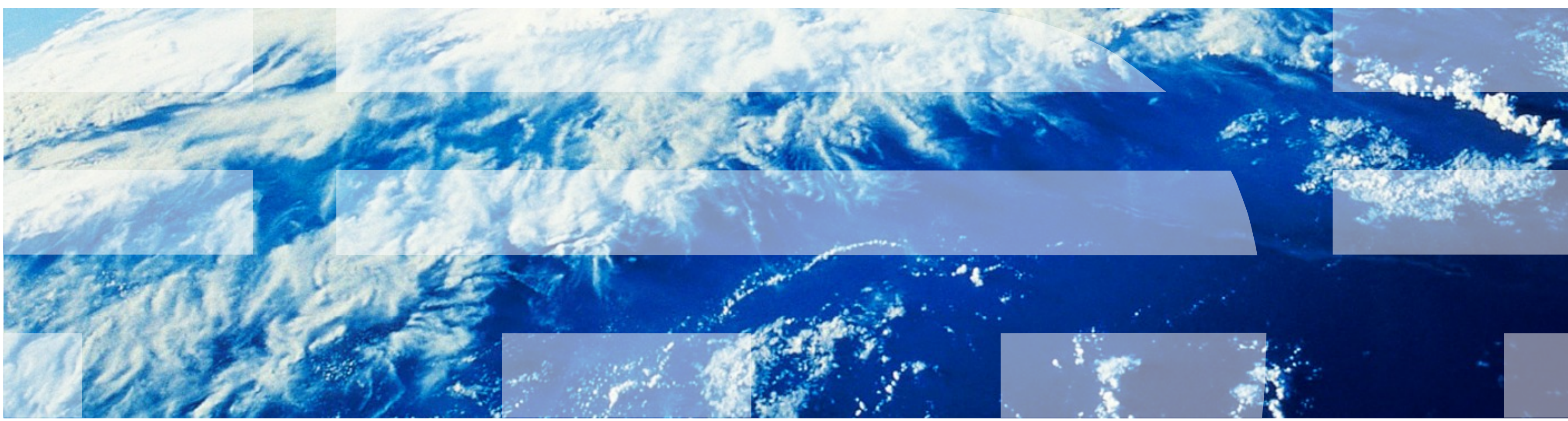
Mario Held mario.held@de.ibm.com
Martin Kammerer martin.kammerer@de.ibm.com

07/07/2010

Linux on System z disk I/O performance



visit us at http://www.ibm.com/developerworks/linux/linux390/perf/index.html

Agenda

- Linux
  - file system
  - special functions
  - logical volumes
  - monitoring
  - High performance FICON
- Storage server
  - internal structure
  - recommended usage and configuration
- Host
  - Linux I/O stack
  - sample I/O attachment
  - FICON/ECKD and FCP/SCSI I/O setups
  - possible bottlenecks
  - recommendations

Linux file system

- Use ext3 instead of reiserfs
- Tune your ext3 file system
  – Select the appropriate journaling mode (journal, ordered, writeback)
  – Consider to turn off atime
  – Consider upcoming ext4
- Temporary files
  – Don't place them on journaling file systems
  – Consider a ram disk instead of a disk device
  – Consider tmpfs for temporary files
- If possible, use direct I/O to avoid double buffering of files in memory
- If possible, use async I/O to continue program execution while data is fetched
  – Important for read operations
  – Applications usually are not depending on write completions

I/O options

- Direct I/O (DIO)
  - Transfer the data directly from the application buffers to the device driver, avoids copying the data to the page cache
  - Advantages:
    - Saves page cache memory and avoids caching the same data twice
    - Allows larger buffer pools for databases
  - Disadvantage:
    - Make sure that no utility is working through the file system (page cache) --> danger of data corruption
    - The size of the I/O requests may be smaller
- 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)
- Recommendation is to use both
  - Database benchmark workloads improved throughput by 50%

I/O schedulers (1)

- Four different I/O schedulers are available
  - noop scheduler
    only request merging
  - deadline scheduler
    avoids read request starvation
  - 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.
- The default in current distributions is deadline
  - Don't change the setting to as in Linux on System z. The throughput will be impacted significantly

HowTo

- How to identify which I/O scheduler is used
  - Searching (grep) in file /var/log/boot.msg for phrase 'io scheduler' will find a line like:
    - <4>Using deadline io scheduler
  - which returns the name of the scheduler in use
- How to select the scheduler
  - The I/O scheduler is selected with the boot parameter elevator in zipl.conf
    - **[ipl]**
    - **target = /boot/zipl**
    - **image = /boot/image**
    - **ramdisk = /boot/initrd**
    - **parameters = "maxcpus=8 dasd=5000 root=/dev/dasda1 elevator=deadline”**
    - **where elevator ::= as | deadline | cfq | noop**
- For more details see /usr/src/linux/Documentation/kernel-parameters.txt.

Logical volumes

- Linear logical volumes allow an easy extension of the file system
- Striped logical volumes
  - Provide the capability to perform simultaneous I/O on different stripes
  - Allow load balancing
  - Are also extendable
- Don't use logical volumes for “/” or “/usr”
  - If the logical volume gets corrupted your system is gone
- Logical volumes require more CPU cycles than physical disks
  - Consumption increases with the number of physical disks used for the logical volume

Monitoring disk I/O

Merged requests
I/Os per second
Throughput
Request size
Queue length
Service times
utilization

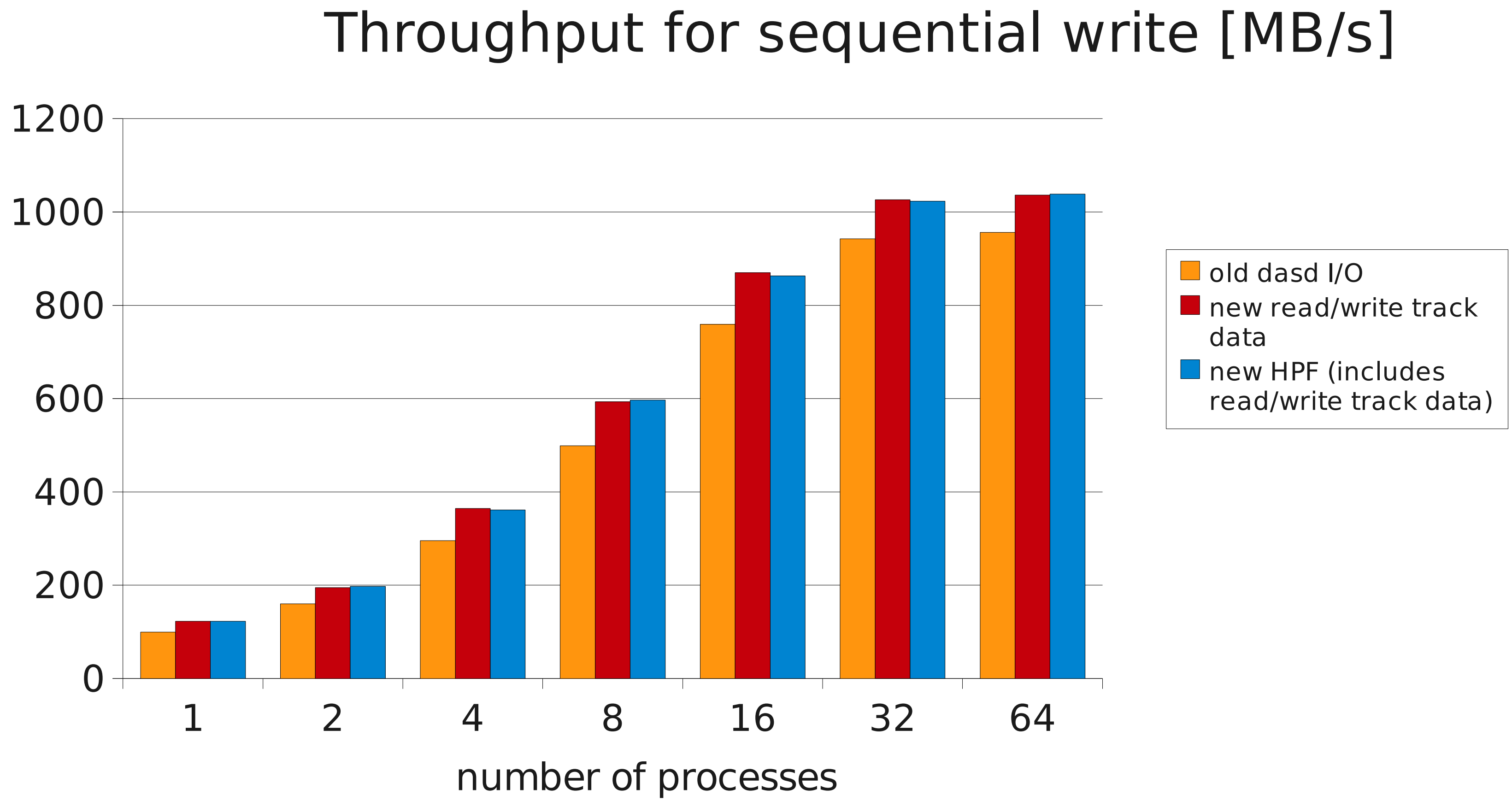
```
Output from iostat -dmx 2 /dev/sda

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await  svctm  %util

sda               0.00 10800.50    6.00 1370.00     0.02    46.70    69.54     7.03    5.11   0.41  56.50
sequential write
---------------------------------------------------------------------------------------------------------

sda               0.00 19089.50   21.50  495.00     0.08    76.59   304.02     6.35   12.31   1.85  95.50
sequential write
---------------------------------------------------------------------------------------------------------

sda               0.00 15610.00  259.50 1805.50     1.01    68.14    68.58    26.07   12.23   0.48 100.00
random write
---------------------------------------------------------------------------------------------------------

sda            1239.50     0.00  538.00    0.00   115.58     0.00   439.99     4.96    9.23   1.77  95.00
sequential read
---------------------------------------------------------------------------------------------------------

sda             227.00     0.00 2452.00    0.00    94.26     0.00    78.73     1.76    0.72   0.32  78.50
random read
```

High Performance FICON

- Advantages:
  - Provides a simpler link protocol than FICON
  - Is capable of sending multiple channel commands to the control unit in a single entity
- Performance expectation:
  - Should improve performance of database OLTP workloads
- Prerequisites:
  - z10 GA2 or newer
  - FICON Express2 or newer
  - DS8000 R4.1 or newer
  - DS8000 High Performance FICON feature
  - A Linux dasd driver that supports read/write track data and HPF
    - Included in SLES11 and future Linux distributions, service packs or updates)

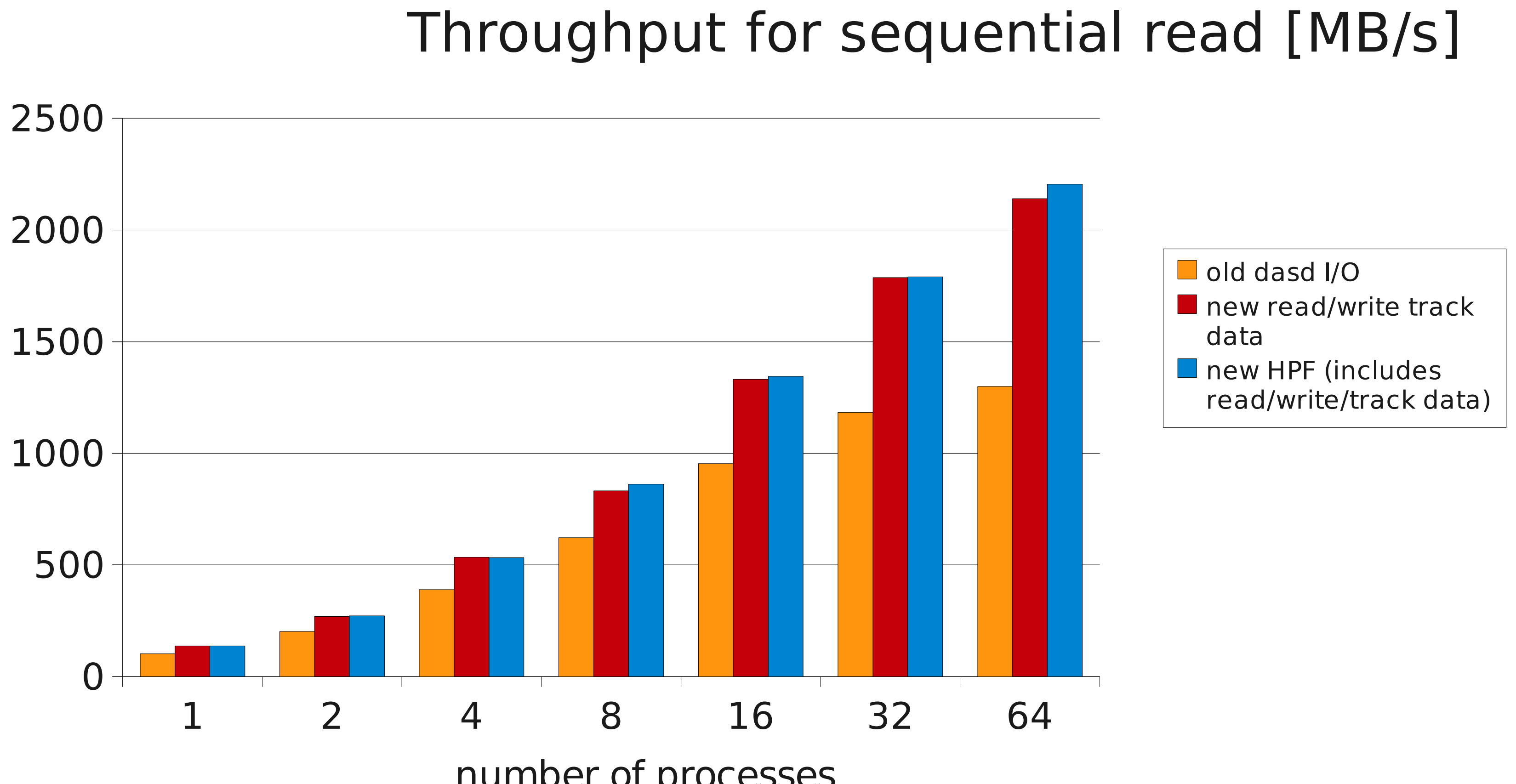
High Performance FICON results (1)

- For sequential write throughput improves with read/write track data up to 1.2x



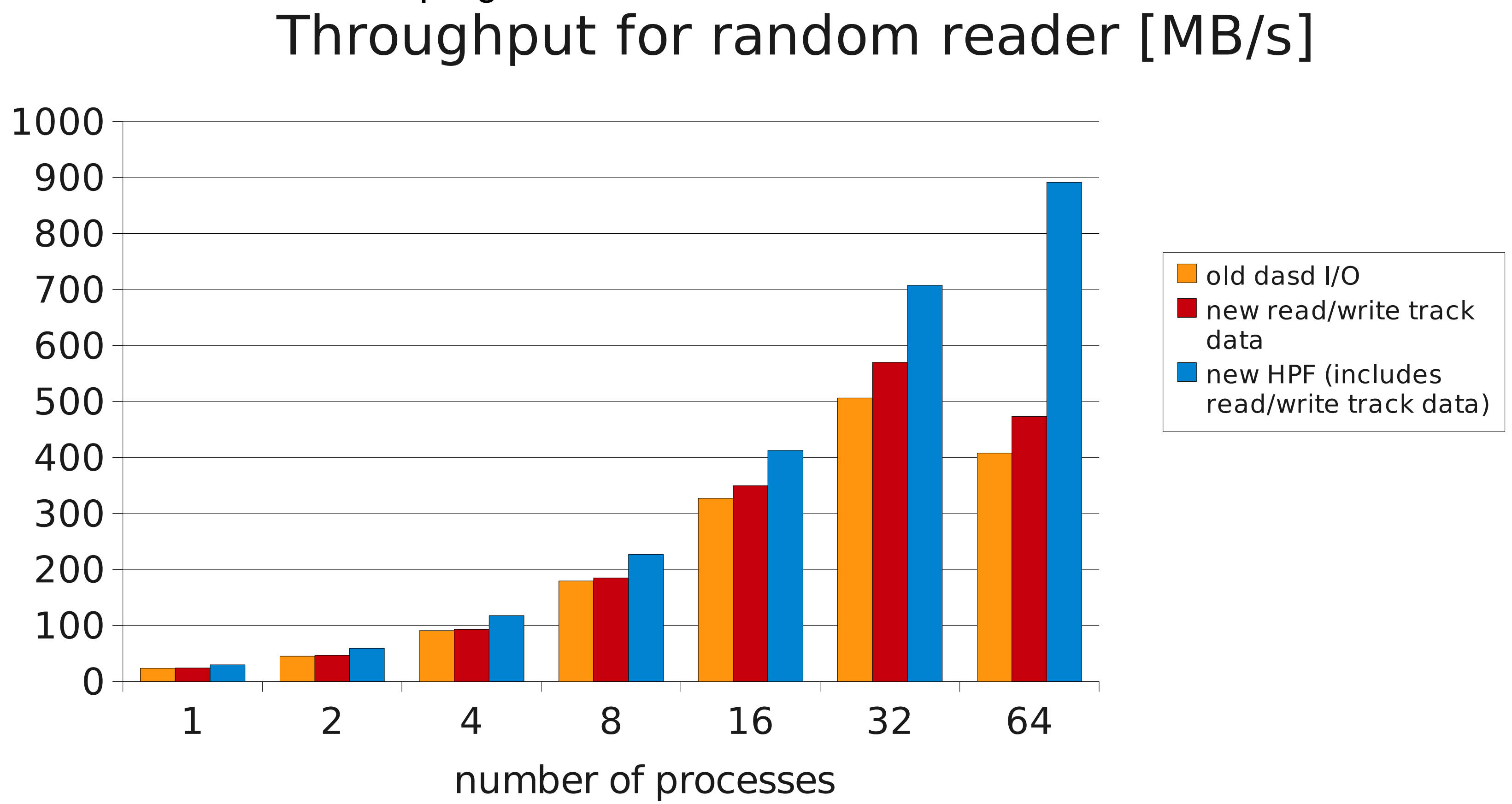
High Performance FICON results (2)

- For sequential read throughput improves with read/write track data up to 1.6x



High Performance FICON results (3)

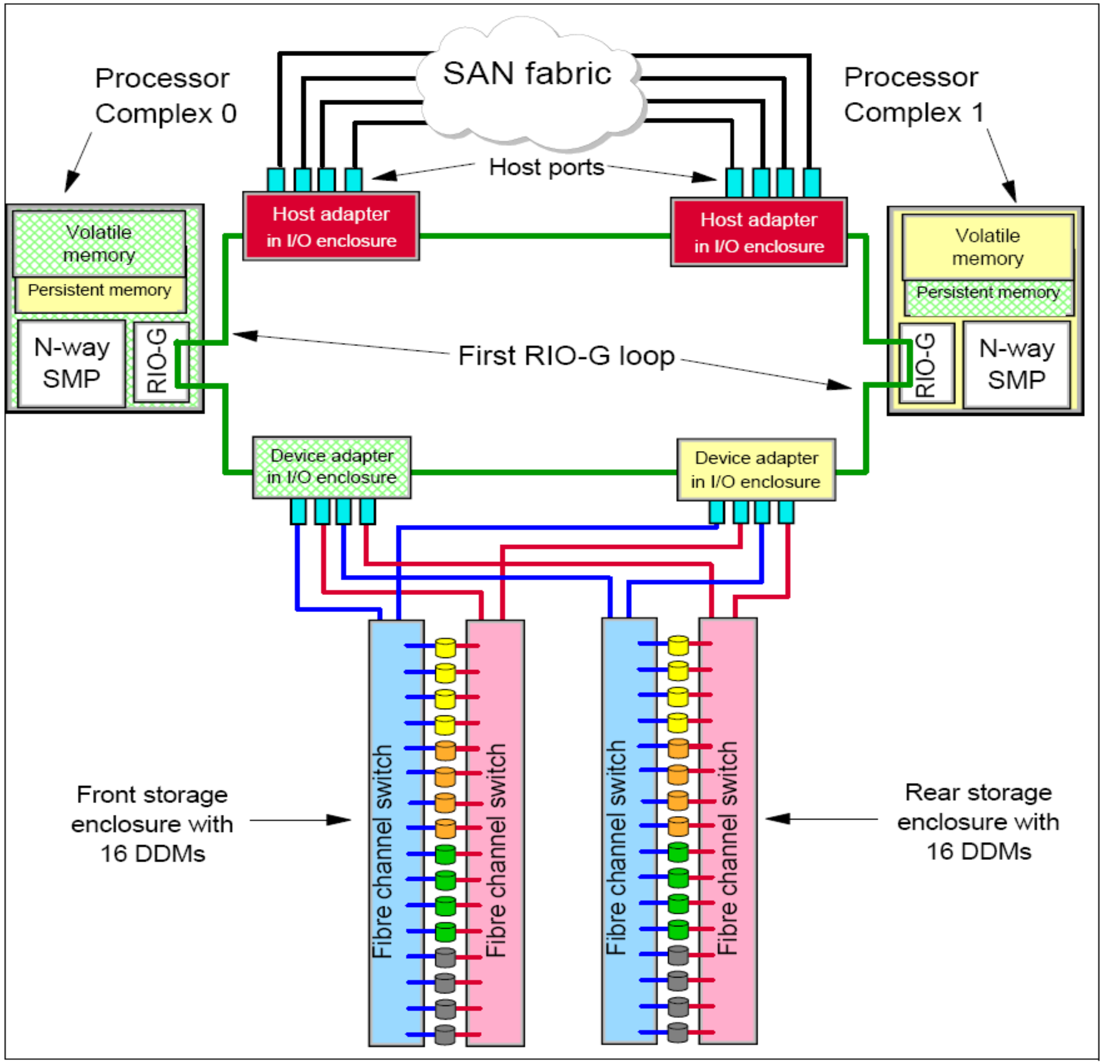
- With small record sizes, as e.g. used by databases, HPF throughput improves up to 2.2x versus old Linux channel programs



DS8000 Disk setup

- Don't treat a storage server as a black box, understand its structure
- Principles apply to other storage vendor products as well
- Several conveniently selected disks instead of one single disk can speed up the sequential read/write performance to more than a triple. Use the logical volume manager to set up the disks.
- Avoid using subsequent disk addresses in a storage server (e.g. the addresses 5100, 5101, 5102,.. in an IBM Storage Server), because
  - they use the same rank
  - they use the same device adapter.
- If 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

DS8000 Architecture



- **structure** is complex
  - disks are 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 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 as for ESS**
  - one disk range resides in one extent pool

Rules for selecting disks

- This makes it fast
  - Use as many paths as possible (CHPID -> host adapter)
  - Spread the host adapters used across all host adapter bays
    - For ECKD switching of the paths is done automatically
    - FCP needs a fixed relation between disk and path
      - Use Linux multipathing for load balancing
  - Select disks from as many ranks as possible!
  - Switch the rank for each new disk in a logical volume
  - Switch the ranks used between servers and device adapters
  - Avoid reusing the same resource (path, server, device adapter, and disk) as long as possible
- Goal is to get a balanced load on all paths and physical disks
- In addition striped Linux logical volumes and / or storage pool striping may help to increase the overall throghput.

Linux kernel components involved in disk I/O

Application program — Issuing reads and writes

Virtual File System (VFS) — VFS dispatches requests to different devices and Translates to sector addressing

Block device layer

- **Logical Volume Manager (LVM)** — LVM defines the physical to logical device relation
- **Multipath** — Multipath sets the multipath policies
- **Device mapper (dm)** — dm holds the generic mapping of all block devices and performs 1:n mapping for logical volumes and/or multipath
- **Page cache** — Page cache contains all file I/O data, direct I/O bypasses the page cache
- **I/O scheduler** — I/O schedulers merge, order and queue requests, start device drivers via (un)plug device

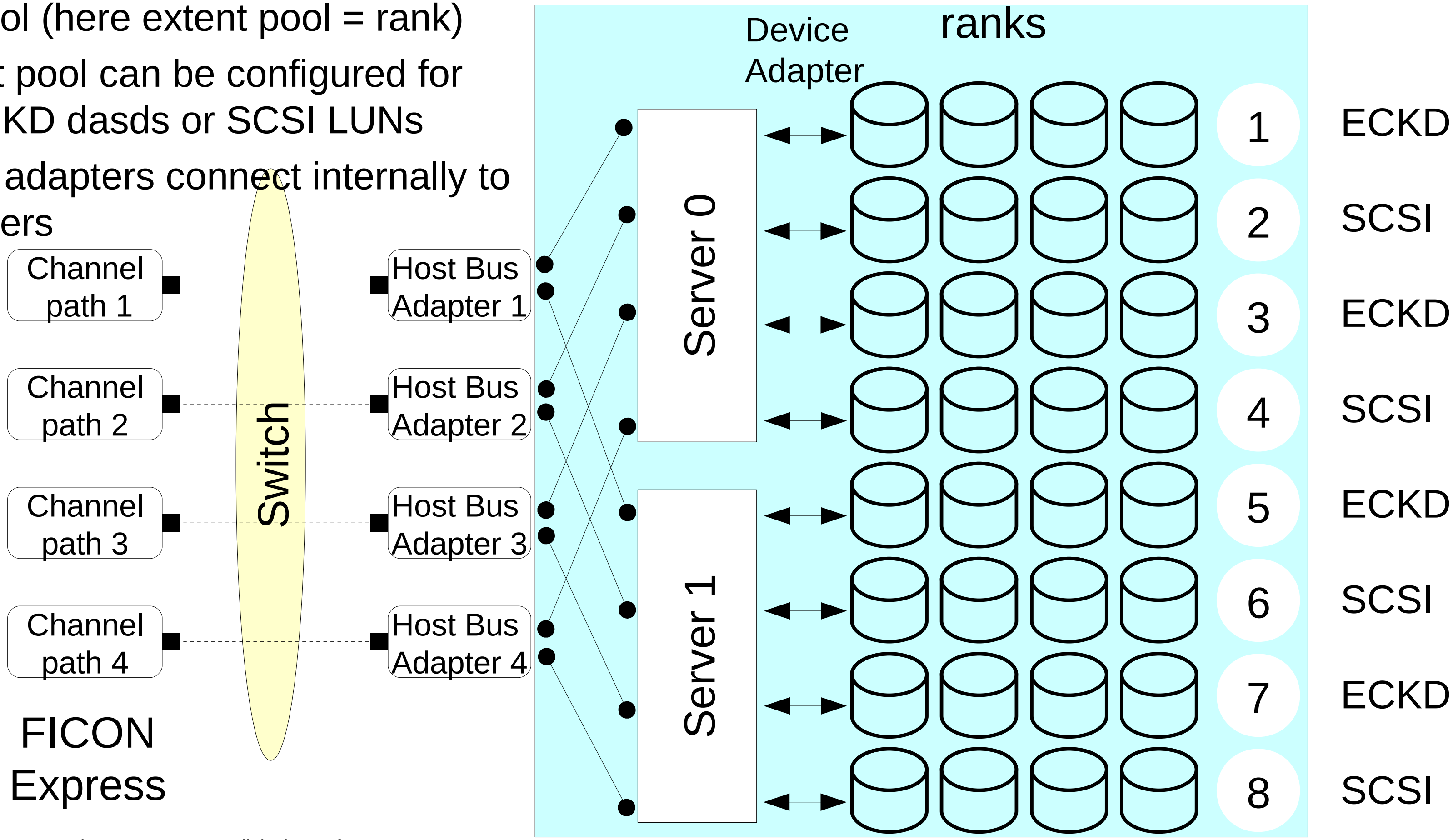
Data transfer handling

Device drivers

- Direct Acces Storage Device driver
- Small Computer System Interface driver (SCSI)
- z Fiber Channel Protocol driver (zFCP)
- Queued Direct I/O driver (qdio)

Disk I/O attachment and storage subsystem

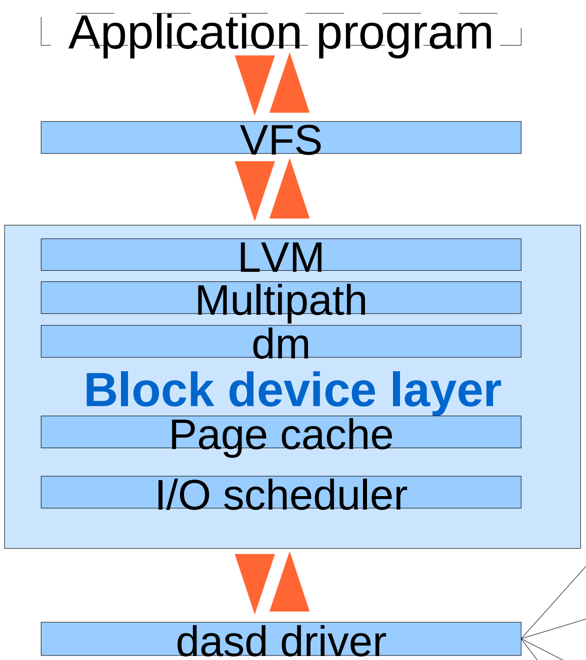
- Each disk is a configured volume in an extent pool (here extent pool = rank)
- An extent pool can be configured for either ECKD dasds or SCSI LUNs
- Host bus adapters connect internally to both servers



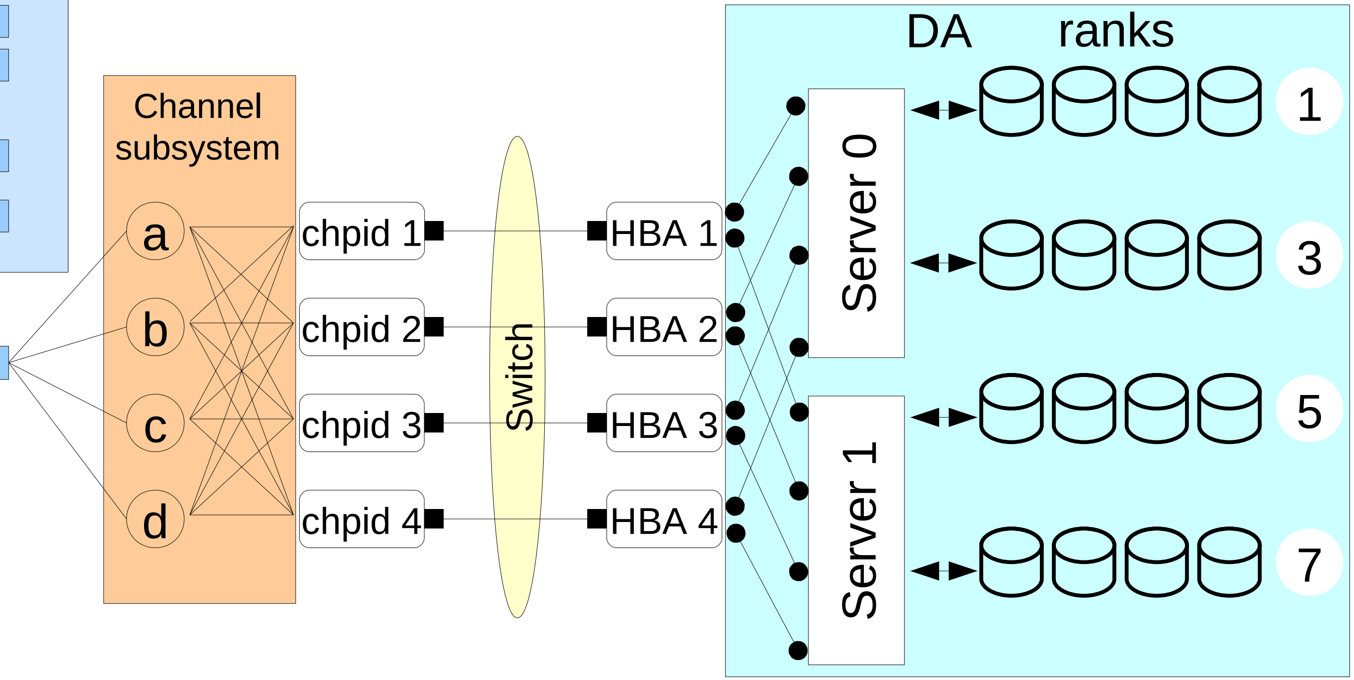
I/O processing characteristics

- FICON/ECKD:
  - 1:1 mapping host subchannel:dasd
  - Serialization of I/Os per subchannel
  - I/O request queue in Linux
  - Disk blocks are 4KB
  - High availability by FICON path groups
  - Load balancing by FICON path groups and Parallel Access Volumes
- FCP/SCSI
  - Several I/Os can be issued against a LUN immediately
  - Queuing in the FICON Express card and/or in the storage server
  - Additional I/O request queue in Linux
  - Disk blocks are 512 bytes
  - High availability by Linux multipathing, type failover or multibus
  - Load balancing by Linux multipathing, type multibus

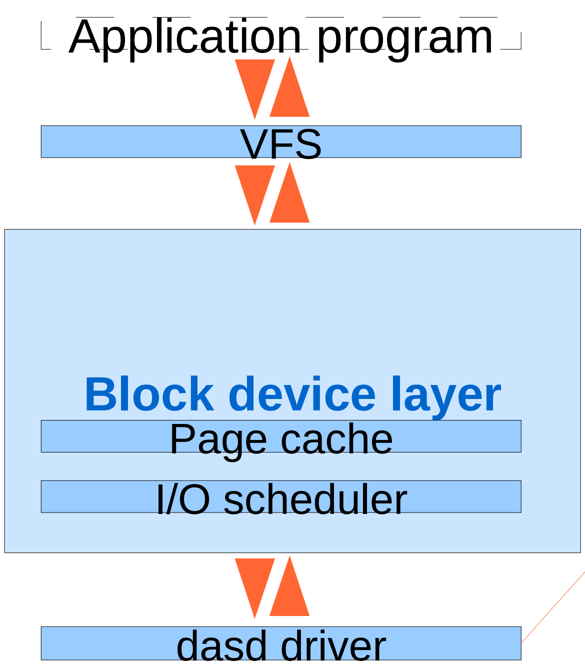
General layout for FICON/ECKD



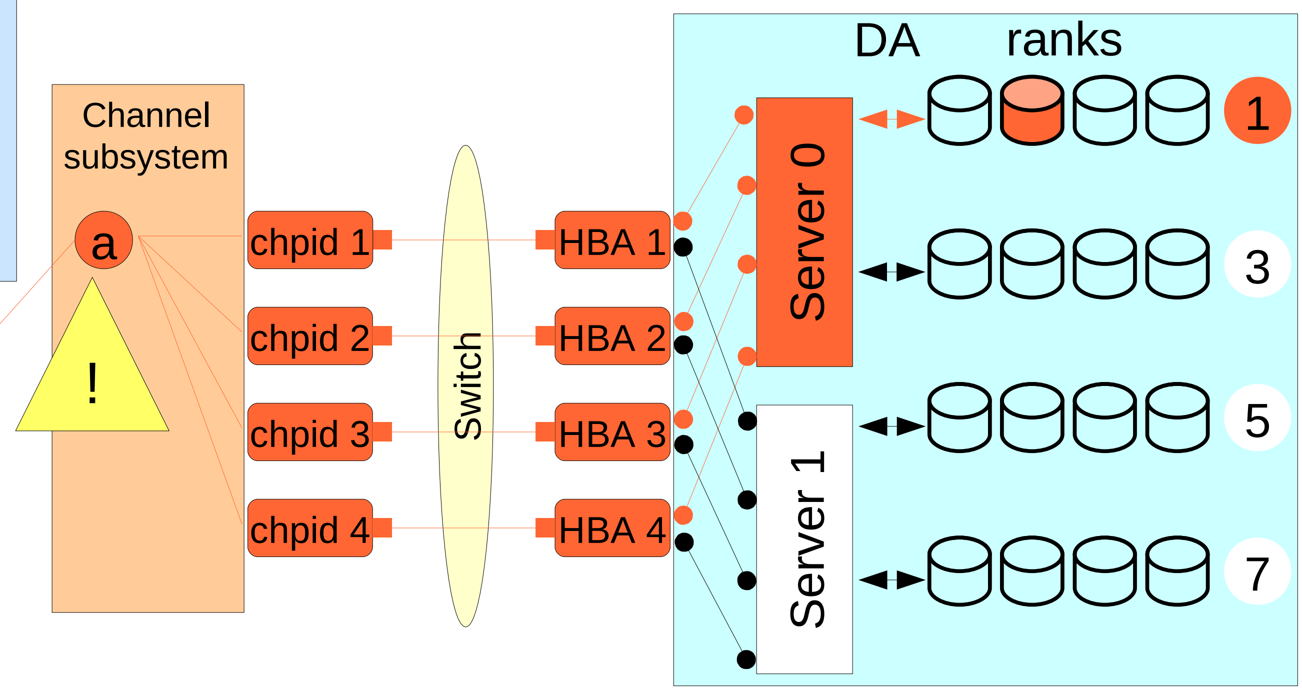
- The dasd driver starts the I/O on a subchannel
- Each subchannel connects to all channel paths in the path group
- Each channel connects via a switch to a host bus adapter
- A host bus adapter connects to both servers
- Each server connects to its ranks



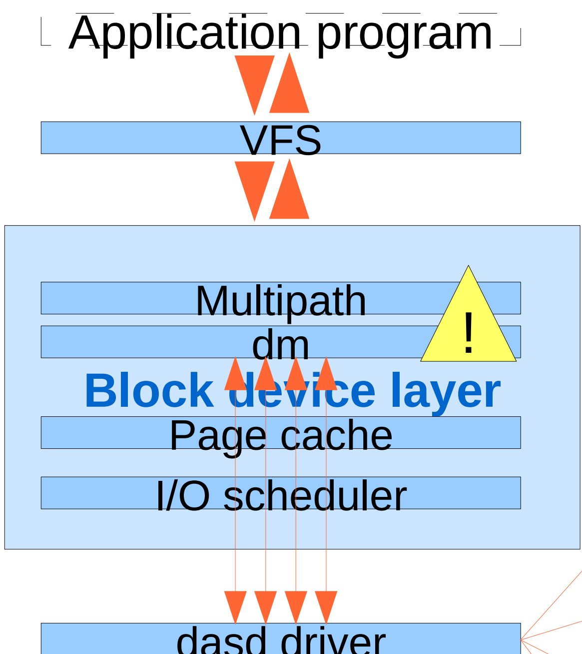
FICON/ECKD dasd I/O to a single disk



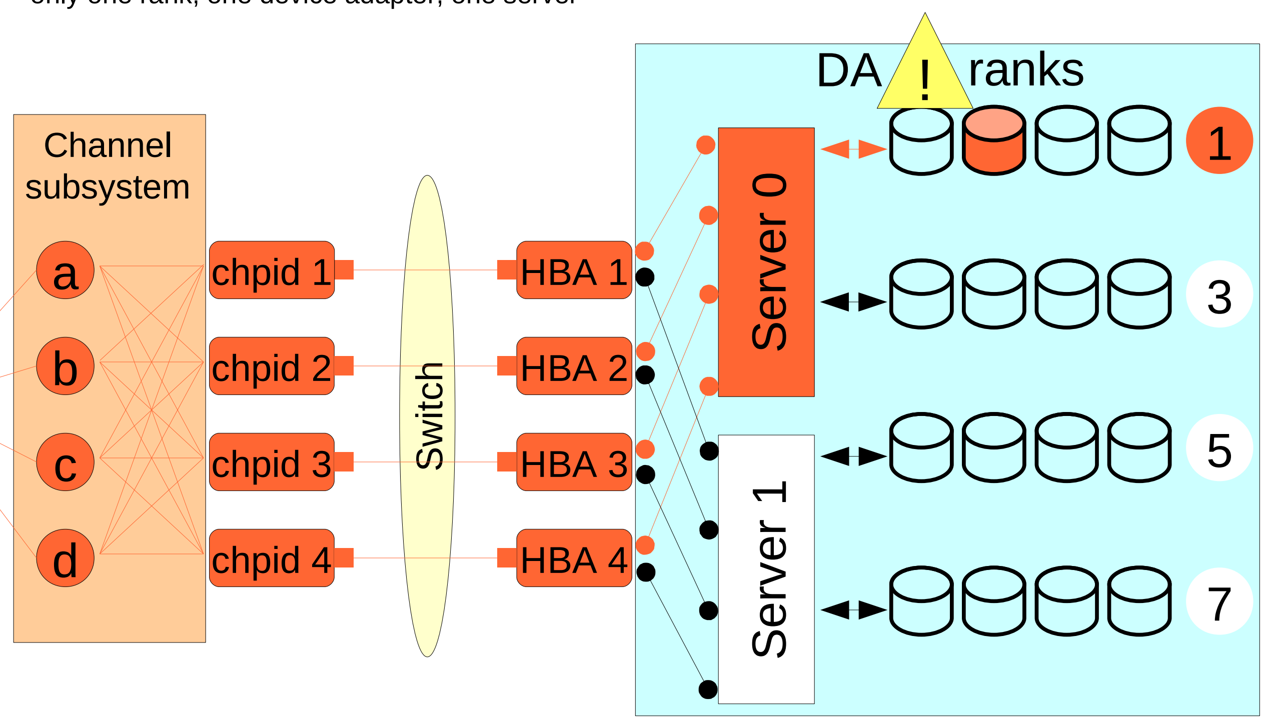
- Assume that subchannel a corresponds to disk 2 in rank 1
- The full choice of host adapters can be used
- Only one I/O can be issued at a time through subchannel a
- All other I/Os need to be queued in the dasd driver and in the block device layer until the subchannel is no longer busy with the preceding I/O



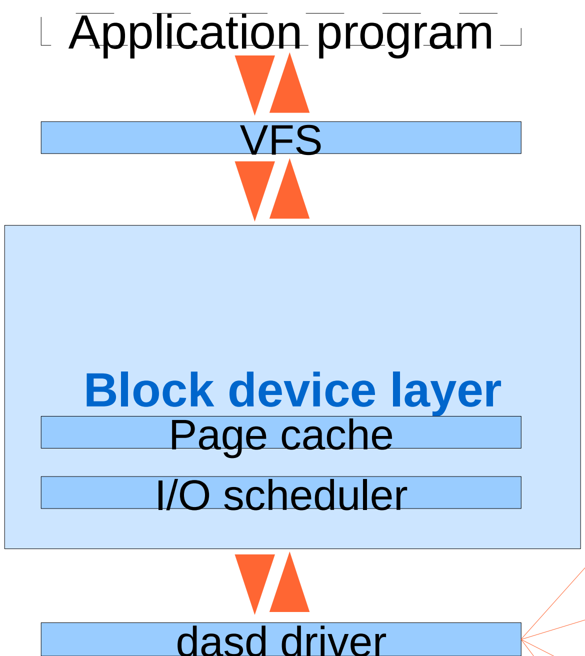
FICON/ECKD dasd I/O to a single disk with PAV (SLES10 / RHEL5)



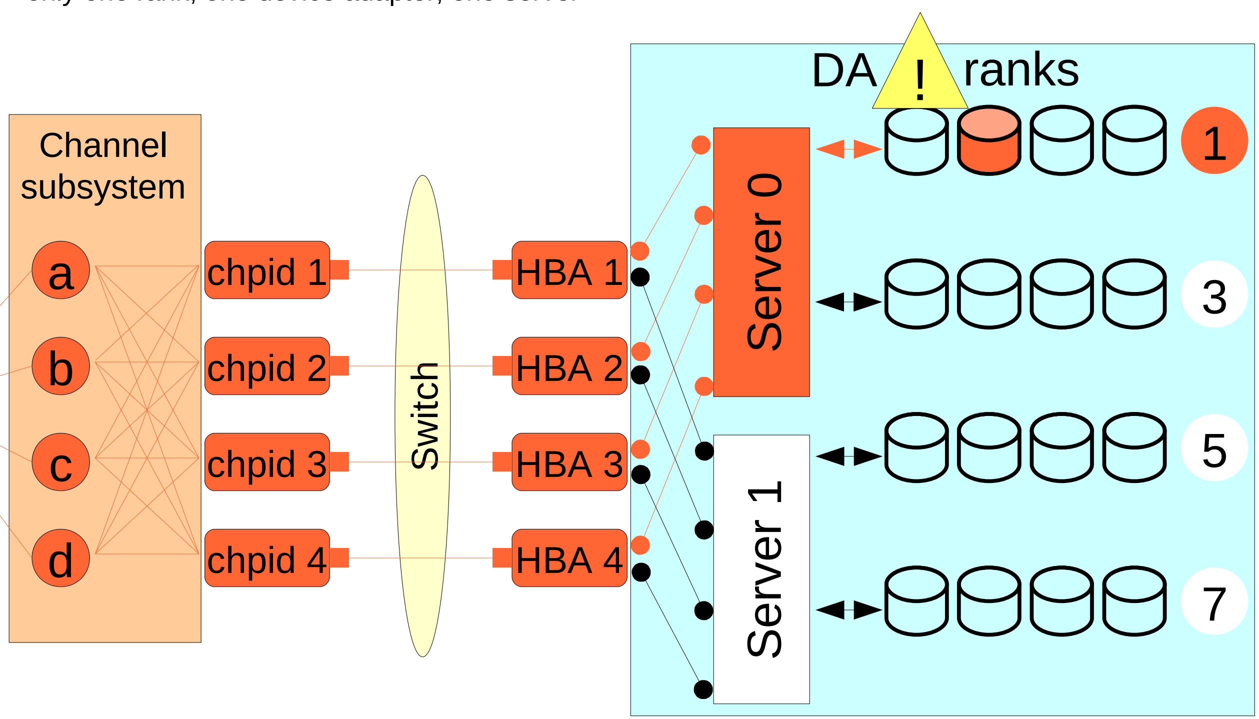
- VFS sees one device
- The device mapper sees the real device and all alias devices
- Parallel Access Volumes solve the I/O queuing in front of the subchannel
- Each alias device uses its own subchannel
- Additional processor cycles are spent to do the load balancing in the device mapper
- The next slowdown is the fact that only one disk is used in the storage server. This implies the use of only one rank, one device adapter, one server



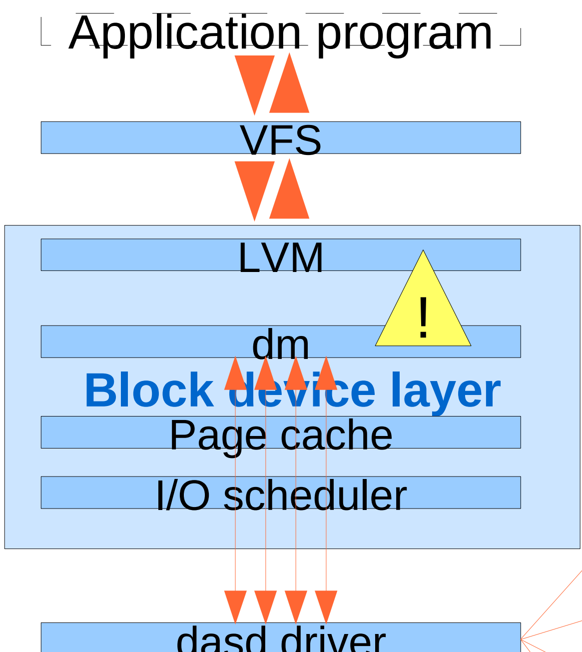
FICON/ECKD dasd I/O to a single disk with HyperPAV (SLES11)



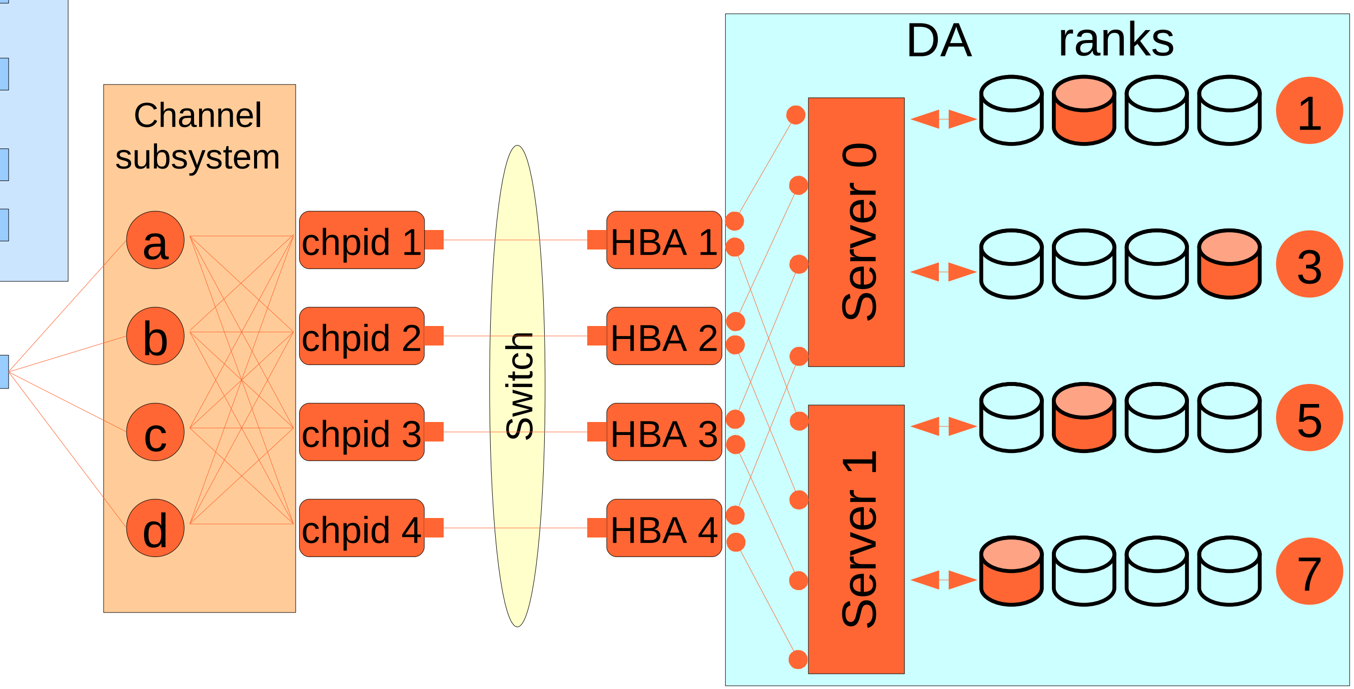
- VFS sees one device
- The dasd driver sees the real device and all alias devices
- Load balancing with HyperPAV and static PAV is done in the dasd driver. The aliases need only to be added to Linux. The load balancing works better than on the device mapper layer.
- Less additional processor cycles are needed than with Linux multipath.
- The next slowdown is the fact that only one disk is used in the storage server. This implies the use of only one rank, one device adapter, one server



FICON/ECKD dasd I/O to a linear or striped logical volume

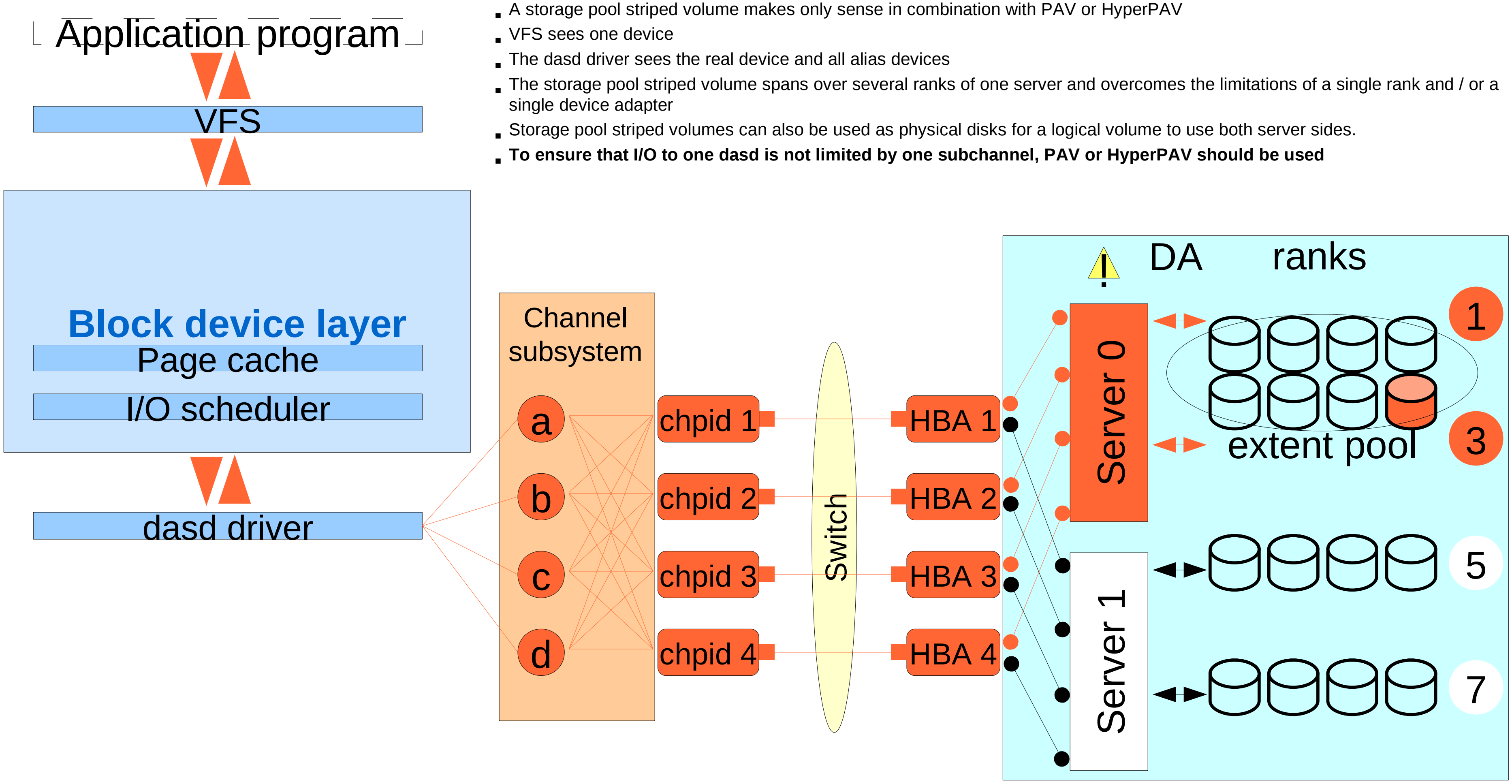


- VFS sees one device (logical volume)
- The device mapper sees the logical volume and the physical volumes
- Additional processor cycles are spent to map the I/Os to the physical volumes.
- Striped logical volumes require more additional processor cycles than linear logical volumes
- With a striped logical volume the I/Os can be well balanced over the entire storage server and overcome limitations from a single rank, a single device adapter or a single server
- **To ensure that I/O to one physical disk is not limited by one subchannel, PAV or HyperPAV should be used in combination with logical volumes**

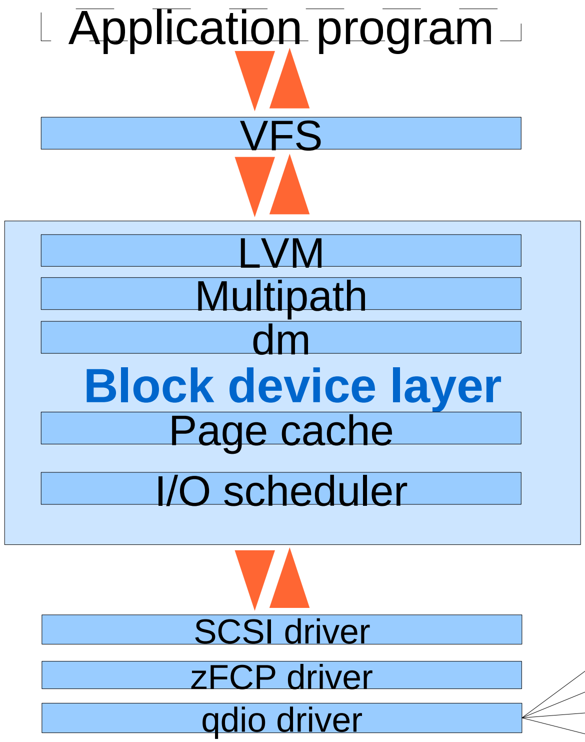


FICON/ECKD dasd I/O to a storage pool striped volume with HyperPAV (SLES11)

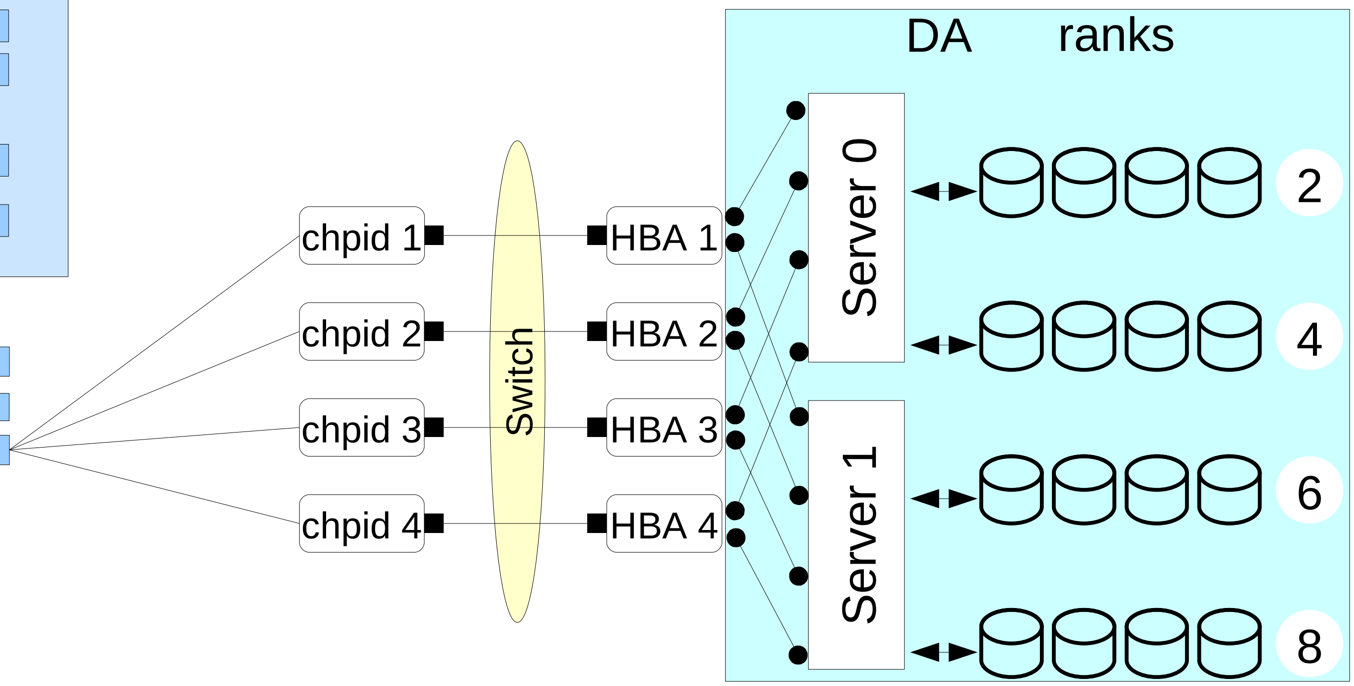
- A storage pool striped volume makes only sense in combination with PAV or HyperPAV
- VFS sees one device
- The dasd driver sees the real device and all alias devices
- The storage pool striped volume spans over several ranks of one server and overcomes the limitations of a single rank and / or a single device adapter
- Storage pool striped volumes can also be used as physical disks for a logical volume to use both server sides.
- **To ensure that I/O to one dasd is not limited by one subchannel, PAV or HyperPAV should be used**



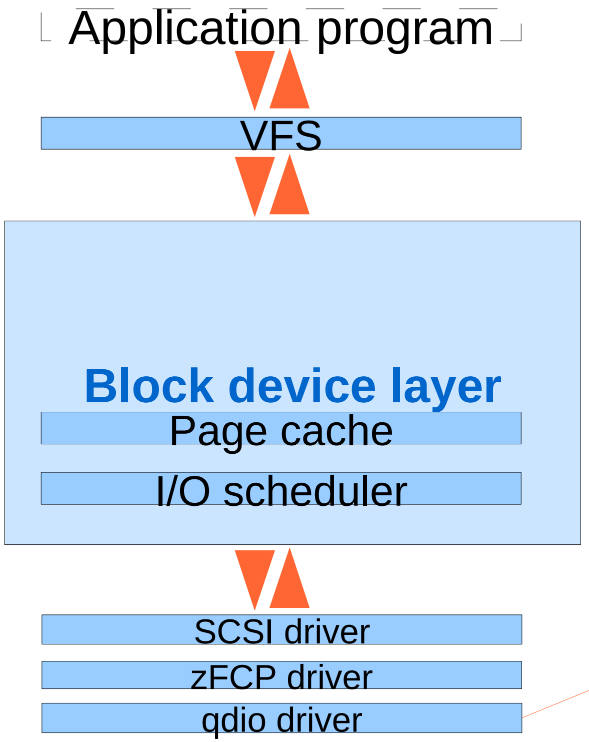
General layout for FCP/SCSI



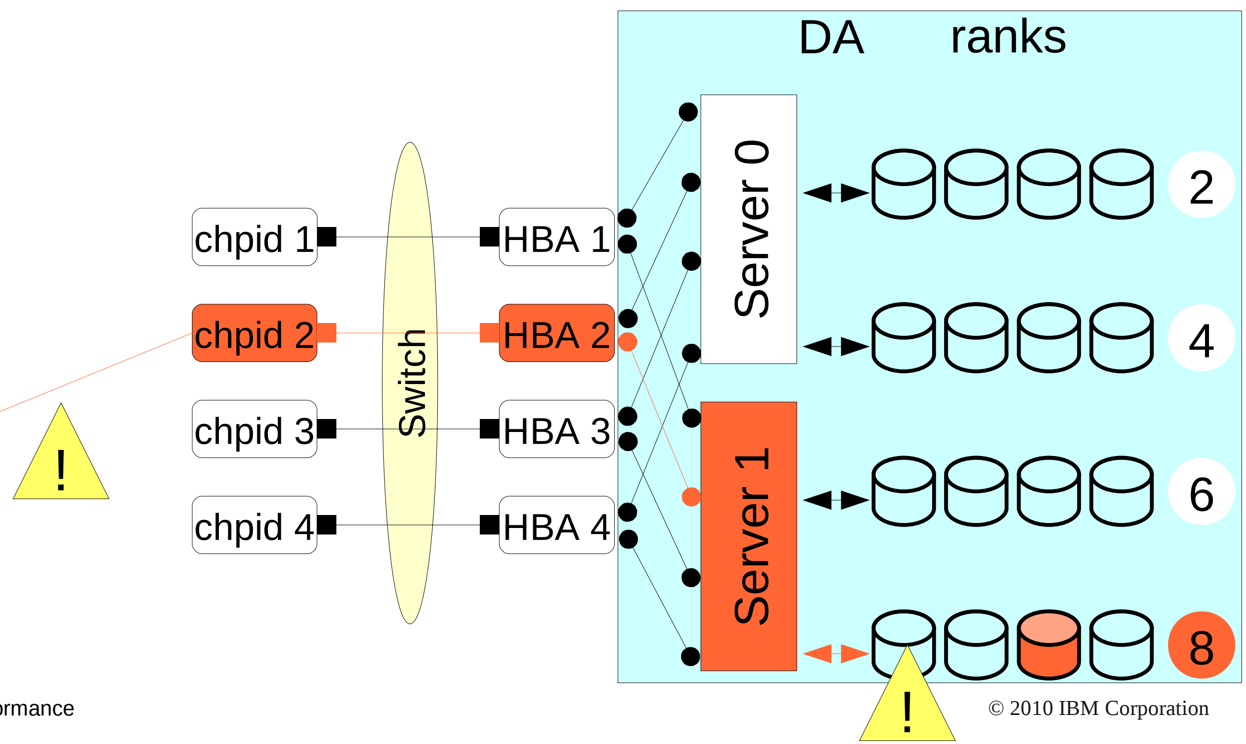
- The SCSI driver finalizes the I/O requests
- The zFCP driver adds the FCP protocol to the requests
- The qdio driver transfers the I/O to the channel
- A host bus adapter connects to both servers
- Each server connects to its ranks



FCP/SCSI LUN I/O to a single disk

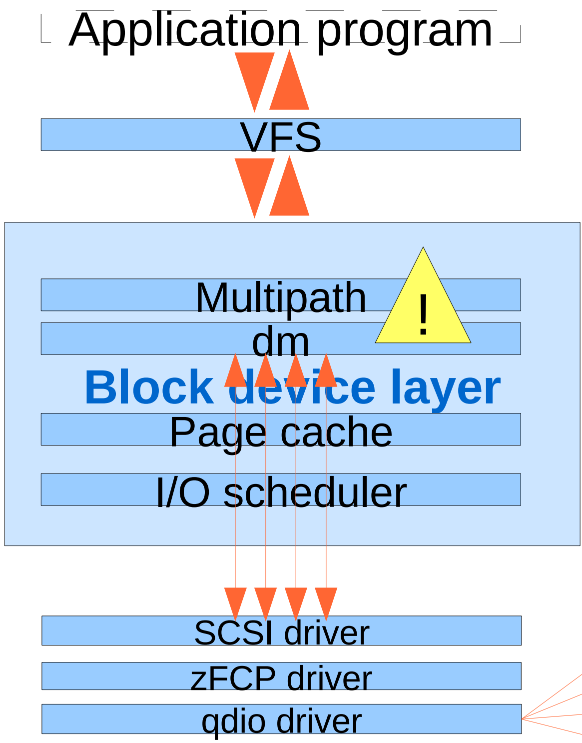


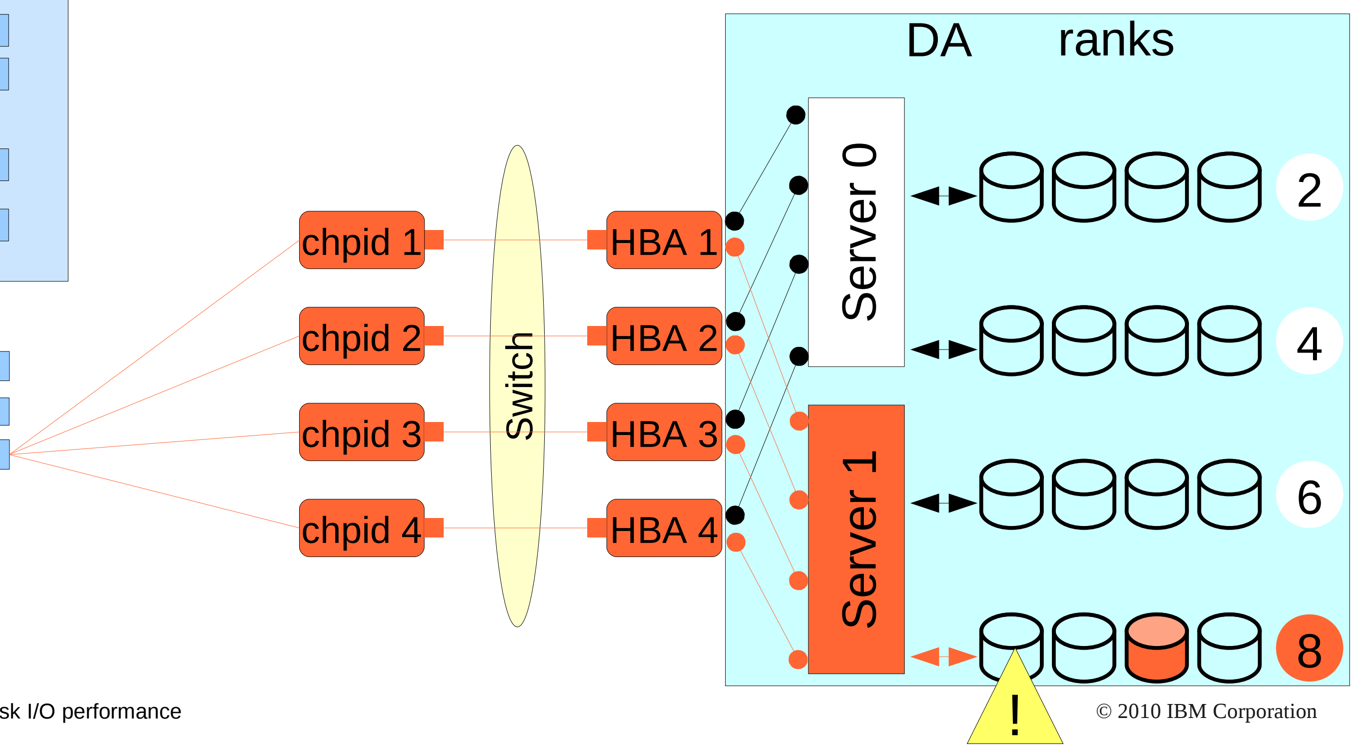
- Assume that disk 3 in rank 8 is reachable via channel 2 and host bus adapter 2
- Up to 32 (default value) I/O requests can be sent out to disk 3 before the first completion is required
- The throughput will be limited by the rank and / or the device adapter
- There is no high availability provided for the connection between the host and the storage server



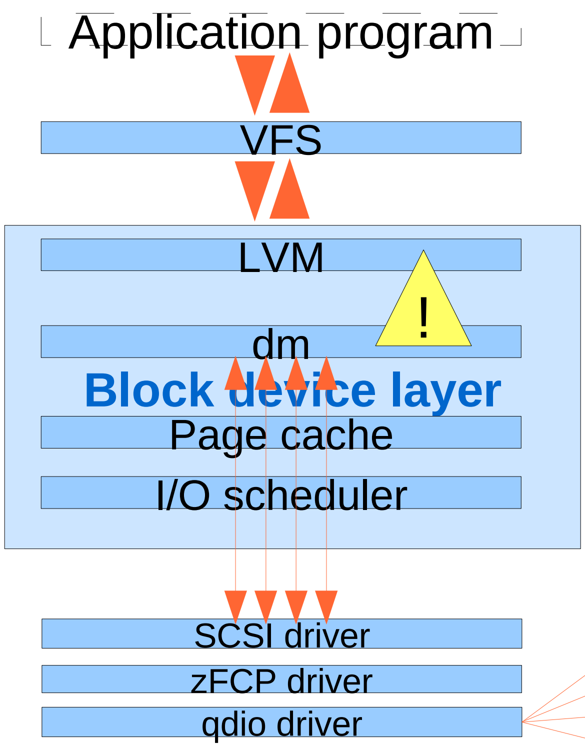
FCP/SCSI LUN I/O to a single disk with multipathing

- VFS sees one device
- The device mapper sees the multibus or failover alternatives to the same disk
- Administrational effort is required to define all paths to one disk
- Additional processor cycles are spent to do the mapping to the desired path for the disk in the device mapper
- Multibus requires more additonal processor cycles than failover

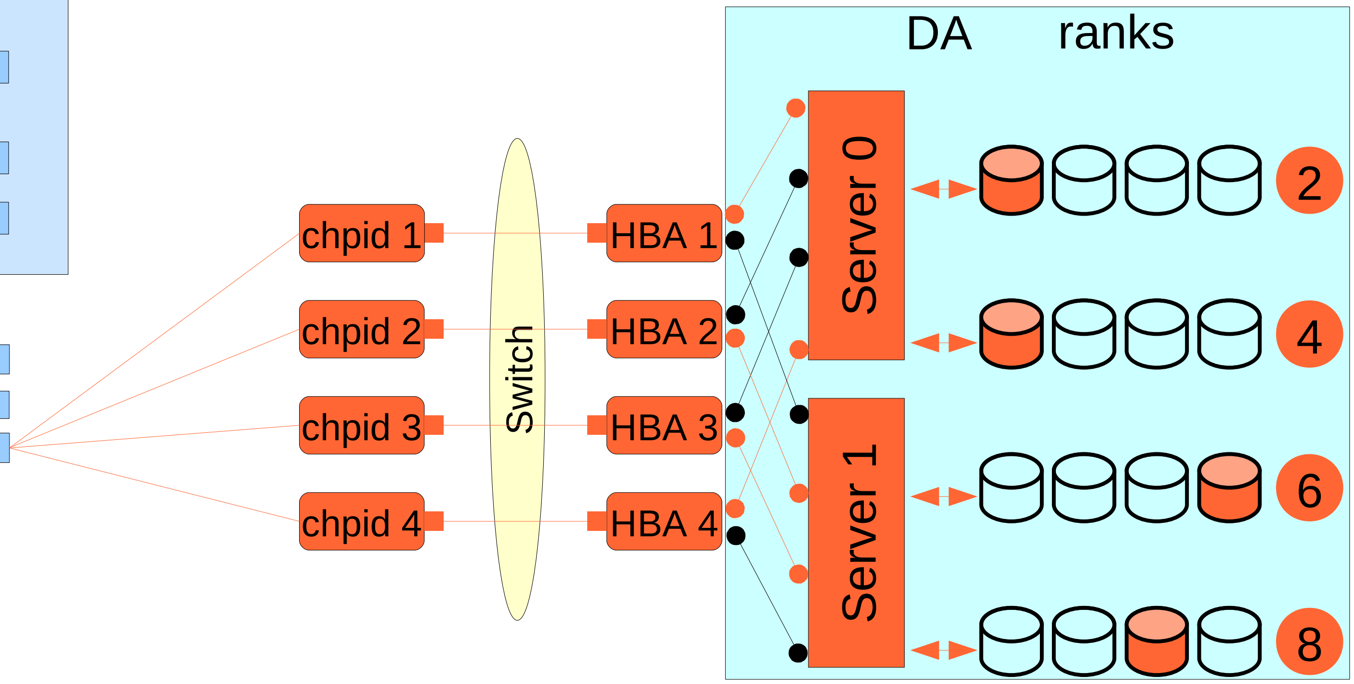




FCP/SCSI LUN I/O to a to a linear or striped logical volume

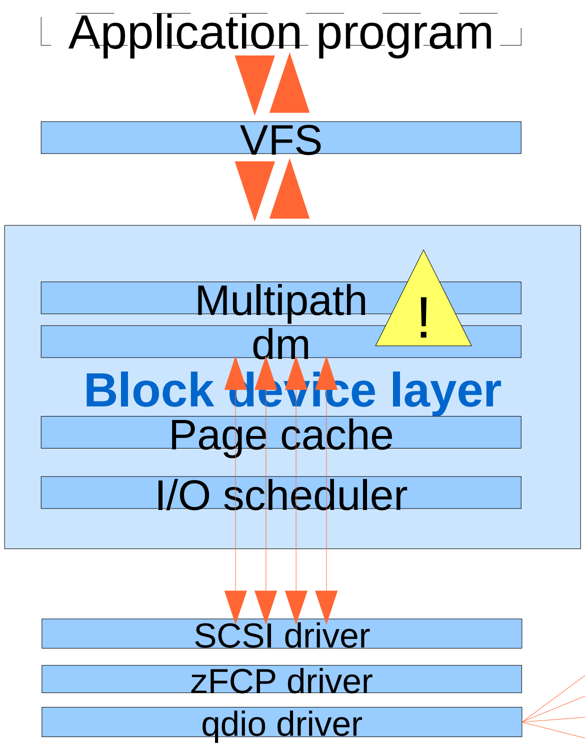


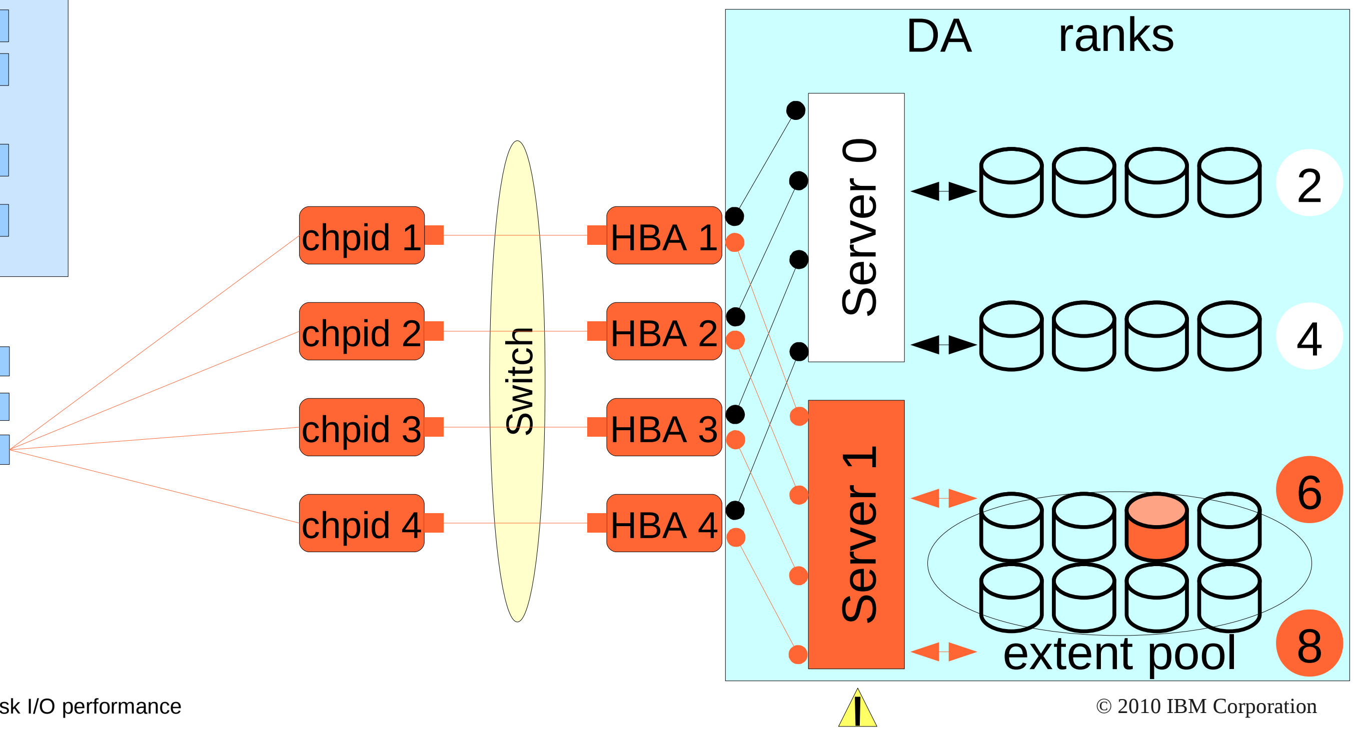
- VFS sees one device (logical volume)
- The device mapper sees the logical volume and the physical volumes
- Additional processor cycles are spent to do map the I/Os to the physical volumes.
- Striped logical volumes require more additional processor cycles than linear logical volumes
- With a striped logical volume the I/Os can be well balanced over the entire storage server and overcome limitations from a single rank, a single device adapter or a single server
- **To ensure high availability the logical volume should be used in combination with multipathing**



FCP/SCSI LUN I/O to a storage pool striped volume with multipathing

- Storage pool striped volumes make no sense without high availability
- VFS sees one device
- The device mapper sees the multibus or failover alternatives to the same disk
- The storage pool striped volume spans over several ranks of one server and overcomes the limitations of a single rank and / or a single device adapter
- Storage pool striped volumes can also be used as physical disks for a logical volume to make use of both server sides.





Summary, recommendations and outlook

- FICON/ECKD
  – Storage pool striped disks (no disk placement)
  – HyperPAV (SLES11)
  – Large volume (RHEL5, SLES11)
  – High Performance FICON (SLES11)
- FCP/SCSI
  – Linux LV with striping (disk placement)
  – Multipath with failover

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.