IBM Linux Technology Center

Performance Experience with Databases on Linux for IBM System z

Eberhard Pasch
epasch@de.ibm.com

Session 9292

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
- Workload
- Disk Setup
- Linux Setup
- Database Setup
- Performance Results
  - Scalability
  - let the database grow
  - z/VM
  - a tuning story

Objectives

- The goal of our work is to get and publish information like
  - How databases on Linux on System z scale
  - How to improve the disk I/O performance
  - What needs to be done in Linux to get best performance
- We did no high end benchmarking!
- Most results are not limited to one database product. Tests have been made with
  - DB2 8.1, 8.2 and 9
  - Informix 9.4.0 and 11
  - Oracle 9i and 10g

Performance tuning should be done at all layers

- "Optimize your stack from the top to the bottom"
  - Application design
  - Application implementation
  - ***Database***
  - ***Operating system***
  - ***Virtualization system***
  - ***Hardware***

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

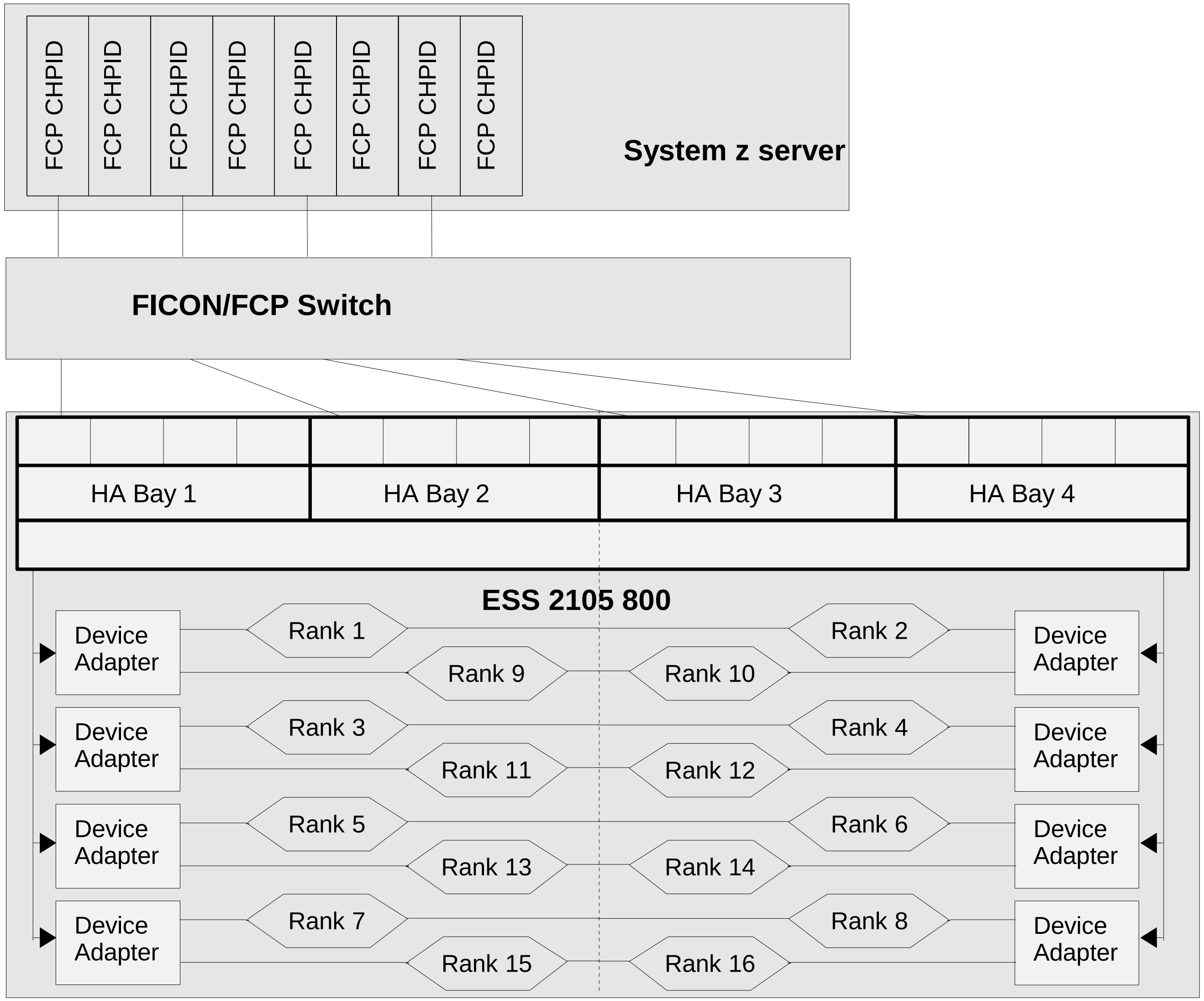
Characteristics of the workload

- very I/O intense
  - The disk utilization is typically at 80% and 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 rate
  - 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
- very cache unfriendly
  - small packets size (typically 4 or 8 KB) and randomly distributed over the disk space
  - relief: large caches,
    avoid cache pollution with unnecessary data,
    use as many physical disks as possible to reduce the impact of disk latencies

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

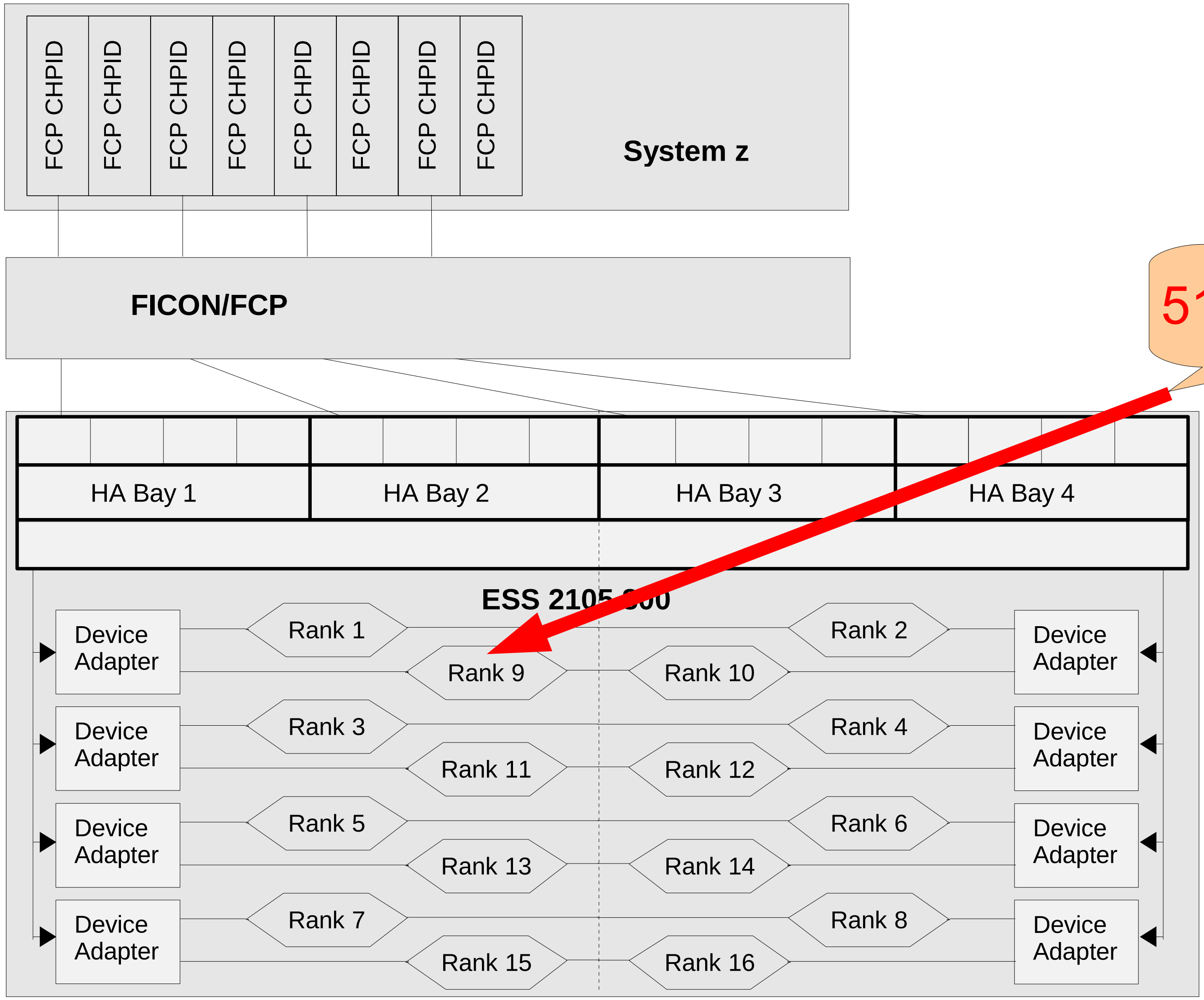
- Don't treat a storage server as a black box, understand its structure
- You ask for 16 disks and your system administrator gives you addresses 5100-510F
- From a performance perspective this is close to the worst case
- So - what's wrong with that?

ESS Architecture



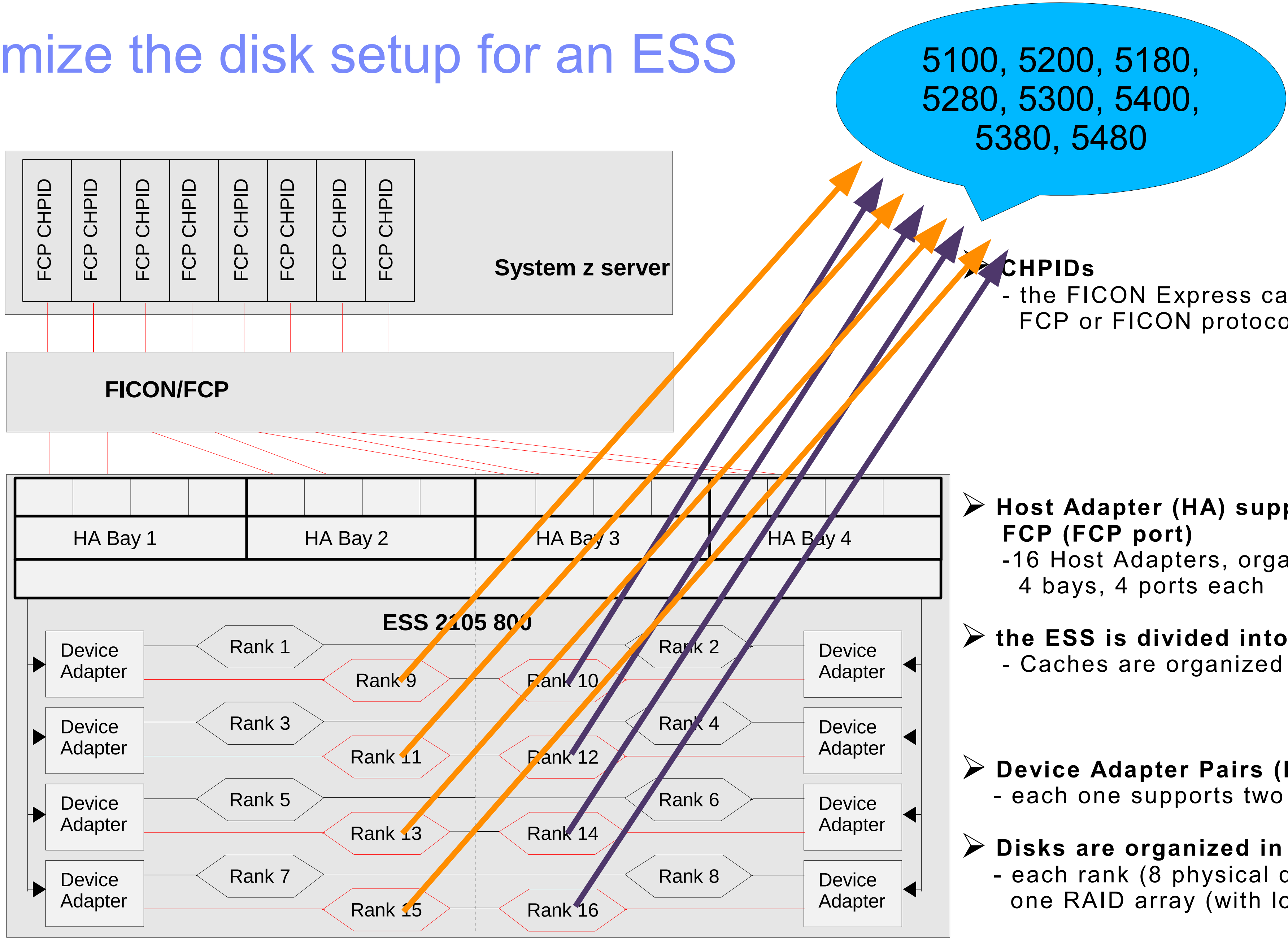
- **CHPIDs**
  - the FICON Express card supports FCP or FICON protocol
- **Host Adapter (HA) supporting FCP (FCP port)**
  - -16 Host Adapters, organized in 4 bays, 4 ports each
- **the ESS is divided into two Clusters**
  - Caches are organized per cluster!
- **Device Adapter Pairs (DA)**
  - each one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

ESS Architecture



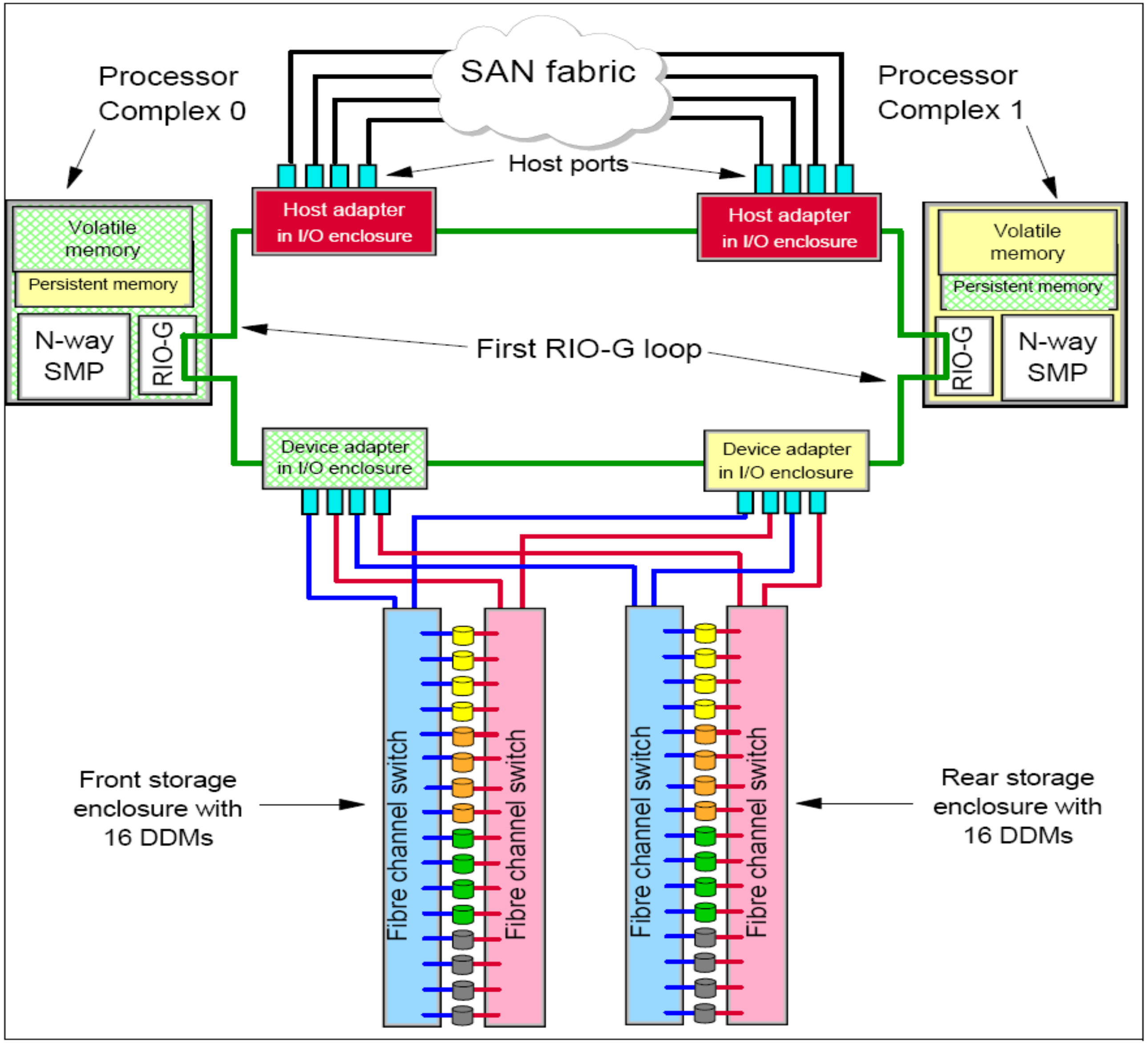
- **CHPIDs**
  - the FICON Express card supports FCP or FICON protocol
- **Host Adapter (HA) supporting FCP (FCP port)**
  - 16 Host Adapters, organized in 4 bays, 4 ports each
- **the ESS is divided into two Clusters**
  - Caches are organized per cluster!
- **Device Adapter Pairs (DA)**
  - each one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

Optimize the disk setup for an ESS



- **CHPIDs**
  - the FICON Express card supports FCP or FICON protocol
- **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 Architecture

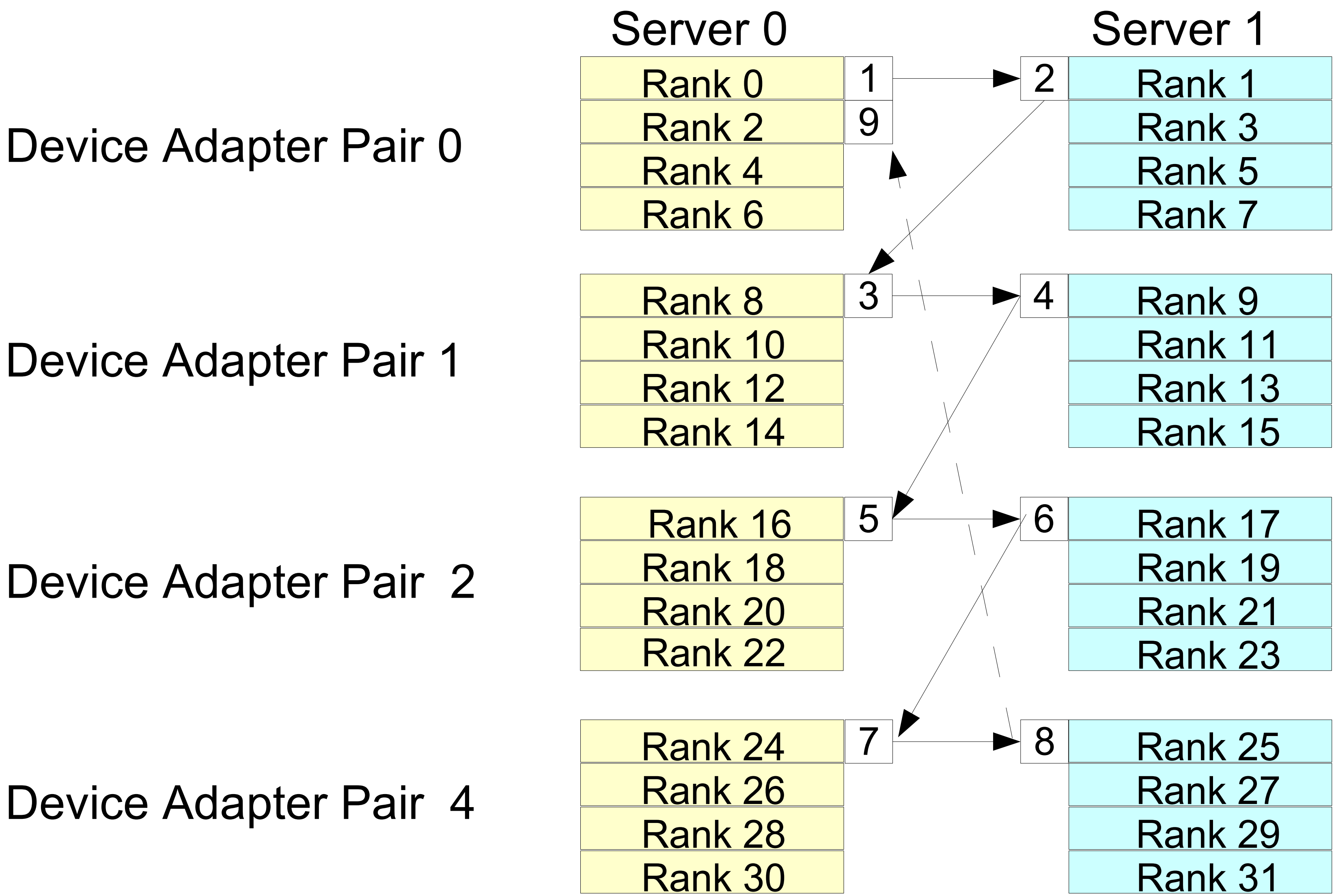


- **structure** is much more complex
  - disks are now connected via two internal FCP switches for higher bandwidth
- the DS8000 is still divided into two parts named **processor complex** or just **server**
  - caches are organized per server
- one **device adapter pair** addresses now 4 array sites
- one **array site** is build 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 9, etc.

Make the disks available for the database

- Use a striped logical volume
  - add the volumes in the right order to the volume group
  - we recommend a stripe size of 32KB for database workloads
- for DB2: use containers
  - ```
    CREATE TABLESPACE TSTEST IN DATABASE PARTITION GROUP
    IBMDEFAULTGROUP PAGESIZE 4096 MANAGED BY DATABASE
      USING (FILE '/TSTEST_cont0/file' 1000,
             FILE '/TSTEST_cont1/file' 1000,
             FILE '/TSTEST_cont2/file' 1000,
             ...
             FILE '/TSTEST_cont15/file' 1000)
      ...
    ```
  - Select the disks in the right order from the ranks
  - the database will distribute the data over the containers automatically
    ```

Read ahead setup – avoid unnecessary I/Os

- Database
  - Recommendation is to disable it. But this is application dependent. However, the database is the only instance which can do meaningful read aheads.
    - in Informix the onconfig parameters `RA_PAGES` and `RA_THRESHOLD` to `0`
    - in DB2 set tablespace parameter `PREFETCHSIZE` to `0`
    - in Oracle set the oracle profile parameter `DB_FILE_MULTIBLOCK_READ_COUNT` to `0`
- LVM
  - Disable it by setting the read ahead to 0 pages with the command
    - `lvchange -r 0 /dev/<volume group>/<logical volume>`
- Linux block device layer
  - Set the value to 0 using the blockdev command,
    - for example: `blockdev --setra 0 /dev/sda`

Availability list

- Certified combinations of database and distribution for Linux on System z

64 bit product	*31 bit product*	*SLES8*	*SLES9*	*SLES10*	*RHEL3*	*RHEL4*	*RHEL5*
	DB2 8.1	X			X		
DB2 8.2		X	X	X	X	X	
DB2 9			X	X		X	X
Informix IDS 9.4.0		X	X		X		
Informix IDS 10		X	X		X	X	
Informix IDS 11				X		X	
	Oracle 9i	X					
Oracle 10g R1		X	X			X	
Oracle 10g R2			X	X		X	

Kernel parameters (1)

- 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`

Kernel parameters (2)

- 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 used for the buffer pools,
    this parameter must be adapted to your specific database configuration

- Strategy:
  - Make shmall and shmmax so large that it is not a limit, e.g. full memory size

Linux memory	shmall	shmmni	shmmax
8 GB	1971200	4096	8074035200

  - Start with a total buffer pool size of 60%
  - Increase buffer pool size and monitor free memory and swapping activity
  - There should be no ongoing swap activity
  - It is recommended to leave at least 5% free memory (free command)

Kernel parameters (3)

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

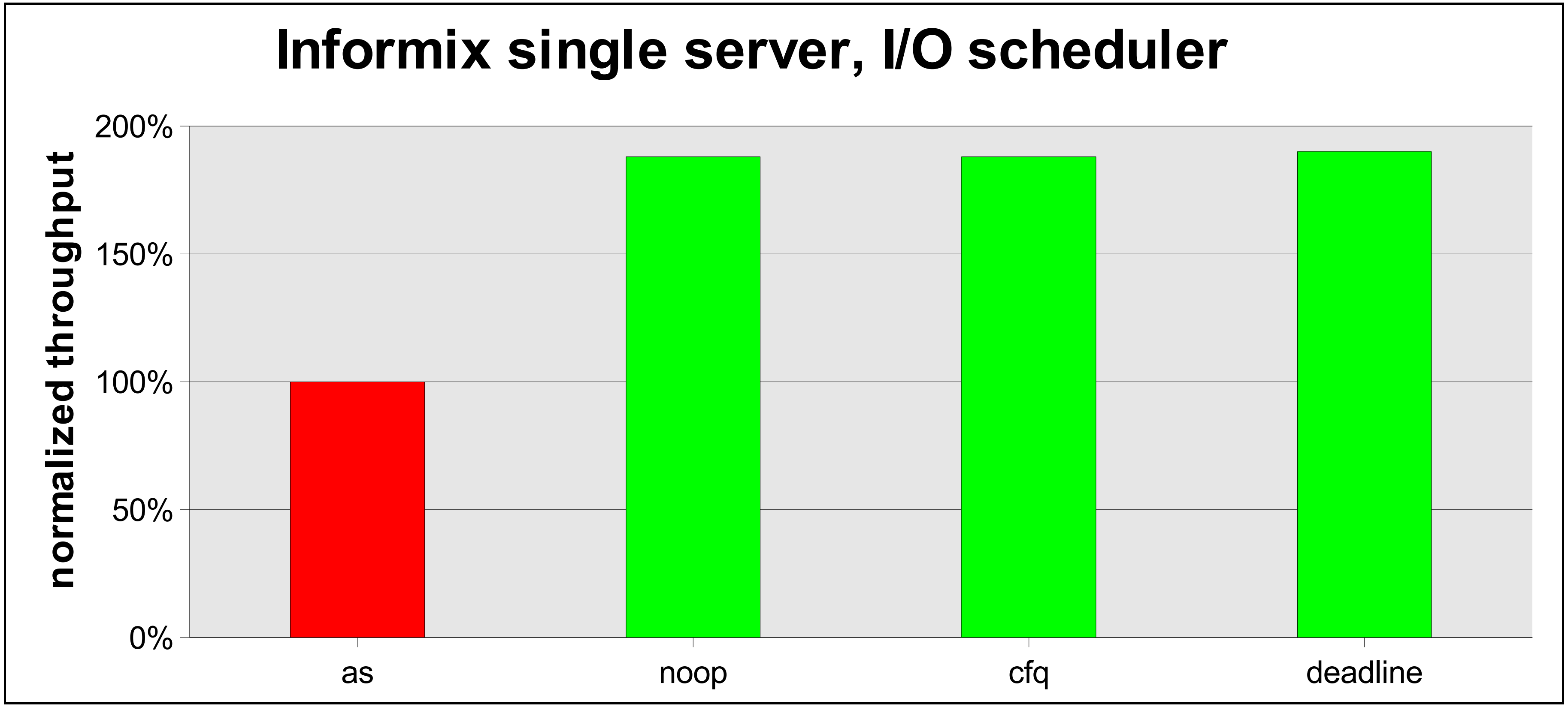
- Kernel semaphores limits
  - kernel.sem:
    Max. semaphores per array / max. Semaphores system wide / max. ops per per semop call / max. number of arrays
- Kernel message limits
  - kernel.msgmni: Maximum queues system wide
  - kernel.msgmax: Maximum size of message (bytes)
  - kernel.msgmnb: Default size of queue (bytes)

Linux 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 2.6 I/O Schedulers - Results

- as scheduler is not a good choice for this environment
- all other schedulers show similar results as the kernel 2.4 scheduling
- Deadline scheduler is used for further tests

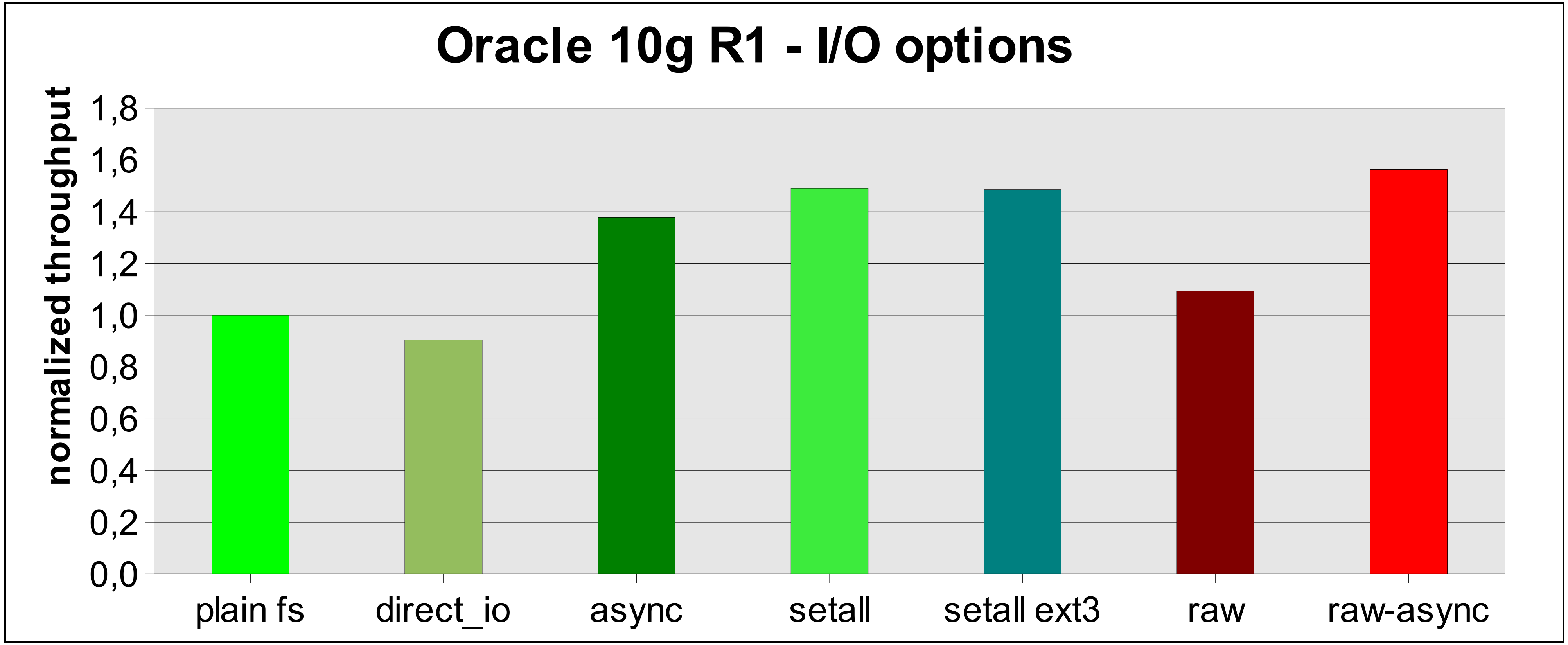


new Disk I/O Options with Linux kernel 2.6

- Direct I/O (DIO)
  - transfer the data directly from the application buffers to the device driver, avoids copying the data to the page cache
  - advantage
    - saves page cache memory and avoids caching the same data twice
    - enables larger buffer pools
  - disadvantage:
    - make sure that no utility is working through the file system (page cache) --> danger of data corruption
- Asynchronous I/O (AIO)
  - The application is not blocked for the time of the I/O operation
  - It resumes its processing and gets notified when the I/O is completed.
  - advantage
    - the issuer of a read/write operation is no longer waiting until the request finishes.
    - reduces the number of I/O processes (saves memory and CPU)
- We recommend to use both features

DIO and AIO – Results

- The combination of direct I/O and async I/O (setall) shows best results when using the Linux file system.
- Best throughput however was seen with raw I/O and async I/O.
- ext2 and ext3 lead to identical throughput



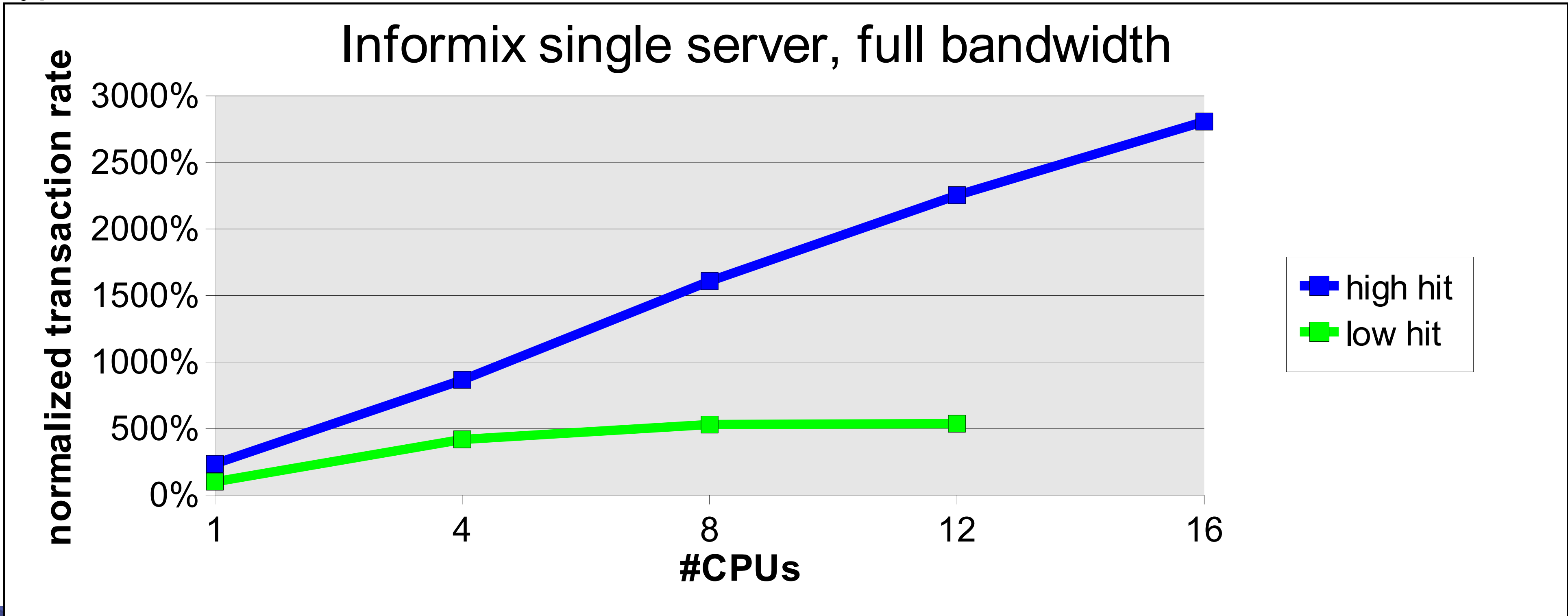
What to do with log files?

- I/O pattern:
  - data access is random I/O, read and write
  - writing a log is sequential write I/O
- When the database log files and the data files are on the same disks (LUN, ECKD device number)
  - the sequential characteristics of the log I/O gets lost
  - the I/O schedulers prefer read request!
  - degrades the transfer rate
  - degrades the priority of writing the logs
  - slows down the transaction rate
- Separate log and data devices, in the best case take
  - other ranks on the same storage server or
  - another storage server

  to guarantee a contiguous flow of the log data at the maximum throughput rate

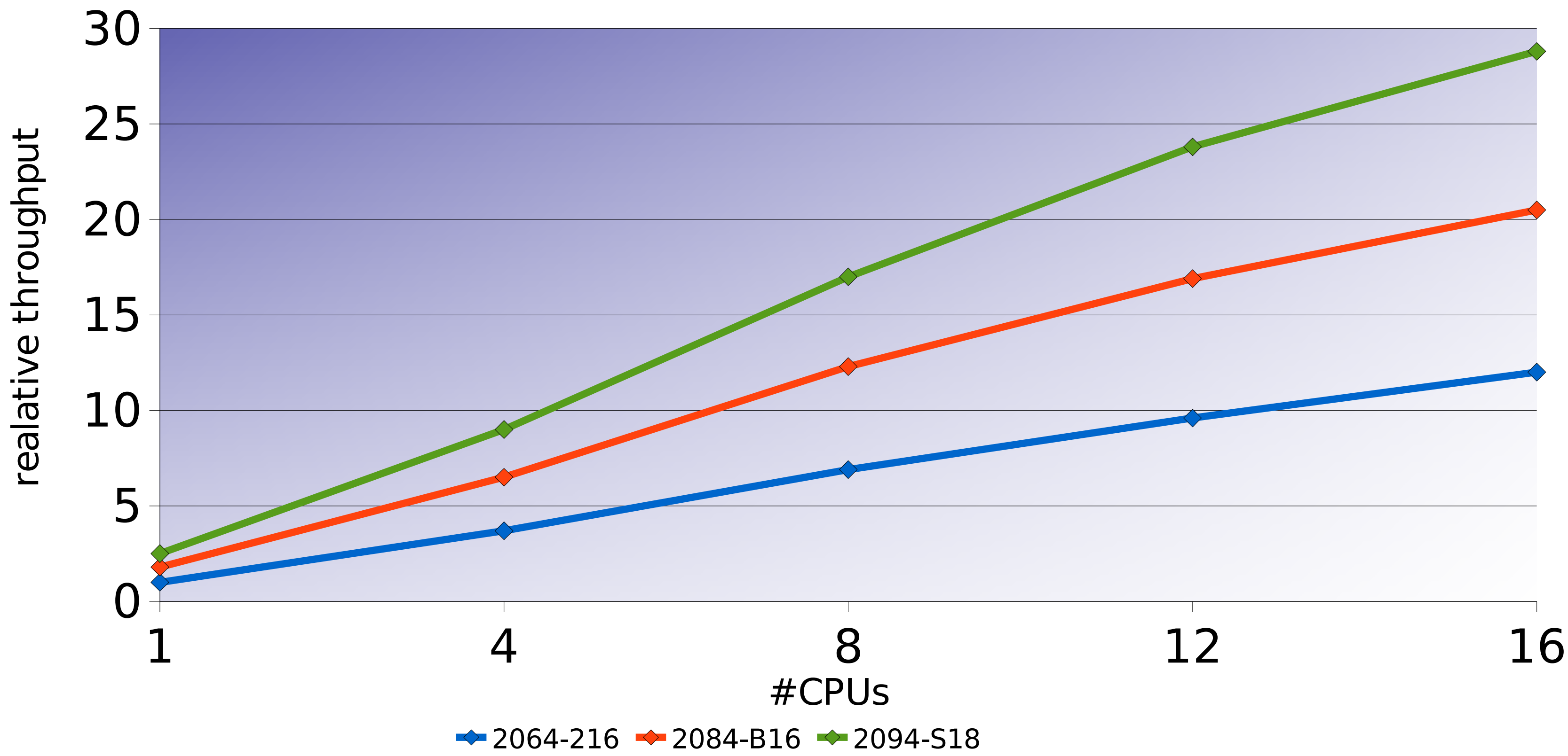
CPU Scalability and the cache hit ratio

- The high cache hit ratio case is a successful implementation for **avoid the I/O**
  - Very good throughput scaling from 1 to 16 CPUs, as long as the workload runs in the buffer pools
- the high and low hit scenarios span the full possible bandwidth, where the high hit scenario marks the upper end and the low hit scenario is the lower end.
- Typical workloads are between the two lines.



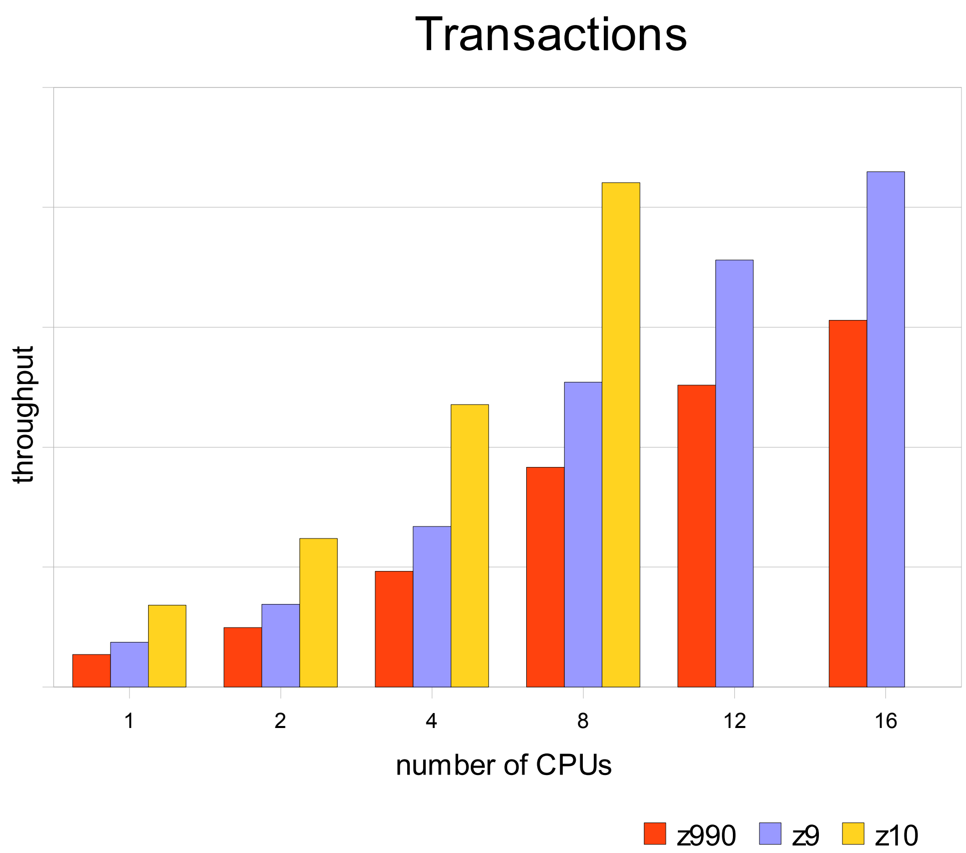
Informix IDS 11 with SLES10 on z990 and z9

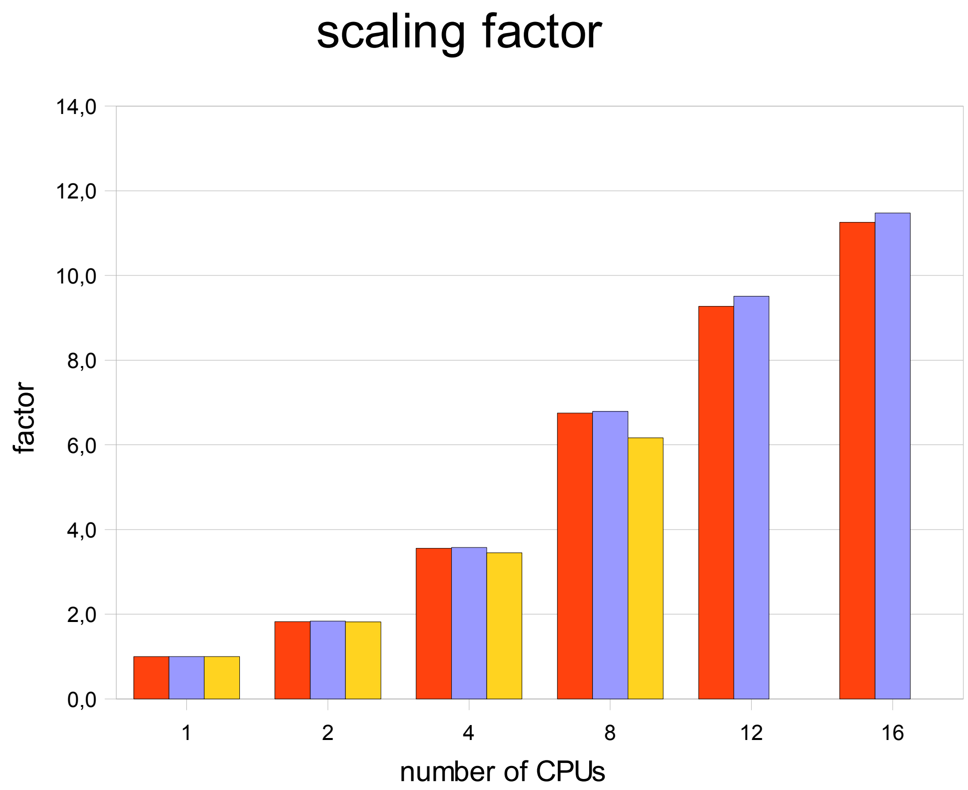
- Relatively constant increase for each measurement point
- 16 z900T CPUs ≙ 8 z990 CPUs ≙ 6 z9 CPUs
- Test with 16 CPUs utilized 15.3 CPUs



z10 with Informix IDS 11 OLTP workload

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

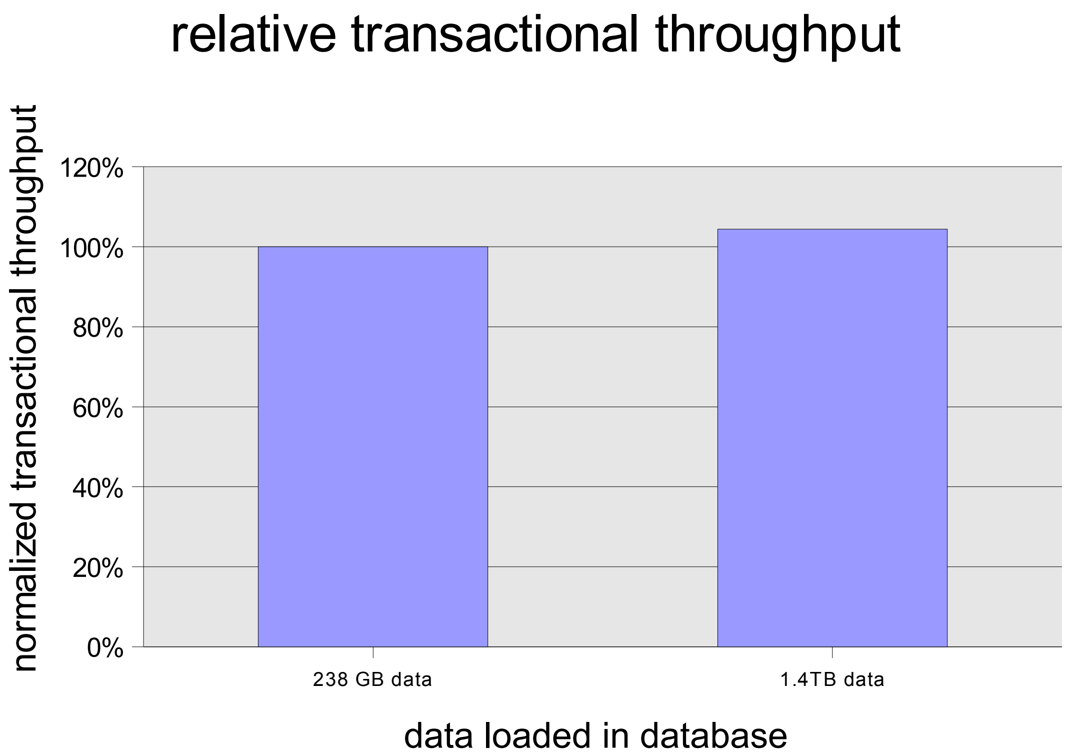




Summary Informix: Relative Performance Increase

- Relatively constant increase for each measurement point
- Average weighted increase of 74% between zSeries 900T and zSeries 990
  - ~ 54% HW related
  - ~ 20% SW related (IDS11.10.FC1 and SLES10SP1)
    - eg. non-blocking check points
- Average weighted increase of 40% between zSeries 990 and System z9
  - 40% HW related
- Improvements between System z9 and System z10 have a greater variance – 65% - 82%

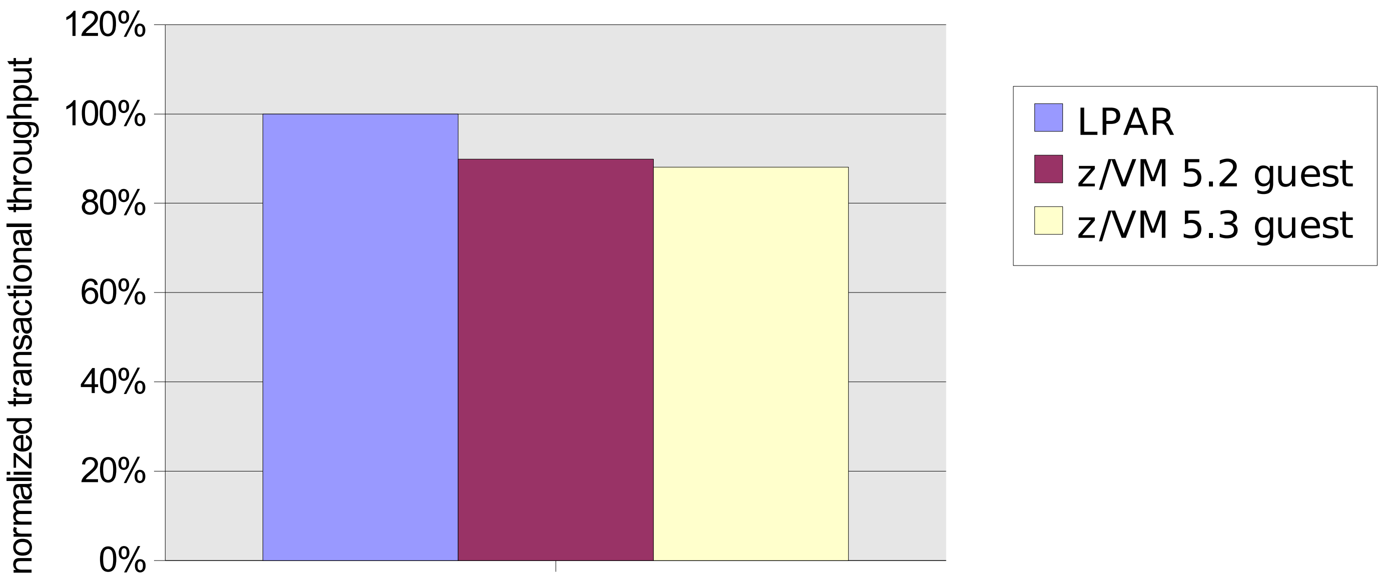
DB2 9 - Let the database grow



- the amount of accessed data was kept constant, and the amount of data loaded was increased by factor 6x
- This emulates a growing database under constant business load
- ... and the large database performs as on the first day

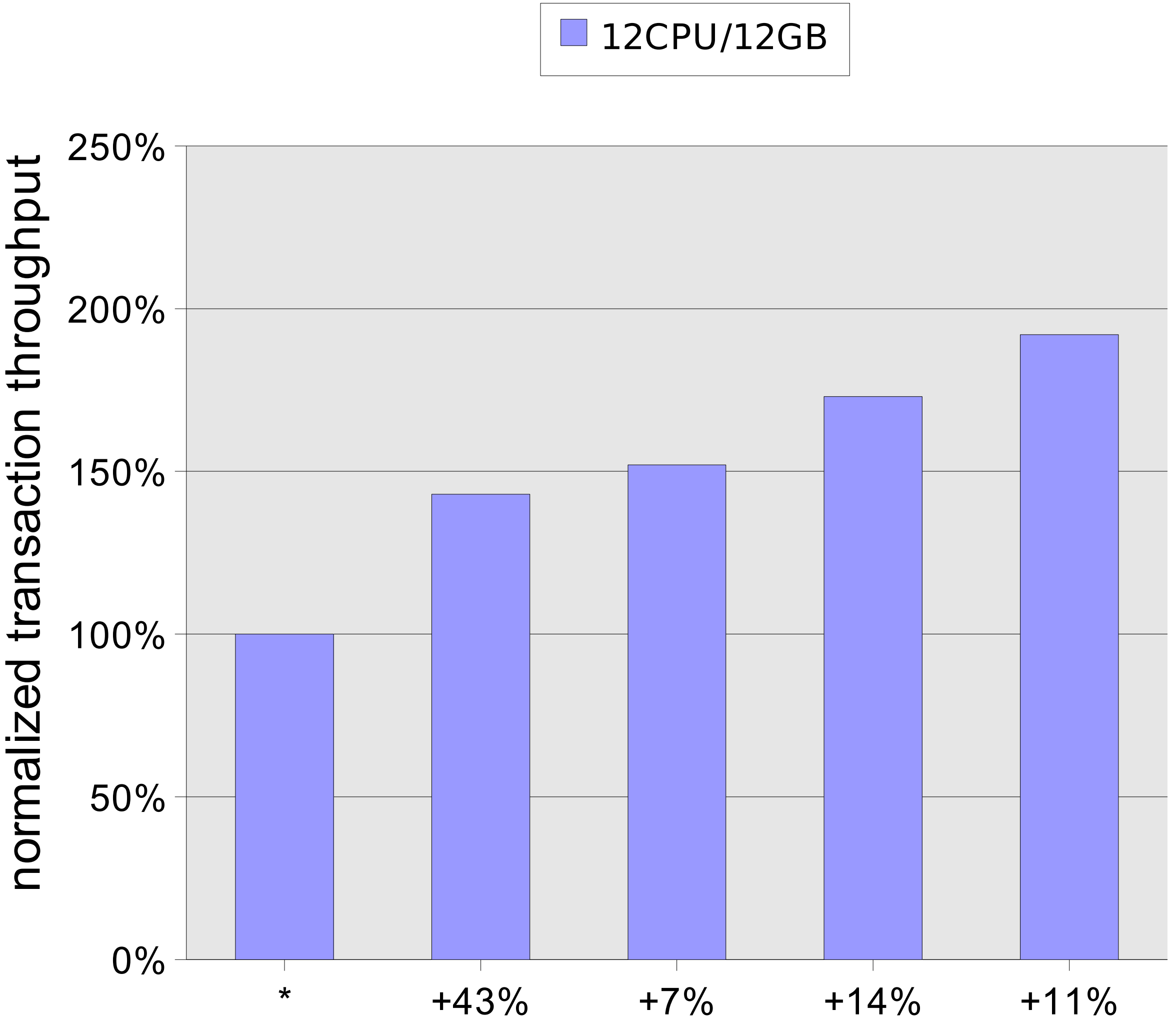
Large Linux guest

Oracle 10g R2 guest under z/VM with 40 GB Linux memory



- Large guests with 40GB memory run under z/VM version 5.2 and higher without any special treatment
- For database workloads use z/VM version 5.2 and higher

DB2 8.2 – A Tuning Story



* starting point

+43%

tablespace prefetch 0
LVM readahead 0

+7%

CHNGPGS_THRESH from 30 to 60

+14%
extra bufferpools (data and index) for tablespace with very large rows

+11%
pagesize 8K for index from the tablespace with very large rows

Finally,
we nearly doubled the throughput compared with the starting point

Optimizing c and c++ code

- Use the highest possible optimization level
  - **"-O" is not enough!**
- Consider other general available options for best performance
- Database products and applications should be compiled with the System z specific architecture setting
  - -mtune=z990 (gcc 3.3 and higher, SLES9 and RHEL4)
  - -mtune=z9-109 (gcc 4.0 and higher, SLES10 and RHE5)
  - if these options are not used, the z990 and z9 processing features are not fully used.
- If source code is available, use "-march" instead of" -mtune"
  - mtune sorts the instructions for best parallel execution by the 2 execution units per CPU
  - march includes mtune and in addition makes use of all machine instructions for the specified machine type

Summary (1)

- Avoid the physical I/O
  - take care on the right buffer pool sizes
  - monitor the cache hit ratio
  - avoid polluting the cache with unnecessary data
  - as long as the data are in the buffer pools the workload scales very well when increasing CPUs on IBM System z

Summary (2)

- If the I/O can't be avoided, make it fast
  - Storage server:
    - Always use disks from many ranks and both clusters/servers
  - Linux
    - Disable read aheads
    - ensure that a suitable I/O scheduler is used (no as-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 stripe the data over the disks
    - use separate disk devices for data files and log files
    - async and direct I/O saves memory and improves database performance
    - take care that all required indexes are available
    - if any instance is doing read ahead, this should be the database

Summary (3)

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

Visit us !

- Linux on zSeries Tuning Hints and Tips
  - http://www.ibm.com/developerworks/linux/linux390/perf/
- Linux-VM Performance Website:
  - http://www.vm.ibm.com/perf/tips/linuxper.html

Questions