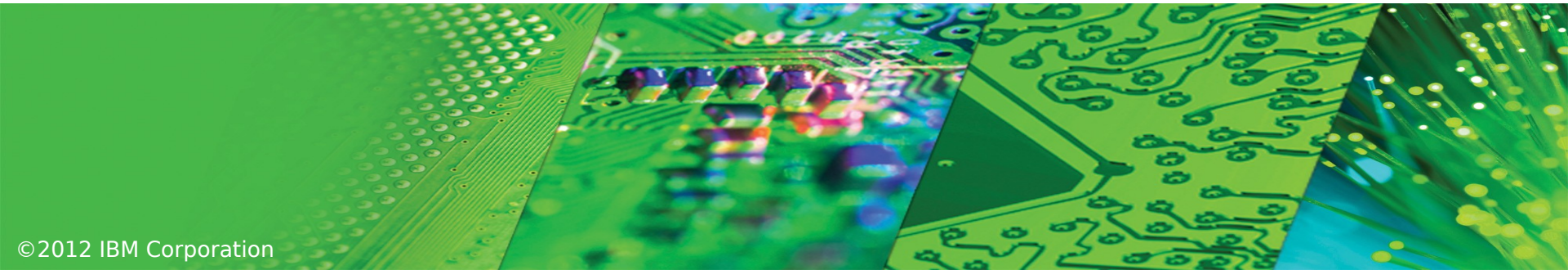
2012
IBM System z Technical University

Enabling the infrastructure for smarter computing

Linux on System z – disk I/O performance

zLG13

Mario Held



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.

Oracle and Java are registered trademarks of Oracle and/or its affiliates in the United States, other countries, or both.

Red Hat, Red Hat Enterprise Linux (RHEL) are registered trademarks of Red Hat, Inc. in the United States and other countries.

SLES and SUSE are registered trademarks of Novell, Inc., in the United States and other countries.

SPEC and the "performance chart" SPEC logo are registered trademarks of the Standard Performance Evaluation Corporation.

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

Agenda

- Terms and characteristics in the disk I/O area
- Disk configurations for FICON/ECKD and FCP/SCSI
- Measurements
  - Setup description
  - Results comparison for FICON/ECKD and FCP/SCSI
- Summary / Conclusion

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 extendable
  - may lead to smaller I/O request sizes compared to linear logical volumes or normal volumes
- Don't use logical volumes for “/” or “/usr”
  - in case 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
- LVM or Logical Volume Manager is the tool to manage logical volumes

Linux multipath

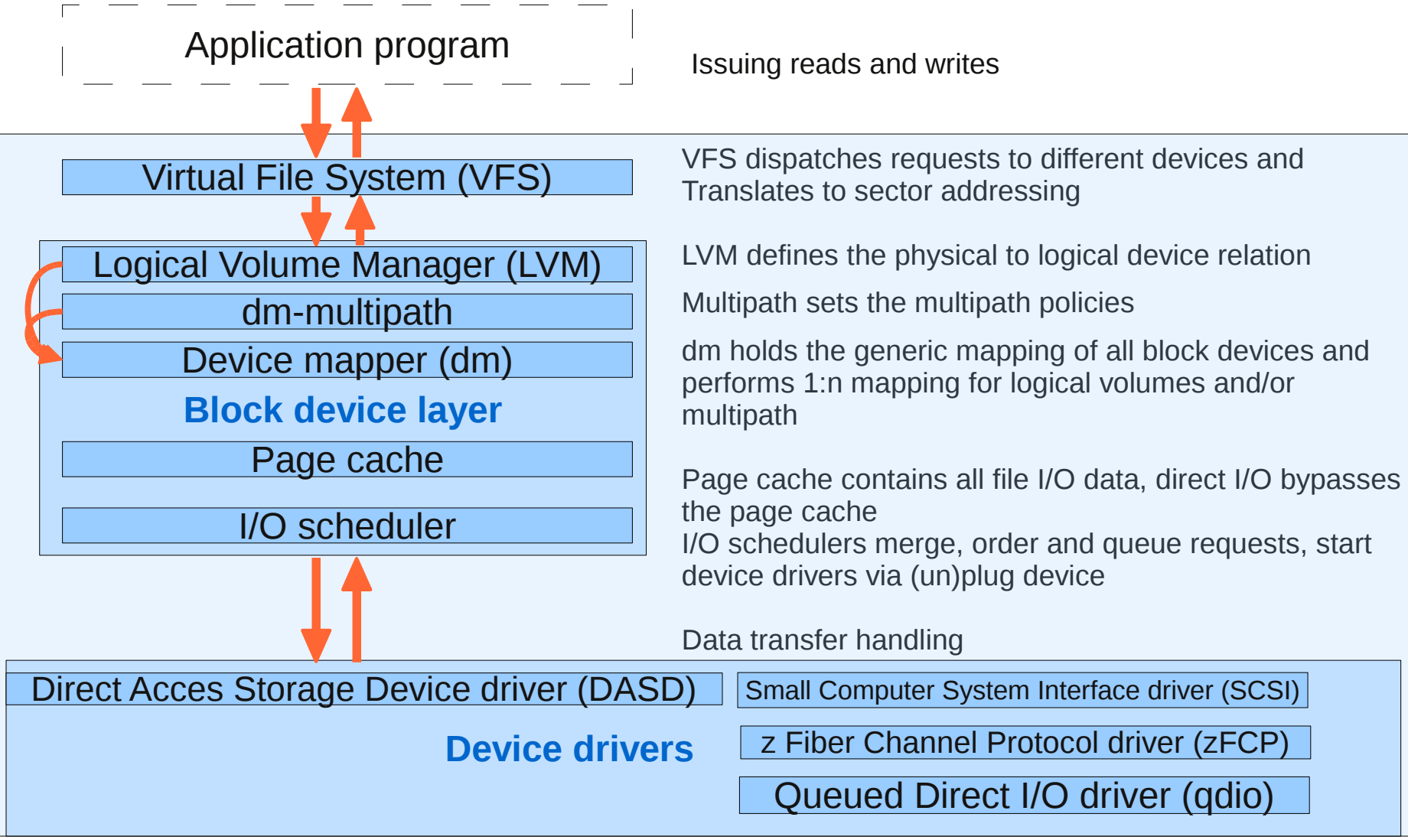
- Connects a volume over multiple independent paths and/or ports
- Multibus mode
  - for load balancing
  - to overcome bandwidth limitations of a single path
  - uses all paths in a priority group in a round-robin manner
    - switches the path after a configurable amount of I/Os

      See rr_min_io in multipath.conf
  - increases throughput for
    - ECKD volumes with static PAV devices in Linux distributions older than SLES11 and RHEL6
    - SCSI volumes in all Linux distributions
  - for high availability
- Failover mode
  - keeps the volume accessible in case of a single failure in the connecting hardware
    - should one path fail, the operating system will route I/O through one of the remaining paths, with no changes visible to the applications
  - will not improve performance
  - for high availability

Page cache considerations

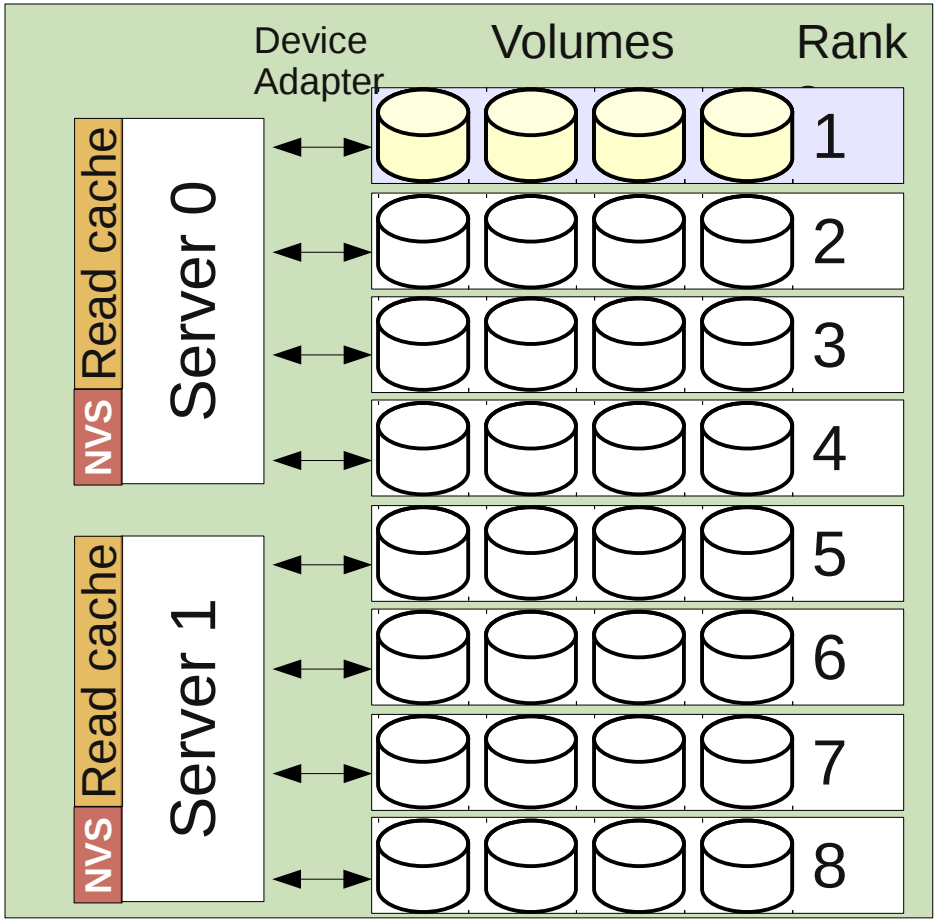
- Page cache helps Linux to economize I/O
- Requires Linux memory pages
- Adopts its size to the amount of memory in use by processes
- Write requests are delayed and data in the page cache can have multiple updates before being written to disk.
- Write requests in the page cache can be merged into larger I/O requests
- Read requests can be made faster by adding a read ahead quantity, depending on the historical behavior of file system accesses by applications
- Settings of the configuration parameters swappiness, dirty_ratio, dirty_background ratio control the page cache
- Linux does not know which data the application really needs next. It makes only a guess

Linux kernel components involved in disk I/O



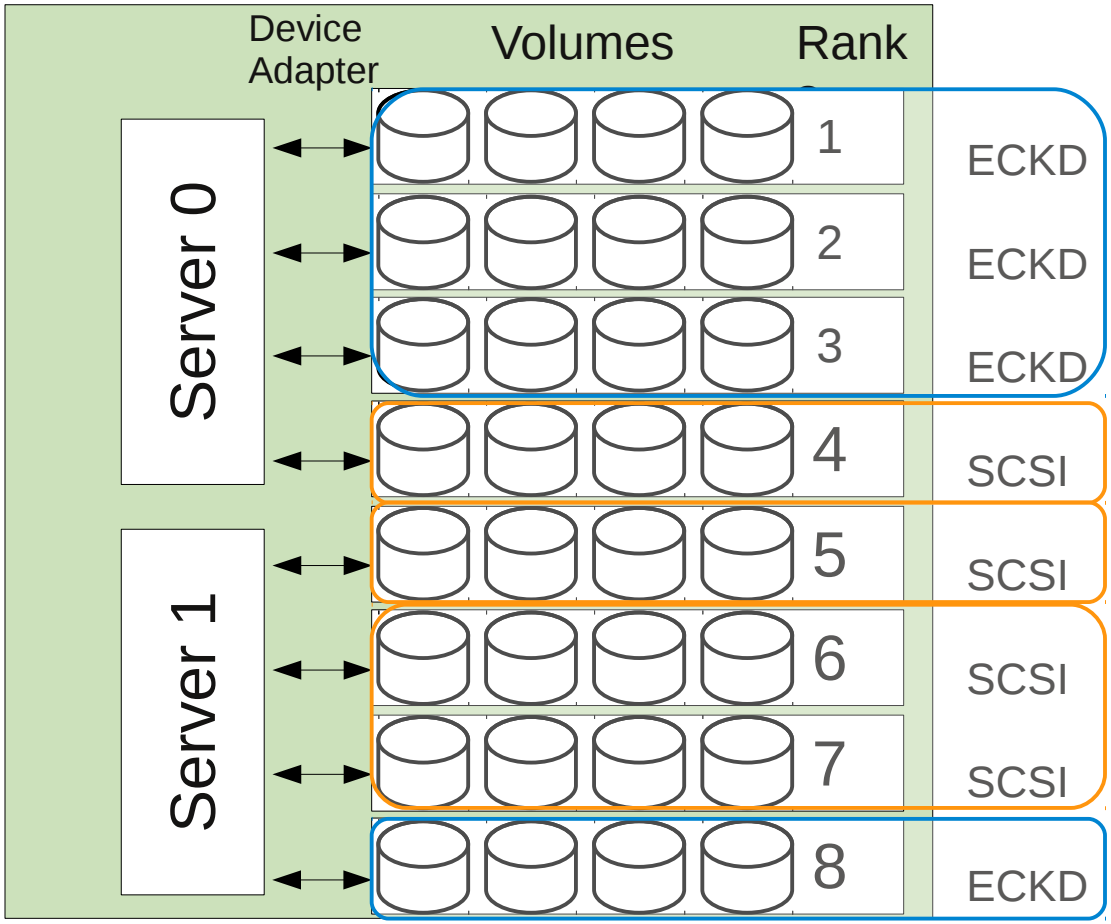
DS8000 storage server – components

- The rank
  - represents a raid array of disk drives
  - is connected to the servers by device adapters
  - belongs to one server for primary access
- A device adapter connects ranks to servers
- Each server controls half of the storage servers disk drives, read cache and non-volatile storage (NVS / write cache)



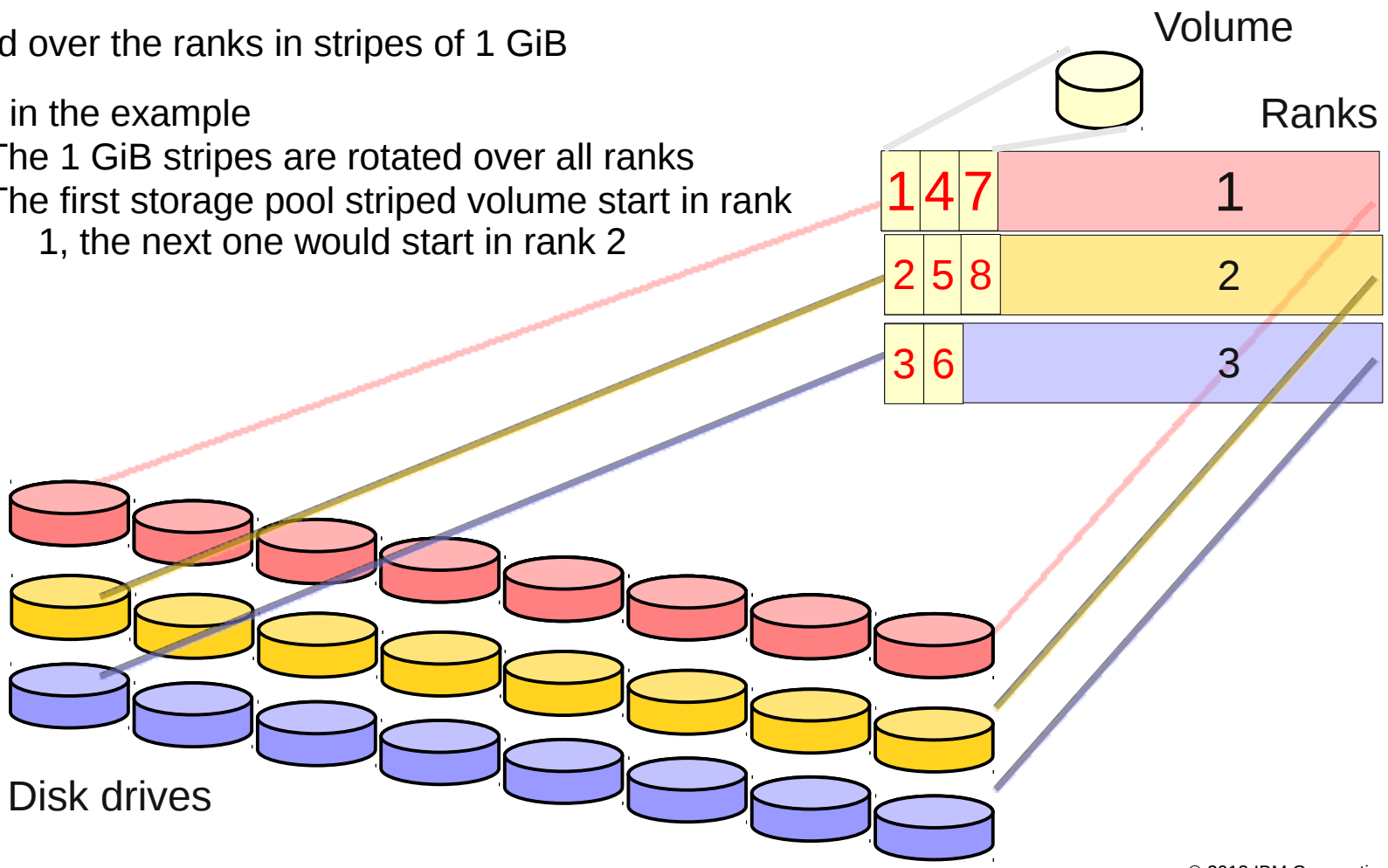
DS8000 storage server – extent pool

- Extent pools
  - consist of one or several ranks
  - can be configured for one type of volumes only: either ECKD DASDs or SCSI LUNs
  - are assigned to one server
- Some possible extent pool definitions for ECKD and SCSI are shown in the example



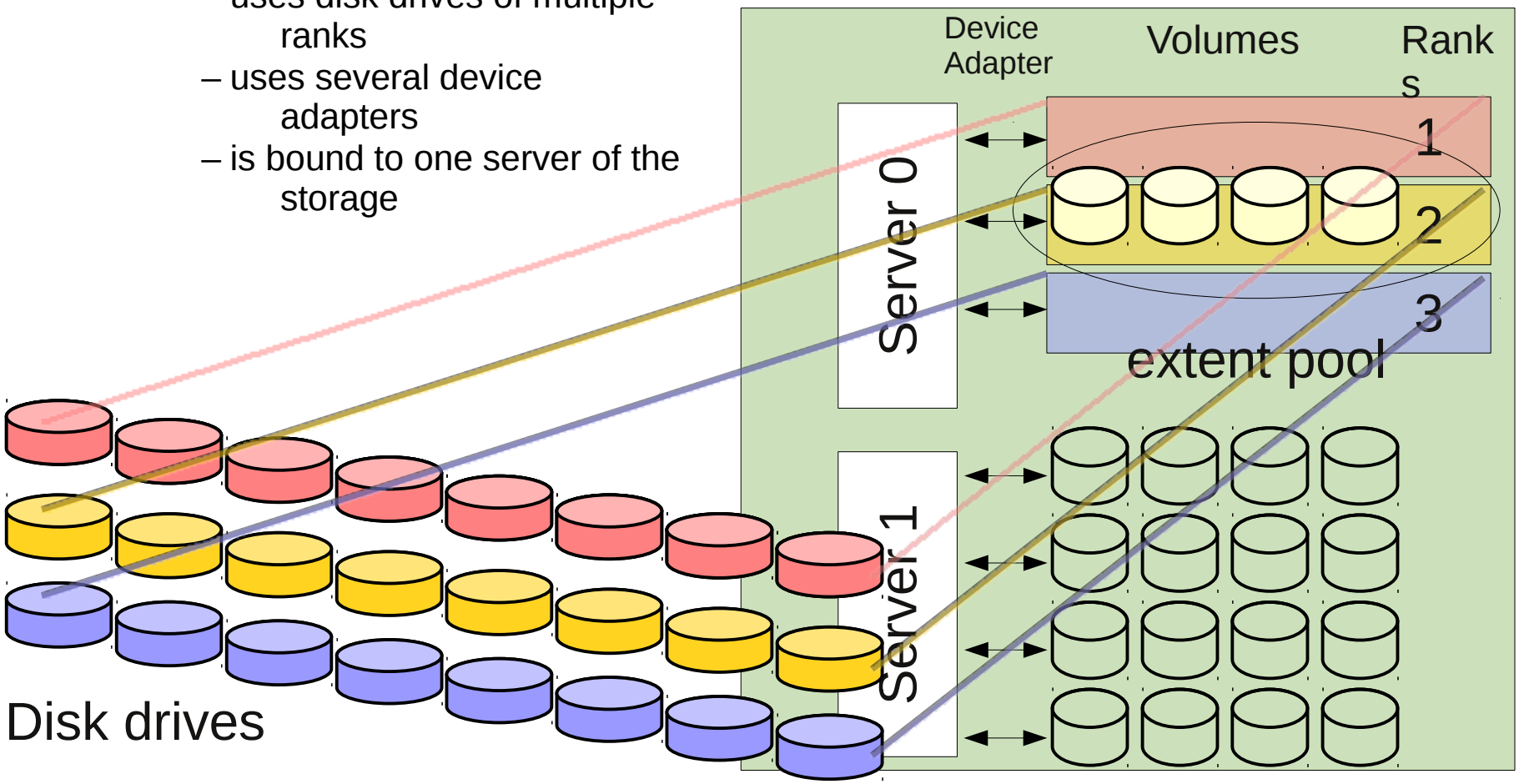
DS8000 storage pool striped volume

- A storage pool striped volume (rotate extents) is defined on a extent pool consisting of several ranks
- It is striped over the ranks in stripes of 1 GiB
- As shown in the example
  - The 1 GiB stripes are rotated over all ranks
  - The first storage pool striped volume start in rank 1, the next one would start in rank 2



DS8000 storage pool striped volume (cont.)

- A storage pool striped volume
  - uses disk drives of multiple ranks
  - uses several device adapters
  - is bound to one server of the storage

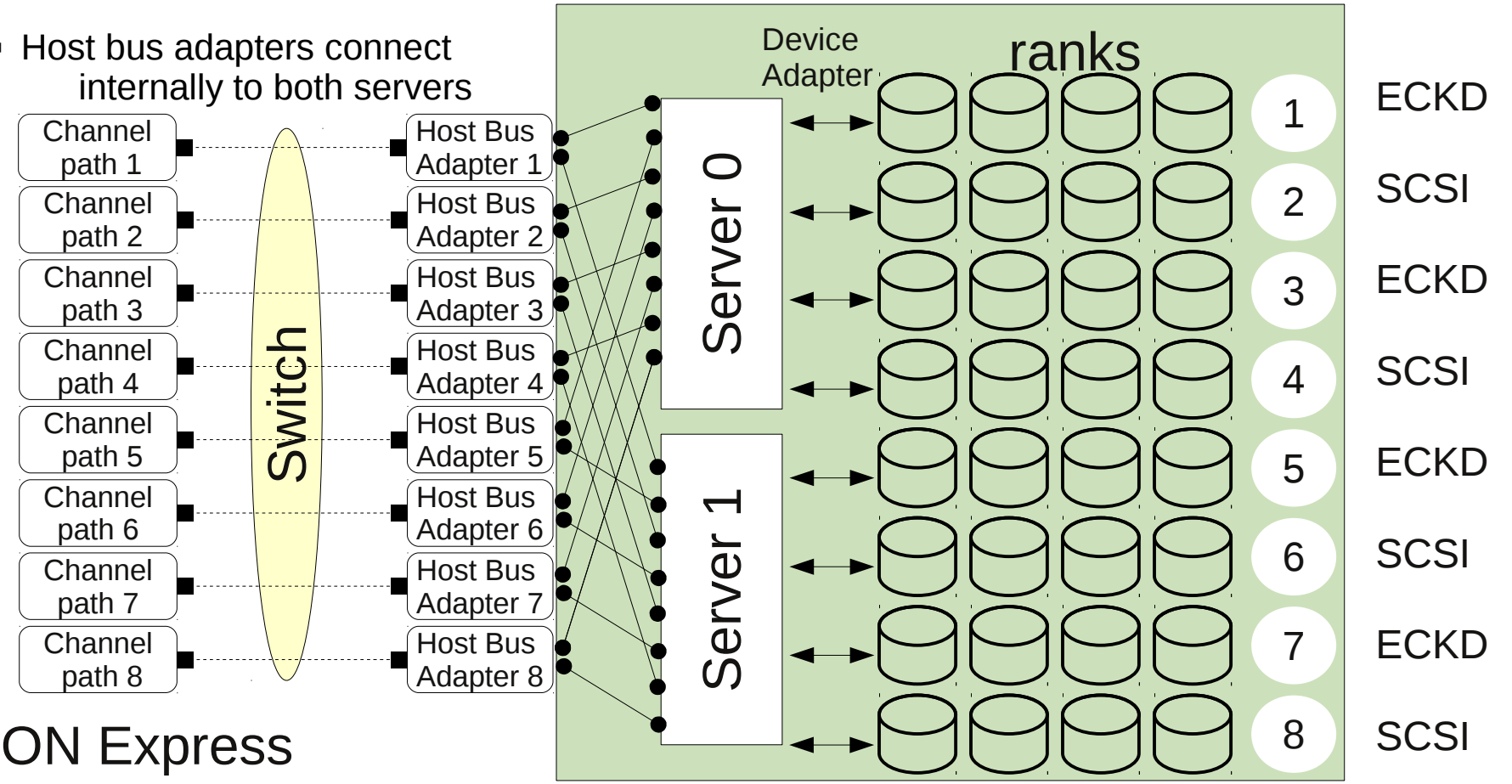


Striped volumes comparison/best practices for best performance

	LVM striped logical volumes	DS8000 storage pool striped volumes
Striping is done by...	Linux (device-mapper)	Storage server
Which disks to choose...	plan carefully	don't care
Disks from one extent pool...	per rank, alternating over servers	out of multiple ranks
Administrating disks is...	complex	simple
Extendable...	yes	no “gluing” disks together as linear LV can be a workaround
Stripe size...	variable, to suit your workload (64KiB, default)	1GiB

Disk I/O attachment and DS8000 storage subsystem

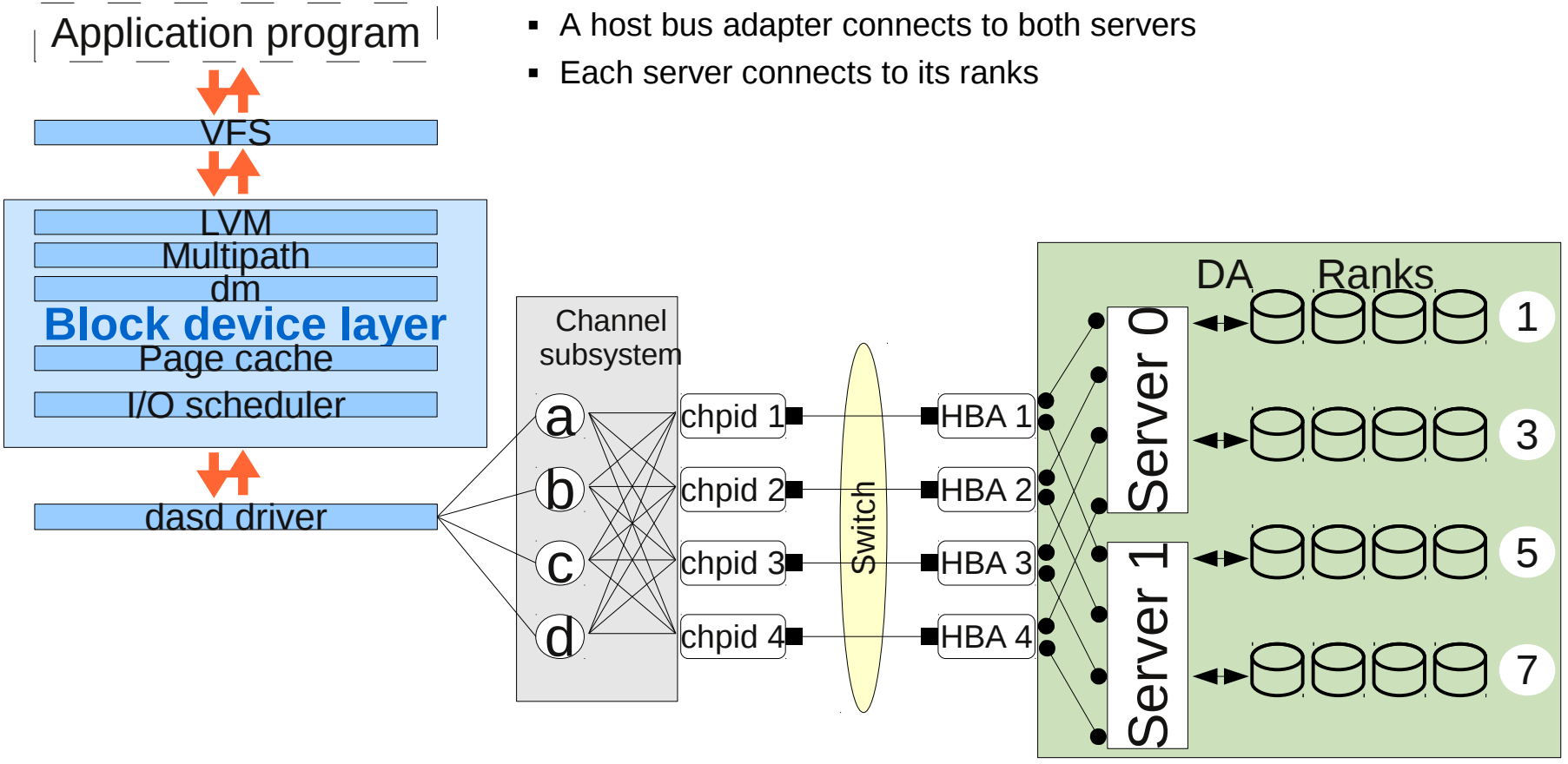
- Each disk is a configured volume in a extent pool (here extent pool = rank)
- Host bus adapters connect internally to both servers



FICON Express

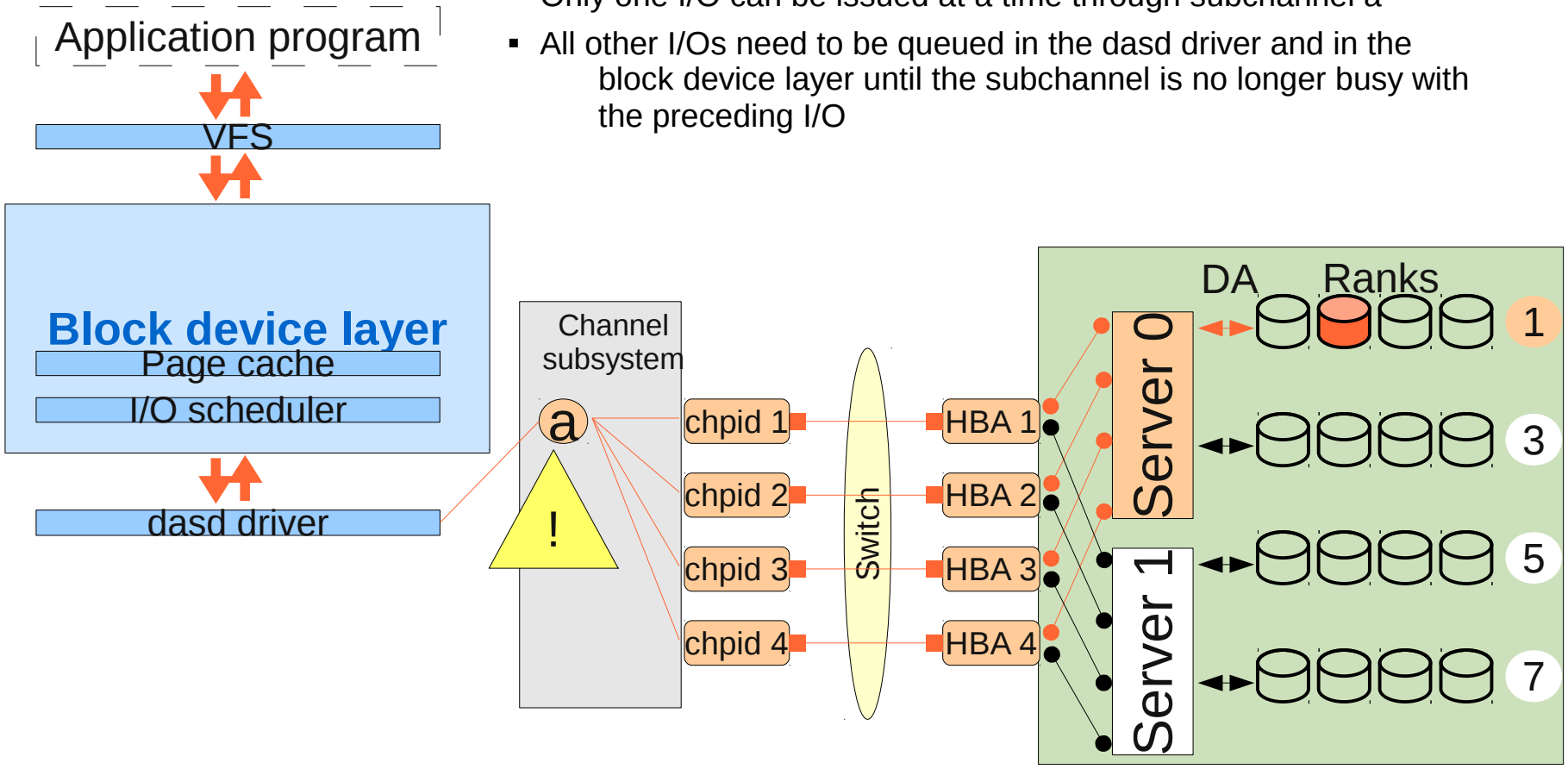
FICON/ECKD layout

- 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



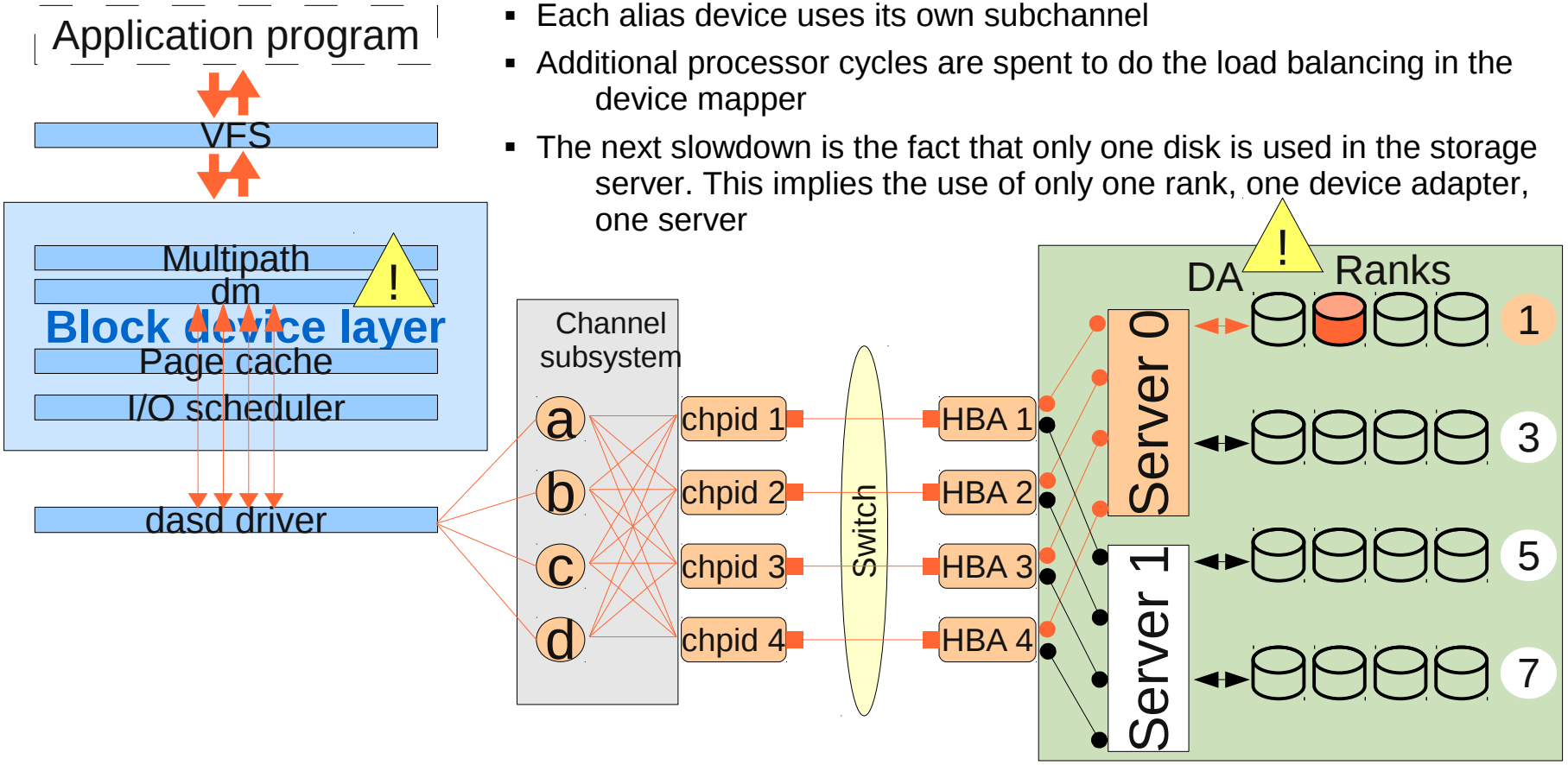
FICON/ECKD single disk I/O

- 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



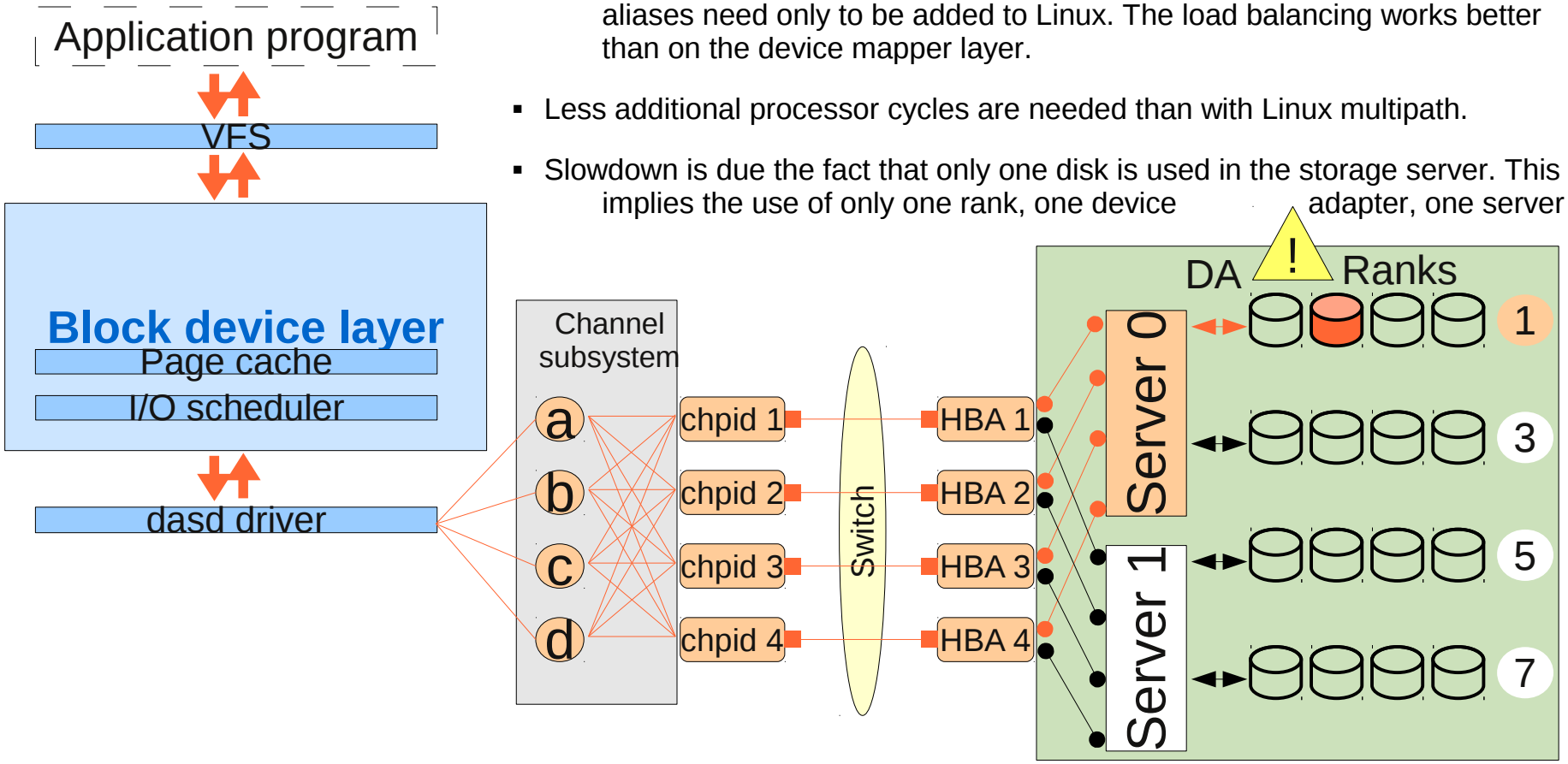
FICON/ECKD single disk I/O with PAV (SLES10/RHEL5)

- 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



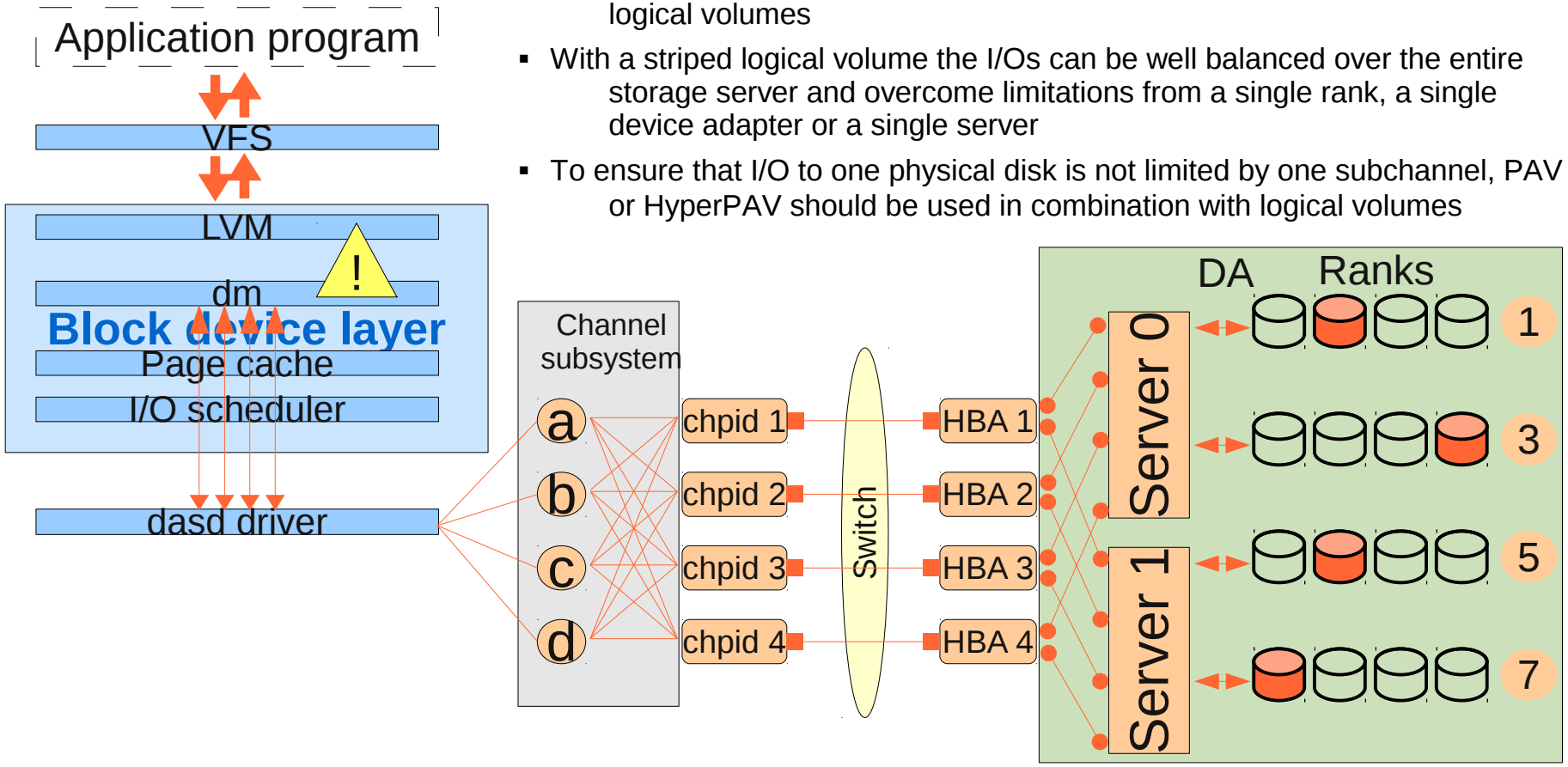
FICON/ECKD single disk I/O with HyperPAV (SLES11/RHEL6)

- 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.
- Slowdown is due 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



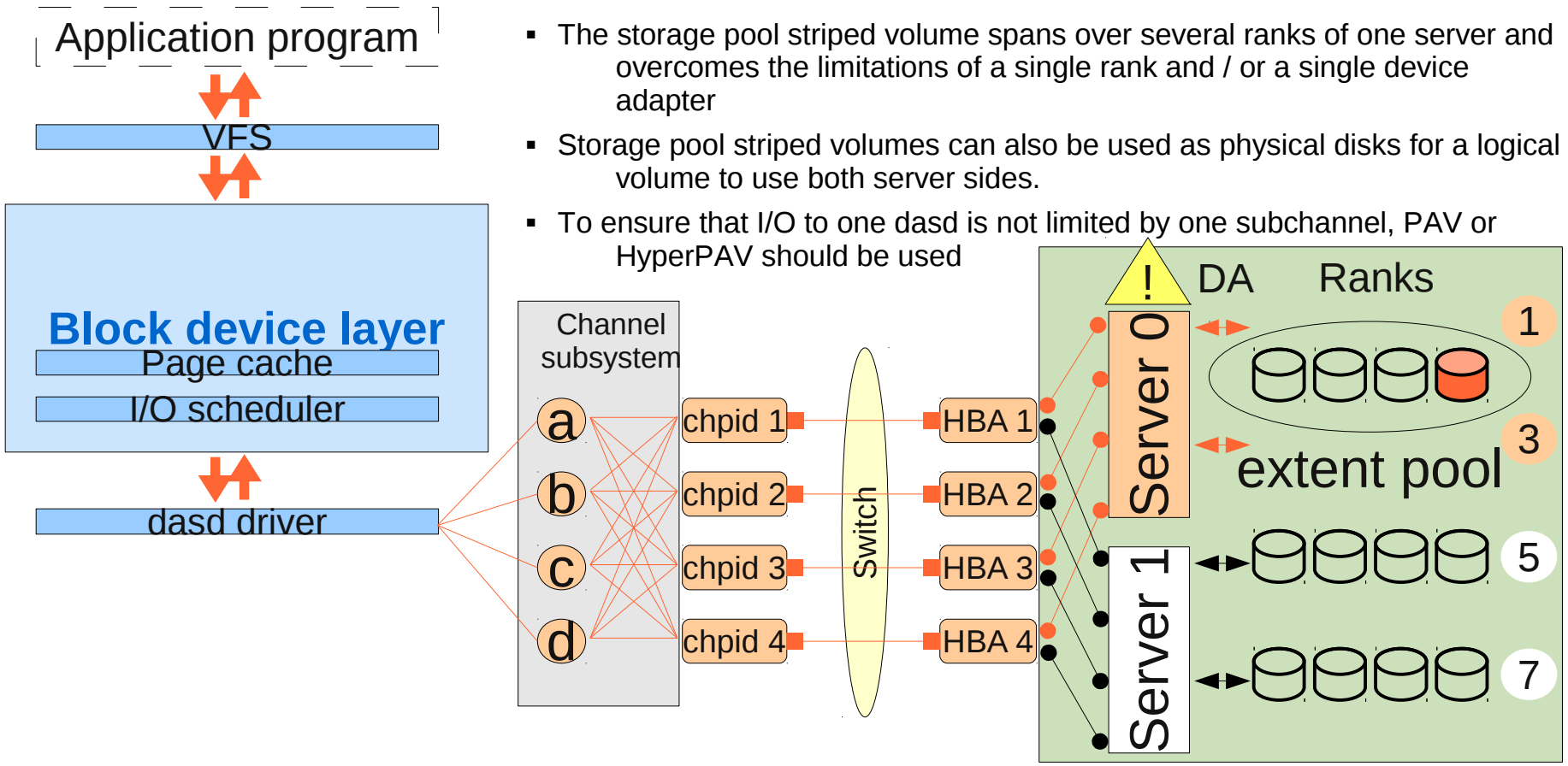
FICON/ECKD DASD I/O to a linear/striped logical volume

- 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



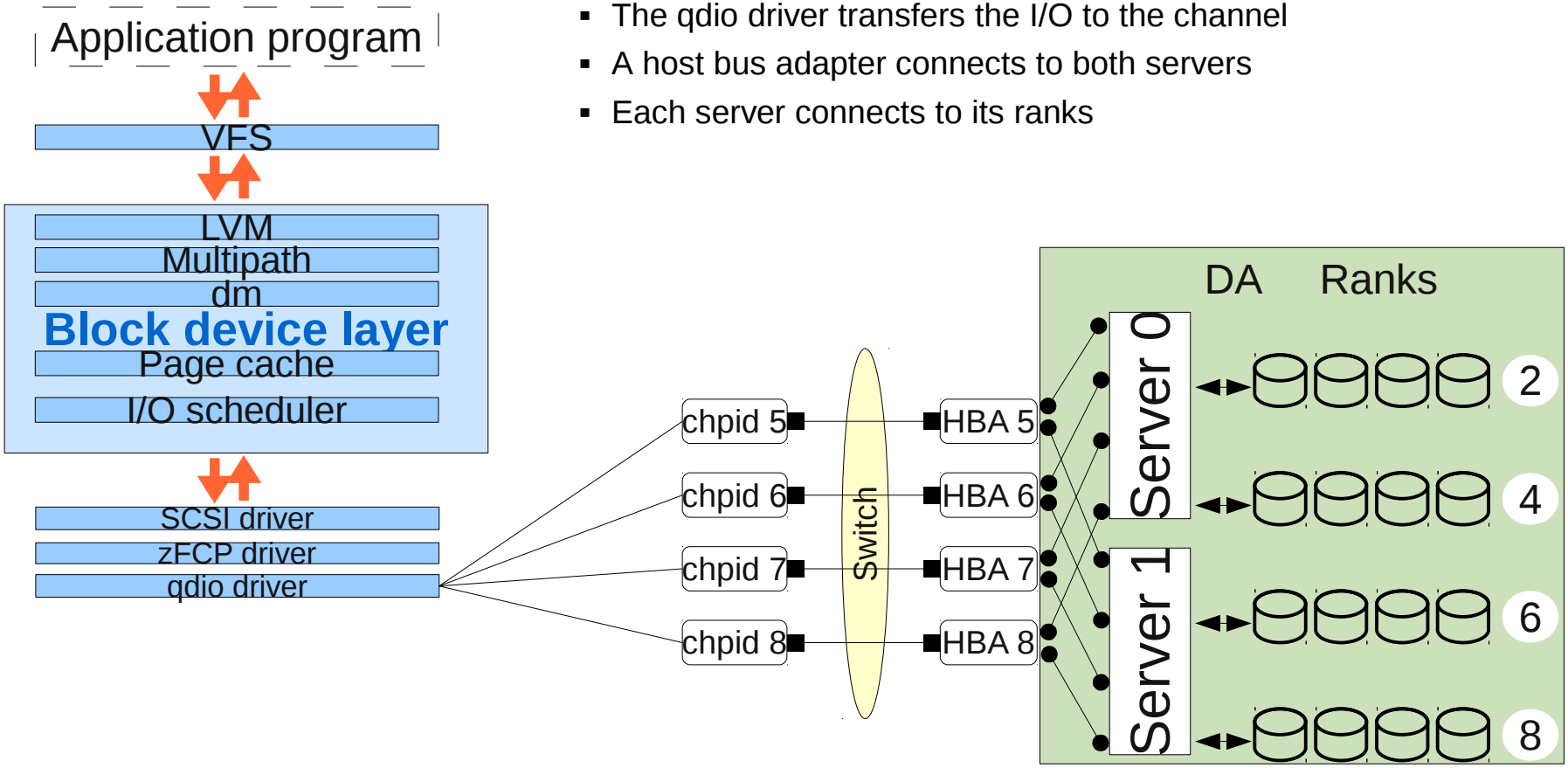
FICON/ECKD dasd I/O to a storage pool striped volume with HyperPAV

- 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



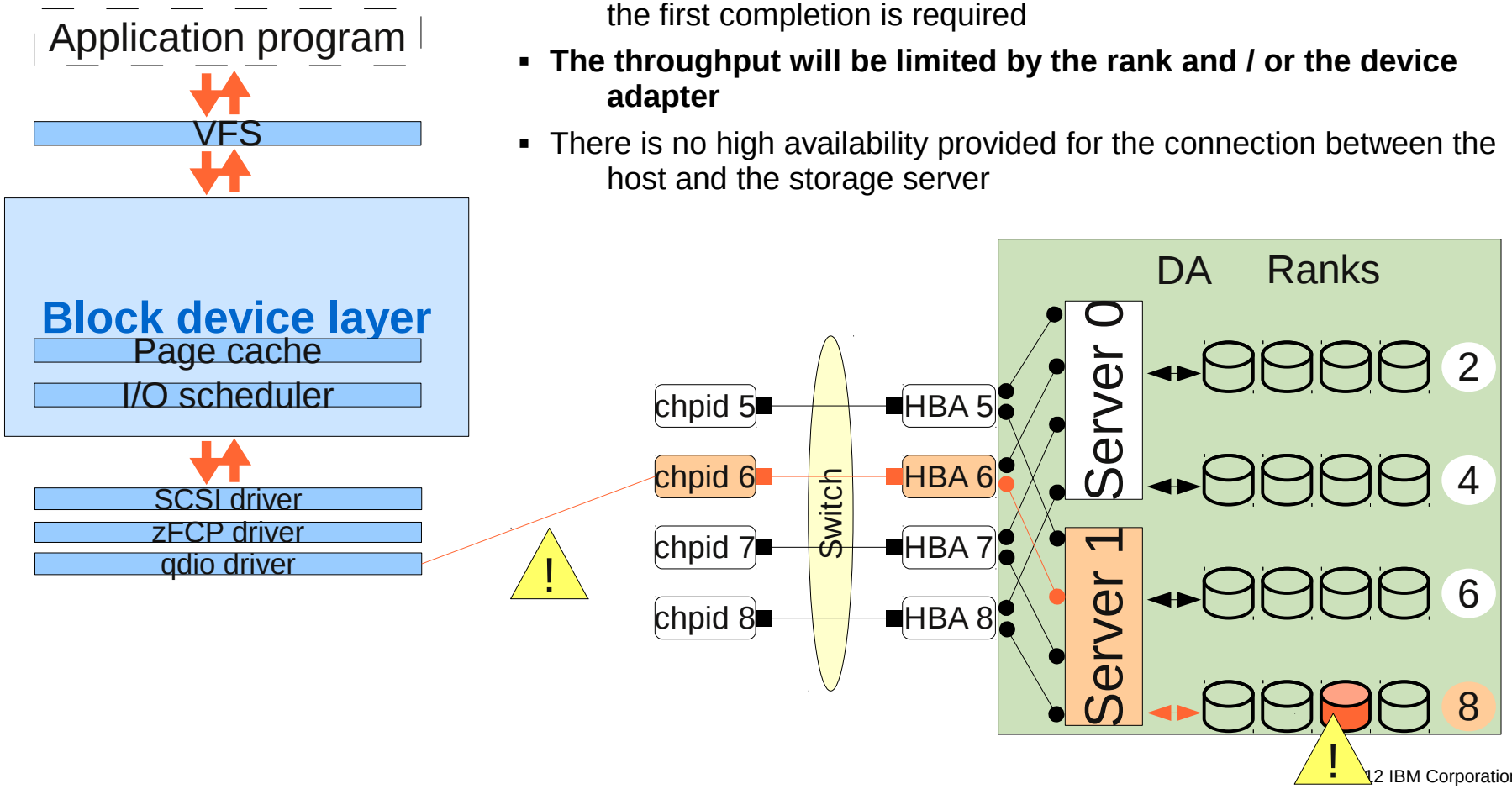
FCP/SCSI layout

- 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



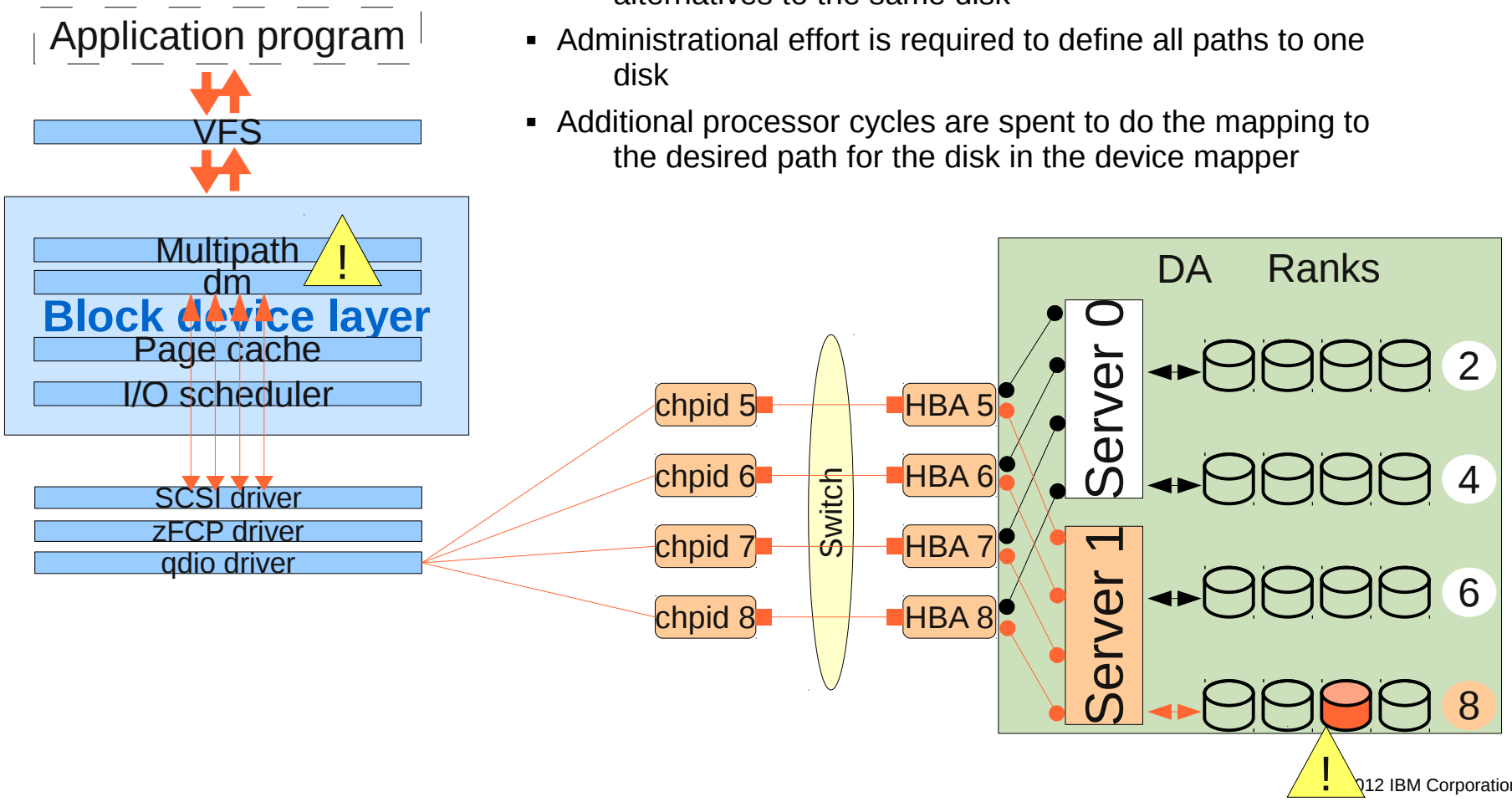
FCP/SCSI LUN I/O to a single disk

- Assume that disk 3 in rank 8 is reachable via channel 6 and host bus adapter 6
- 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



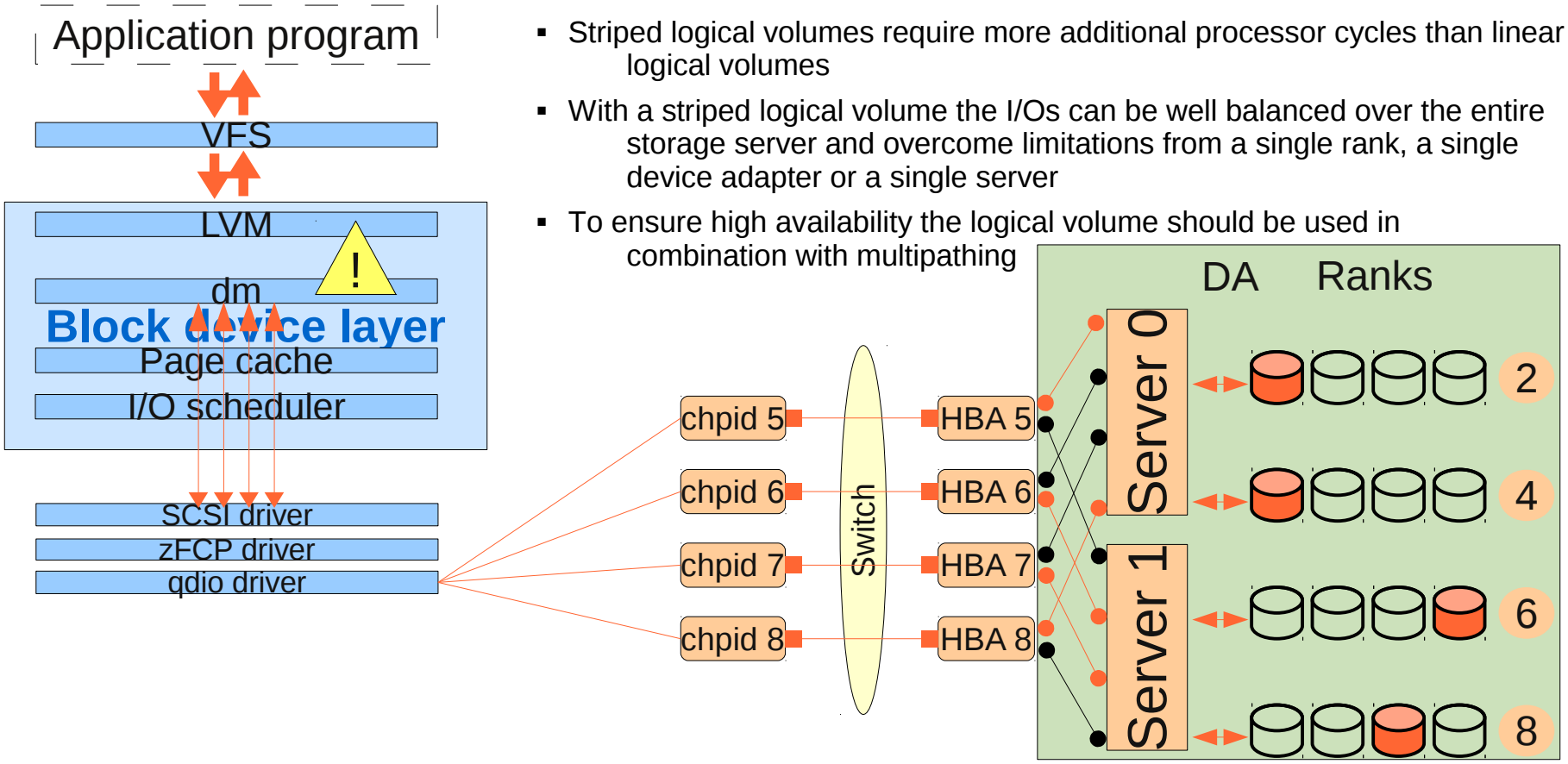
FCP/SCSI LUN I/O to a single disk with multipathing

- 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



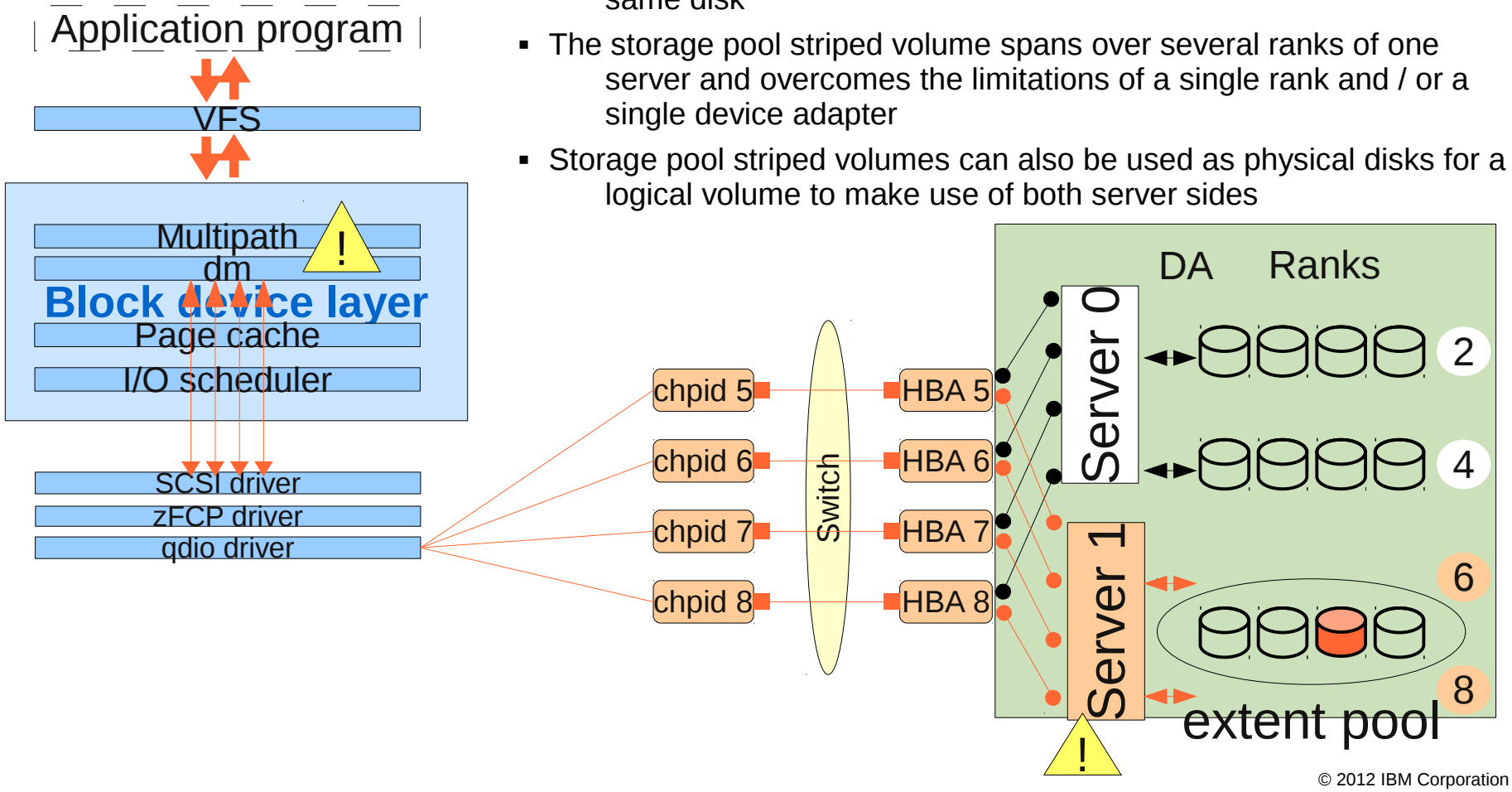
FCP/SCSI LUN I/O to a linear/striped logical volume

- 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 high availability the logical volume should be used in combination with multipathing



FCP/SCSI LUN I/O to 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 – 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 4 KiB
  - High availability by FICON path groups
  - Load balancing by FICON path groups and Parallel Access Volumes (PAV) or HyperPAV

- 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

Summary – Performance considerations

- Use more than 1 FICON Express channel

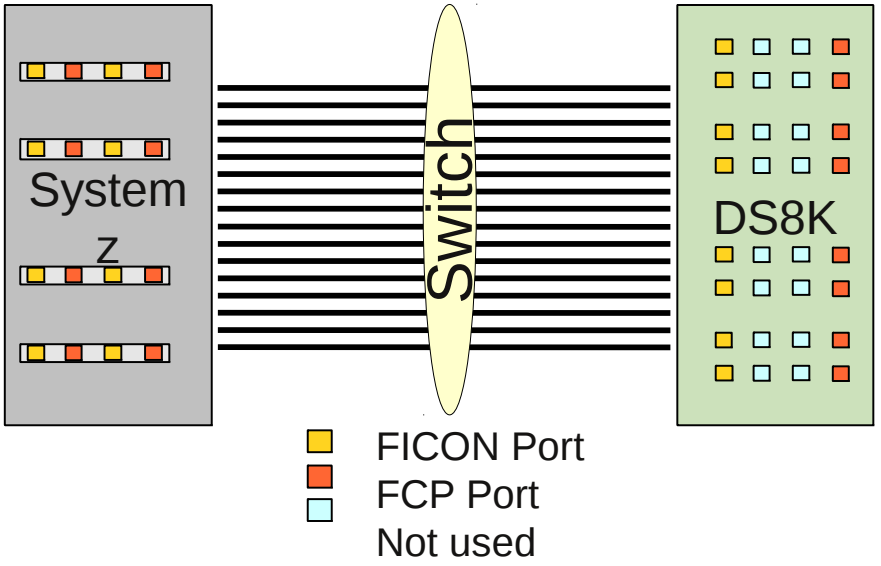
- Speed up techniques
  - Use more than 1 rank: storage pool striping, Linux striped logical volume
  - Use more channels for ECKD
    - Logical path groups
    - Use more than 1 subchannel: PAV, HyperPAV
    - High Performance FICON and Read Write Track Data
  - Use more channels for SCSI
    - SCSI Linux multipath multibus

Configuration – host and connection to storage server

z9 LPAR:
- 8 CPUs
- 512MiB
- 4 FICON Express4
  - 2 ports per feature used for FICON
  - 2 ports per feature used for FCP
  - Total 8 paths FICON, 8 paths FCP

DS8700:
- 8 FCION Express4
  - 1 port per feature used for FICON
  - 1 port per feature used for FCP

Linux:
- SLES11 SP1 (with HyperPAV, High Performance FICON)
- Kernel: 2.6.32.13-0.5-default (+dm stripe patch)
- Device-mapper:
  - multipath: version 1.2.0
  - striped: version 1.3.0
  - Multipath-tools-0.4.8-40.23.1
- 8 FICON paths defined in a channel path group



IOZone Benchmark description

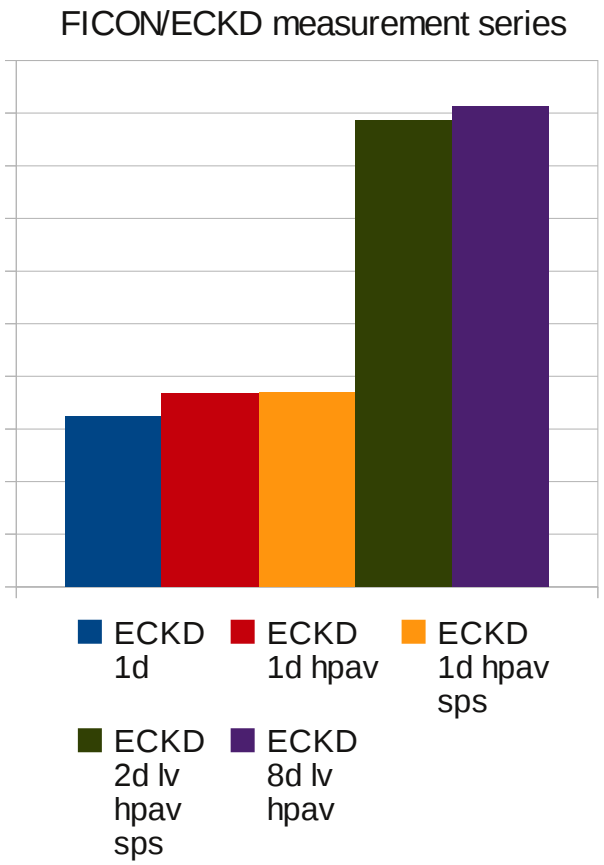
- Workload
  - Threaded I/O benchmark (IOzone)
  - Each process writes or reads a single file
  - Options to bypass page cache, separate execution of sequential write, sequential read, random read/write

- Setup
  - Main memory was restricted to 512 MiB
  - File size: 2 GiB, Record size: 8 KiB or 64 KiB
  - Run with 1, 8 and 32 processes
  - Sequential run: write, rewrite, read
  - Random run: write, read (with previous sequential write)
  - Runs with direct I/O and Linux page cache
  - Sync and drop caches prior to every invocation of the workload to reduce noise

Measured scenarios: FICON/ECKD

FICON/ECKD (always 8 paths)

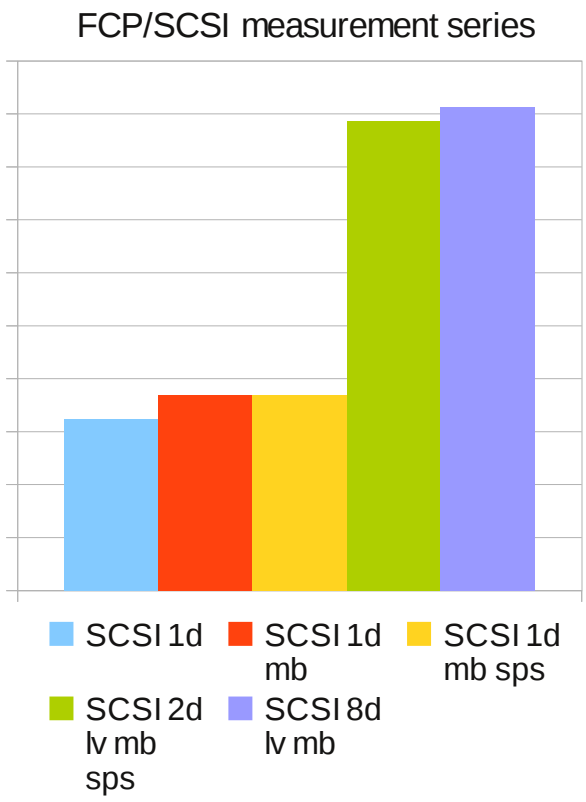
- 1 single disk
  - configured as 1 volume in a extent pool, containing 1 rank
- 1 single disk
  - configured as 1 volume in a extent pool, containing 1 rank
  - using 7 HyperPAV alias devices
- 1 storage pool striped disk
  - configured as 1 volume in a extent pool, containing 4 ranks
  - using 7 HyperPAV alias devices
- 1 striped logical volume
  - built from 2 storage pool striped disks, 1 from sever0 and 1 from server1
  - each disk
    - configured as 1 volume in a extent pool, containing 4 ranks
    - using 7 HyperPAV alias devices
- 1 striped logical volume
  - built from 8 disks, 4 from sever0 and 4 from server1
  - each disk
    - configured as 1 volume in a extent pool, containing 1 rank
    - using 7 HyperPAV alias devices



Measured scenarios: FCP/SCSI

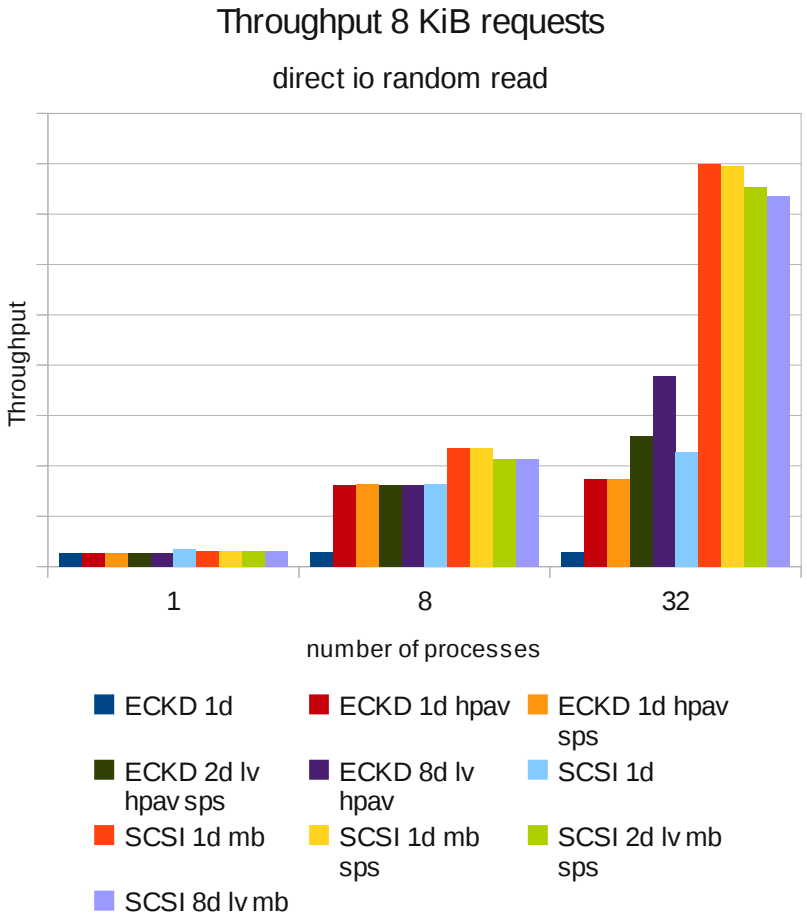
FCP/SCSI

- 1 single disk
  - configured as 1 volume in a extent pool, containing 1 rank
  - using 1 path
- 1 single disk
  - configured as 1 volume in a extent pool, containing 1 rank
  - using 8 paths, multipath multibus, rr_min_io = 1
- 1 storage pool striped disk
  - configured as 1 volume in a extent pool, containing 4 ranks
  - using 8 paths, multipath multibus, rr_min_io = 1
- 1 striped logical volume
  - built from 2 storage pool striped disks, 1 from sever0 and 1 from server1
  - each disk
    - configured as 1 volume in a extent pool, containing 4 ranks
    - using 8 paths, multipath multibus, rr_min_io = 1
- 1 striped logical volume
  - built from 8 disks, 4 from sever0 and 4 from server1
  - each disk
    - configured as 1 volume in a extent pool, containing 1 rank
    - using 8 paths, multipath multibus, rr_min_io = 1



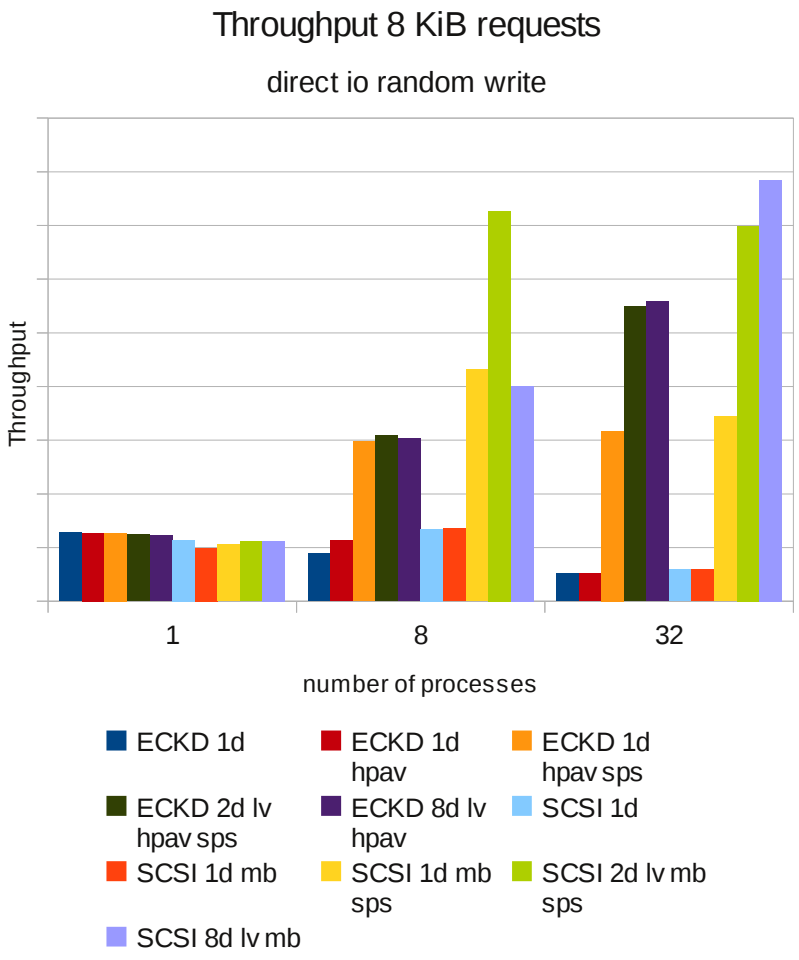
Database like scenario, random read

- ECKD
  - For 1 process the scenarios show equal throughput
  - For 8 processes HyperPAV improves the throughput by up to 5.9x
  - For 32 processes the combination Linux logical volume with HyperPAV dasd improves throughput by 13.6x
- SCSI
  - For 1 process the scenarios show equal throughput
  - For 8 processes multipath multibus improves throughput by 1.4x
  - For 32 processes multipath multibus improves throughput by 3.5x
- ECKD versus SCSI
  - **Throughput for corresponding scenario is always higher with SCSI**



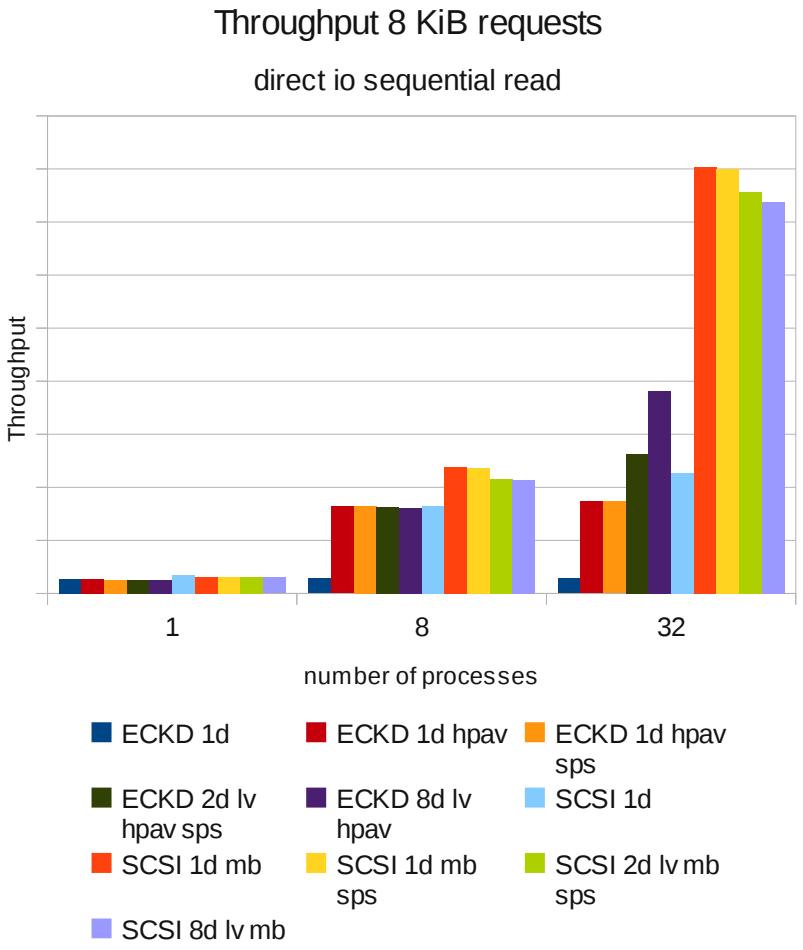
Database like scenario, random write

- ECKD
  - For 1 process the throughput for all scenarios show minor deviation
  - For 8 processes HyperPAV + storage pool striping or Linux logical volume improve the throughput by 3.5x
  - For 32 processes the combination Linux logical volume with HyperPAV or storage pool striped dasd improves throughput by 10.8x
- SCSI
  - For 1 process the throughput for all scenarios show minor deviation
  - For 8 processes the combination storage pool striping and multipath multibus improves throughput by 5.4x
  - For 32 processes the combination Linux logical volume and multipath multibus improves throughput by 13.3x
- ECKD versus SCSI
  - ECKD is better for 1 process
  - **SCSI is better for multiple processes**
- General
  - More NVS keeps throughput up with 32 processes



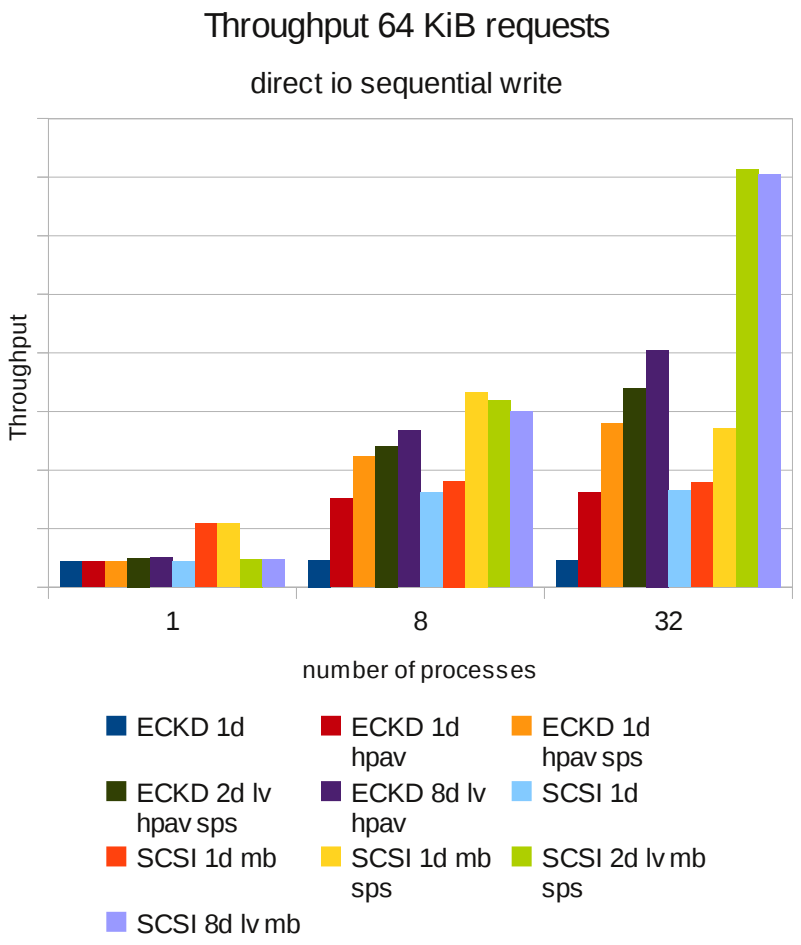
Database like scenario, sequential read

- ECKD
  - For 1 process the scenarios show equal throughput
  - For 8 processes HyperPAV improves the throughput by up to 5.9x
  - For 32 processes the combination Linux logical volume with HyperPAV dasd improves throughput by 13.7x
- SCSI
  - For 1 process the throughput for all scenarios show minor deviation
  - For 8 processes multipath multibus improves throughput by 1.5x
  - For 32 processes multipath multibus improves throughput by 3.5x
- ECKD versus SCSI
  - Throughput for corresponding scenario is always higher with SCSI
- General
  - Same picture as for random read



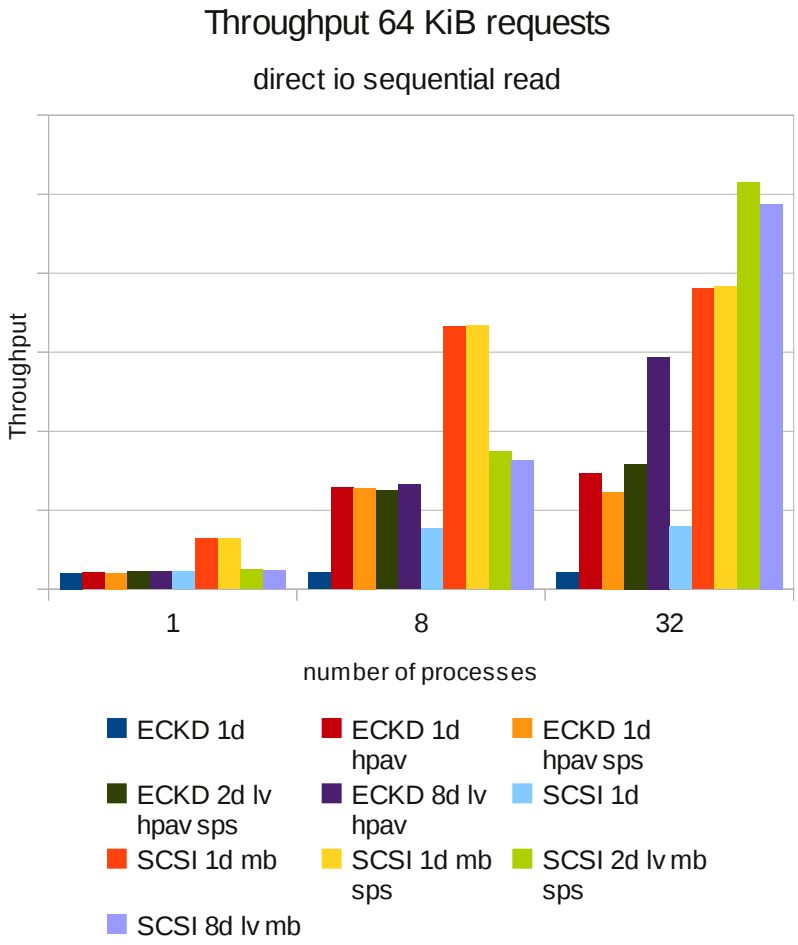
File server, sequential write

- ECKD
  - For 1 process the throughput for all scenarios show minor deviation
  - For 8 processes HyperPAV + Linux logical volume improve the throughput by 5.7x
  - For 32 processes the combination Linux logical volume with HyperPAV dasd improves throughput by 8.8x
- SCSI
  - For 1 process multipath multibus improves throughput by 2.5x
  - For 8 processes the combination storage pool striping and multipath multibus improves throughput by 2.1x
  - For 32 processes the combination Linux logical volume and multipath multibus improves throughput by 4.3x
- ECKD versus SCSI
  - For 1 process sometimes advantages for ECKD, sometimes for SCSI
  - **SCSI is better in most cases for multiple processes**



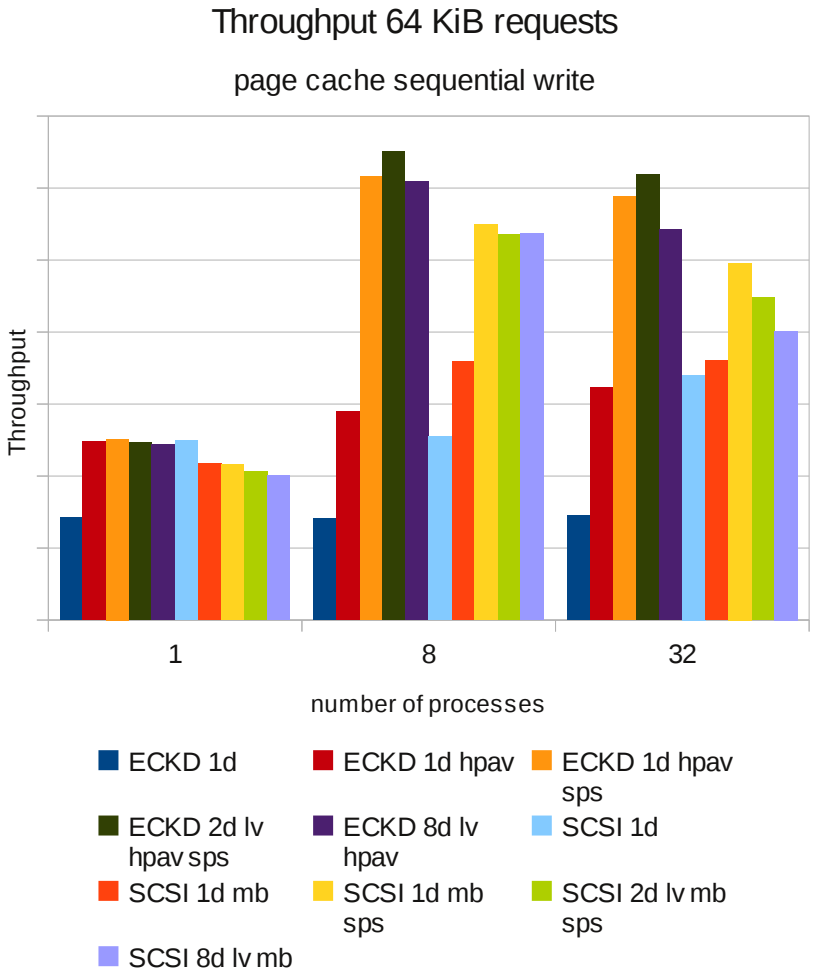
File server, sequential read

- ECKD
  - For 1 process the throughput for all scenarios show minor deviation
  - For 8 processes HyperPAV improves the throughput by up to 6.2x
  - For 32 processes the combination Linux logical volume with HyperPAV dasd improves throughput by 13.8x
- SCSI
  - For 1 process multipath multibus improves throughput by 2.8x
  - For 8 processes multipath multibus improves throughput by 4.3x
  - For 32 processes the combination Linux logical volume and multipath multibus improves throughput by 6.5x
- ECKD versus SCSI
  - SCSI is better in most cases



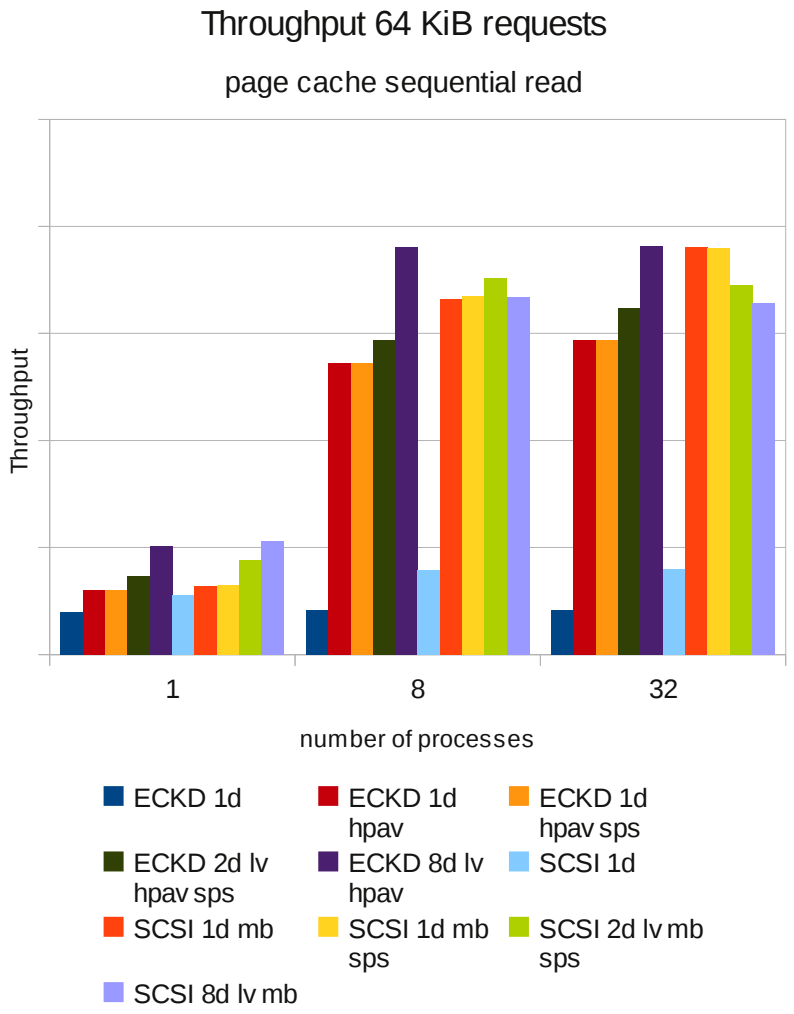
Effect of page cache with sequential write

- General
  - Compared to direct I/O:
    - Helps to increase throughput for scenarios with 1 or a few processes
    - Limits throughput in the many process case
  - Advanatage of SCSI scenarios with additional features no longer visible
- ECKD
  - HyperPAV,storage pool striping and Linux logical volume still improves throughput up to 4.6x
- SCSI
  - Multipath multibus with storage pool striping and/or Linux logical volume still improves throughput up to 2.2x



Effect of page cache with sequential read

- General
  - The SLES11 read ahead setting of 1024 helps a lot to improve throughput
  - Compared to direct I/O:
    - Big throughput increase with 1 or a few processes
    - Limits throughput in the many process case for SCSI
  - Advantage of SCSI scenarios with additional features no longer visible
    - The number of available pages in the page cache limits the throughput at a certain rate
- ECKD
  - HyperPAV,storage pool striping and Linux logical volume still improves throughput up to 9.3x
- SCSI
  - Multipath multibus with storage pool striping and/or Linux logical volume still improves throughput up to 4.8x



Summary

Linux options	Hardware options

▪ Choice of Linux distribution
▪ Appropriate number and size of I/Os
▪ File system
▪ Placement of temp files
▪ direct I/O or page cache
▪ Read ahead setting
▪ Use of striped logical volume
  • ECKD
  • HyperPAV
  • High Performance FICON for small I/O requests
  • SCSI
  • Single path configuration is not supported!
  • Multipath multibus – choose rr_min_io value
	▪ FICON Express4 or 8
	▪ number of channel paths to the storage server
	▪ Port selection to exploit maximum link speed
	▪ No switch interconnects with less bandwidth
	▪ Storage server configuration
	  • Extent pool definitions
	  • Disk placement
	  • Storage pool striped volumes

Conclusions

- Small sets of I/O processes benefit from Linux page cache in case of sequential I/O
- Larger I/O requests from the application lead to higher throughput
- Reads benefit most from using HyperPAV (FICON/ECKD) and multipath multibus (FCP/SCSI)
- Writes benefit from the NVS size
  - Can be increased by the use of
    - Storage pool striped volumes
    - Linux striped logical volumes to disks of different extent pools
- The results may vary depending on used
  - Storage servers
  - Linux distributions
  - Number of disks
  - Number of channel paths

CPU consumption

- Linux features, like page cache, PAV, striped logical volume or multipath consume additional processor cycles
- CPU consumption
  - grows with number of I/O requests and/or number of I/O processes
  - depends on the Linux distribution and versions of components like device mapper or device drivers
  - depends on customizable values as e.g. Linux memory size (and implicit page cache size), read ahead value, number of alias devices, number of paths, rr_min_io setting, I/O request size from the applications
  - is similar for ECKD and SCSI in the 1 disk case with no further options
- HyperPAV and static PAV in SLES11 consume less CPU cycles compared to static PAV in older Linux distributions
- The CPU consumption in the measured scenarios
  - has to be normalized to the amount of transferred data
  - differs between a simple and a complex setup
    - for ECKD up to 2x
    - for SCSI up to 2.5x

Questions?

Mario Held

Linux on System z
System Software Performance Engineer

IBM Deutschland Research & Development
Schoenaicher Strasse 220
71032 Boeblingen, Germany

Phone +49 (0)7031–16–4257
Email mario.held@de.ibm.com