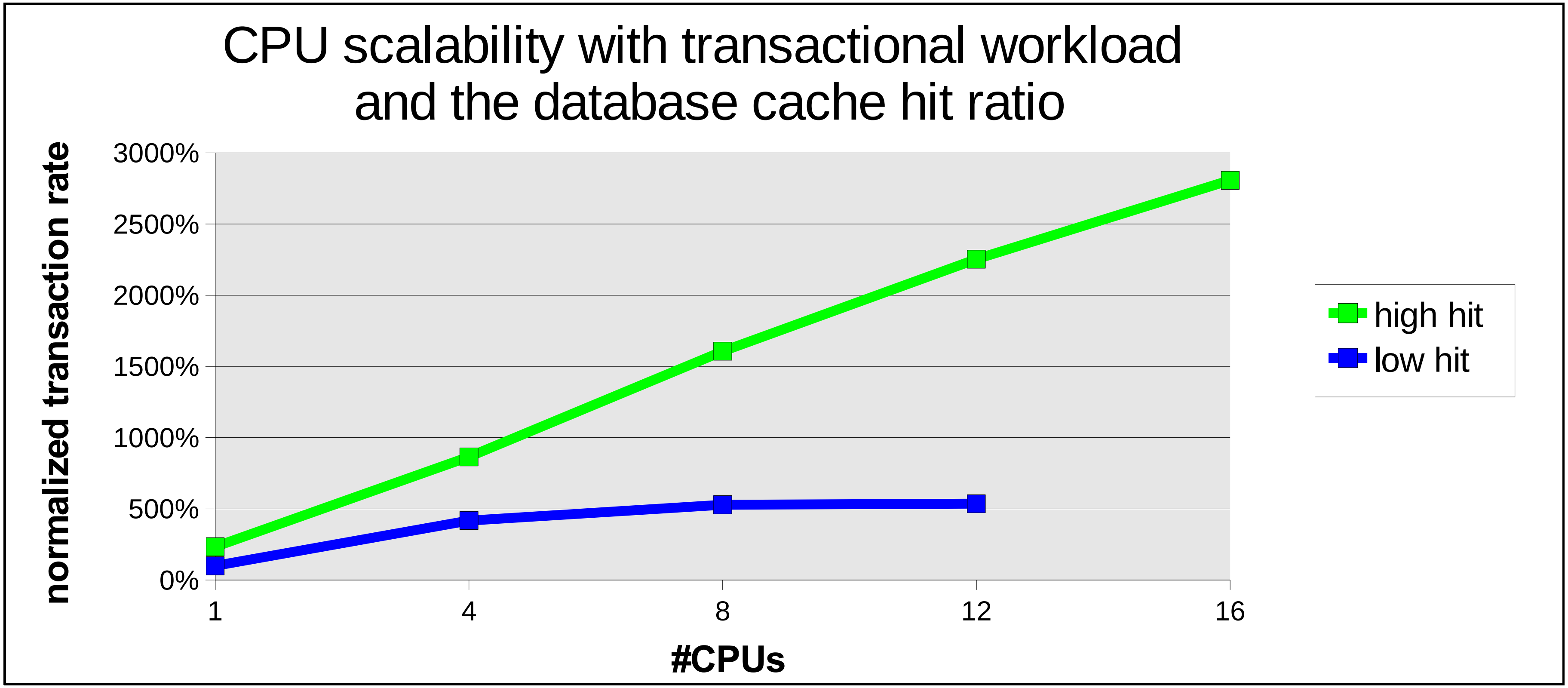
Performance tuning at all layers

- **Optimize your stack from the TOP to the BOTTOM**
  - Application design
  - Application implementation
  - Database
  - Operating system
  - Virtualization system
  - Hardware

Covered in this presentation



CPU Scalability with transactional workloads



- **The best disk I/O is the one which does not need to be done!**
  - High database cache hit ratios are a successful implementation for avoiding disk I/O
  - throughput rate scales from 1 to 16 CPUs very well
- **Low cache hit scenario depends on the possible throughput bandwidth.**
  - make the I/O fast!
- **The real world is mostly somewhere in between.**

Agenda

- Objectives
- **Avoid unnecessary disk I/O!**
  - Buffer Pool
  - Read Ahead
- If you can't avoid it - make it fast!
  - Disk Setup
  - Log Files
  - I/O Schedulers
  - Direct I/O and Asynchronous I/O
  - Hyperpav
  - Storage Pool Striping
- z9/z10 comparison
- A little tuning story
- Summary

Optimize Caching – adjust the buffer pool size

- **Reducing the amount of I/Os is heavily related to**
  - Application design, don't ask for data not needed
  - Application implementation, for example having the right indexes

- **The key component on database level are correctly sized buffer pools**
  - Which pools are needed depends on the application
  - The larger the better
  - Critical point: avoid that the system starts swapping (monitor with vmstat/sar)

- **Recommendations:**
  - Rule of thumb for database buffer pool size (start value)
    when using
    - page cache for file I/O: 60% of the memory size
    - direct I/O for file I/O: 70% of the memory size
  - you might increase the database buffer pool size, but stop before swapping starts
  - it is recommended to leave at least 5% free memory (free command)

Read ahead setup

- **On Linux block device layer level**
  Set the read ahead value to 0 using the blockdev command!
  – for example: blockdev --setra 0 /dev/sda

- **On Logical Volume Manager (LVM) level (same for ASM)**
  Disable it by setting the read ahead with the commands lvcreate or lvchange
  – LVM2: -r, --readahead none   (instead of auto)

- **On database level**
  The database is the only instance which can do a meaningful read ahead!
  – For OLTP workloads we recommend to disable the read ahead, compare results w/ and w/o read ahead
  – Oracle: set profile parameter DB_FILE_MULTIBLOCK_READ_COUNT to 0
  – DB2: set the tablespace parameter PREFETCHSIZE to 0
  – Informix: set the onconfig parameters RA_PAGES and RA_THRESHOLD to 0

Agenda

- Objectives
- Avoid unnecessary disk I/O!
  - Buffer Pool
  - Read Ahead
- **If you can't avoid it - make it fast!**
  - Disk Setup
  - Log Files
  - I/O Schedulers
  - Direct I/O and Asynchronous I/O
  - Hyperpav
  - Storage Pool Striping
- z9/z10 comparison
- A little tuning story
- Summary

What is the characteristics of the OLTP workload?

- **Cache "unfriendly"**
  - Small packets size (typically 4 or 8 KB) and randomly distributed over the disk space
  - Relief: larger caches
    avoid cache pollution with unnecessary data

- **Disk I/O intensive**
  - Disk utilization is typically at 80% or higher
  - Physical disk access times are limiting the throughput
  - Relief: use as many physical disks as possible
    make the buffer pools as large as possible

- **High write I/O portion**
  - Exceeds the non-volatile storage cache (NVS) from the storage server frequently
  - Interrupts the data flow to flush the cache
  - Relief: make sure to use as much of the NVS as possible

What can we do to get the best disk I/O performance?

- **Don't treat a storage server as a black box, understand its internal structure!**

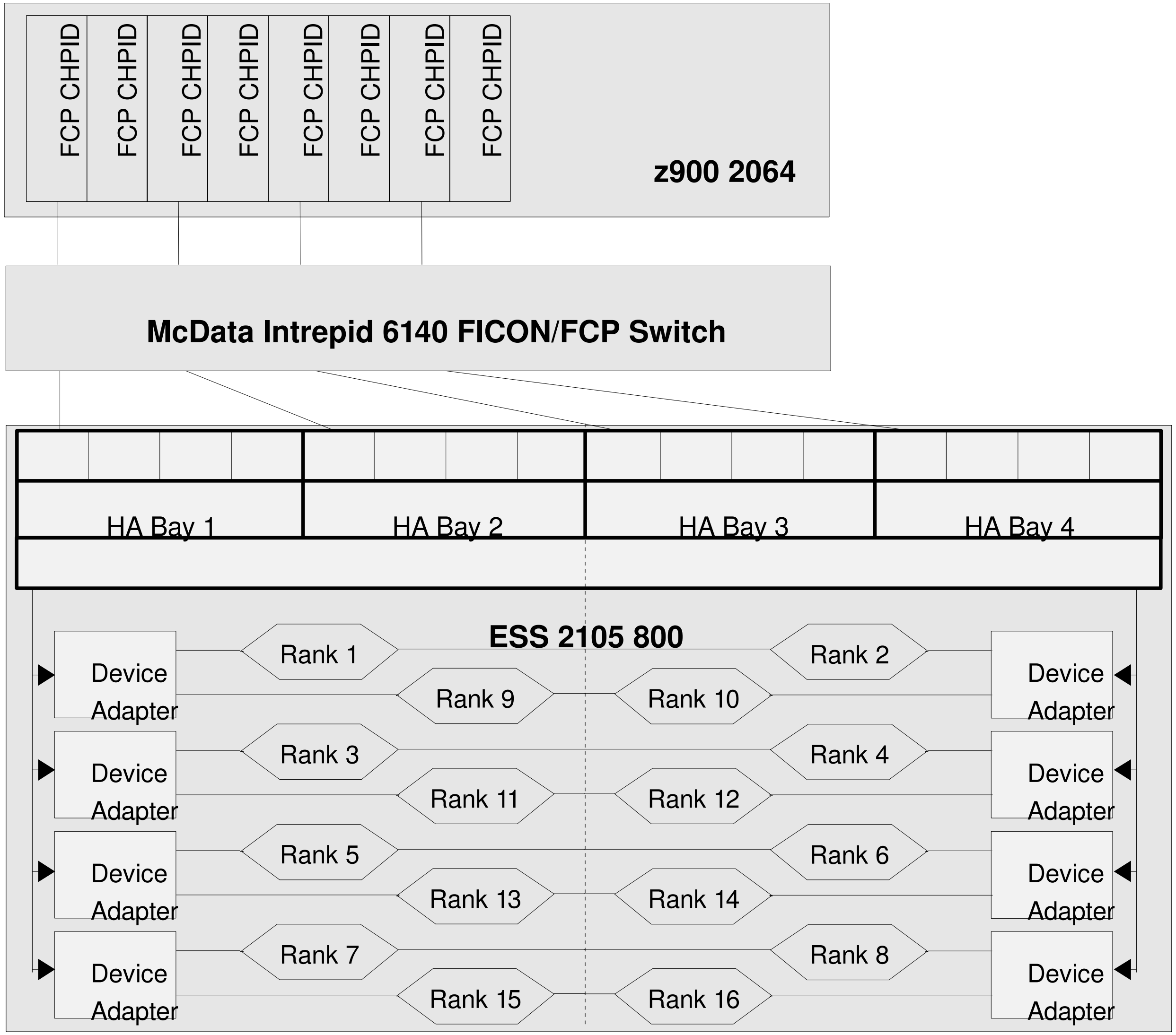
- A typical task:
  - You ask for 16 disks and
  - your system administrator gives you disks with addresses like 5100-510F...

- This will bring you close to the worst case in terms of disk performance...

➢ **What's wrong with that?**

Storage Server Architecture (1)

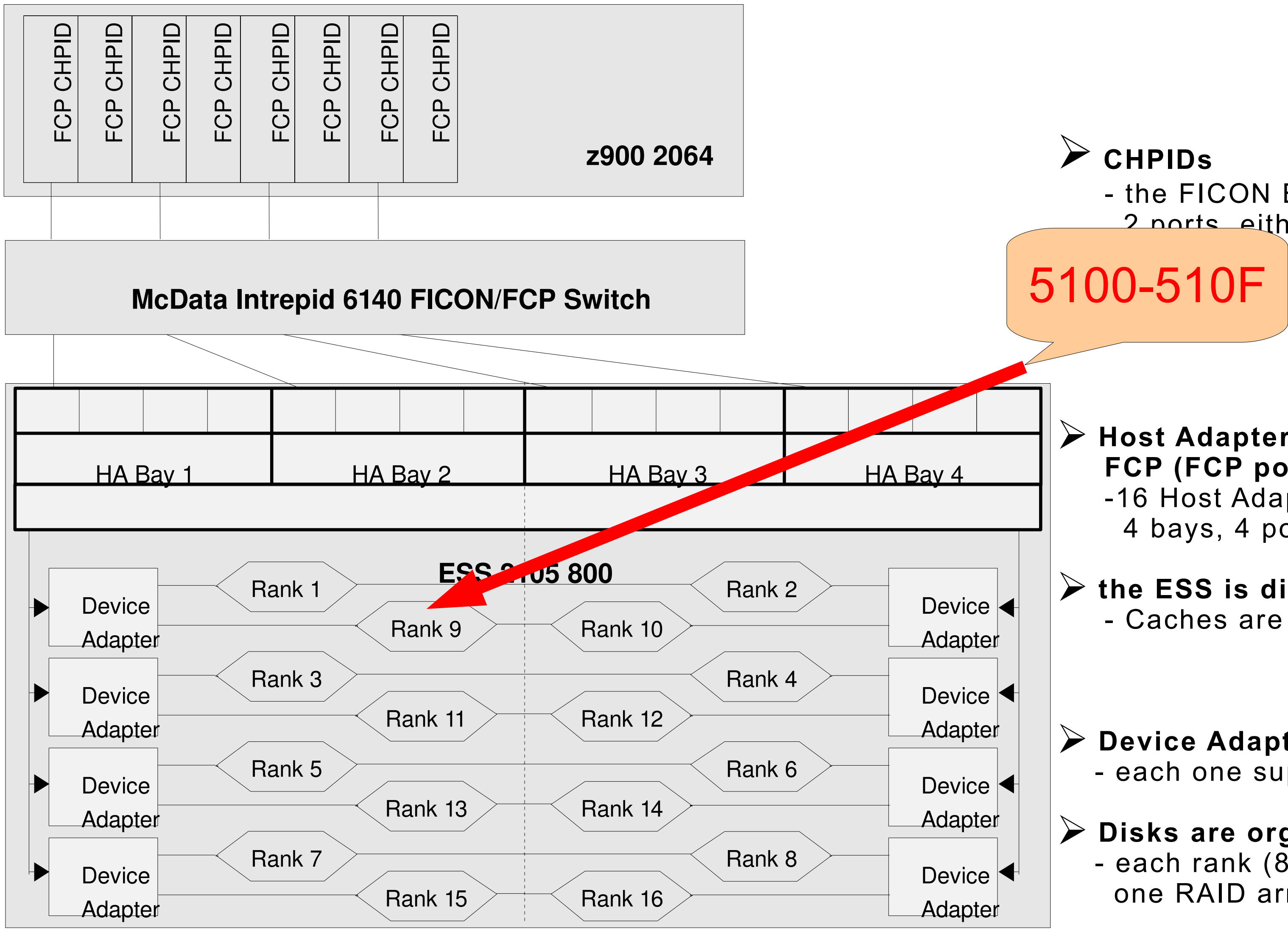
– Let's have a closer look at the elements of the scenario:



- **CHPIDs**
  - the FICON Express card supports 2 ports, either FCP or FICON
- **Host Adapter (HA) supporting FCP (FCP port)**
  - -16 Host Adapters, organized in 4 bays, 4 ports each
- **the ESS is divided into two Clusters**
  - Caches are organized per cluster!
- **Device Adapter Pairs (DA)**
  - each one supports two loops
- **Disks are organized in ranks**
  - each rank (8 physical disks) implements one RAID array (with logical disks)

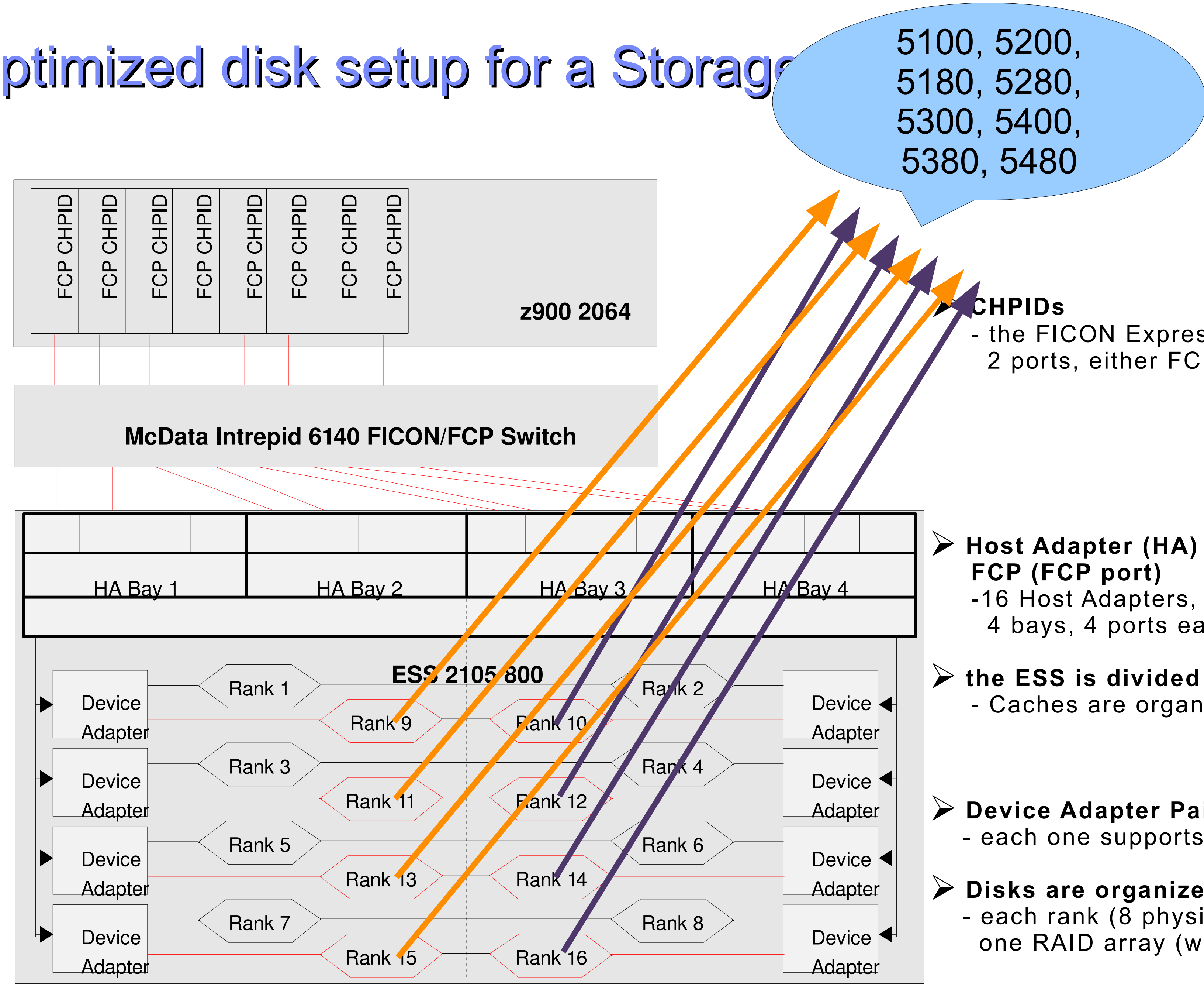
Storage Server Architecture (2)

– Let's have a closer look at the elements of the scenario:



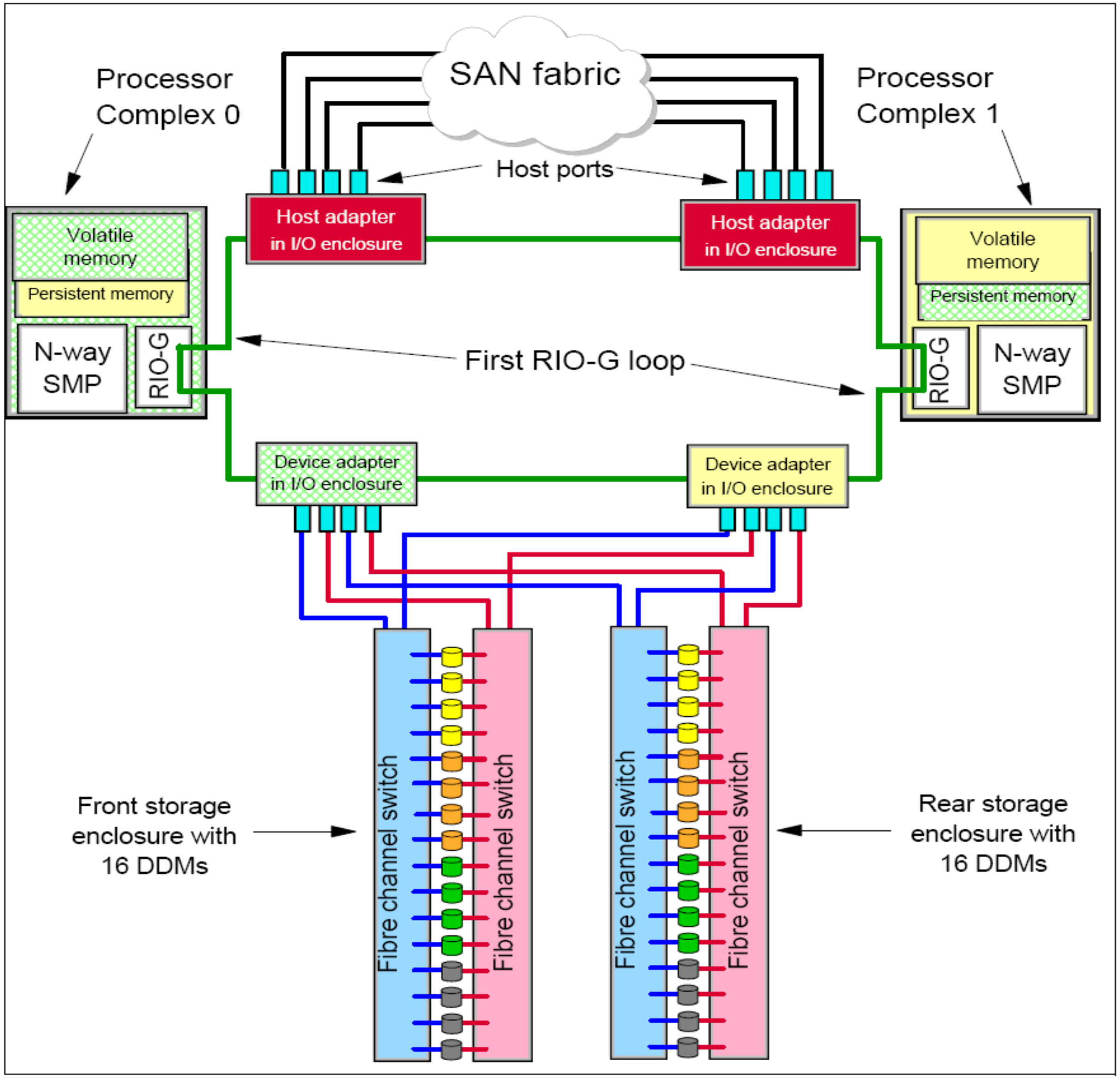
- **CHPIDs**
  - the FICON Express card supports 2 ports, either FCP or FICON
- **Host Adapter (HA) supporting FCP (FCP port)**
  - -16 Host Adapters, organized in 4 bays, 4 ports each
- **the ESS is divided into two Clusters**
  - Caches are organized per cluster!
- **Device Adapter Pairs (DA)**
  - each one supports two loops
- **Disks are organized in ranks**
  - each rank (8 physical disks) implements one RAID array (with logical disks)

Optimized disk setup for a Storage



- **CHPIDs**
  - the FICON Express card supports 2 ports, either FCP or FICON
- **Host Adapter (HA) supporting FCP (FCP port)**
  - 16 Host Adapters, organized in 4 bays, 4 ports each
- **the ESS is divided into two Clusters**
  - Caches are organized per cluster!
- **Device Adapter Pairs (DA)**
  - each one supports two loops
- **Disks are organized in ranks**
  - each rank (8 physical disks) implements one RAID array (with logical disks)

DS8000 Architecure

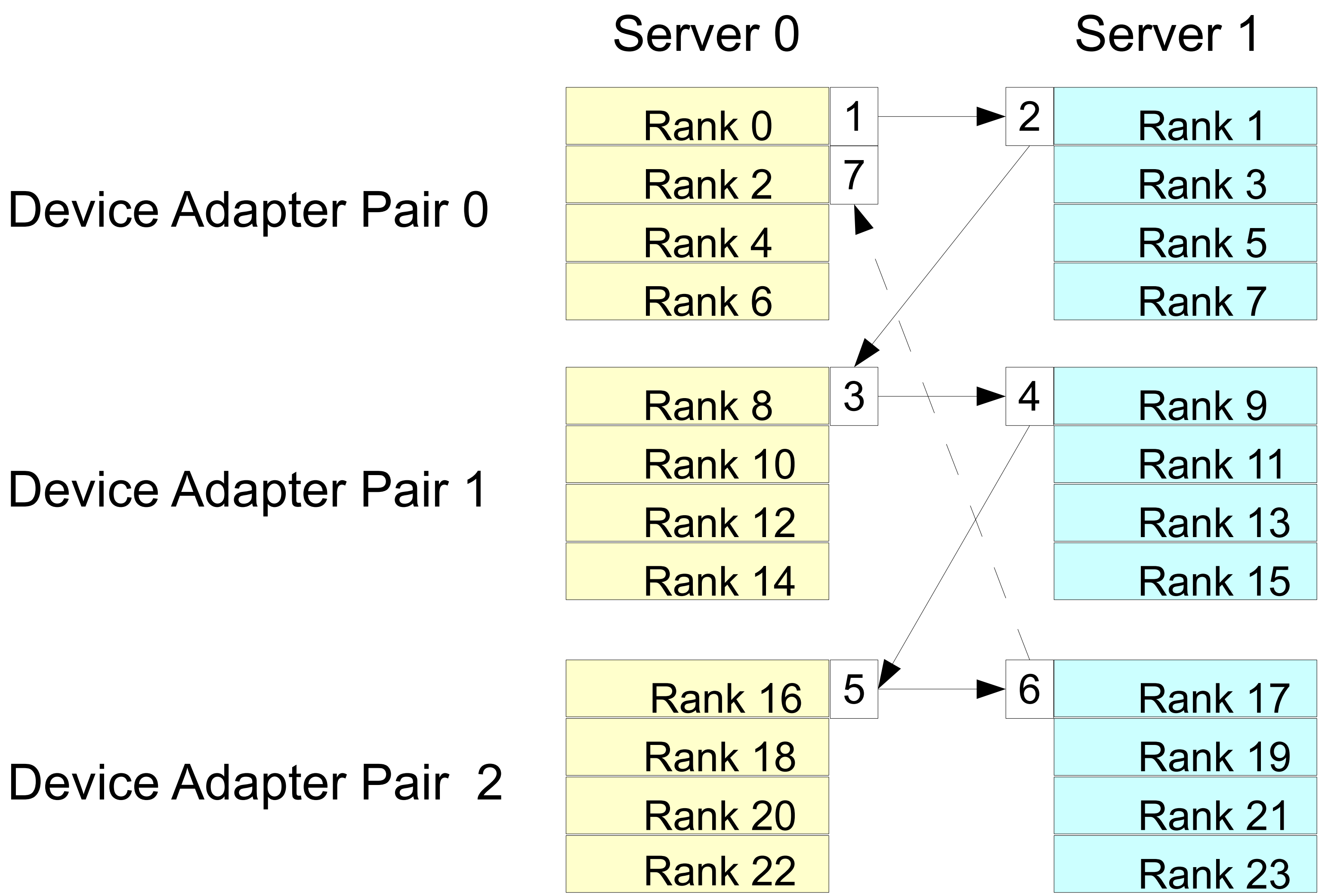


- **structure** is much more complex
  - disks are now connected via two internal FCP switches for higher bandwidth
- the DS8000 is still divided into two parts named **processor complex** or just **server**
  - caches are organized per server
- one **device adapter pair** addresses now 4 array sites
- one **array site** is built from 8 disks
  - disks are distributed over the front and rear storage enclosures
  - have the same color in the chart
- one **RAID array** is defined using one array site
- one **rank** is built using one RAID array
- ranks are assigned to an **extent pool**
- extent pools are assigned to **one of the servers**
  - this assigns also the caches
- **the rules are the same**
  - one disk range resides in one extent pool

Rules for selecting disks

- **Target is to get a balanced load on all paths and physical disks**
  - use as many paths as possible (CHPID -> host adapter)
  - for ECKD switching the paths is done automatically
  - FCP needs a fixed relation between disk and path
    - we establish a fix mapping between path and rank in our environment
    - taking a disk from another rank will then use another path
  - switch the rank for each new disk
  - switch the ranks used between servers and device adapters
  - select disks from as many ranks as possible!
  - avoid reusing the same resource (path, server, device adapter, and disk) as long as possible

Sample for optimal disk selection



- **assign the disks to the system in the order from 0 to 7, etc.**

How to make the disks available for the database

- **use a striped Logical Volume (LV)**
  - add the volumes appropriate to Volume Group (VG)
  - we recommend a stripe size of 32KB for OLTP workloads
  - number of stripes equal to the number of disks in VG

- **let the database do the striping:**
  - e.g. for Oracle: use ASM managed storage
  - for DB2: use multiple containers
    - ```
      CREATE TABLESPACE dms1 MANAGED BY DATABASE
        USING (FILE '/TSTEST_cont0/file' 1000,
               FILE '/TSTEST_cont1/file' 1000,
               …
      ```
    - ```
      CREATE TABLESPACE dms2 MANAGED BY DATABASE
        USING (DEVICE '/dev/sda2' 1170736,
               DEVICE '/dev/sda3' 1170736,
               …
      ```
    - Select the disks in the right order from the ranks
    - The database will stripe over the containers automatically

What to do with database LOG files?

- **I/O pattern:**
  - OLTP database access is random read/write I/O
  - writing to a database log file is pure sequential write I/O (except for rollback)

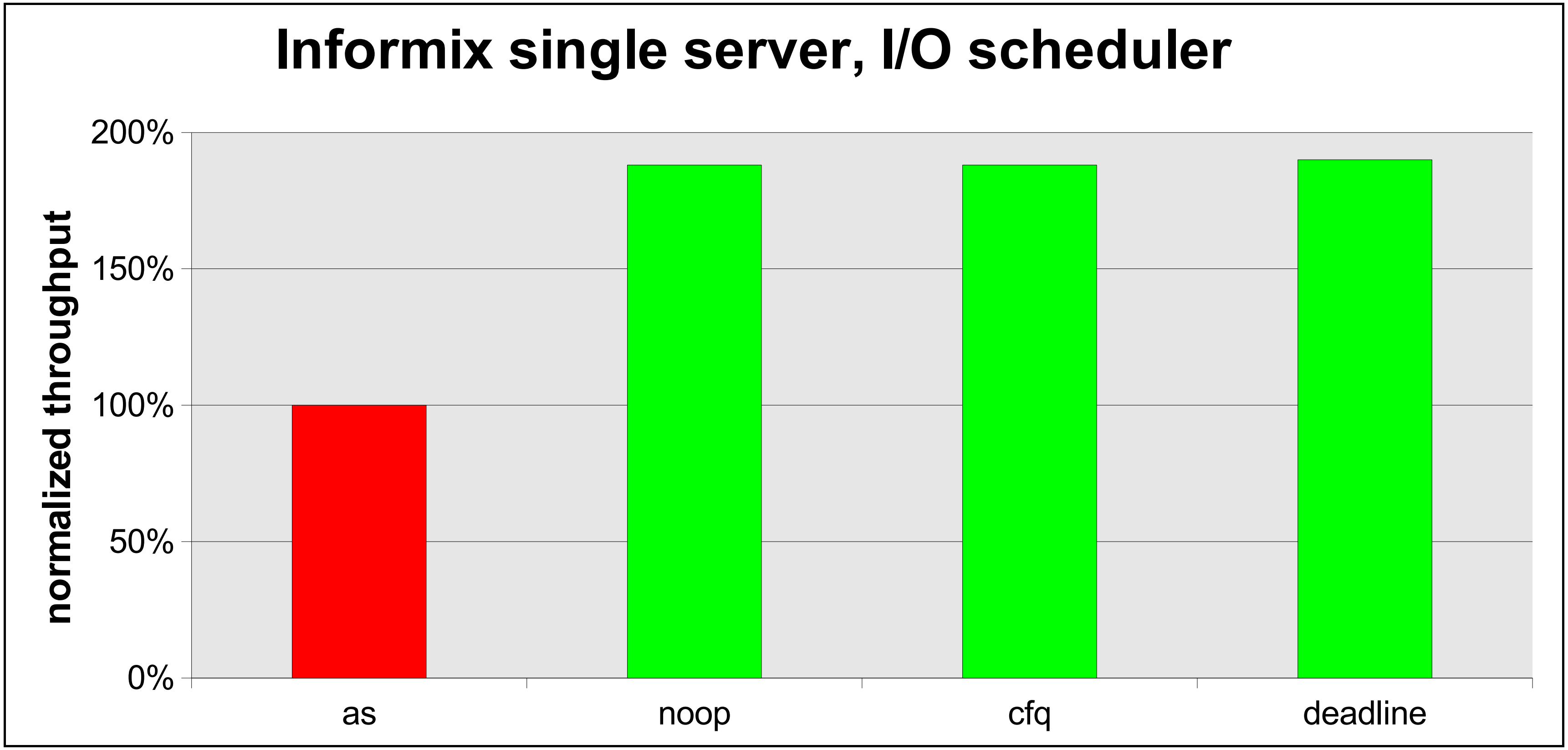
- **Mixing database log files and the database files on the same disks (either SCSI LUN, ECKD device or Logical Volume)**
  - the sequential characteristics of the log I/O gets lost
  - the I/O schedulers prefer read requests!
  - → degradation of the speed for writing log records to the disk
  - → degradation of the priority when writing logs
  - → overall a performance degradation of the application throughput rate

➢ **make separate log and data devices, use if possible**
  - different ranks on the same storage server
  - use different storage servers
  - → This ensures a contiguous log disk I/O and good transaction rates.

Linux kernel 2.6 I/O schedulers

- **four different I/O schedulers are available**
  - **noop** scheduler
    does only request merging
  - **deadline** scheduler
    avoids read request starvation, offers the possibility to give write requests the same priority like reads
  - anticipatory scheduler (**as** scheduler)
    designed for the usage with physical disks, not intended for storage subsystems
  - complete fair queuing scheduler (**cfq** scheduler)
    all users of a particular drive would be able to execute about the same number of I/O requests over a given time

Linux kernel 2.6 I/O schedulers - Results



- as scheduler is not a good choice for OLTP environments
- all other schedulers show similar results
- **deadline scheduler is used for further tests**

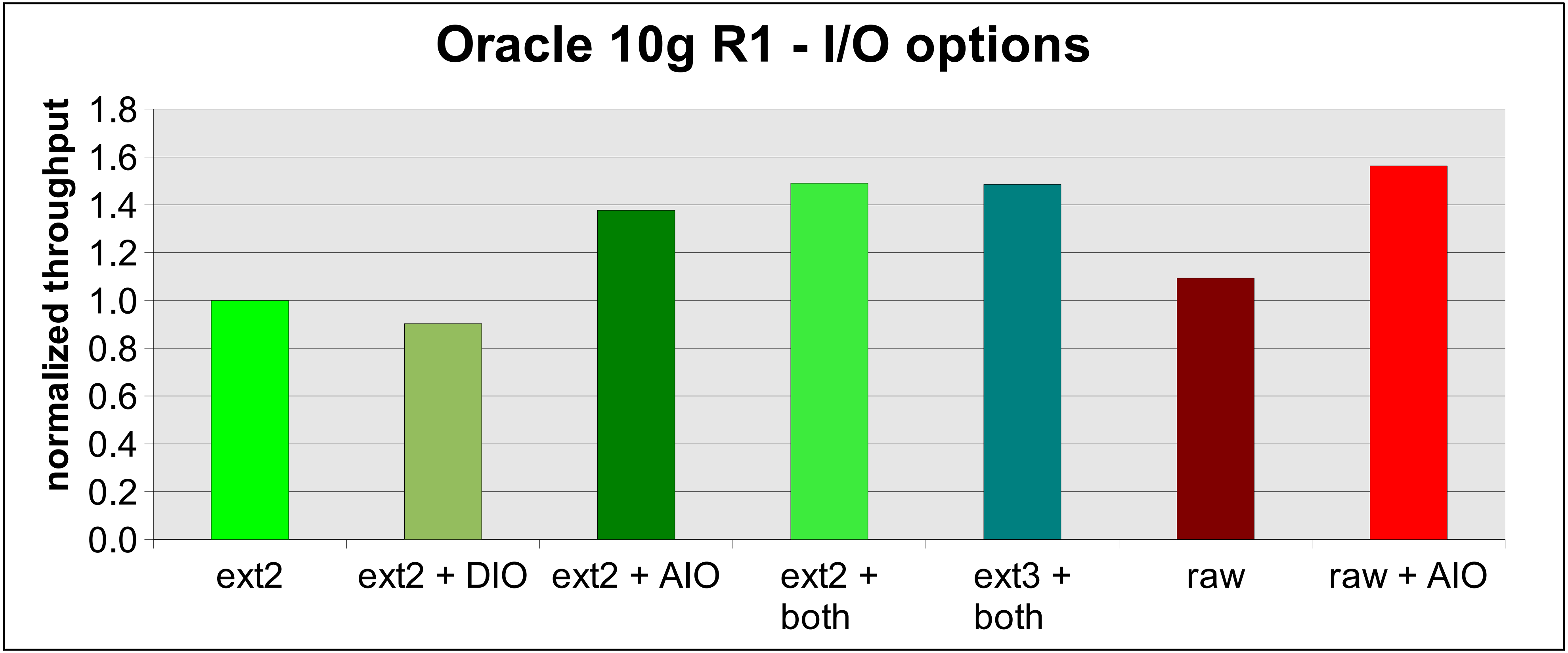
Disk I/O Options with Linux kernel 2.6

- **Direct I/O (DIO)**
  - transfer the data directly from the application buffers to the device driver
  - advantage
    - saves page cache memory
    - same data are not cached twice (no copying of the data to the Linux page cache)
    - use larger buffer pools instead
  - disadvantage
    - ensure that no utility is accessing the same data through the file system (page cache) --> danger of data corruption

- **Asynchronous I/O (AIO)**
  - I/O requests are issued asynchronously by the application
  - the application does not have to wait for I/O request completion
  - application can immediately continue processing
  - advantage
    - the number of parallel I/O processes can be reduced (this saves memory and CPU)

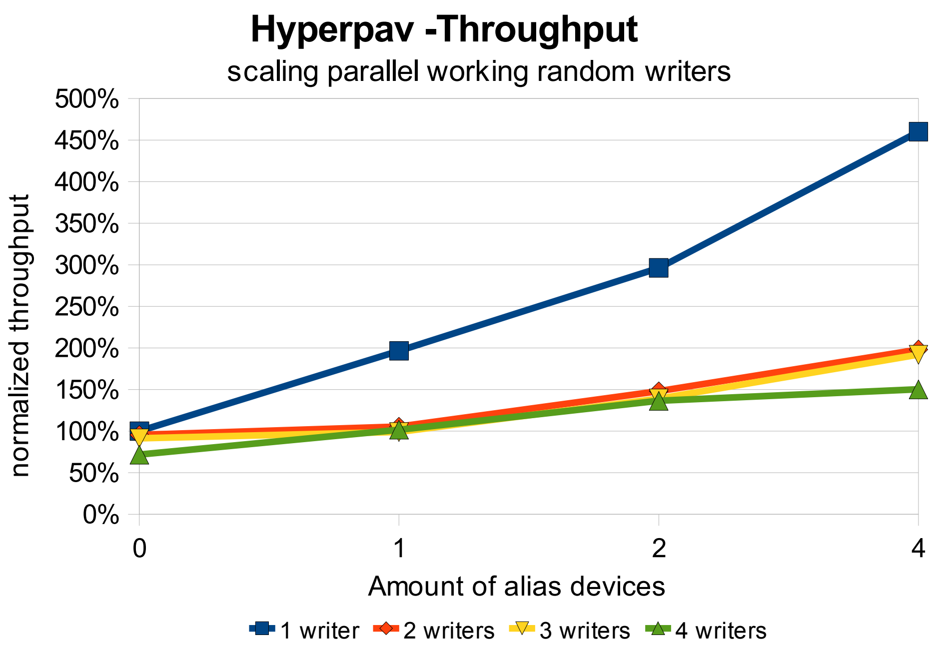
➢ **use both features together**

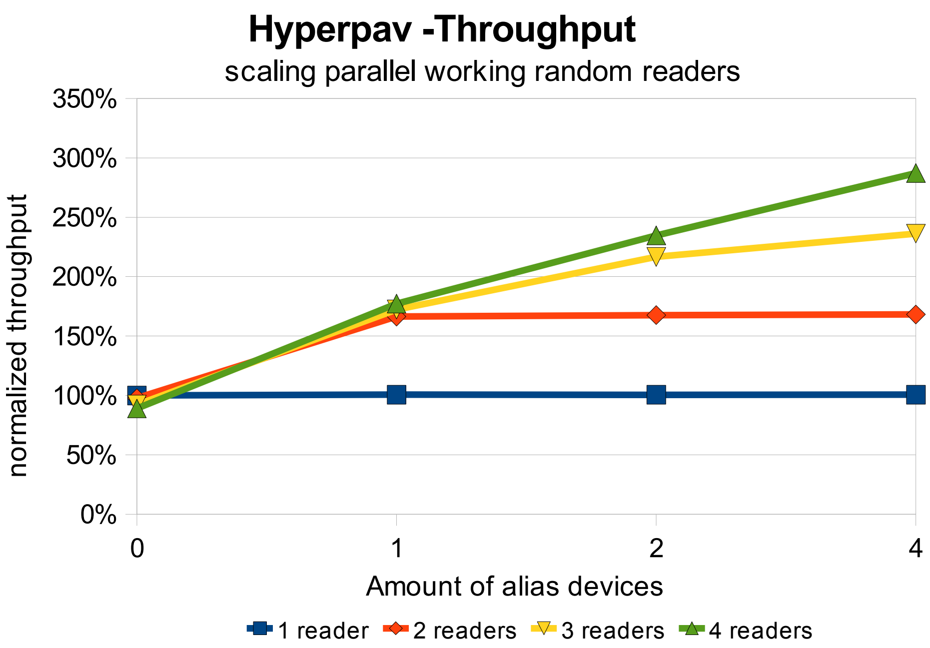
DIO and AIO – Results



- **The combination of DIO and AIO shows the best results for the Linux file system**
  - filesystemio_options=setall
- **The best throughput rate is provided by raw I/O + AIO**
  - raw I/O uses implicitly DIO

new feature with SLES 11 – Hyperpav





- **Access to ECKD DASD could become contented when used in parallel from multiple processes (subchannel busy)**
- **Solution: having multiple subchannels per device = alias device**
  - HYPERPAV: pool of aliases assigned temporarily to a device as needed
- **usage of Hyperpav can be highly recommended!**

DS8000 Storage Pool Striping

- **requires License Machine Code 5.30xx.xx**
- **Stripe the extents of a DS8000 Volume across multiple RAID arrays/ranks**
- **Pro**
  - Storage pool striped volumes are simple to set up and to administrate as a few large disks
  - Provides a similar performance advantage as the disk setup described earlier (striping over ranks)
  - Striping on the storage device lowers CPU consumption (LVM) on the Linux side
- **Con**
  - Stripe size is 1 GB (relatively large)
  - Losing control how the extents are selected from the ranks
  - Cannot span internal servers

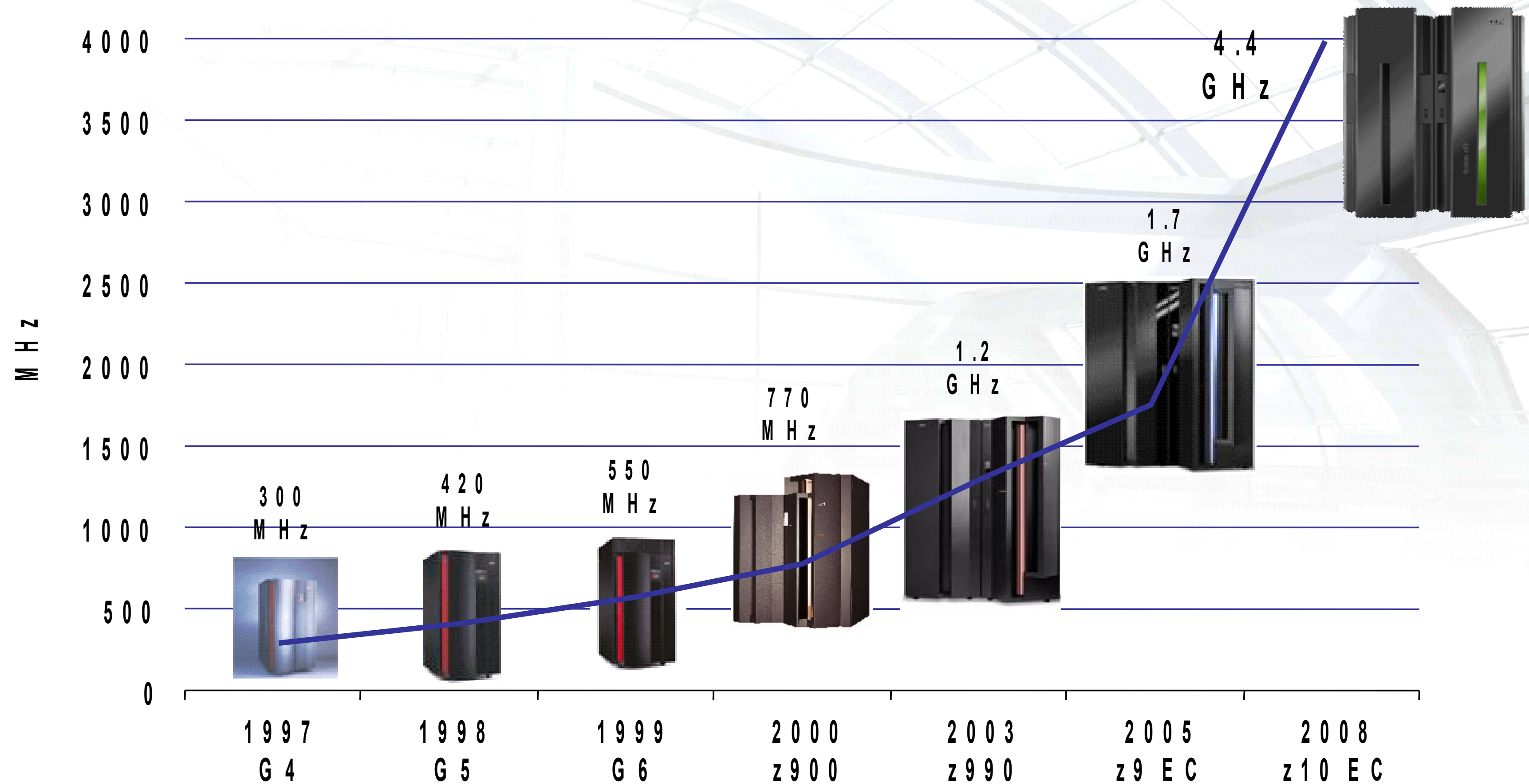
- **ECKD disks**
  - Without HiperPAV only one I/O per DASD can be processed which limits the performance!
  - FICON path groups do the load balancing

➢ **Using storage pool striping makes administration easy and provides a very good overall performance**

Agenda

- Objectives
- Avoid unnecessary disk I/O!
  - Buffer Pool
  - Read Ahead
- If you can't avoid it - make it fast!
  - Disk Setup
  - Log Files
  - I/O Schedulers
  - Direct I/O and Asynchronous I/O
  - Hyperpav
  - Storage Pool Striping
- **z9/z10 comparison**
- A little tuning story
- Summary

IBM z10 EC Continues the CMOS Mainframe Heritage

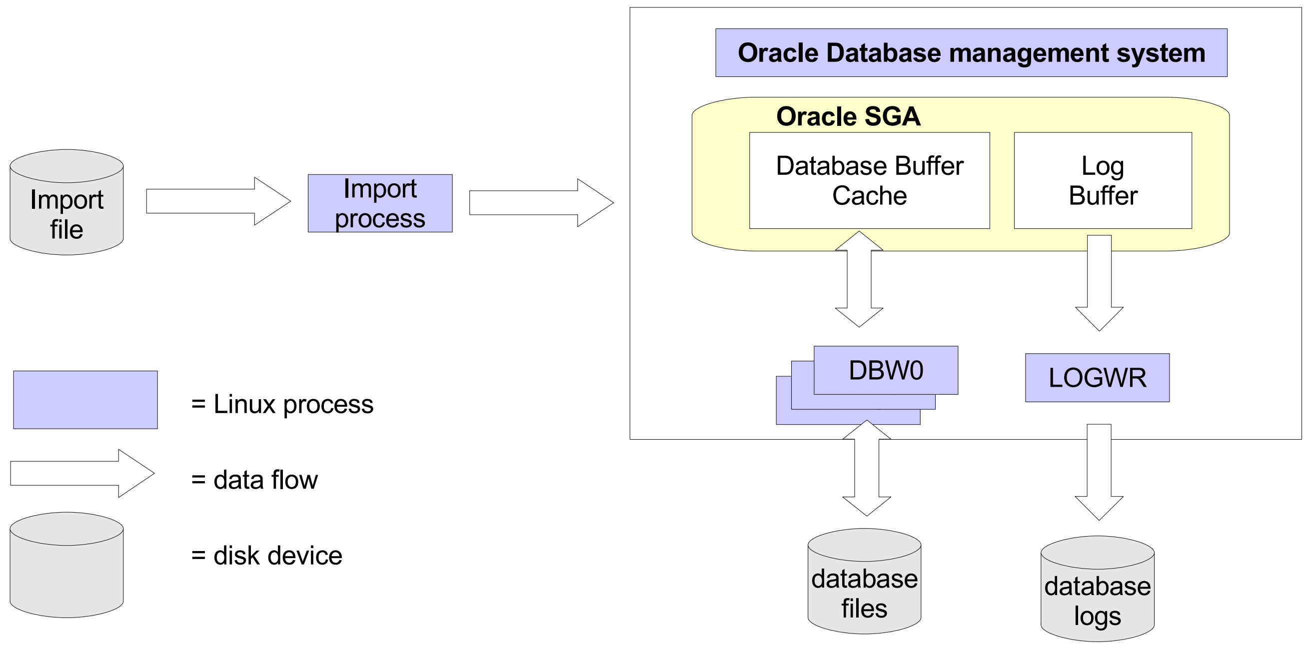


- G4 - 1st full-custom CMOS S/390®
- G5 - IEEE-standard BFP; branch target prediction
- G6 - Cu BEOL

- IBM eServer zSeries 900 (z900) - Full 64-bit z/Architecture®
- IBM eServer zSeries 990 (z990) - Superscalar CISC pipeline
- z9 EC - System level scaling

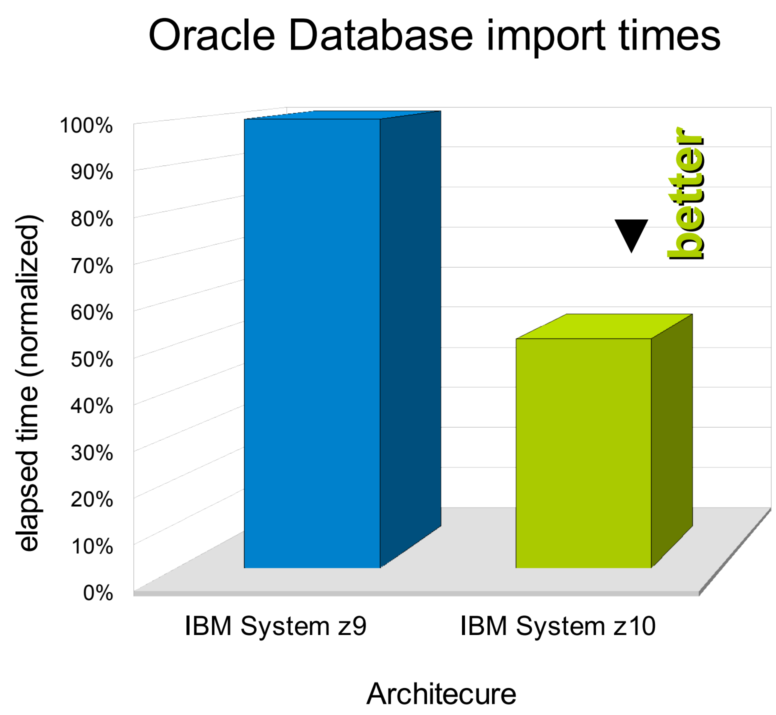
- z10 EC - Architectural extensions

Oracle Import – Environment



- Importing 20 GB data into the database from a single import file
- The whole performance depends on the speed of the single import process
  - A fast disk I/O subsystem is required, e.g. IBM DS8000

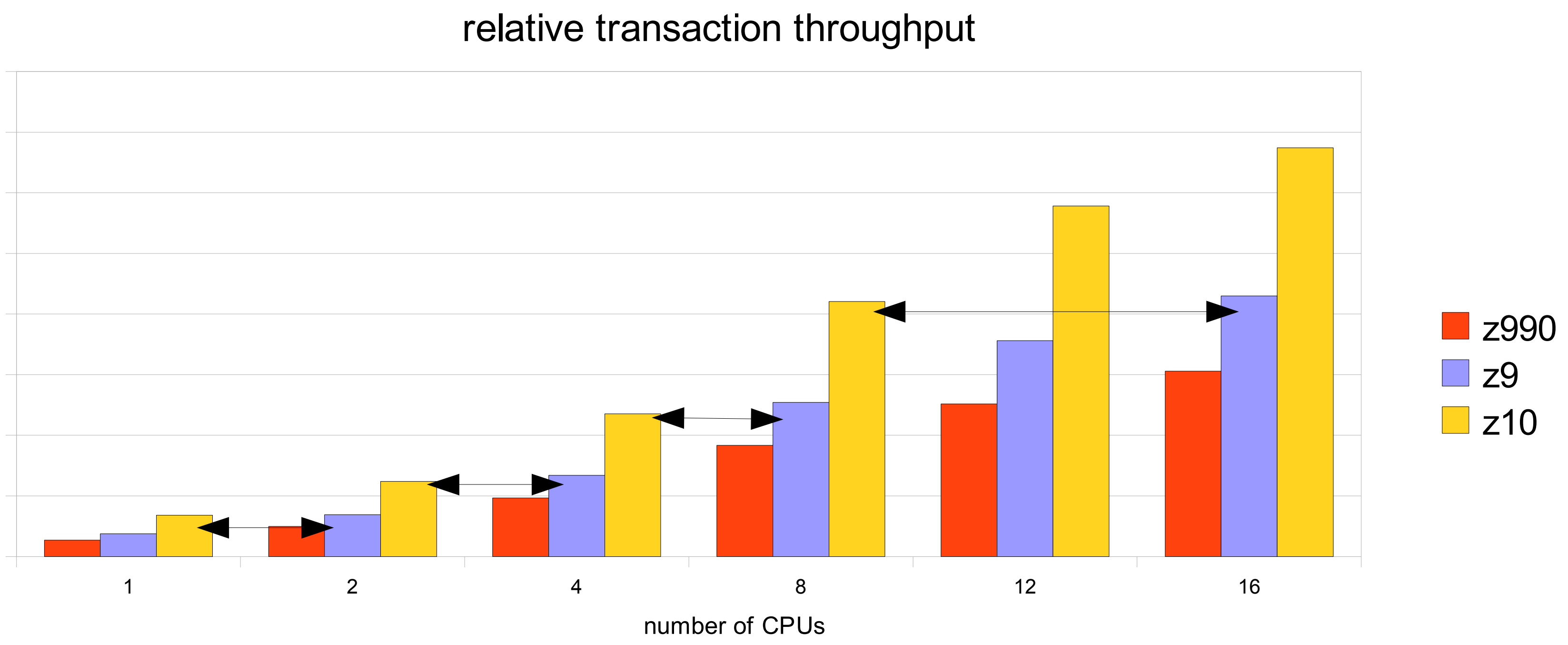
Oracle Import – Results



- IBM System z10 provides a reduction of the runtime by 50%

➢ **The import of the data runs twice as fast!**

z10 with Informix IDS 11 OLTP workload

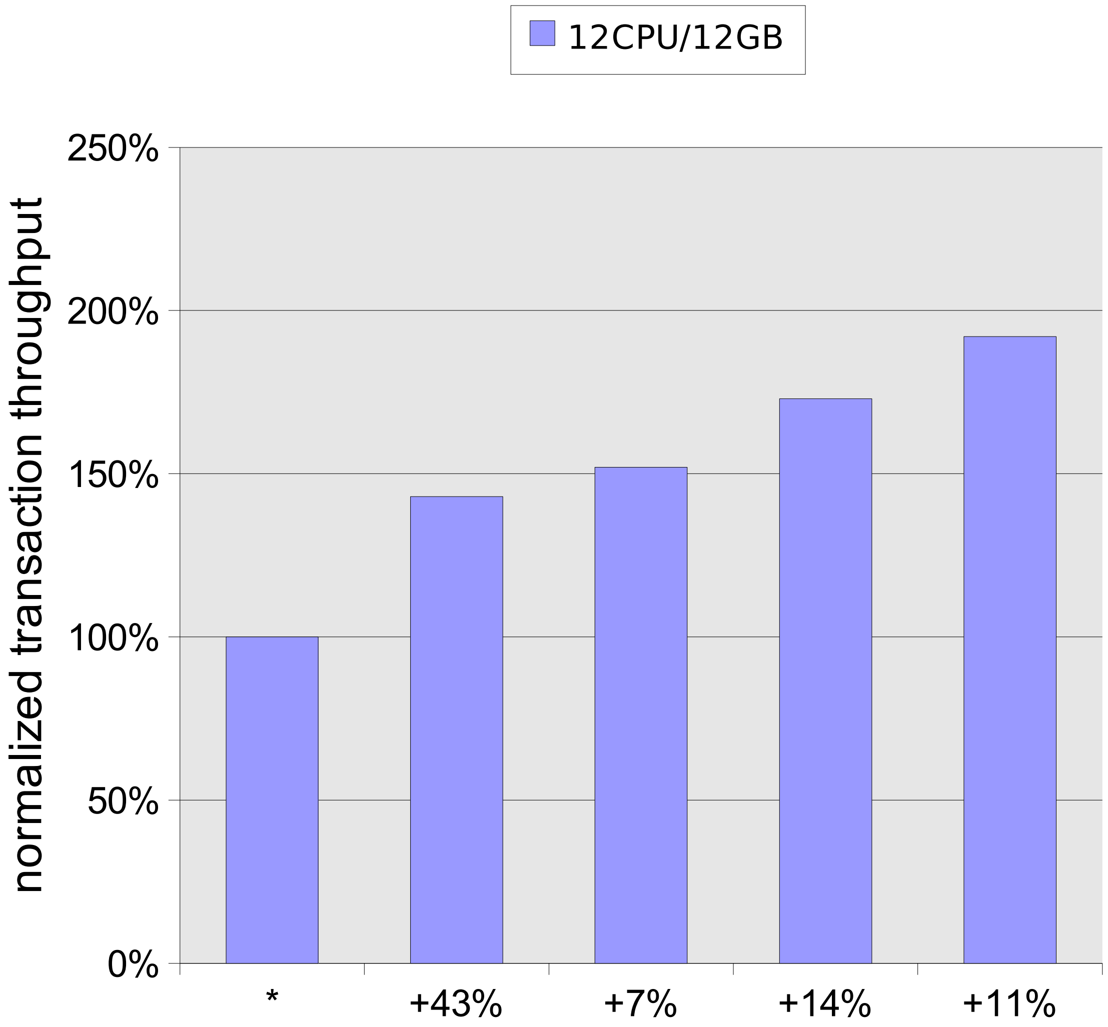


- **Throughput improvements**
  - z9 to z10: 65% - 82%
  - *n* numbers of z10 CPUs can do the same work as 2*n* z9 CPUs

Agenda

- Objectives
- Avoid unnecessary disk I/O!
  - Buffer Pool
  - Read Ahead
- If you can't avoid it - make it fast!
  - Disk Setup
  - Log Files
  - I/O Schedulers
  - Direct I/O and Asynchronous I/O
  - Hyperpav
  - Storage Pool Striping
- z9/z10 comparison
- **A little tuning story**
- Summary

A little DB2 Tuning Story



* started tuning here

tablespace prefetch 0
LVM readahead 0
→ **+43%**

Changed CHNGPGS_THRESH from 30 to 60
→ **+7%**

separate data and index bufferpools

for tablespaces with very large rows

→ **+14%**

8K pagesize for the index from the tablespaces with very large rows
→ **+11%**

Overall Tuning Result:
The throughput rate is nearly doubled compared to the starting point.

Agenda

- **Objectives**
- **Avoid unnecessary disk I/O!**
  - Buffer Pool
  - Read Ahead
- **If you can't avoid it - make it fast!**
  - Disk Setup
  - Log Files
  - I/O Schedulers
  - Direct I/O and Asynchronous I/O
  - Hyperpav
  - Storage Pool Striping
- **z9/z10 comparison**
- **Summary**

Summary (1)

- **Avoid physical disk I/O**
  - **<u>Database:</u>**
    - carefully choose the right buffer pool sizes
    - if any instance is doing read ahead, this should be done by the database
  - **<u>Linux:</u>**
    - disable readaheads for OLTP workloads
    - choose the right kernel parameter settings (shared memory)
- **Monitor the database cache hit ratio**

➢ **high cache hit scenarios scale well with the number of CPUs on IBM System z**

Summary (2)

- **If you can't avoid a lot of disk I/O, make it fast...**
  - Storage server:
    - use disks out of all ranks
    - alternate between the device adapter pairs and servers on the storage server
    - use storage pool striping
  - Linux:
    - ensure that a suitable I/O scheduler is used (e.g. deadline scheduler)
    - use Hiperpav with ECKD devices
  - z/VM:
    - use version 5.2 or higher
  - Database:
    - use striped storage structures provided by ASM or LVM to access disks in parallel
    - use separate disks for Data and Log files
    - async and direct I/O save memory and improve database performance

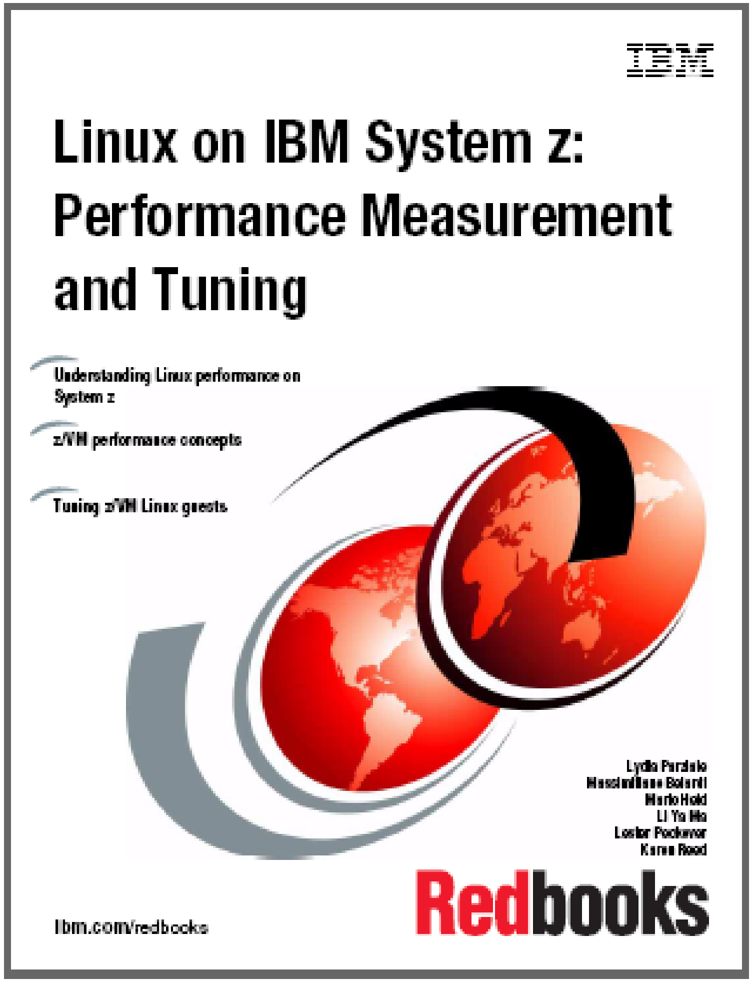
- **System z10 provides a performance improvement up to factor 2x**

Related Topics at 2009 System z Expo

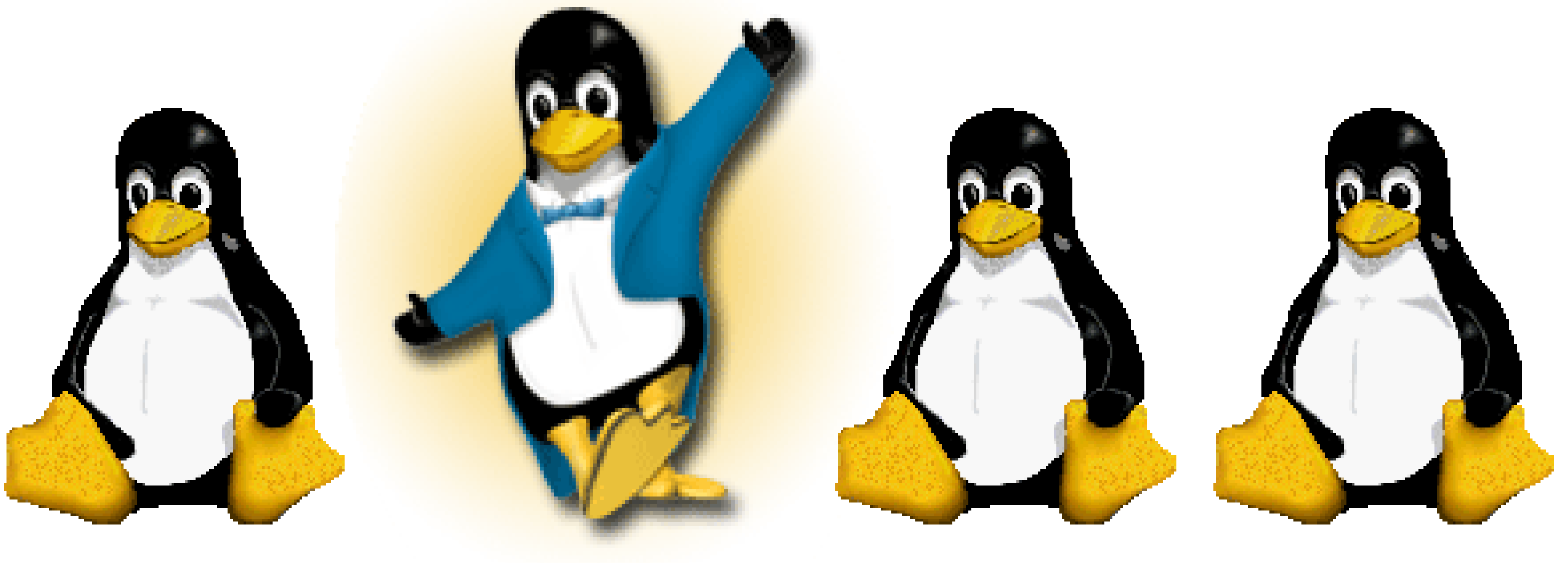
- **Performance Tuning and Monitoring: DB2 for Linux, Unix and Windows (LUW) for Linux**
  zLA08 Wednesday 4:10 PM

Visit us !

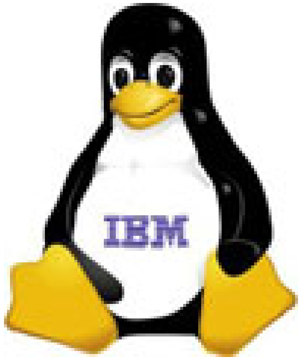
- **Linux on System z: Tuning Hints & Tips**
  - http://www.ibm.com/developerworks/linux/linux390/perf/
- **Linux-VM Performance Website:**
  - http://www.vm.ibm.com/perf/tips/linuxper.html
- **IBM Redbooks**
  - http://www.redbooks.ibm.com/



Questions



BACKUP



Workload description

- **OLTP workload, simulating an order entry system**
- **Five different transaction types, executed randomly within a defined mix**
  - new order
  - payment
  - order status
  - delivery
  - stock status
- **High and low database buffer read hit ratios simulate different production environment conditions**

Linux Kernel parameters (1)

- **Linux kernel parameters can limit the amount of shared memory!**
  Shared memory is needed for database buffer pools!
  - kernel.shmall available memory for shared memory in 4 K pages
  - kernel.shmmax maximum size of one shared memory segment in bytes
  - kernel.shmmni maximum number of shared memory segments

- **shm parameter recommendations:**
  - set shmall and shmmax equal to the current memory size, so that they're not a limiting factor.

Linux memory	shmall	shmmni	shmmax
8 GB	1971200	4096	8074035200

- **Kernel parameter changes were made in /etc/sysctl.conf**
  - Enable sysctl service with `chkconfig boot.sysctl on`
  - sysctl.conf is read during boot time by the sysctl command
  - Insert a line for each kernel parameter according to

```
kernel.parameter = value
```

Linux Kernel parameters (2)

- **Take care for the database specific recommendations for following kernel parameters:**

- **Kernel semaphores limits**
  - kernel.sem (Kernel semaphores limits)
    Max. semaphores per array / max. Semaphores system wide / max. ops per per semop call / max. number of arrays
  - e.g. kernel.sem = 250 32000 32 128

- **Kernel message limits**
  - kernel.msgmni maximum queues system wide
  - kernel.msgmax maximum size of message (bytes)
  - kernel.msgmnb default size of queue (bytes)