Performance Experiences with Databases on Linux for System z

Thomas Weber (tweber@de.ibm.com)

WAVV Conference 2009

Orlando, FL, May 15-19

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.

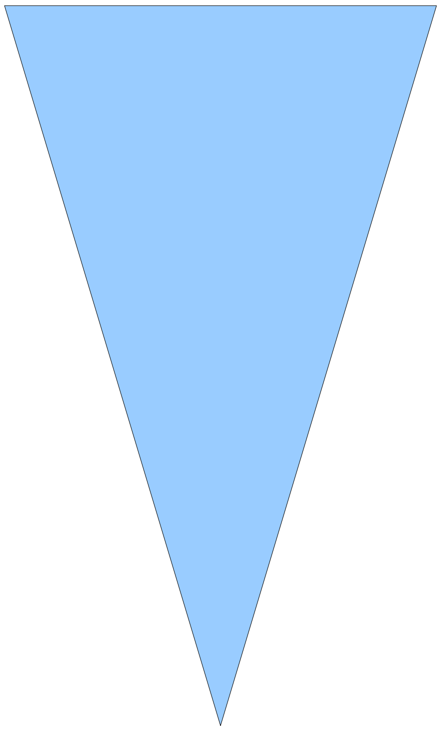
Agenda

- Performance Experiences with Databases on Linux for System z
  - Workload
  - Storage Server Internals
  - Linux kernel
  - Disk I/O options
  - Some database test scenarios

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



Workload description

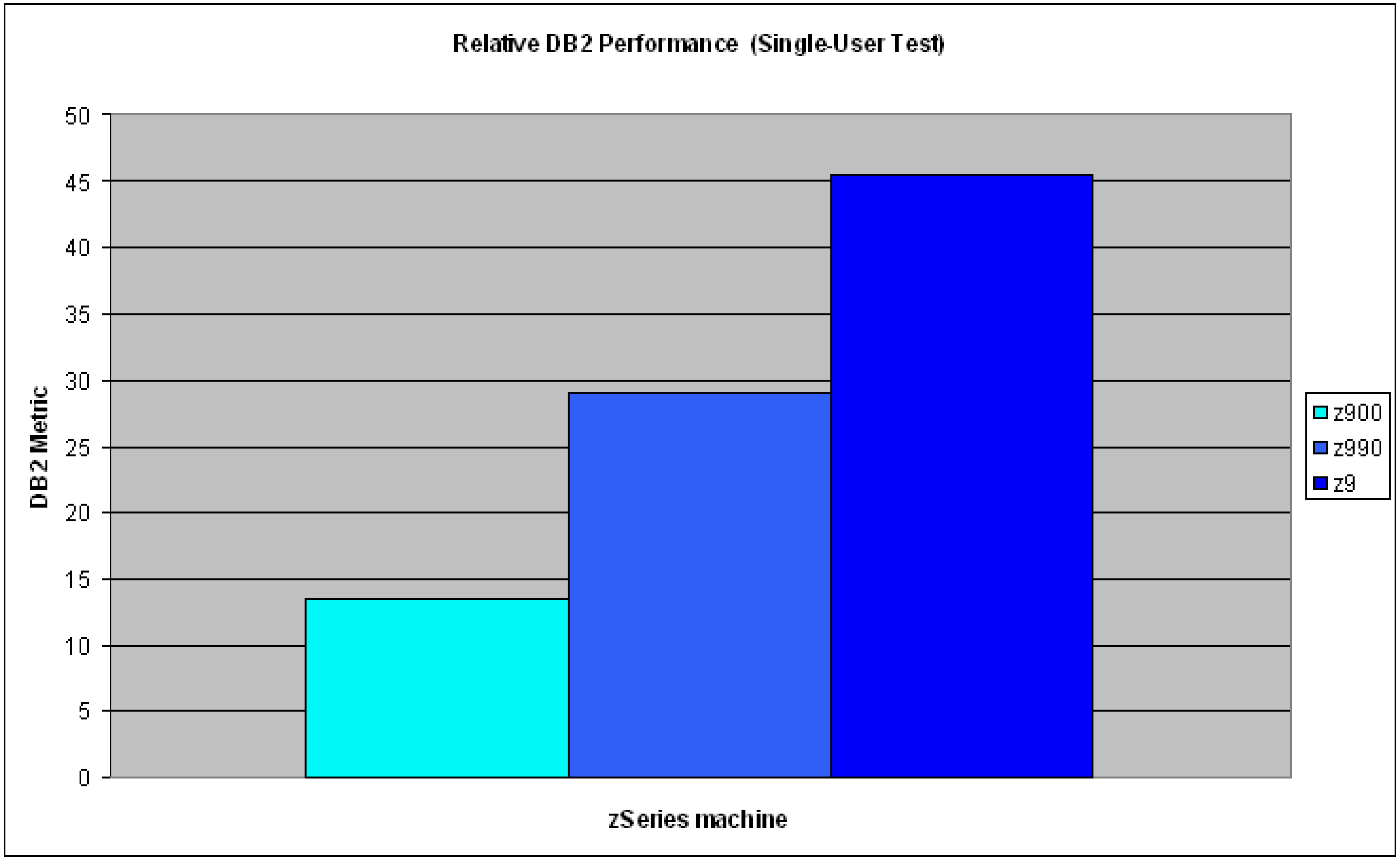
- OLTP workload, simulating an order entry system
- Five different transaction types, executed randomly within a defined mix
  - new order
  - payment
  - order status
  - delivery
  - stock status
- High and low database buffer read hit ratios simulate different production environment conditions

How does this workload impact on performance?

The workload characteristics are:

- 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
- 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

System z machine evolution



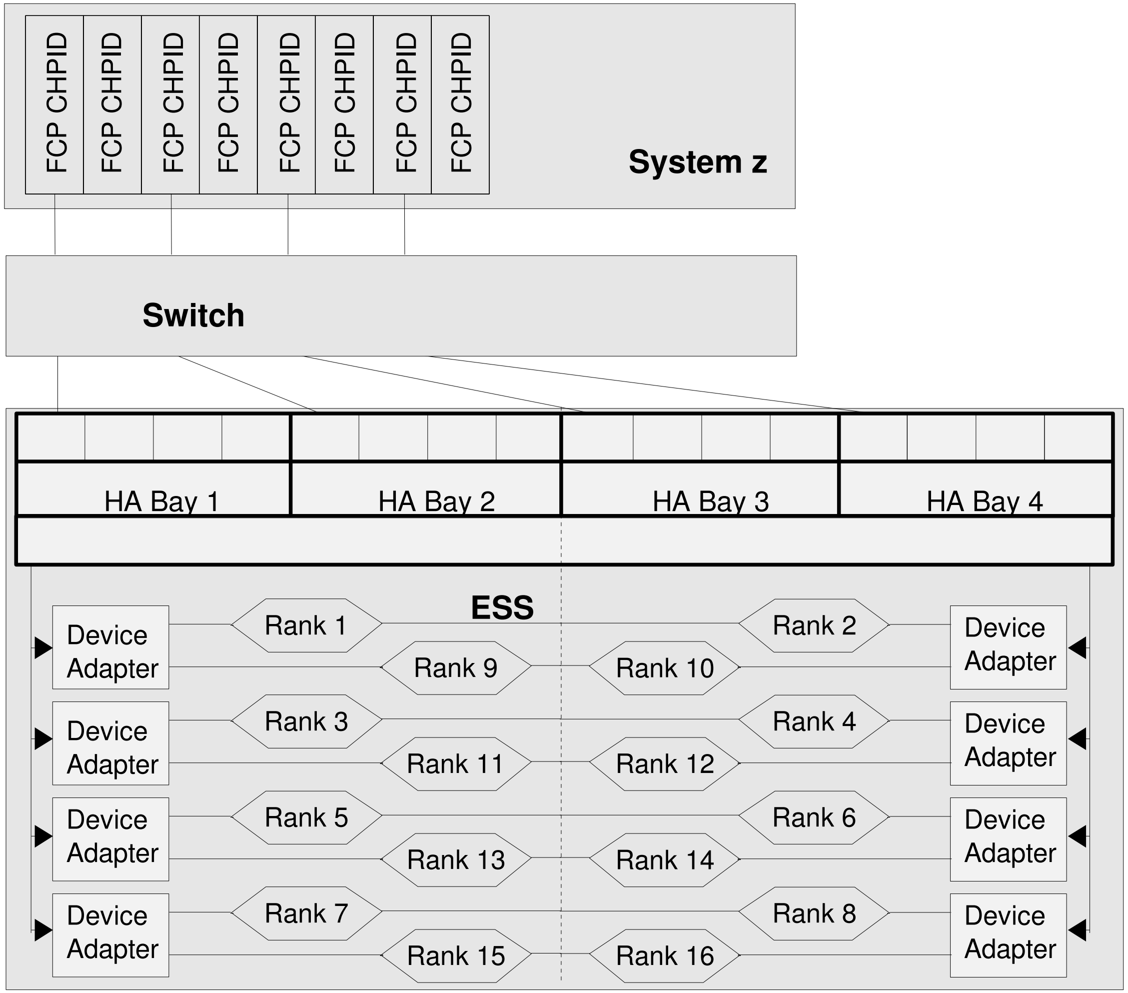
- DB2 metric: z900 to z990 = 2.2x → z990 to z9 = 1.6x
- Clock speed: z900 to z990 = 1.6x → z990 to z9 = 1.4x

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!
- Problem: You ask for 16 disks and your system administrator gives you addresses 5100-510F...
- This is close to the worst case in terms of disk performance...
- So - what's wrong with that?

Storage Server Architecture (1)

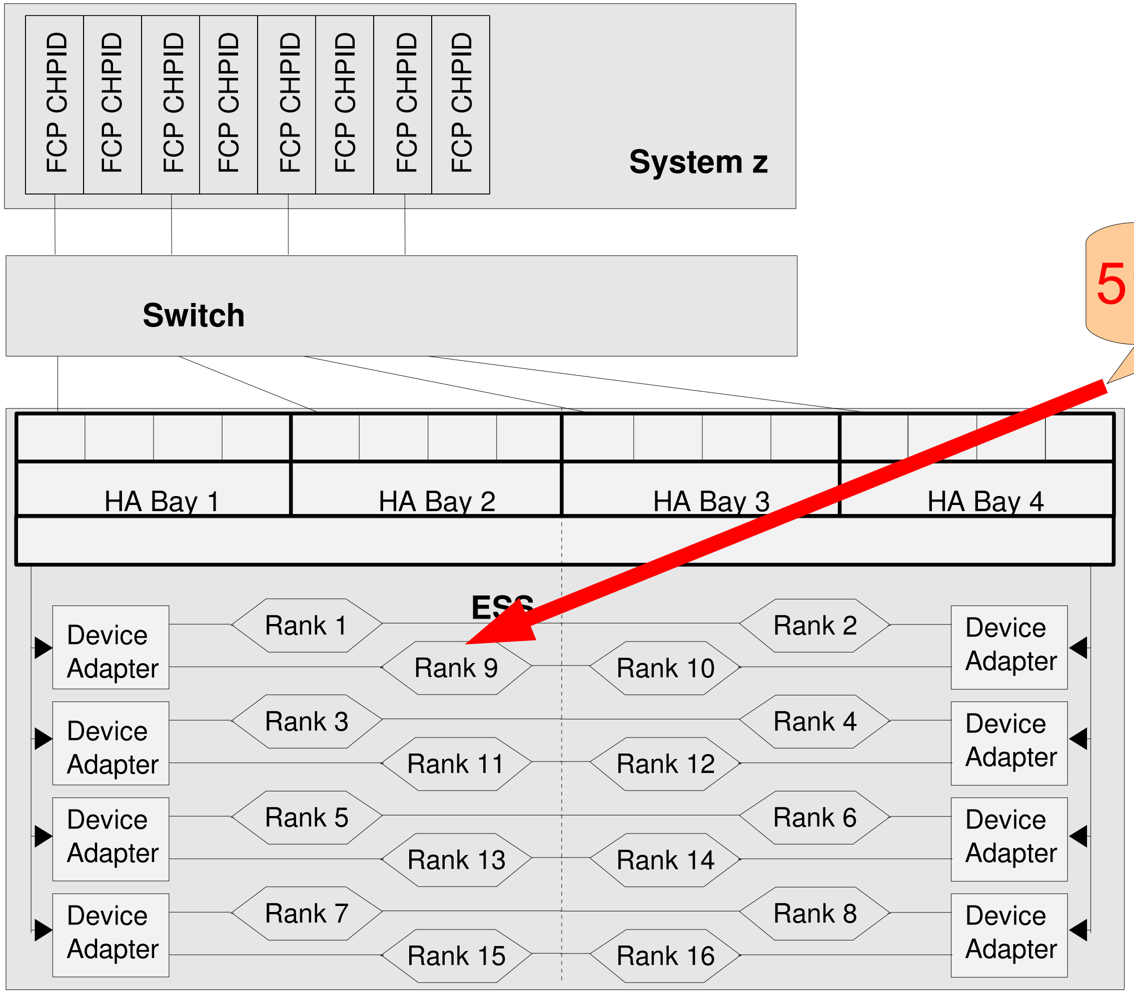
–Let's have a closer look at the elements involved



- **CHPIDs**
  - 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
- **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 implements a RAID array (with logical disks)

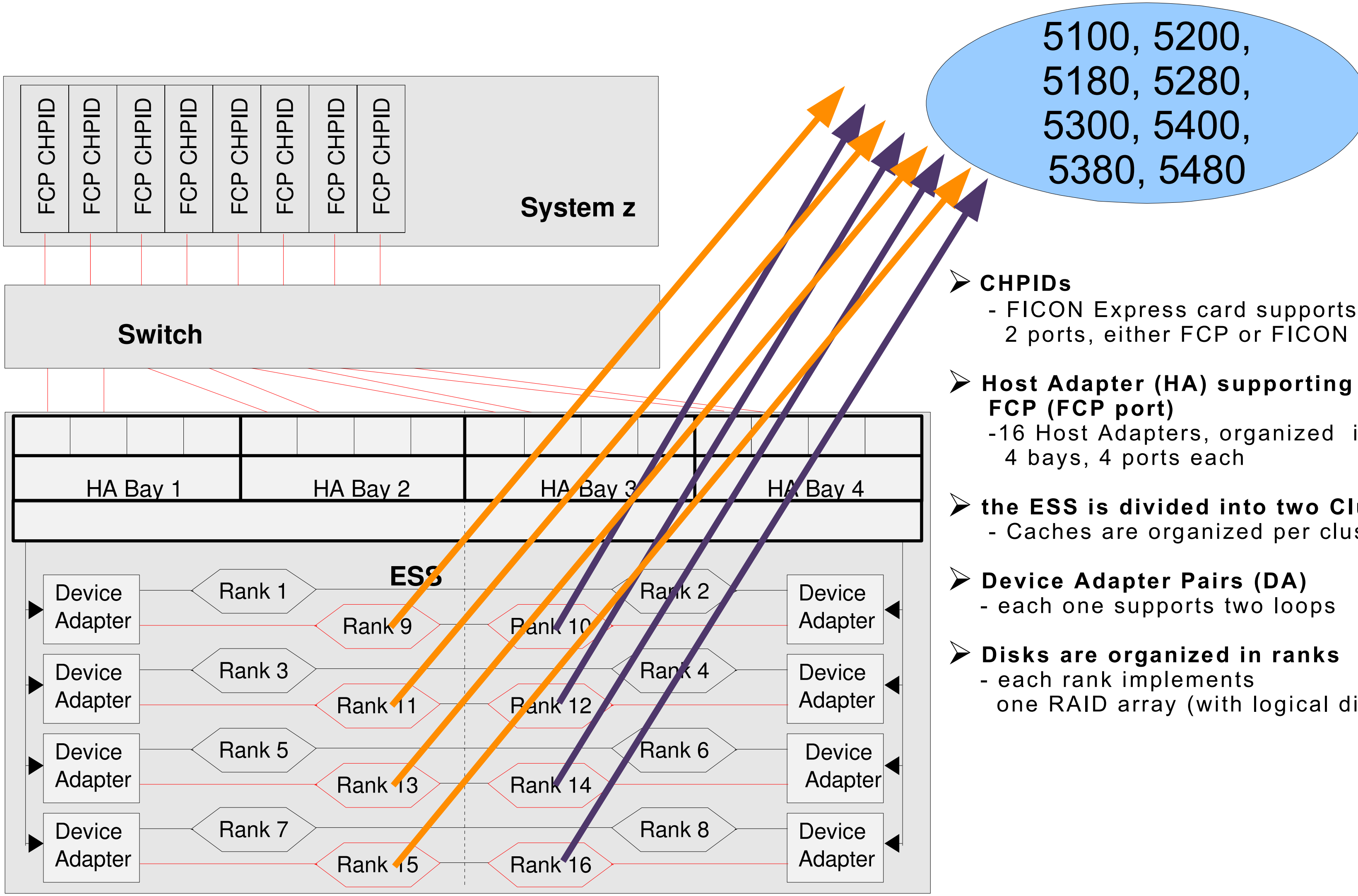
Storage Server Architecture (2)

– Let's have a closer look at the elements involved



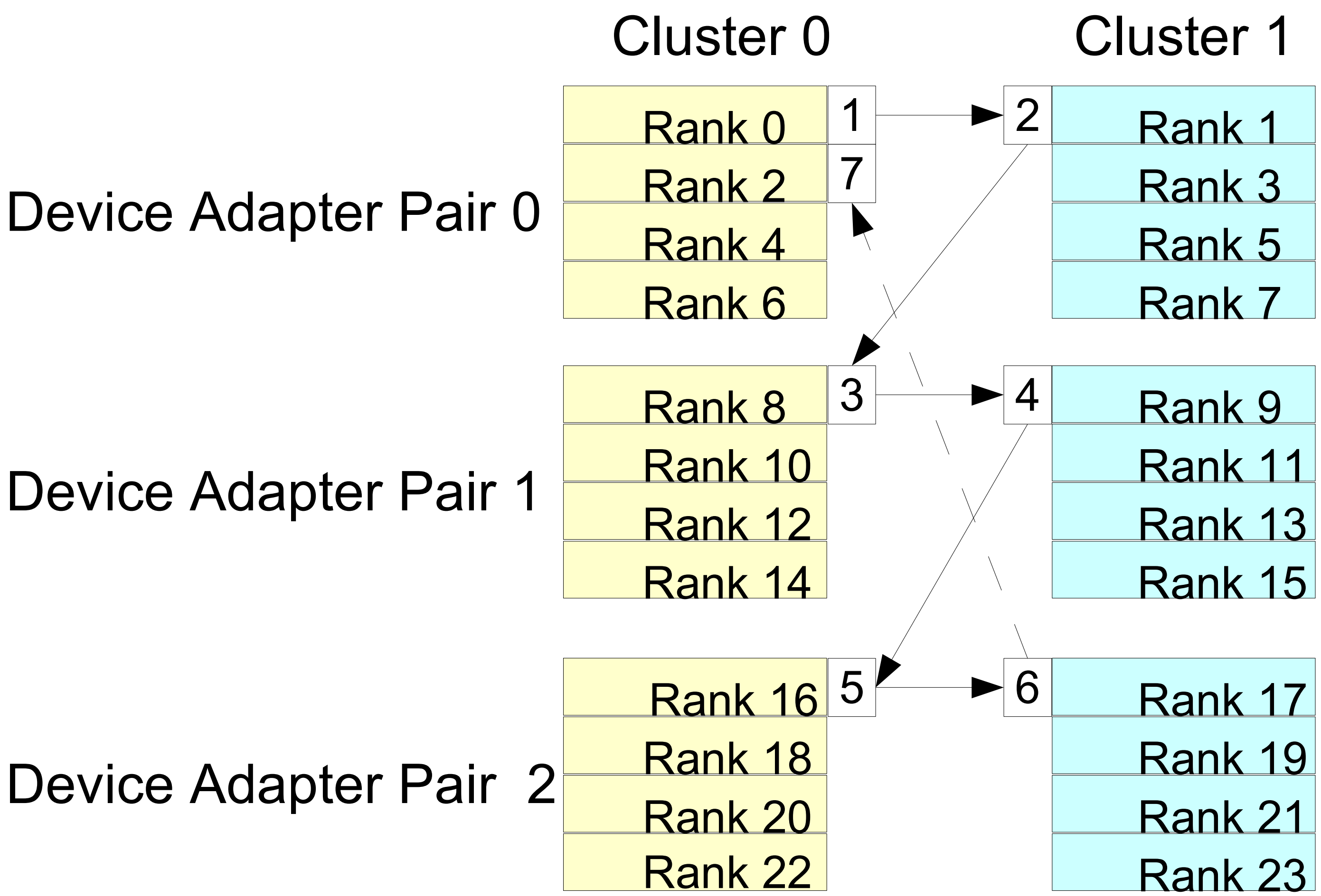
- **CHPIDs**
  - FICON Express card supports ... either FCP or FICON
- ... **(HA) supporting ... rt)**
  - ... adapters, organized in 4 bays, 4 ports each
- **ESS is divided into two Clusters**
  - Caches are organized per cluster!
- **Device Adapter Pairs (DA)**
  - each one supports two loops
- **Disks are organized in ranks**
  - each rank implements a RAID array (with logical disks)

Optimized disk setup for a Storage Server (3)



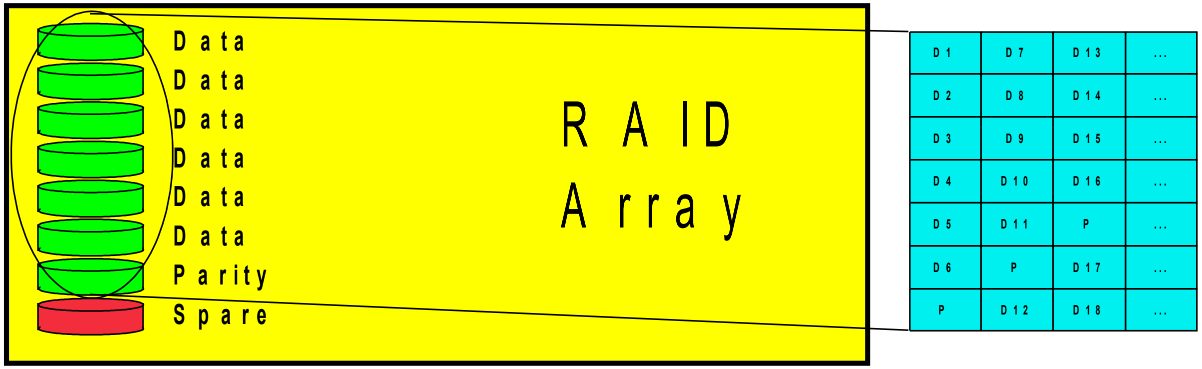
- **CHPIDs**
  - FICON Express card supports 2 ports, either FCP or FICON
- **Host Adapter (HA) supporting FCP (FCP port)**
  - 16 Host Adapters, organized in 4 bays, 4 ports each
- **the ESS is divided into two Clusters**
  - Caches are organized per cluster!
- **Device Adapter Pairs (DA)**
  - each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

Sample for an optimal disk selection (4)

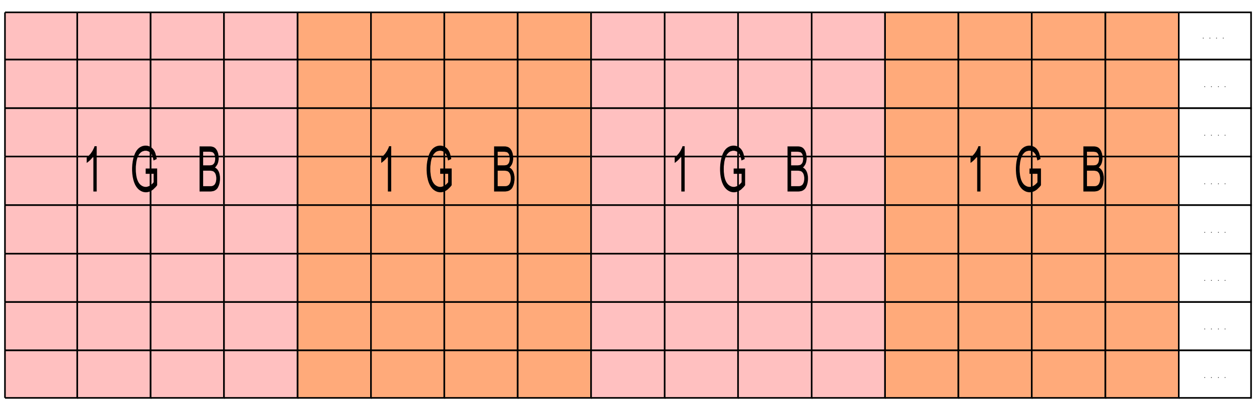


– Select the disks in the order from 1 to 7 for your Linux system.

DS8000 - Storage Pool Striping (1) - Rank

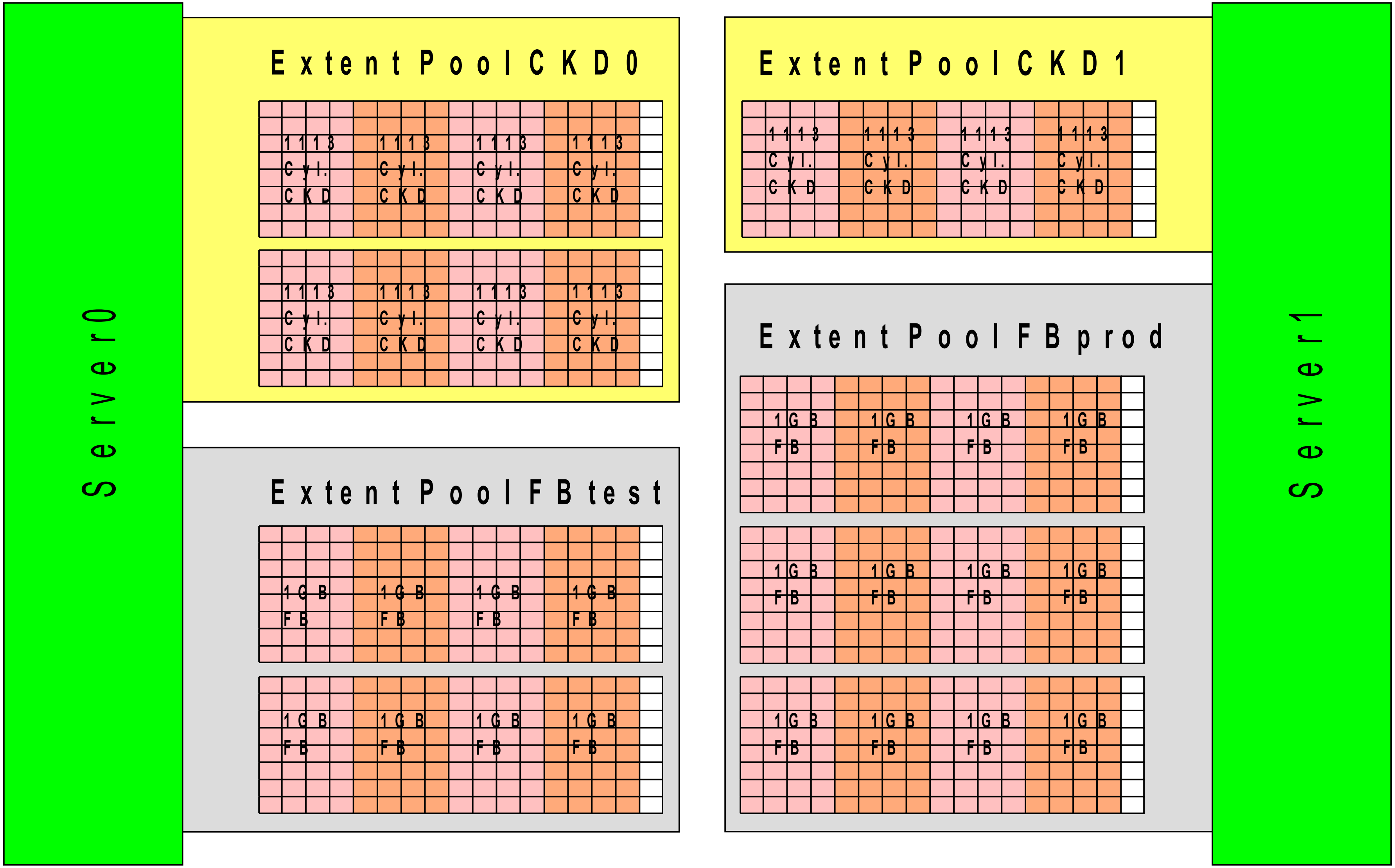


Creation of a Rank



FB Rank of 1GB extents or CKD Ranks with 3390-1 extents

DS8000 - Storage Pool Striping (2) - Extent Pool



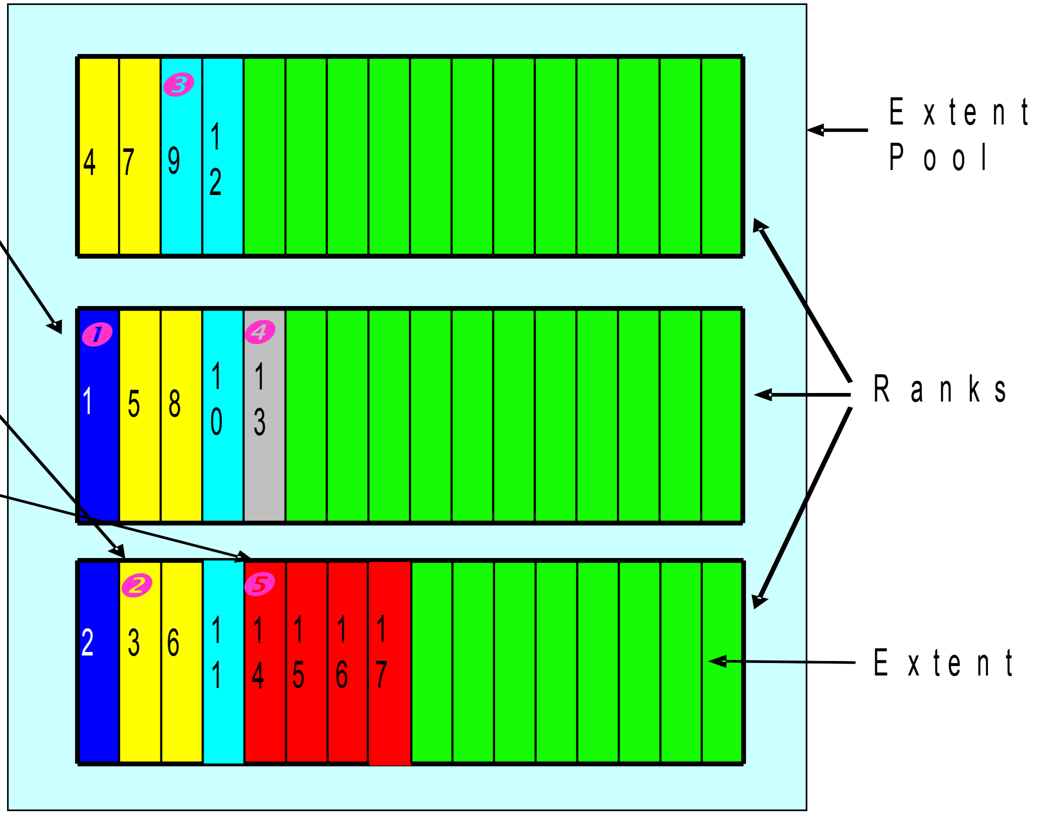
DS8000 - Storage Pool Striping (3) – Striped Volumes

1 LUN / CKD volume is created

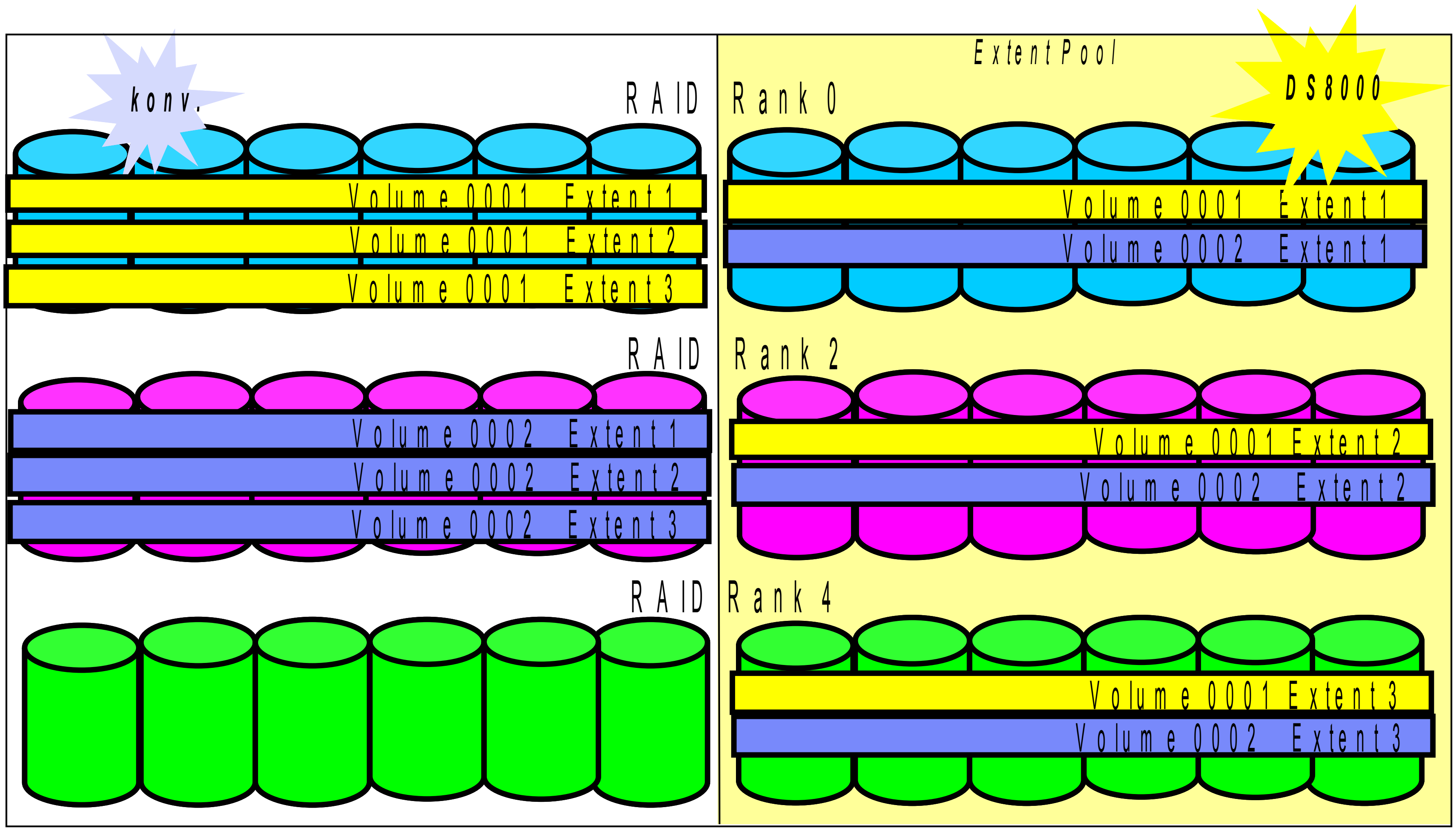
Striped volume with two Extents created

2 Next striped volume starts at next rank (six extents in this example)

5 Non-striped volume created Starts at next rank



DS8000 - Storage Pool Striping (4)



DS8000 - Storage Pool Striping (5) - Overview

	LVM striping	DS8000 storage pool striping
striping is done in	Linux	storage server
effort to construct the volume	take care of picking subsequent disks from different ranks	configure storage server
administrating disks within Linux	can be challenging, e.g. several hundred for a database	simple
volume extendable ?	yes	no
maximum I/O request size	stripe size (e.g. 64KB)	maximum provided by the device driver (e.g. 512KB)
multipathing	SCSI: assign paths round robin to disks, multipath failover ECKD: path group	SCSI: multipath multibus ECKD: path group
usual disk sizes	LV = many disks SCSI 10GB to 20GB, ECKD mod9 or mod27	Volume = 1 disk, SCSI unlimited, e.g. 300GB, ECKD max. mod54
extent pool	1 rank	multiple ranks
maximum number of ranks for the constructed	total number of ranks	Total number of one server side (50%)

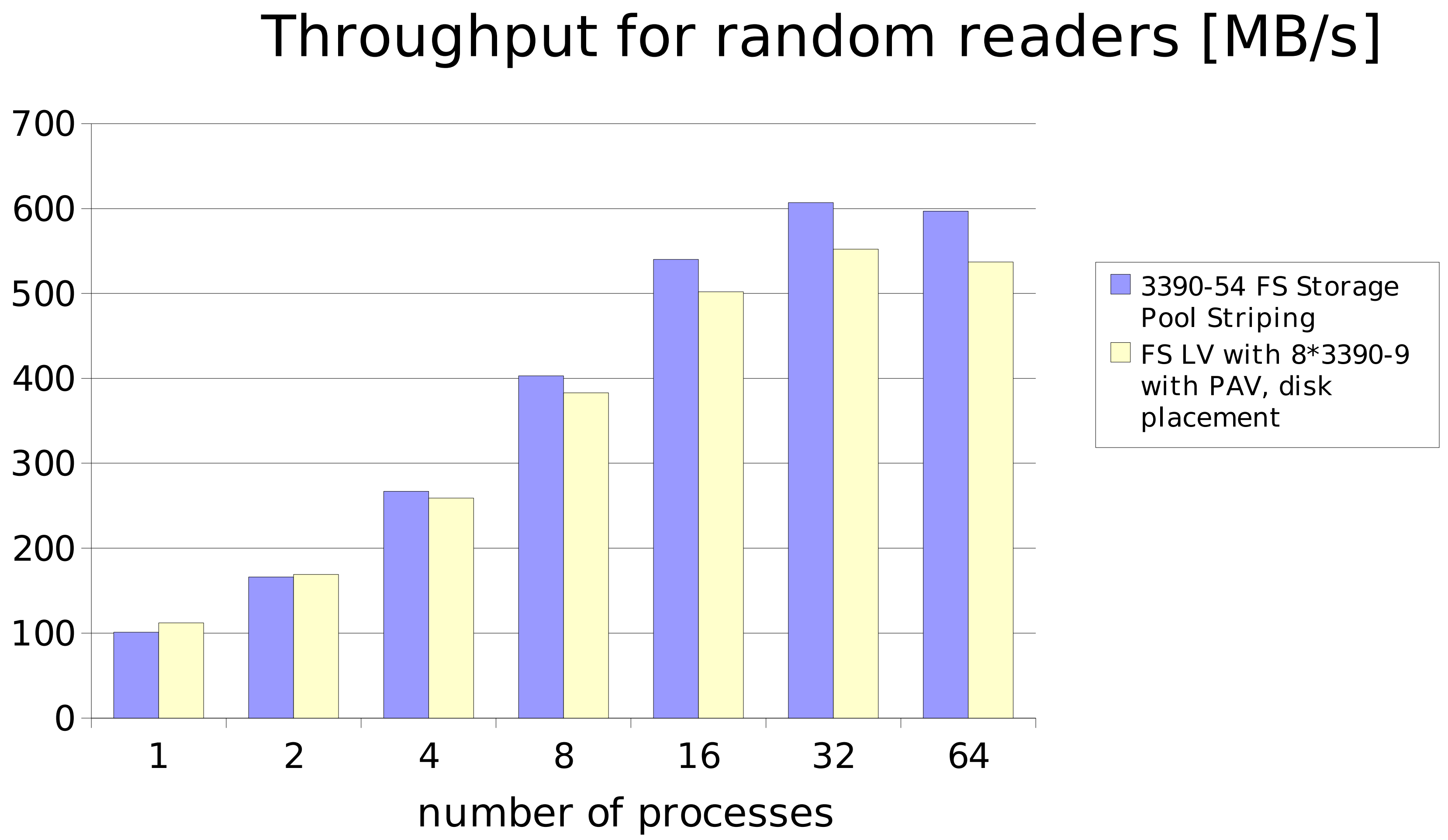
DS8000 - Storage Pool Striping (6) - Impact

- License Machine Code 5.30xx.xx
- Stripe the extents of a DS8000 Logical Volume across multiple RAID arrays
- Will improve throughput for some workloads
- 1 GB granularity, random workloads will generally benefit more than sequential ones
- Cannot span servers
- Can be combined with LVM striping or DB2 database container striping
- Risk: Losing one rank
- Tip: 4 – 8 ranks / extent pool
- Assumption: Disk placement no longer necessary

Storage Pool Striping (7) – ECKD Measurement Setup

- z9 LPAR, 8 CPUs, 256 MB
- DS8000 Server 0: 1 Extent-Pool with 8 ranks
  Server 1: 8 Extent-Pools, 1 per rank
- 4 x 4 Gb/s Ficon card
- Internal Linux Driver
- HiperPAV, 63 Aliases per Server

Storage Pool Striping (8) – Iozone Workload Test



DS8000 - Storage Pool Striping (8) – Summary

- General pros / cons
  - Storage pool striped volumes are as simple to set up and to administrate as a few large disks
  - Striping on the storage device lowers CPU consumption (LVM) on the Linux side
  - Stripe size is 1 GB
  - Rank failure will hit all disks
- Results with ECKD disks
  - Combination with HiperPAV reaches nearly the same performance as Linux solution
  - Without HiperPAV only one I/O outstanding per DASD is possible, which limits the performance
  - FICON path groups doing the load balancing
- Results with SCSI disks
  - Linux striped logical volumes are faster but the Logical Volume Manager (LVM) takes more CPU cycles than e.g. the multipath daemon
  - For random workloads the multipath daemon used to distribute workload to the FCP channels needs improvements (work in progress)
- If you don't use striping in Linux today, consider to enable it at least in the storage server – your performance won't become worse

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 database workloads
  - number of stripes equal to the number of disks in VG
- let the database do the striping: e.g. 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 then

Read ahead setup – avoid unnecessary I/Os (random OLTP)

- on database level
  - read ahead can be disabled for random OLTP, compare results w/ and w/o read ahead
  - logically only the database is the instance which can do a meaningful read ahead
    - Informix: set the onconfig parameters RA_PAGES and RA_THRESHOLD to 0
    - DB2: set the tablespace parameter PREFETCHSIZE to 0
    - Oracle: set the oracle profile parameter DB_FILE_MULTIBLOCK_READ_COUNT to 0
- on Logical Volume Manager (LVM) level
  - disable it by setting the read ahead with the commands lvcreate or lvchange
    - LVM2: -r, --readahead none   (instead of auto)
    - LVM1: -r, --readahead 0
- on Linux block device layer level
  - set the value to 0 using the blockdev command
    - for example: blockdev --setra 0 /dev/sda

Linux Kernel parameters (1)

- Shared memory kernel parameters:
  - **kernel.shmall** available memory for shared memory in **4 K pages**
  - **kernel.shmmax** maximum size of one shared memory segment in **byte**
  - **kernel.shmmni** maximum number of shared memory segments
  - Shared memory is needed for database buffer pools!

- 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

  - start with a database buffer pool size of 60% from the current memory
  - increase database buffer pool size and monitor free memory and swapping activity
  - stop until the desired size is reached and right before swapping starts
  - it is recommended to leave at least 5% free memory (free command)

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

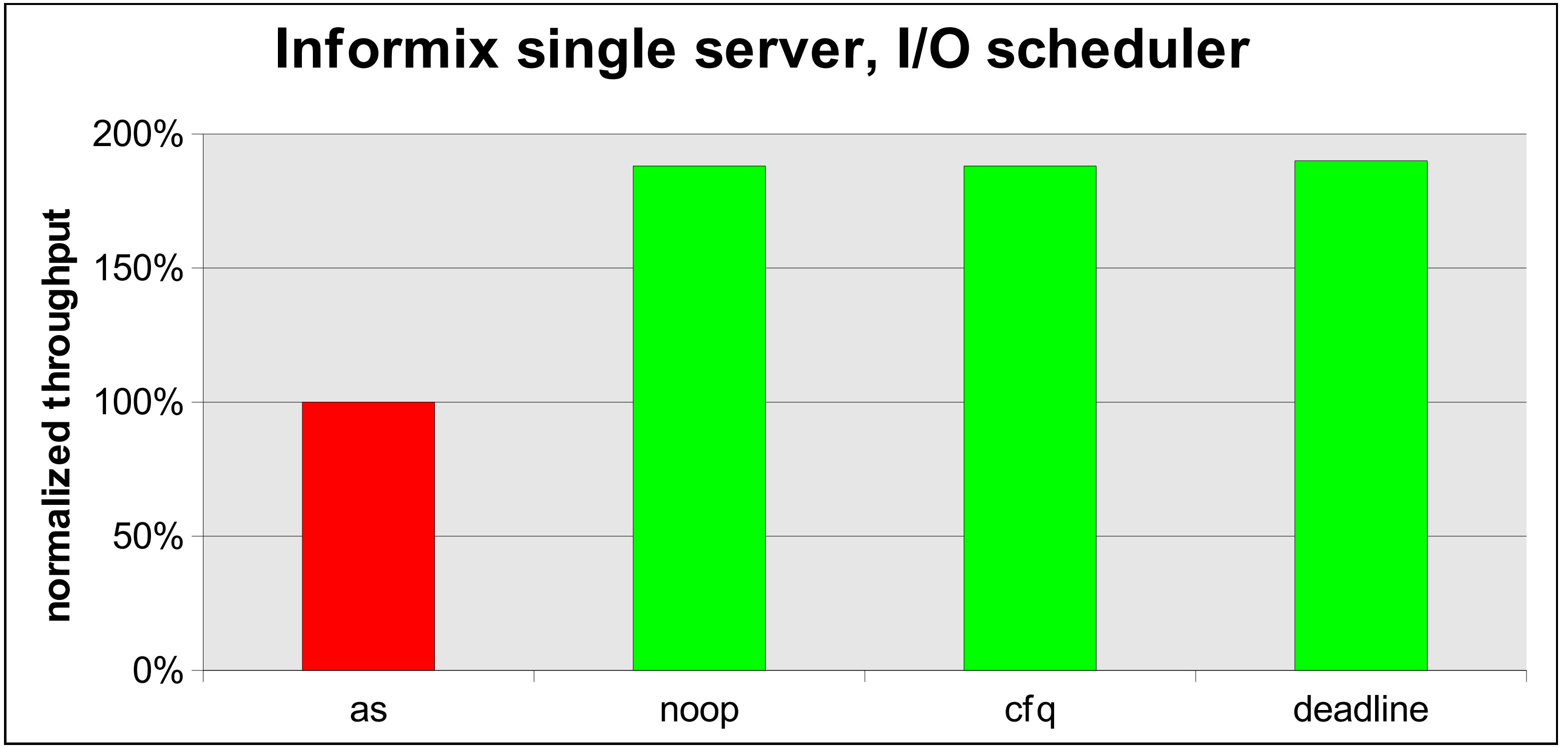
- Kernel message limits
  - **kernel.msgmni** maximum queues system wide
  - **kernel.msgmax** maximum size of message (bytes)
  - **kernel.msgmnb** default size of queue (bytes)

- Permanent Kernel parameter changes should be set 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 2.6 I/O schedulers

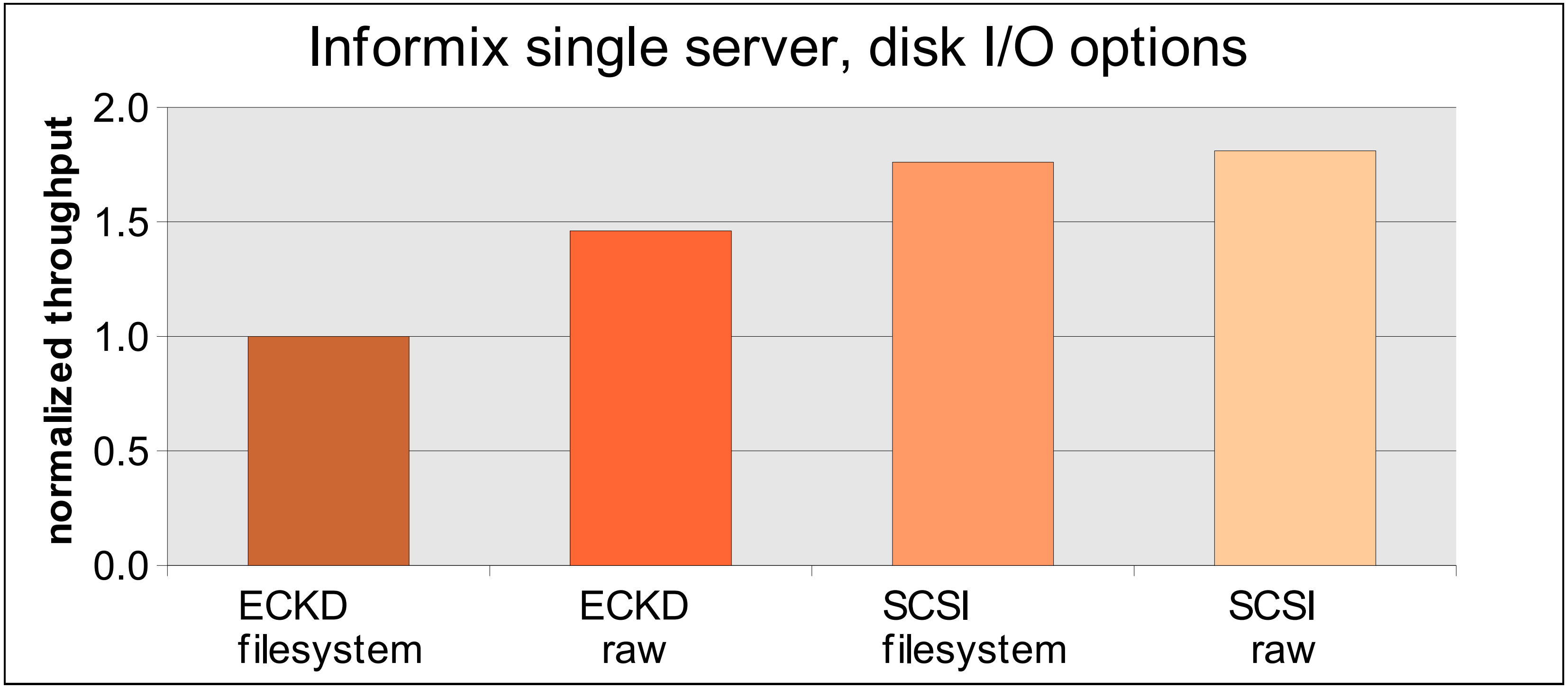
- four different I/O schedulers are available
  - **noop** scheduler
    does only request merging
  - **deadline** scheduler
    avoids read request starvation, offers the possibility to give write requests the same priority like reads
  - anticipatory scheduler (**as** scheduler)
    designed for the 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 type attachments

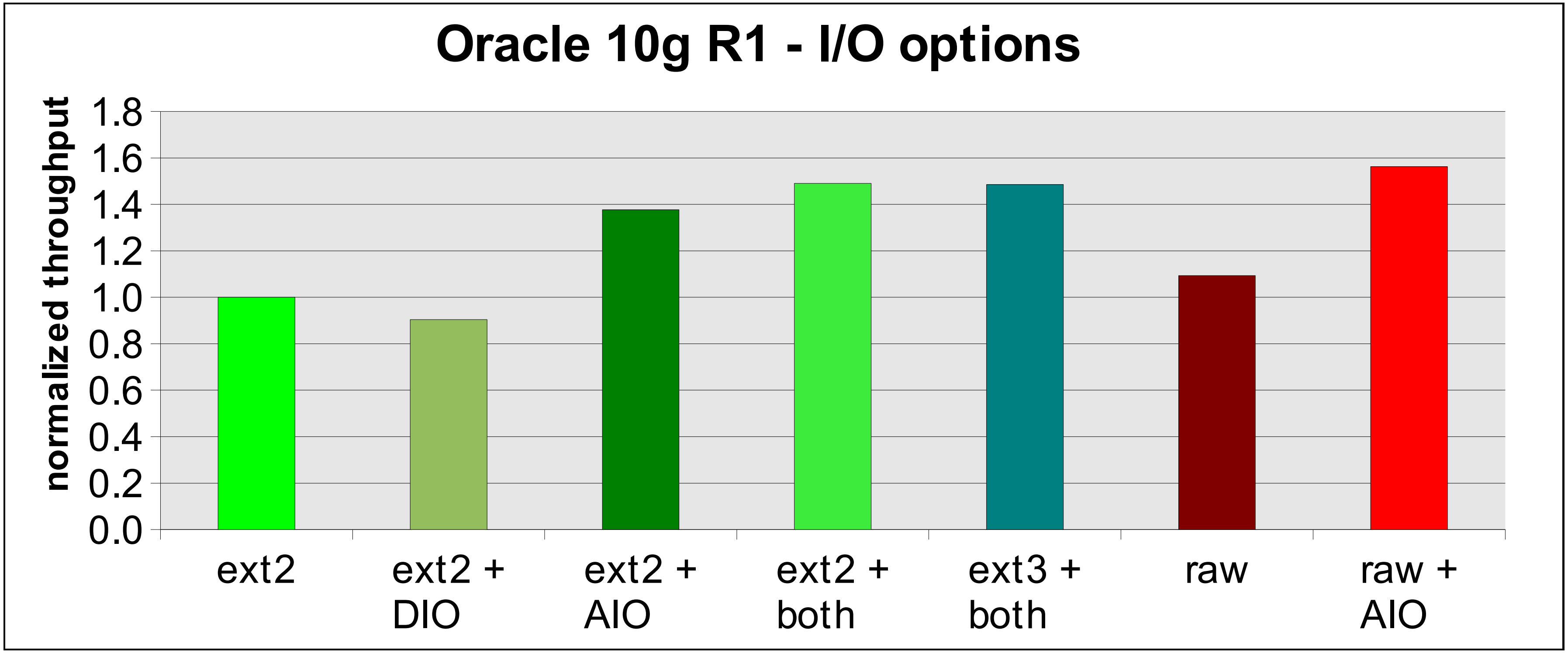


- all tests were done with ext2 filesystem
- best results with SCSI file system and ECKD raw
- SCSI file system was used for all following scaling tests with Informix

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
  - no copying of the data to the Linux page cache
  - advantage
    - saves page cache memory
    - same data is not cached twice
    - 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 for database processing if available

DIO and AIO – Results



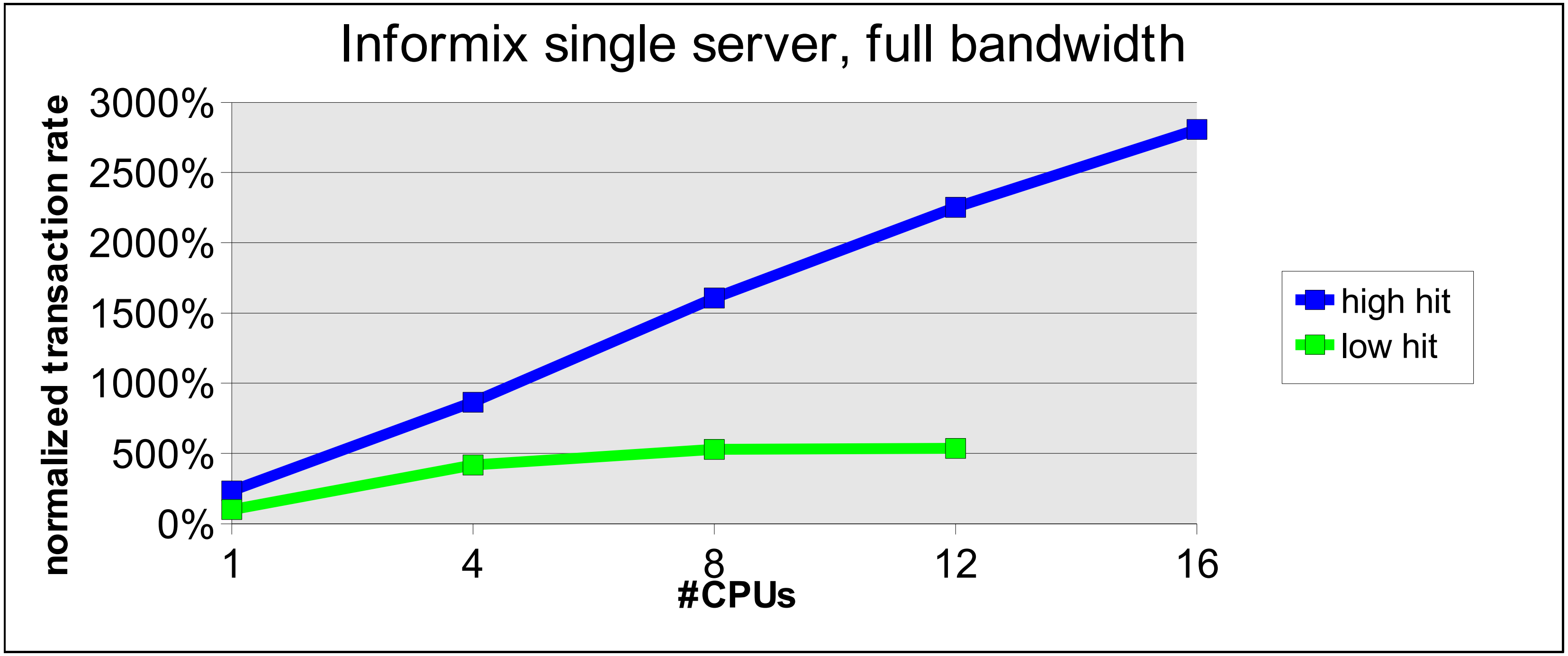
- combination of DIO and AIO shows the best results for the Linux file system (ext2 + both and ext3 + both)
- best throughput rate with raw I/O + AIO

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 usually sequential I/O
- database log files and the database files on the same disks (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 disk IO transfer rate
    - degradation of the priority when writing logs
    - overall a performance degradation of the throughput rate
- make separate log and data devices, use if possible
  - different ranks on the same storage server
  - use different storage servers

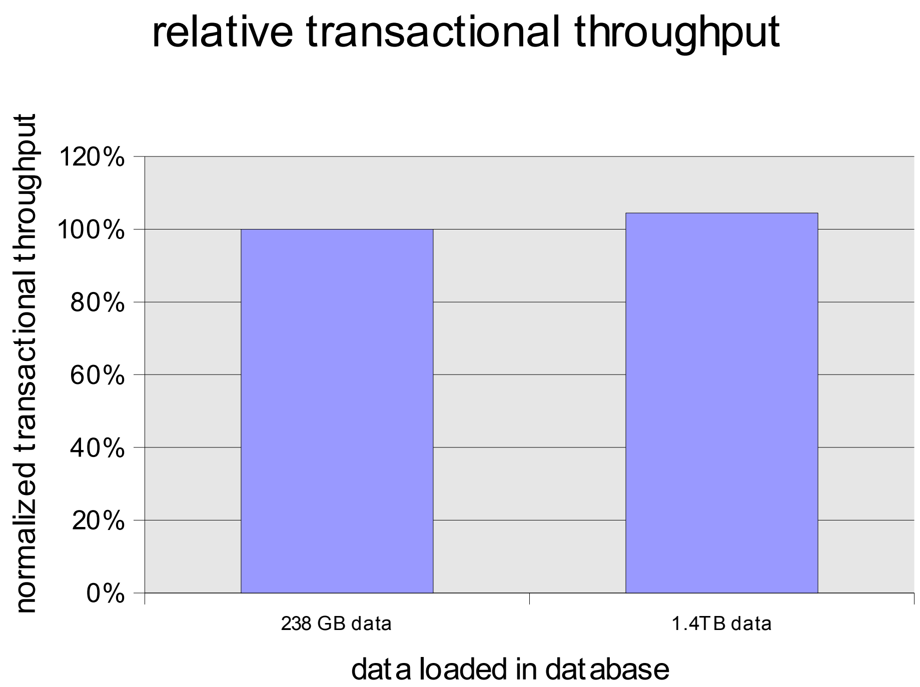
→ This ensures a contiguous log disk IO and good transaction rates.

CPU Scalability and the cache hit ratio



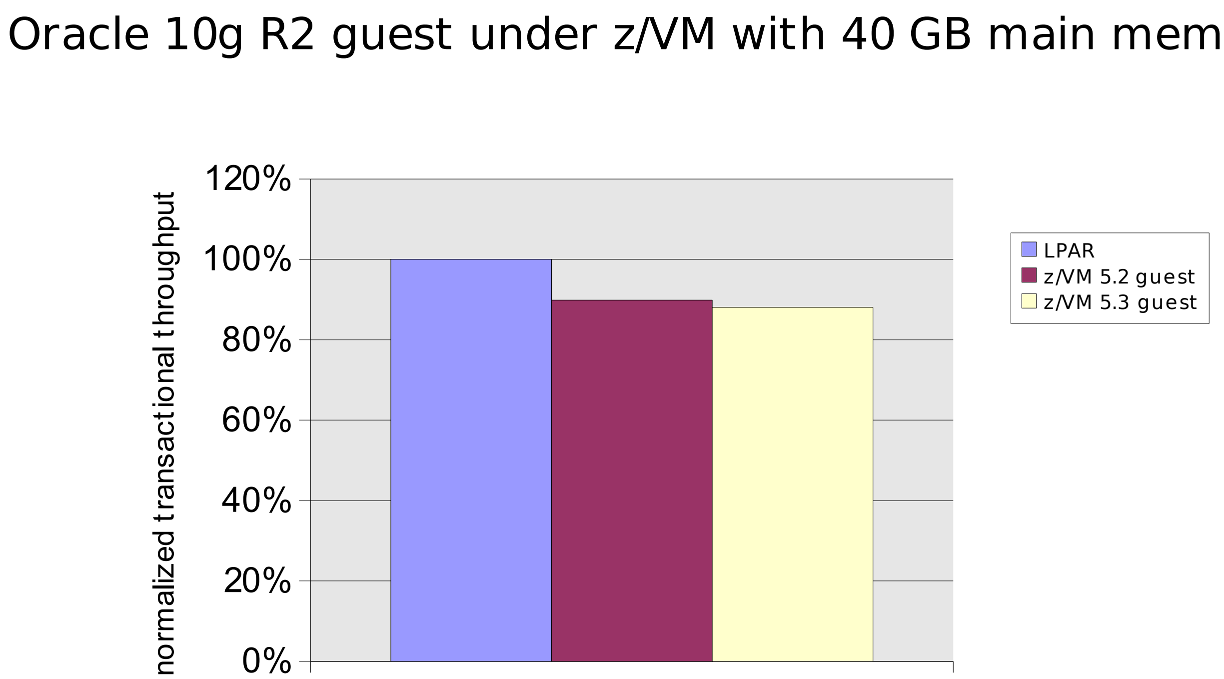
- High hit cache ratio scenario is a successful implementation for **avoiding disk I/O**
  - throughput rate scales from 1 to 16 CPUs as long as the complete database fits into the bufferpools (99% buffer hit ratio)
- Low (below 90%) and high hit scenario mark the possible throughput bandwidth
  - where the low hit scenario is the lower and the high hit scenario is the upper limit
  - a typical workload is somewhere in between

DB2 v9 - Let the database grow



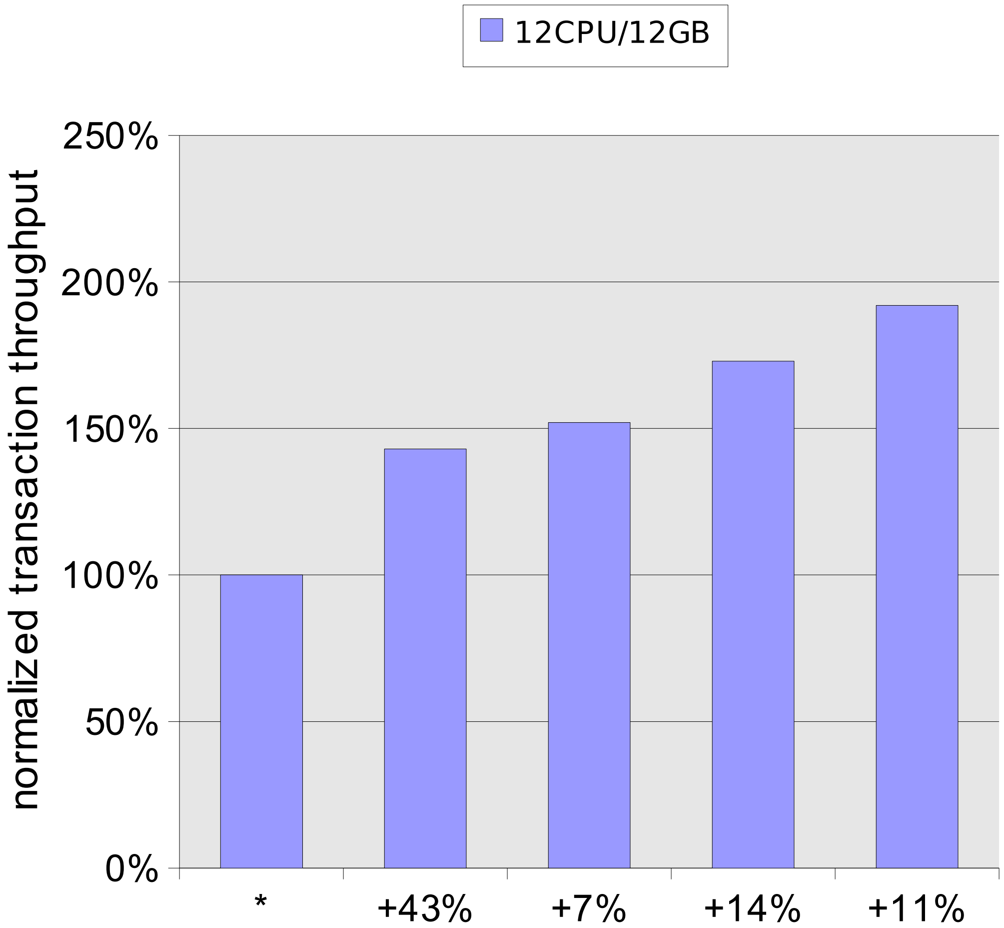
- the amount of accessed data was kept constant
- the amount of loaded data was increased by factor 6x (!)
- this test emulates a growing database like it happens in real life
- Finally... the larger database does not show any performance degradations!

TEST – Very large Linux guest



- very large Linux guests run under z/VM 5.2 or higher without any special treatment
- hence use for database workloads at least z/VM 5.2 or higher

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.

Summary (1)

- avoid physical disk I/O
  - take care on the right buffer pool sizes
  - monitor the cache hit ratio
  - avoid polluting the file system cache with unnecessary data
  - high hit scenarios scale well with the number of available CPUs on IBM system z

Summary (2)

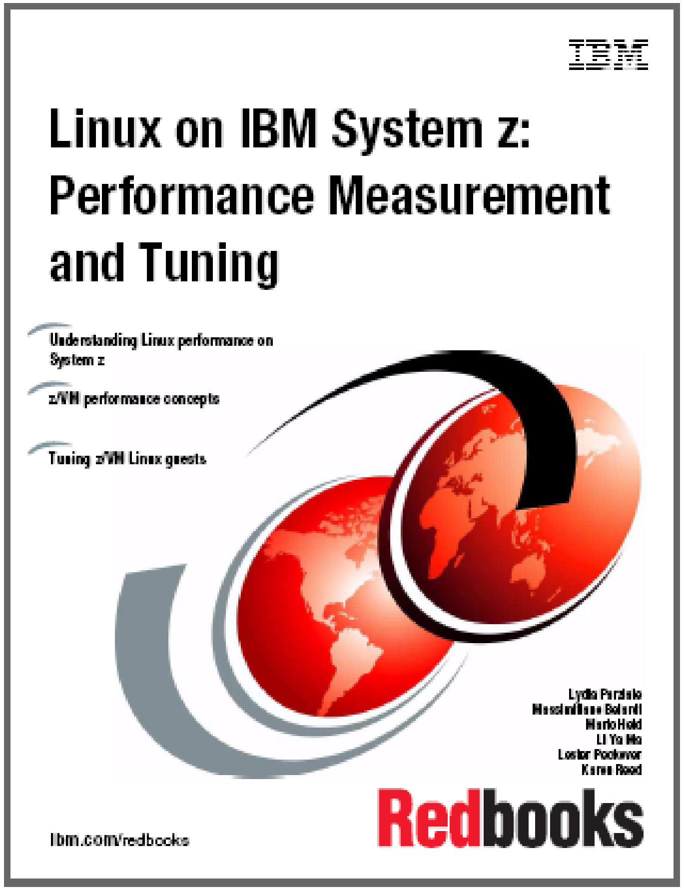
- If you can't avoid a lot of disk I/O, make it fast...
  - Storage server:
    - use disks out of all ranks
    - alternate between the device adapter pairs and servers on the storage server
  - Linux:
    - disable readahead for OLTP workloads
    - ensure that a suitable I/O scheduler is used (e.g. deadline scheduler)
    - take care on the right kernel parameter settings (shared memory, semaphores, message queues)
  - z/VM:
    - use version 5.2 or higher
  - Database:
    - monitor buffer pool usage
    - use striped Logical Volumes or Container like structures to access disks in parallel
    - use separate disks for Data and Log files
    - async and direct I/O save memory and improve database performance
    - if any instance is doing read ahead, this should be done by the database

Summary (3)

- Overall
  - very large database servers are well supported under Linux for System z
  - Database size: there are no limitations

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