Dr. Eberhard Pasch (epasch@de.ibm.com)

IBM

A Hitchhikers Guide to Linux Performance Issues

Trademarks

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

BlueMix	ECKD	IBM*	Maximo*	Smarter Cities*	WebSphere*	z Systems
BigInsights	FICON*	Ibm.com	MQSeries*	Smarter Analytics	XIV*	z/VSE*
Cognos*	FileNet*	IBM (logo)*	Performance Toolkit for VM	SPSS*	z13	z/VM*
DB2*	FlashSystem	IMS	POWER*	Storwize*	zEnterprise*	
DB2 Connect	GDPS*	Informix*	Quickr*	System Storage*	z/OS*	
Domino*	GPFS	InfoSphere	Rational*	Tivoli*		
DS8000*			Sametime*			

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Windows Server and the Windows logo are trademarks of the Microsoft group of countries.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

* Other product and service names might be trademarks of IBM or other 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. **All z13 numbers have been measured on pre GA hardware with pre GA software.**

Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here. All z13 numbers have been measured on pre GA hardware.

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.

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

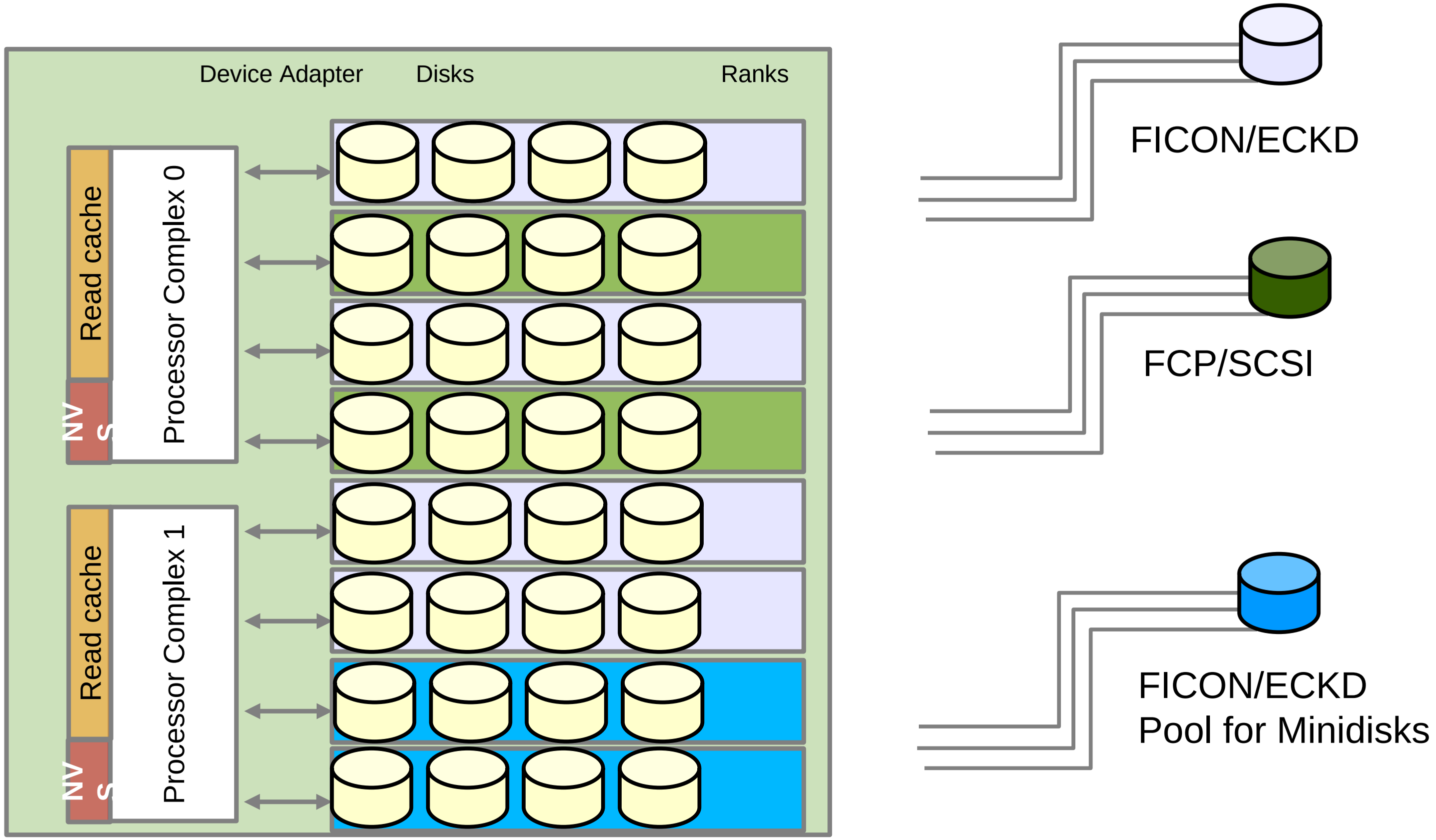
Agenda

▪ **Disk performance**	approximately 40% of external support requests
▪ Network performance	approximately 25% of external support requests
▪ Compiler	ISVs and RYO C/C++ applications
▪ Huge pages	beneficial in almost every huge installation
▪ Java	without basic tuning always a trouble maker

- In any environment from which we got support requests at least one of these areas was set up sub-optimally wasting performance or efficiency
  - So lets derive optimistically:
    "maybe those people following this guide never have significant issues"
  - Let us work on making you one of those

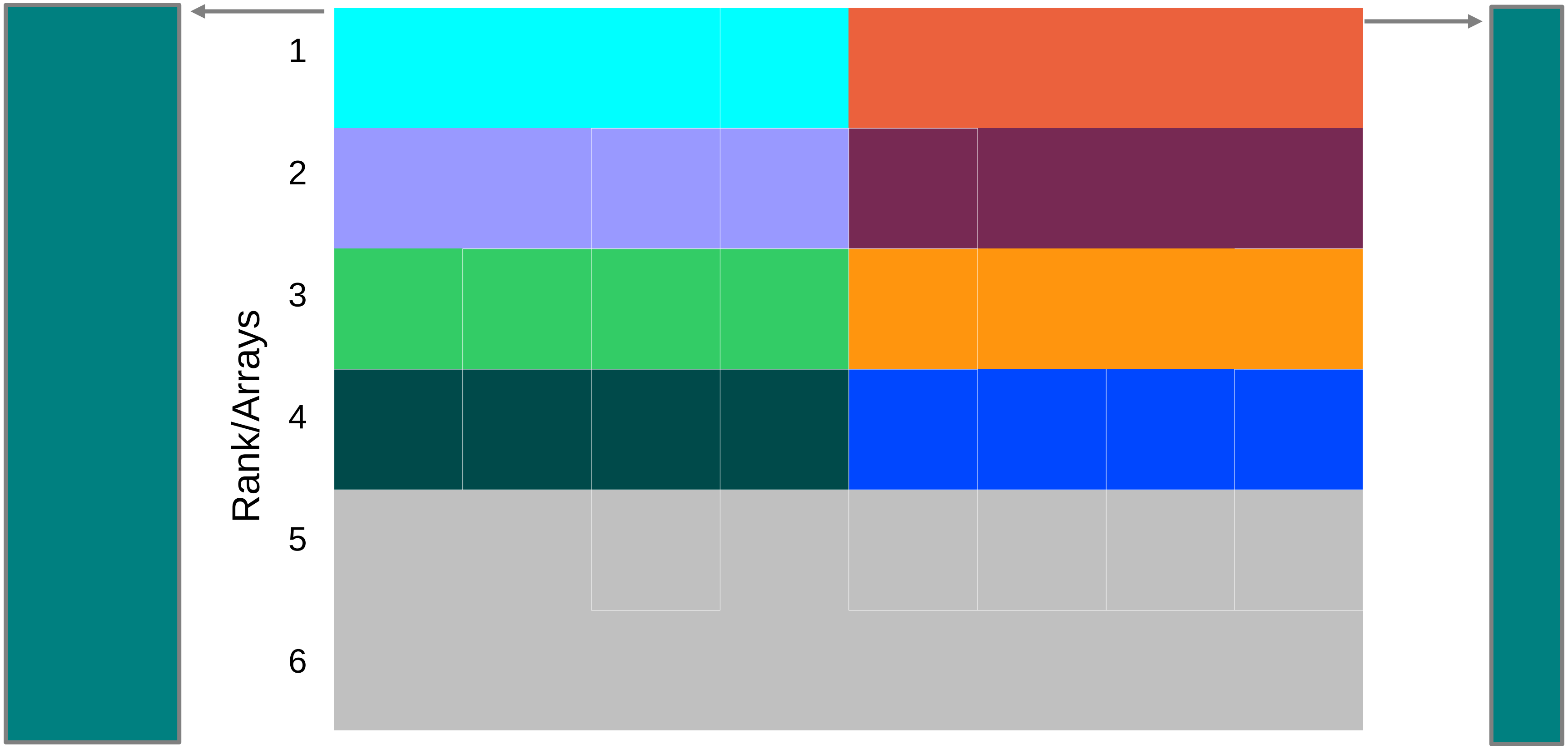
Disk I/O – Storage Server DS8x00

- Storage server basics – various configurations possible
  – Preferable many ranks into a extent pool with Storage Pool Striping (extents striped over all ranks within extent pool)



Disk I/O – Volumes

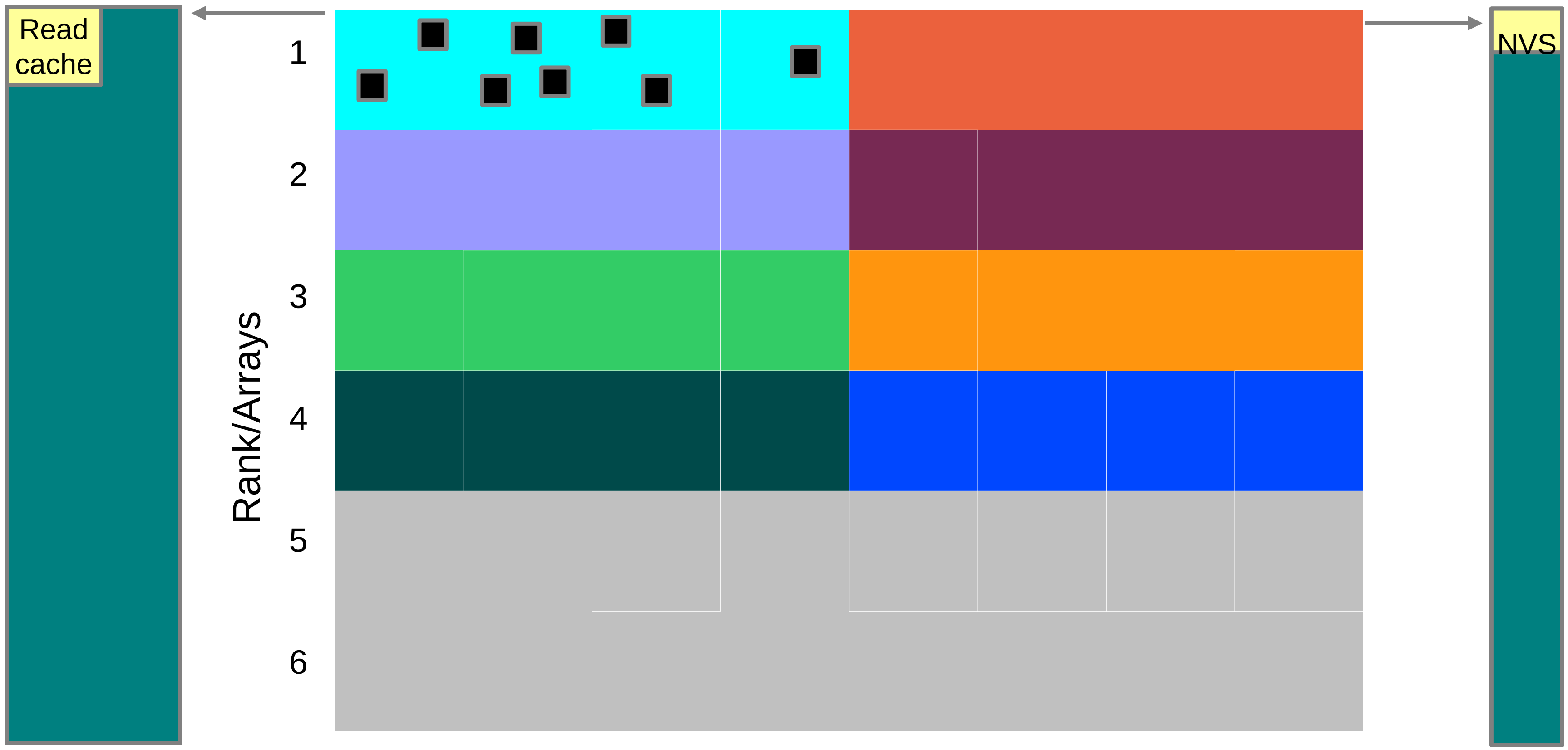
- Extent pool with 8 disks a 4 GB defined
  - Each rank has access to an adequate portion of the read cache and non-volatile storage (NVS – write cache)
- Example: random access to one volume
  - Usable portions of read cache and NVS very limited because just one rank is involved
  - Only one Device Adapter (DA) in use



Disk I/O – Volumes

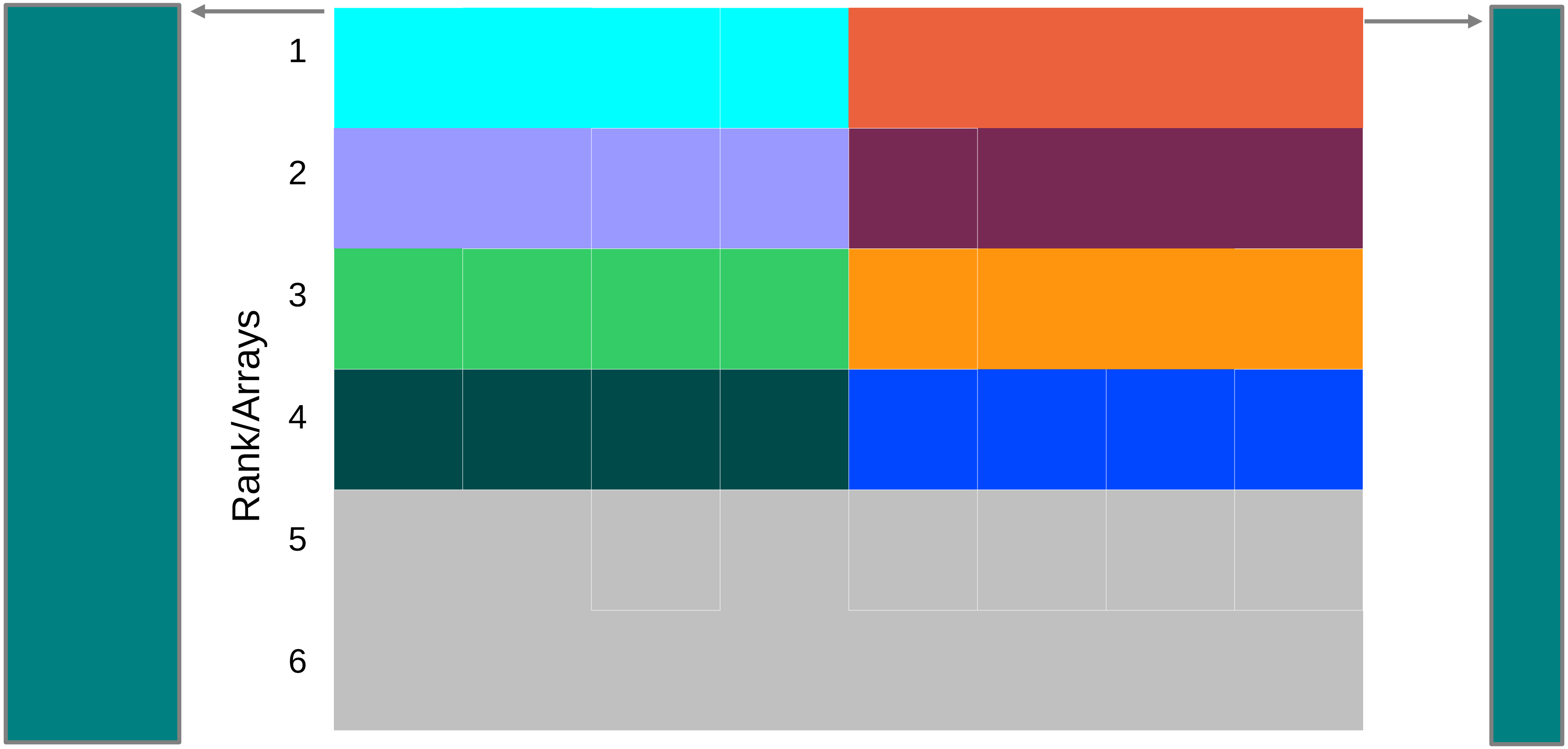
- Extent pool with 8 disks a 4 GB defined
  – Each rank has access to an adequate portion of the read cache and non-volatile storage (NVS – write cache)

- Example: random access to one volume
  – Usable portions of read cache and NVS very limited because just one rank is involved
  – Only one Device Adapter (DA) in use



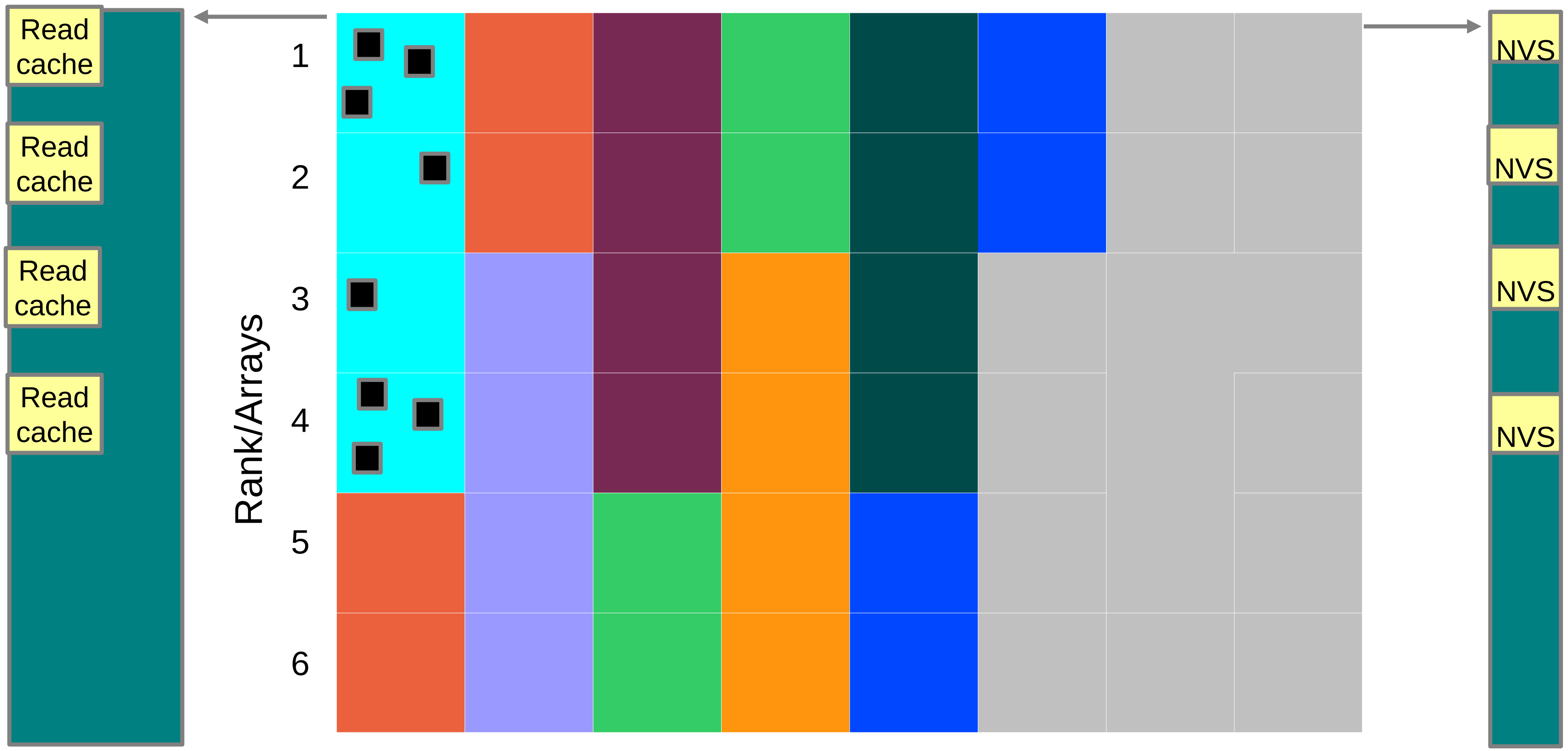
Disk I/O – Volumes with Storage Pool Striping (SPS)

- Extent pool example with 8 disks a 4 GB, with Storage Pool Striping (SPS)
  - Each rank has access to an adequate portion of the read cache and non-volatile storage (NVS – write cache)
- Example: random access to one SPS volume
  - Usable portions of read cache and NVS much bigger because four ranks are involved
  - Up to four Device Adapters (DA) are in use



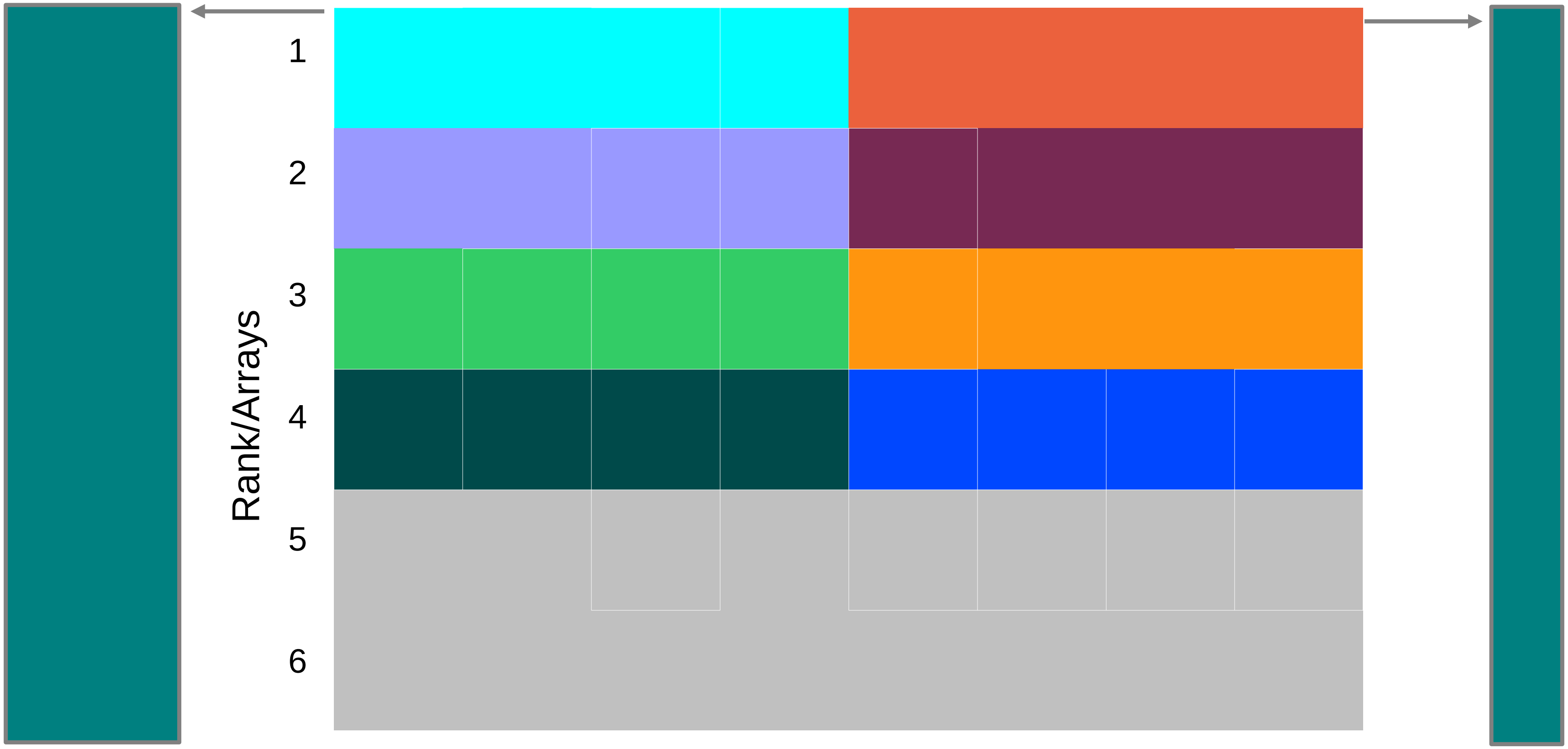
Disk I/O – Volumes with Storage Pool Striping (SPS)

- Extent pool example with 8 disks a 4 GB, with Storage Pool Striping (SPS)
  - Each rank has access to an adequate portion of the read cache and non-volatile storage (NVS – write cache)
- Example: random access to one SPS volume
  - Usable portions of read cache and NVS much bigger because four ranks are involved
  - Up to four Device Adapters (DA) are in use



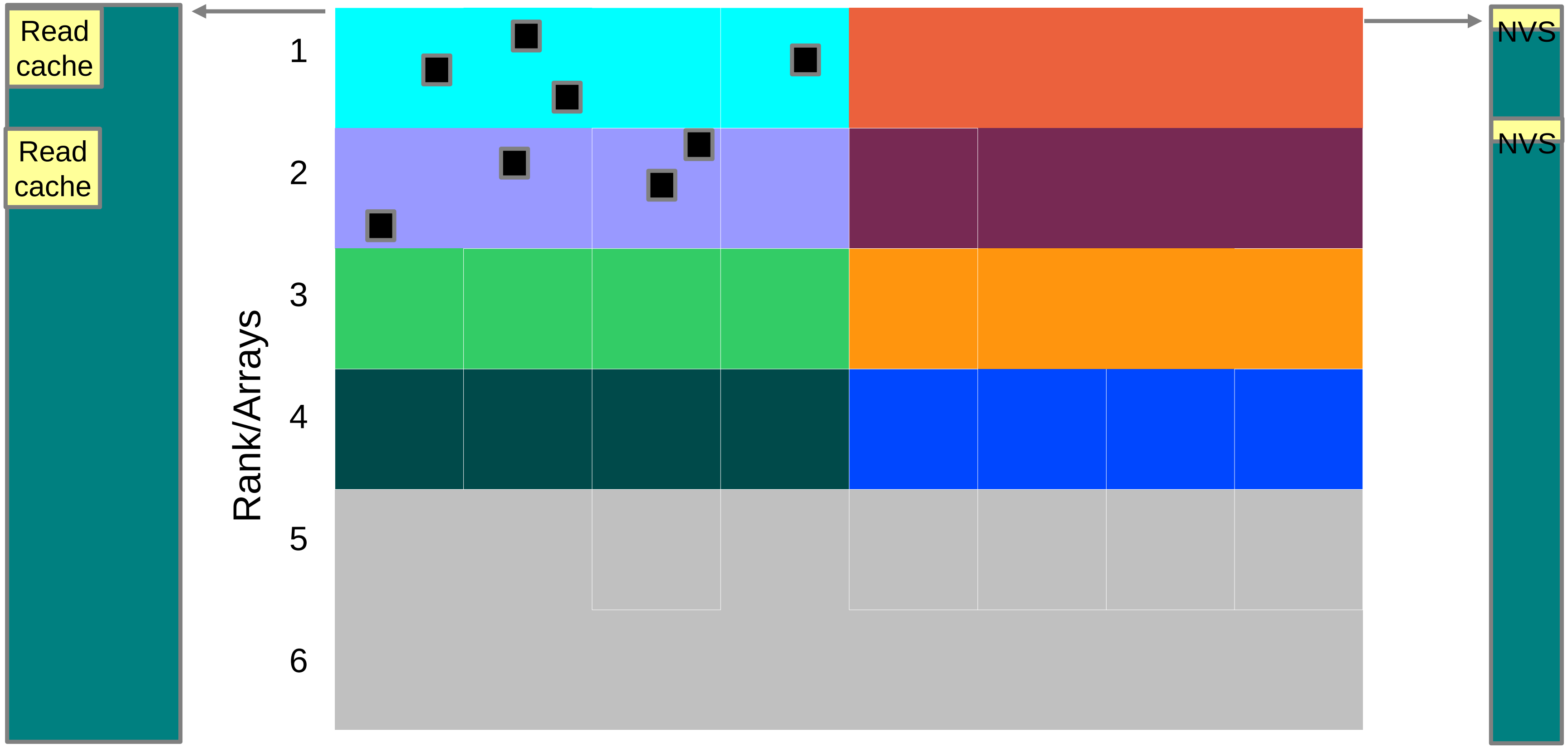
Disk I/O – two volumes in a striped LVM

- Extent pool example with 8 disks of 4 GB size
  - Each rank has access to an adequate portion of the read cache and non-volatile storage (NVS – write cache)
- Two volumes are used for the LVM
  - Usable portions of read cache and NVS very limited because only two ranks are involved
  - Up to two Device Adapters (DA) are used for the connection to cache and NVS



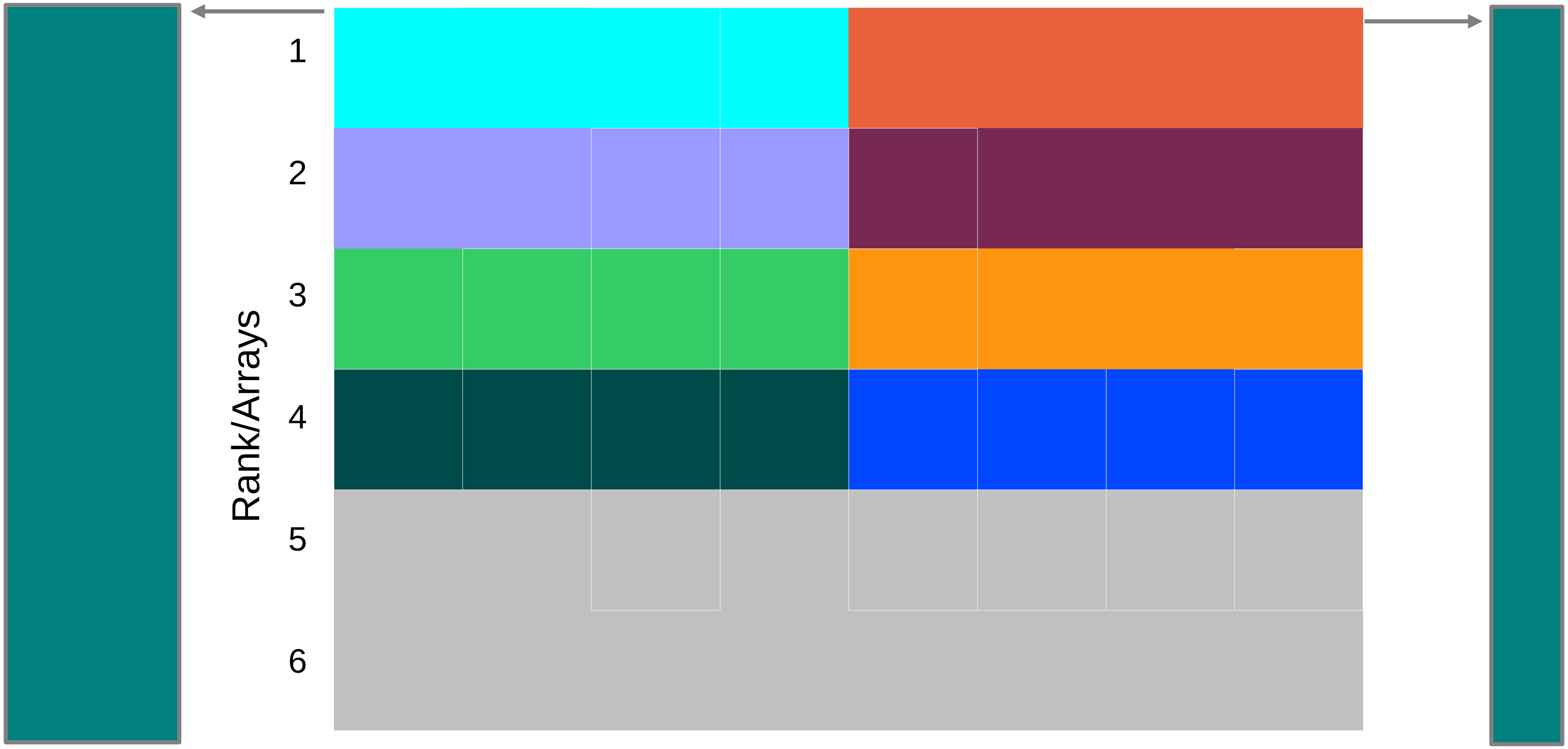
Disk I/O – two volumes in a striped LVM

- Extent pool example with 8 disks of 4 GB size
  - Each rank has access to an adequate portion of the read cache and non-volatile storage (NVS – write cache)
- Two volumes are used for the LVM
  - Usable portions of read cache and NVS very limited because only two ranks are involved
  - Up to two Device Adapters (DA) are used for the connection to cache and NVS



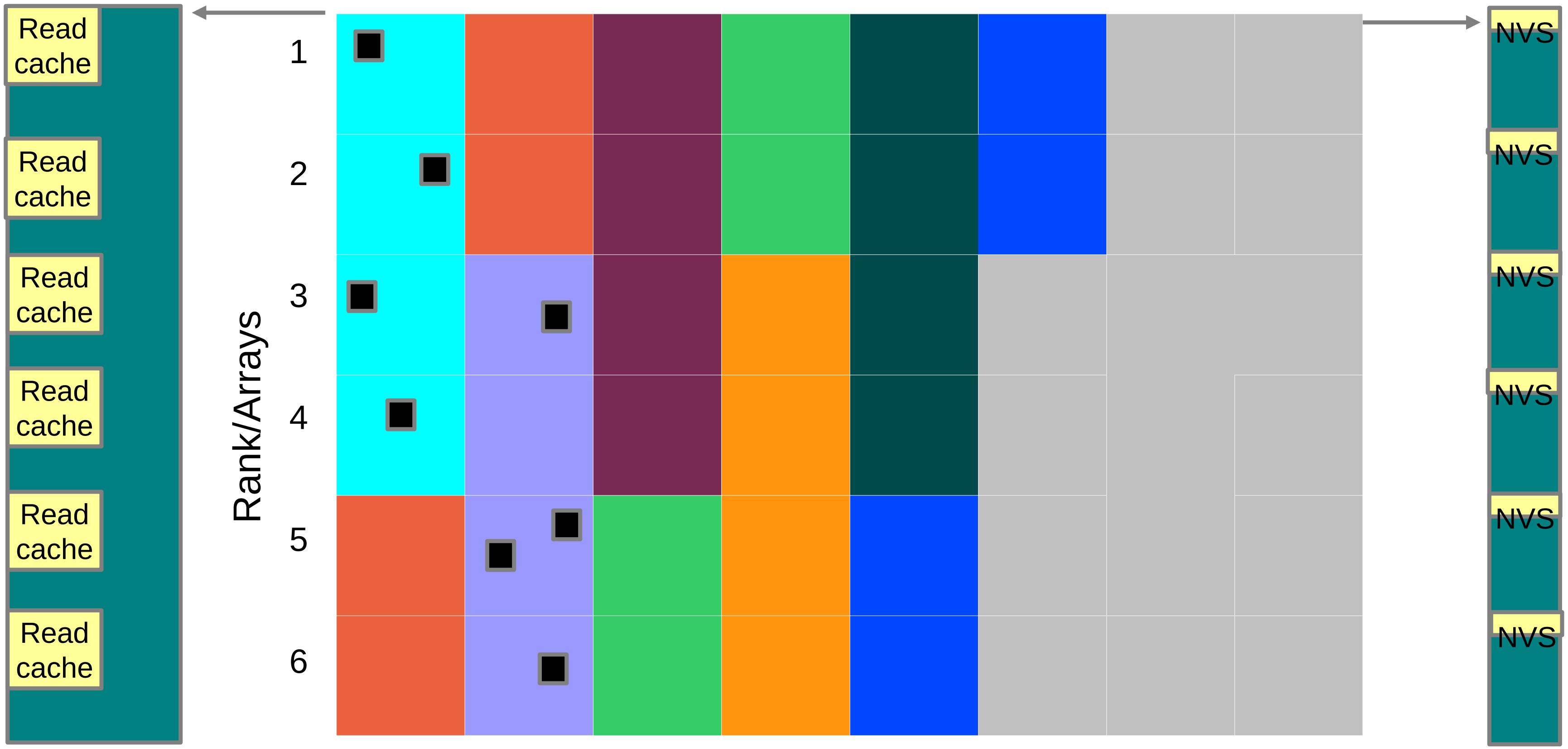
Disk I/O – two SPS volumes in a striped LVM

- Extent pool example with 8 disks a 4 GB, with Storage Pool Striping (SPS)
  - Each rank has access to an adequate portion of the overall amount of read cache and non-volatile storage (NVS – write cache)
- Two SPS volumes are used for the LVM
  - Usable portions of read cache and NVS much bigger because six ranks are involved
  - Up to six Device Adapters (DA) are used for the connection to cache and NVS



Disk I/O – two SPS volumes in a striped LVM

- Extent pool example with 8 disks a 4 GB, with Storage Pool Striping (SPS)
  - Each rank has access to an adequate portion of the overall amount of read cache and non-volatile storage (NVS – write cache)
- Two SPS volumes are used for the LVM
  - Usable portions of read cache and NVS much bigger because six ranks are involved
  - Up to six Device Adapters (DA) are used for the connection to cache and NVS



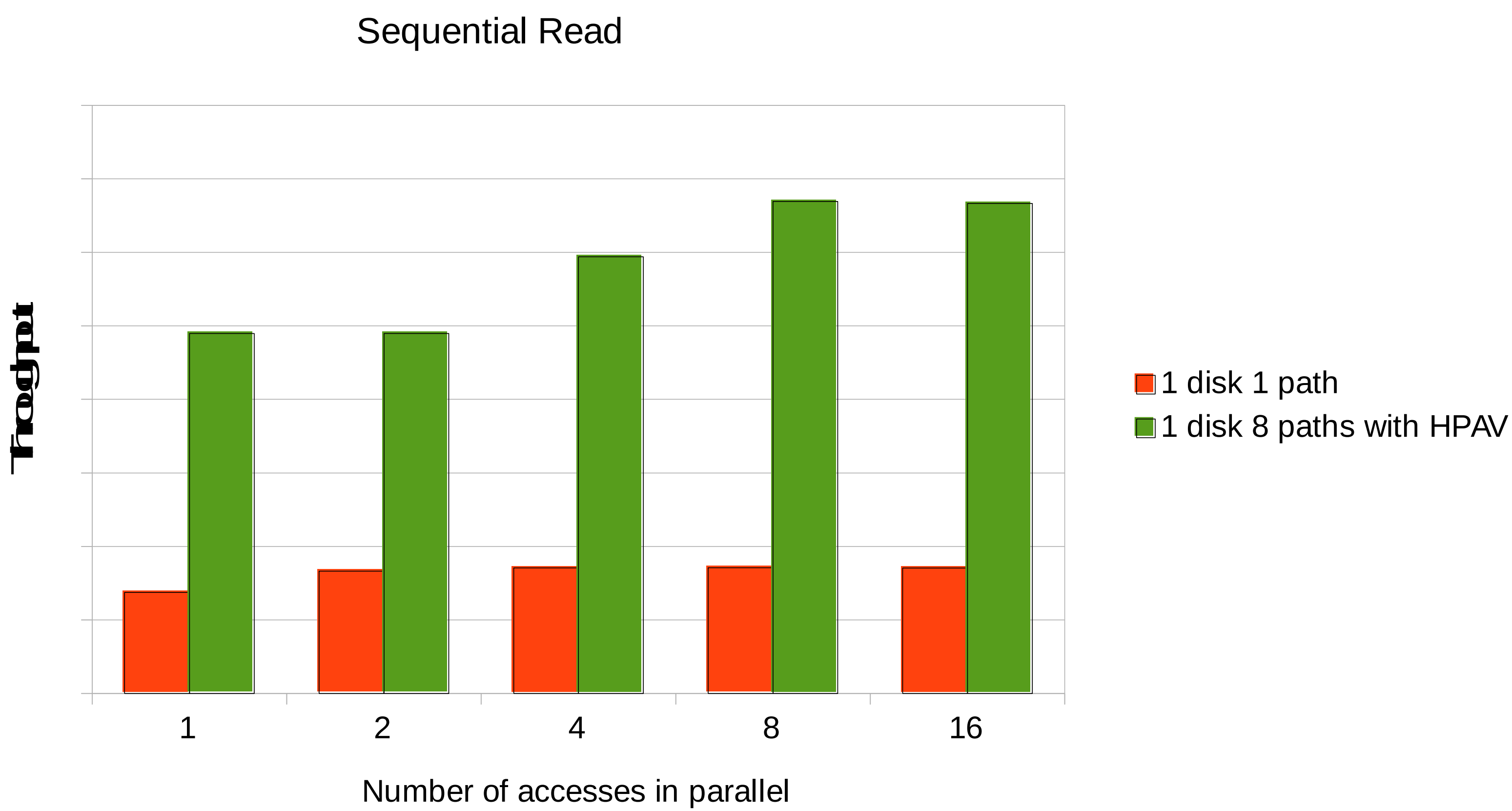
Disk I/O - striping options

- Striping is recommended and will result in higher throughput
  - Storage Pool Striped (SPS) disks with linear LV will perform better on many disk I/O processes
  - Device mapper striping on SPS disks will have good performance with few disk I/O processes

	Storage Pool Striping (SPS) or equivalent	Device mapper LV striping	No striping
Performance improvement	yes	yes	no
Processor consumption in Linux	no	yes	no
Complexity of administration	low	high	no

Disk I/O FICON / ECKD – number of paths in use

- Comparison of a single used subchannel to HyperPAV
  – Multiple (in example eight) paths perform much better
  – For reliable production systems you should use a multipath setup



Disk I/O FICON / ECKD – number of paths in use (cont.)

- iostat comparison (case 16 jobs in parallel)

```
...
04/10/14 23:52:20
Device:         rrqm/s   wrqm/s     r/s     w/s     rkB/s     wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
dasda             0.00     0.20    0.00    0.20      0.00      1.60    16.00     0.00    0.00    0.00    0.00   0.00   0.00
dasdb             0.00     0.00    0.00    0.00      0.00      0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dasdc          2830.60     0.00  750.60    0.00 340915.20      0.00   908.38    36.06   48.03   48.03    0.00   1.33 100.00
…
```

```
...
04/11/14 01:15:31
Device:         rrqm/s   wrqm/s     r/s     w/s     rkB/s     wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
dasda             0.00     0.00    0.00    0.00      0.00      0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dasdb             0.00     0.00    0.00    0.00      0.00      0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dasdc         10243.20     0.00 2700.40    0.00 1229968.00      0.00   910.95    32.87   12.16   12.16    0.00   0.34  92.20
...
```

Disk I/O FICON / ECKD – number of paths in use (cont.)

- DASD statistics comparison (case 16 accesses in parallel)

- One CCW program must be finished before the next can executed in one path case
  - DASD driver queue size limited to maximal five entries
    - First table shows the distribution in statistics of one to five requests queued

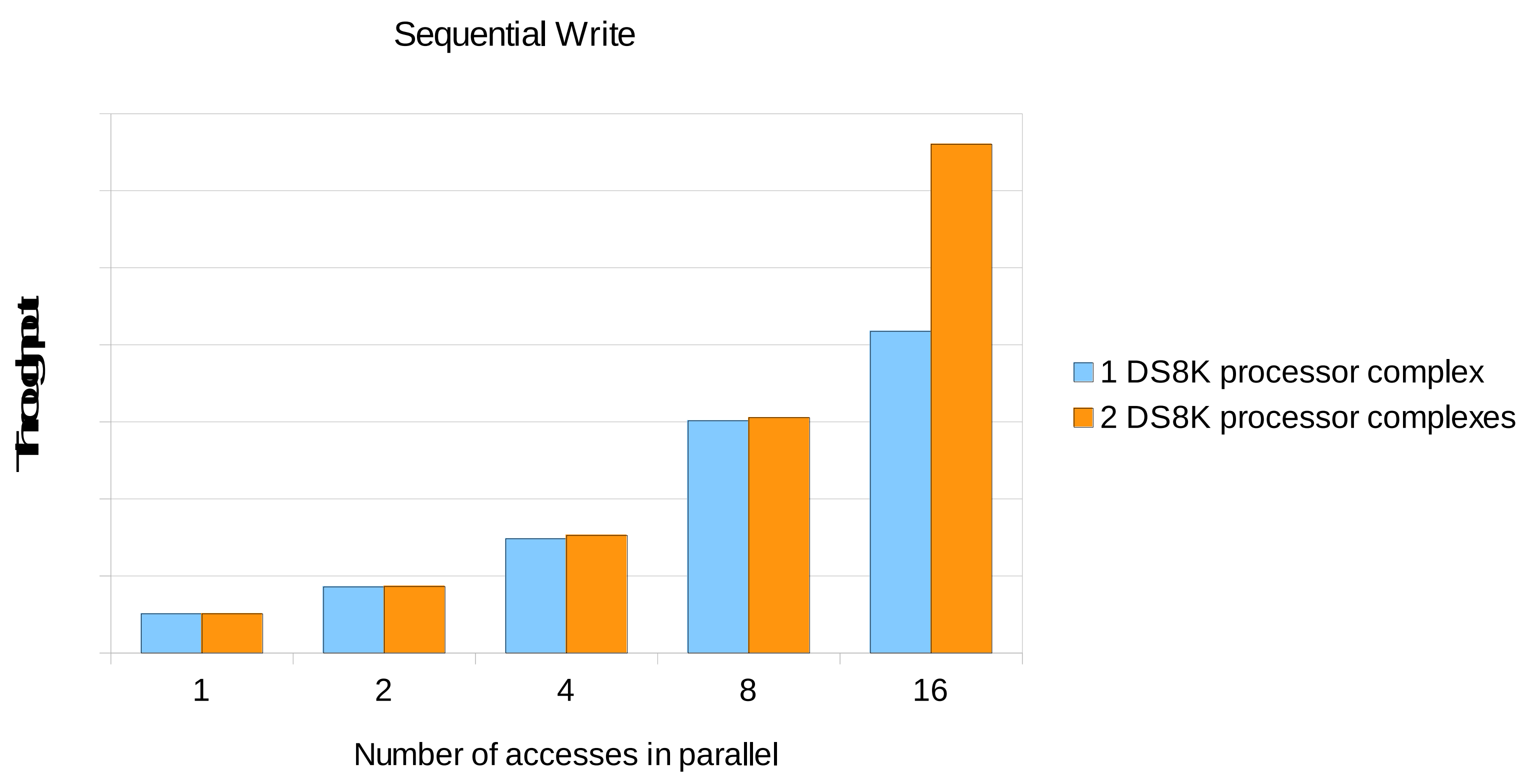
- When more paths are used the requests gets distributed and parallel execution is possible
  - No more limitation to maximal five entries
    - Second table shows a distribution in statistics with up to seventeen requests queued
    - Most of the time eight to twelve requests queued

```
14513 dasd I/O requests
with 13108456 sectors(512B each)
Scale Factor is  1
 __<4   ___8  __16   __32  __64  _128  _256  _512  __1k  __2k  __4k  __8k  _16k  _32k  _64k  128k
 _256   _512  __1M   __2M  __4M  __8M  _16M  _32M  _64M  128M  256M  512M  __1G  __2G  __4G  _>4G



# of req in chanq at enqueuing (1..32)
    0    29  5396  7643  1445     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

```
...
# of req in chanq at enqueuing (1..32)
    0    14     8    28    95    85   181  1265  2958  3329  3755  1796   620   126    28    18
    9     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
...
```

Disk I/O FICON / ECKD – usage of DS8K processor complexes

- Comparison one DS8K processor complex versus both processor complexes with LVM and HyperPAV
  - Recommendation if throughput matters: redistribute workload over both processor complexes
  - Write performance depends on available non-volatile write cache (NVS)



Disk I/O FICON / ECKD – usage of DS8K processor complexes (cont.)

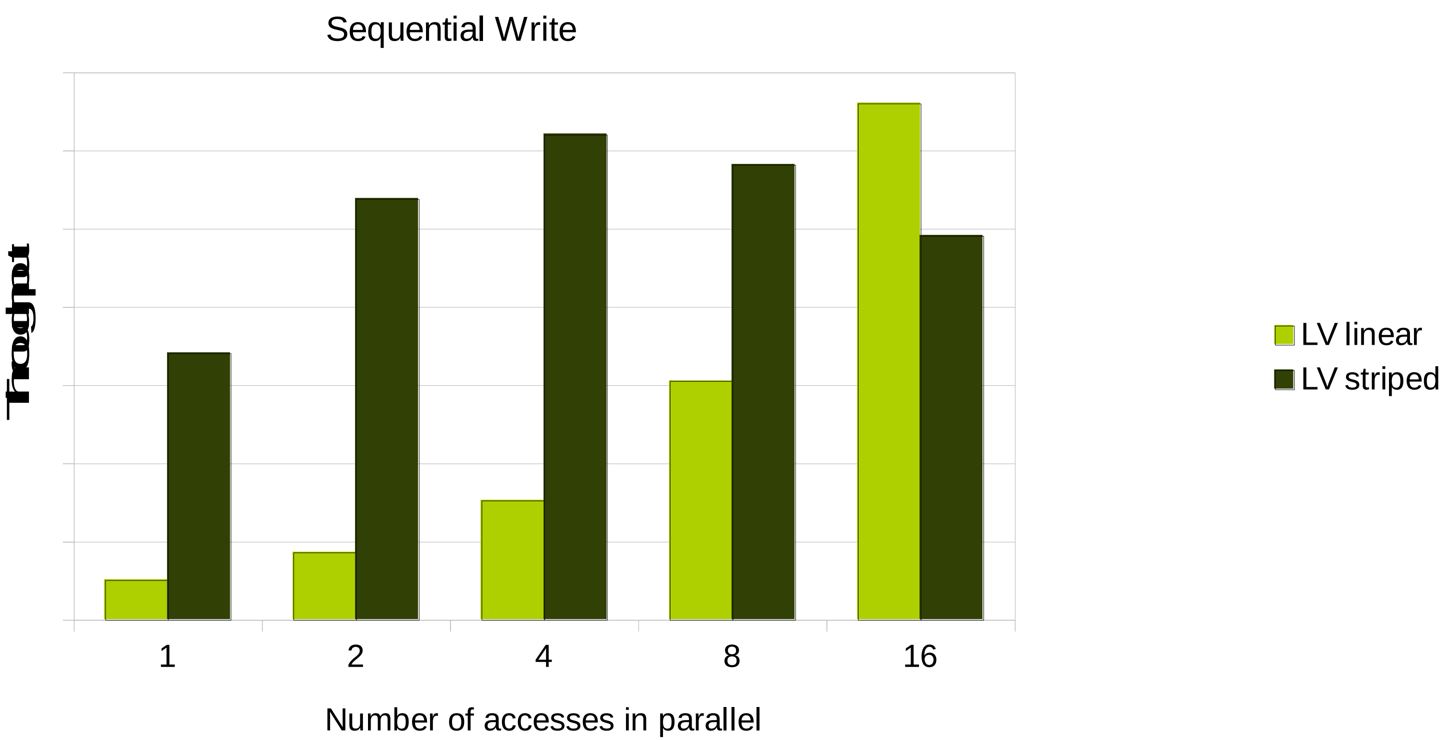
- Run iostat using command "`iostat -xtdk 10`"
- iostat results for sequential write using one DS8K processor complex compared to both processor complexes (16 streams write in parallel)
  - Much more throughput for both processor complexes with more NVS available
  - Less await and service time with both processor complexes

```
04/11/14 04:29:07
Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
dasda             0.00     0.20    0.00    0.20     0.00     1.60    16.00     0.00    0.00    0.00    0.00   0.00   0.00
...
...
dasddz            0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-0              0.00     0.00    0.00 15577.60     0.00 1482777.60   190.37   139.00    9.41    0.00    9.41   0.06 100.00
...
```

```
04/11/14 20:58:22
Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
dasda             0.00     0.00    0.00    0.20     0.00     0.80     8.00     0.00    0.00    0.00    0.00   0.00   0.00
...
...
dm-0              0.00     0.00    0.00 33563.60     0.00 3194752.00   190.37   161.00    4.80    0.00    4.80   0.03  98.60
```

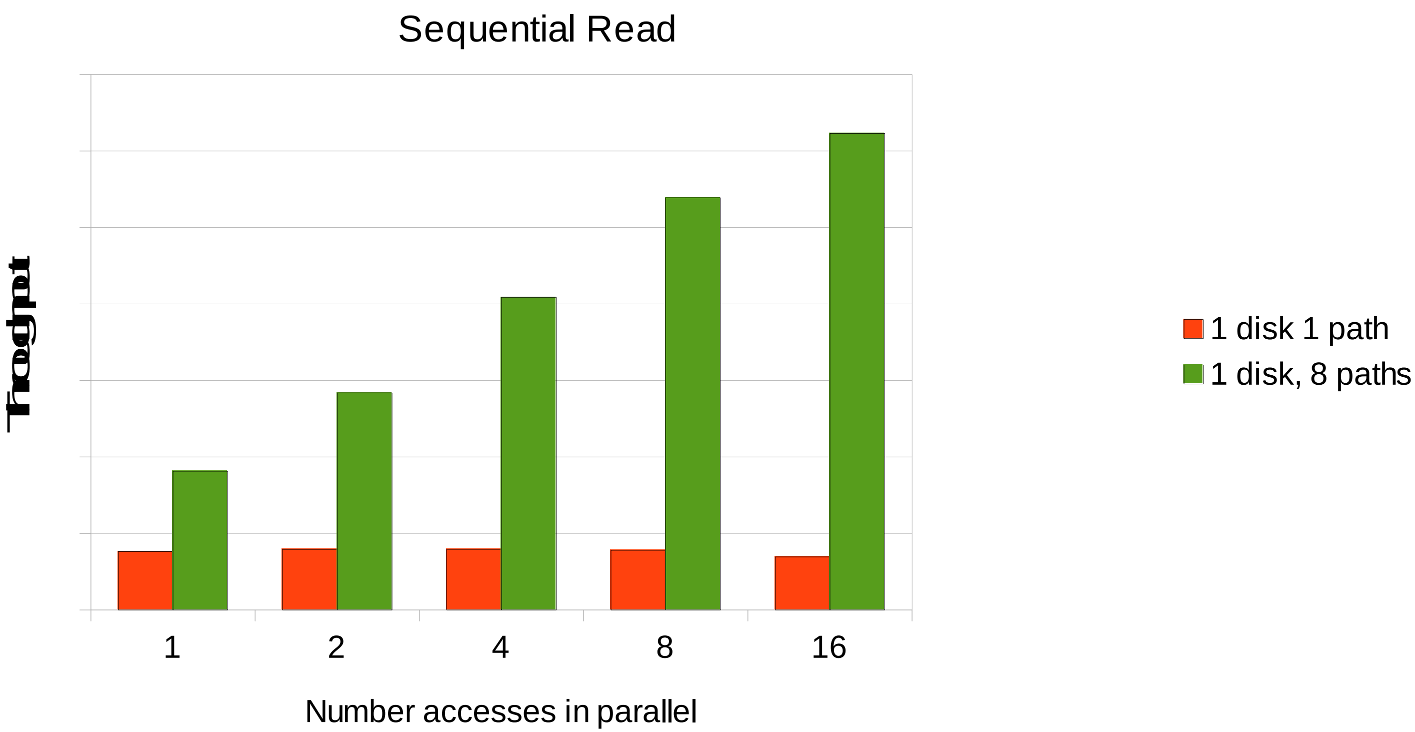
Disk I/O FICON / ECKD - LVM linear versus LVM striped

- Comparison Logical Volume linear versus Logical Volume striped
  - Much more parallelism when using striping with a few jobs running
  - Striping with sizes of 32kiB / 64 kiB may split up single big I/Os (bad)
    - This applies especially to sequential workloads where read-ahead scaling take place
  - Striping adds extra effort / processor consumption to the system
    - Eventually can consume the benefits of striping by cpu induced latencies



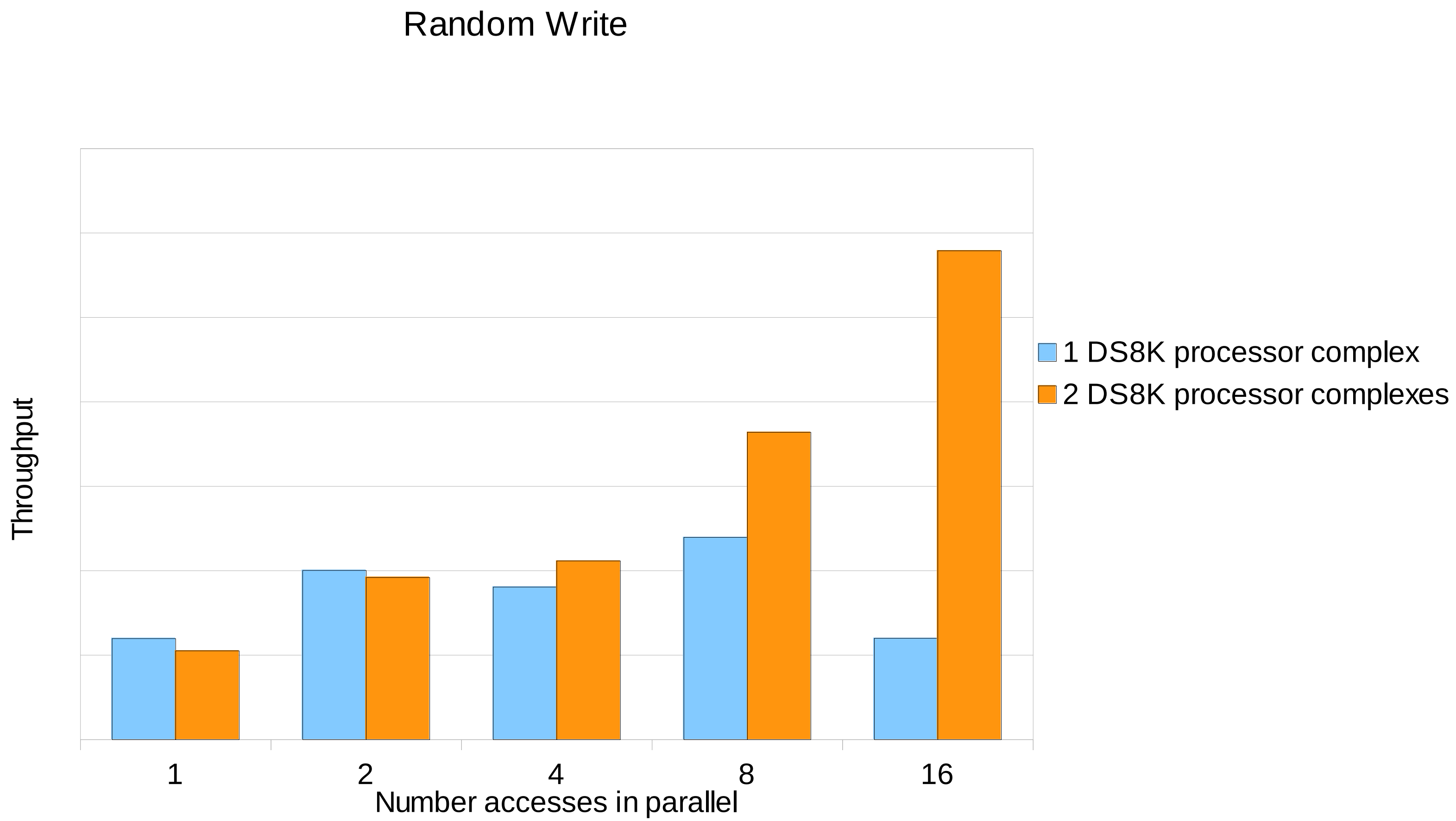
Disk I/O FCP / SCSI – number of paths in use

- Comparison single path setup to many paths
  - Multipath solution allows much more throughput
    - Multipath requires some extra processor cycles
  - Similar behavior to comparison single subchannel versus HyperPAV with ECKD / FICON
- **For reliable production systems you should use a multipath setup anyway**
  - Failover does not increase the capacity available to a path group, while multibus does



Disk I/O FCP / SCSI - usage of DS8K processor complexes

- Comparison usage of one processor complex versus both processor complexes with LVM
  - Usage of both processor complexes has an advantage if NVS became the limiting factor



Disk I/O – more tuning options

- Use latest hardware if throughput is important
  - Currently FICON Express 16S
- Use direct I/O and asynchronous I/O
  - Requires support by your used software products
  - More throughput at less processor consumption
  - In most cases advantageous if combined
- Use advanced FICON/ECKD techniques such as
  - High Performance FICON
  - Read Write Track Data
- Use the FCP/SCSI datarouter technique for further speedup (~5-15%)
  - Kernel parmline zfcp.datarouter=1, default on in more recent distribution releases
  - Requires 8S cards or newer
    - Feature similar to the store-forward architecture of recent OSA Cards
  - Allows the driver to avoid extra buffering in the card
    - No in card buffering also means there can't be a stalling buffer shortage

Disk I/O – performance considerations summary

- Use as much paths as possible
  – ECKD logical path groups combined with HyperPAV
  – SCSI Linux multipath multibus
- Use all advanced software, driver and Hardware features
- Storage Server
  – Use Storage Pool Striping (SPS) as a convenient tool
  – Define extent pools spanning over many ranks
  – Use both storage server complexes of the storage server (DS8x00)
- If you use Logical Volumes (LV)
  – Linear: with SPS and random access
  – Linear: with SPS and sequential access and many processes
  – Striped: for special setups that proved to be superior to SPS

- **So long story short let nothing idle and use all you got**

Agenda

▪ Disk performance	approximately 40% of external support requests
▪ **Network performance**	approximately 25% of external support requests
▪ Compiler	ISVs and RYO C/C++ applications
▪ Huge pages	beneficial in almost every huge installation
▪ Java	without basic tuning always a trouble maker

Network performance tuning

- It's not that hard actually...

```
net.core.netdev_max_backlog = 25000
net.ipv4.tcp_dsack = 1
net.ipv4.tcp_sack = 1
net.core.somaxconn = 1024
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_max_syn_backlog = 10000
net.ipv4.ip_local_port_range = 15000 65000
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_fin_timeout = 1
net.ipv4.tcp_rmem = 4096 87380 524288
net.core.rmem_max = 524288
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.core.wmem_max = 524288
net.ipv4.tcp_wmem = 4096 16384 524288
```

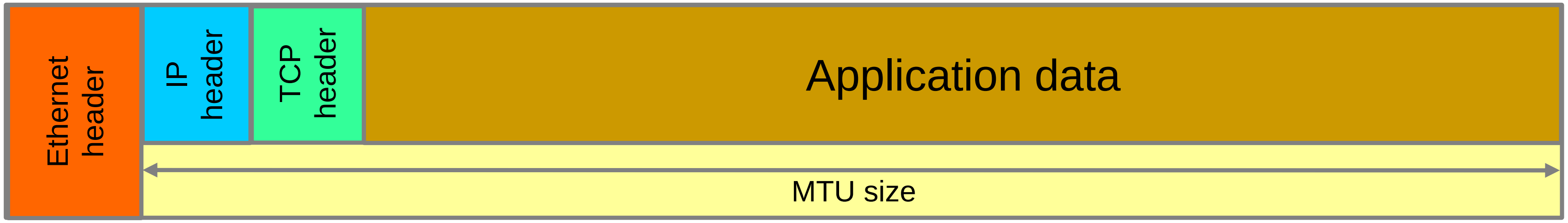
Network performance tuning

- But seriously...
- We won't go into all the gritty details here
  - Instead, we're going to introduce you to the concepts you can use to improve your network performance
  - If you really want to get into all the details (and especially *how* to do it), there are slides that go into that in the appendix of this presentation

Tuning parameters - MTU size

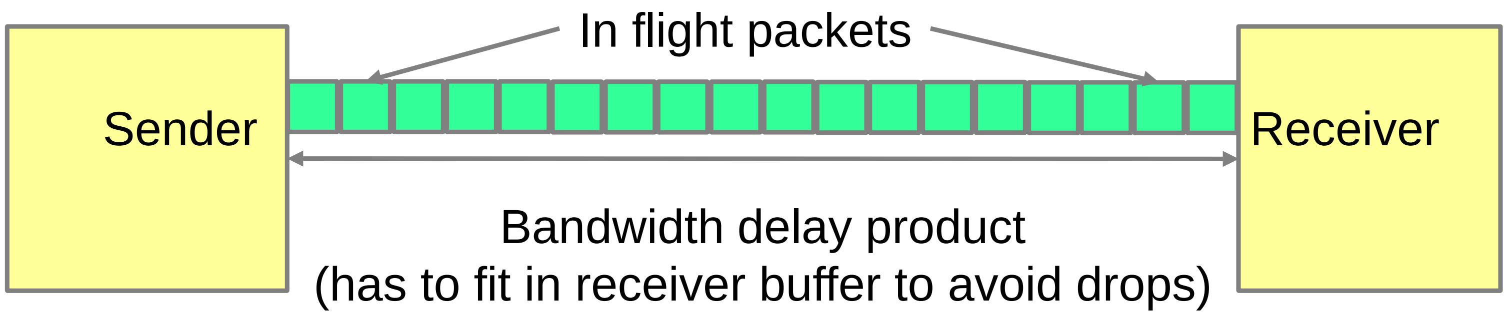
- The maximum size usable for payload data in a single IP packet
  - Minus protocol headers
- The default for Ethernet is 1500
  - 1492 for OSA in layer 3 mode
- You can increase this to reduce segmentation overhead and thus CPU cycles
  - Those frames are called “jumbo frames”
  - Your infrastructure (switches, routers, …) must support those
  - Normally up to 9000, for OSA in layer 3 mode up to 8992
- Ideally, your MTU should not exceed the MTUs used on all the hops your packets pass through on their way to their target

Ethernet frame



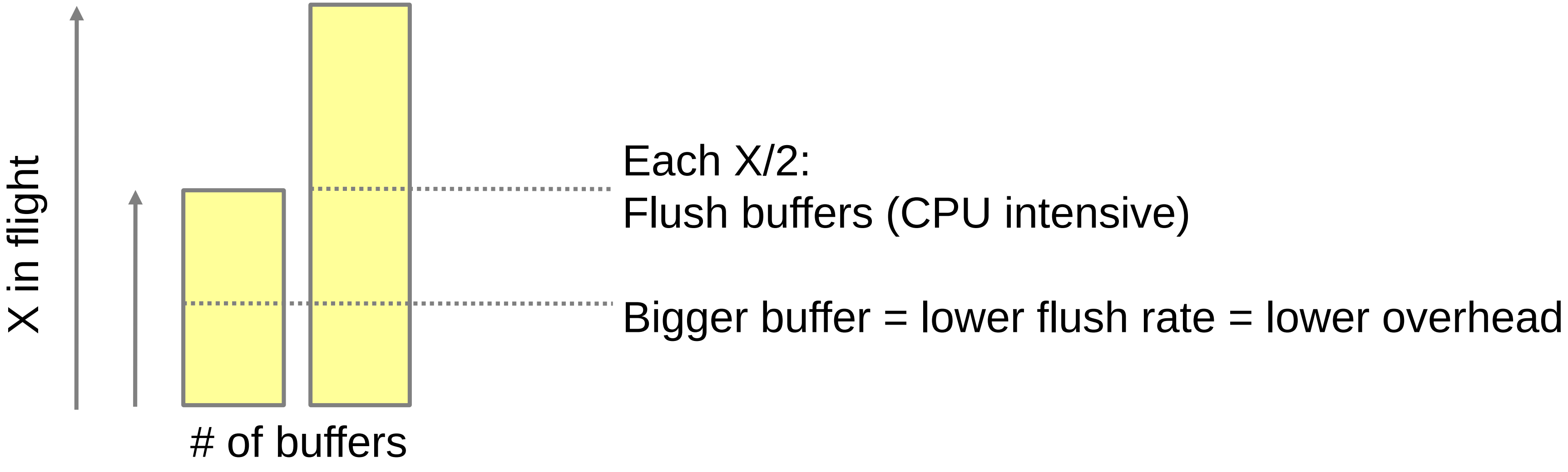
Tuning parameters - send / receive buffer size

- Buffer packets to accommodate for bandwidth mismatches between sender and receiver
  – Both could be a source of latencies if they are not drained fast enough (buffer bloat)
- Linux automatically manages the size of these buffers
  – You can set some bounds respected by the auto-tuning mechanism
- Depending on your scenario, bigger or smaller buffers work better
  – HiperSockets vs. OSA
    • For HiperSockets with a MTU > 8000, the buffer size should not exceed 524288
    • For OSA, larger buffer sizes like 4194304 are preferred for optimal performance
  – LAN vs. WAN
    • Generally, if either your link speed or your round-trip latency (or both) increases, you'll need bigger buffers (based on the bandwidth delay product).



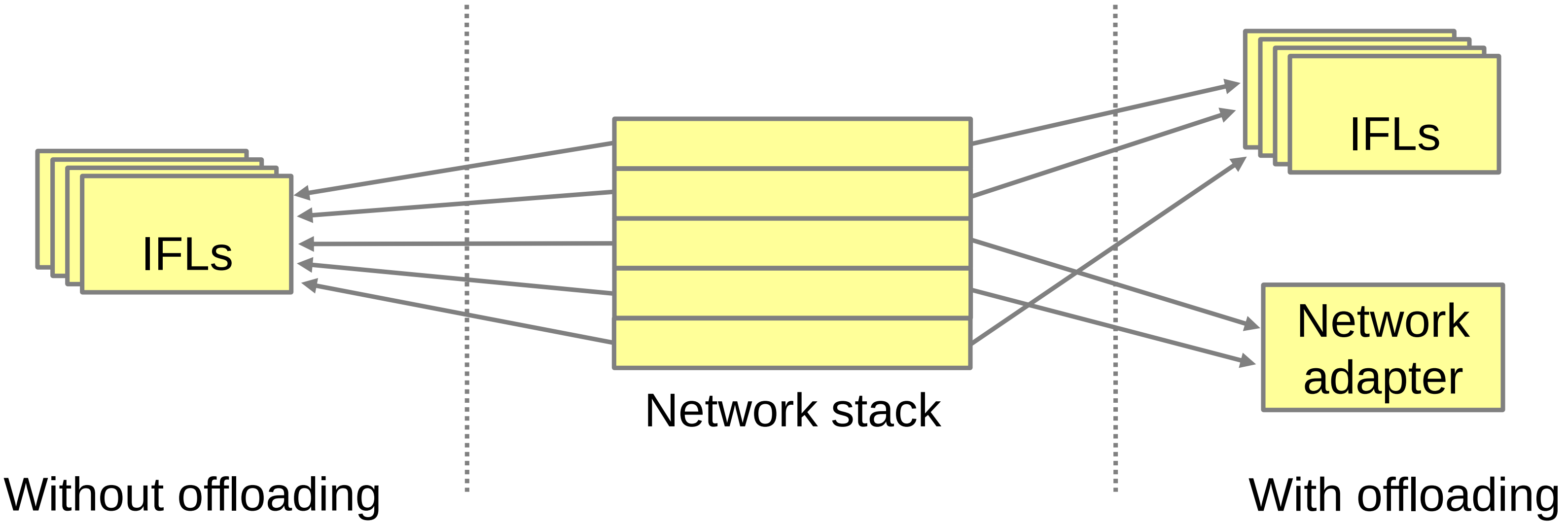
Tuning parameters - OSA inbound buffer count

- You can limit the number of buffers the OSA adapter uses for inbound connections
- The default here is 64
- For maximum performance, this should be increased to 128
- Caveat: this increases your memory consumption by 64 KiB per additional buffer



Tuning parameters - offloads

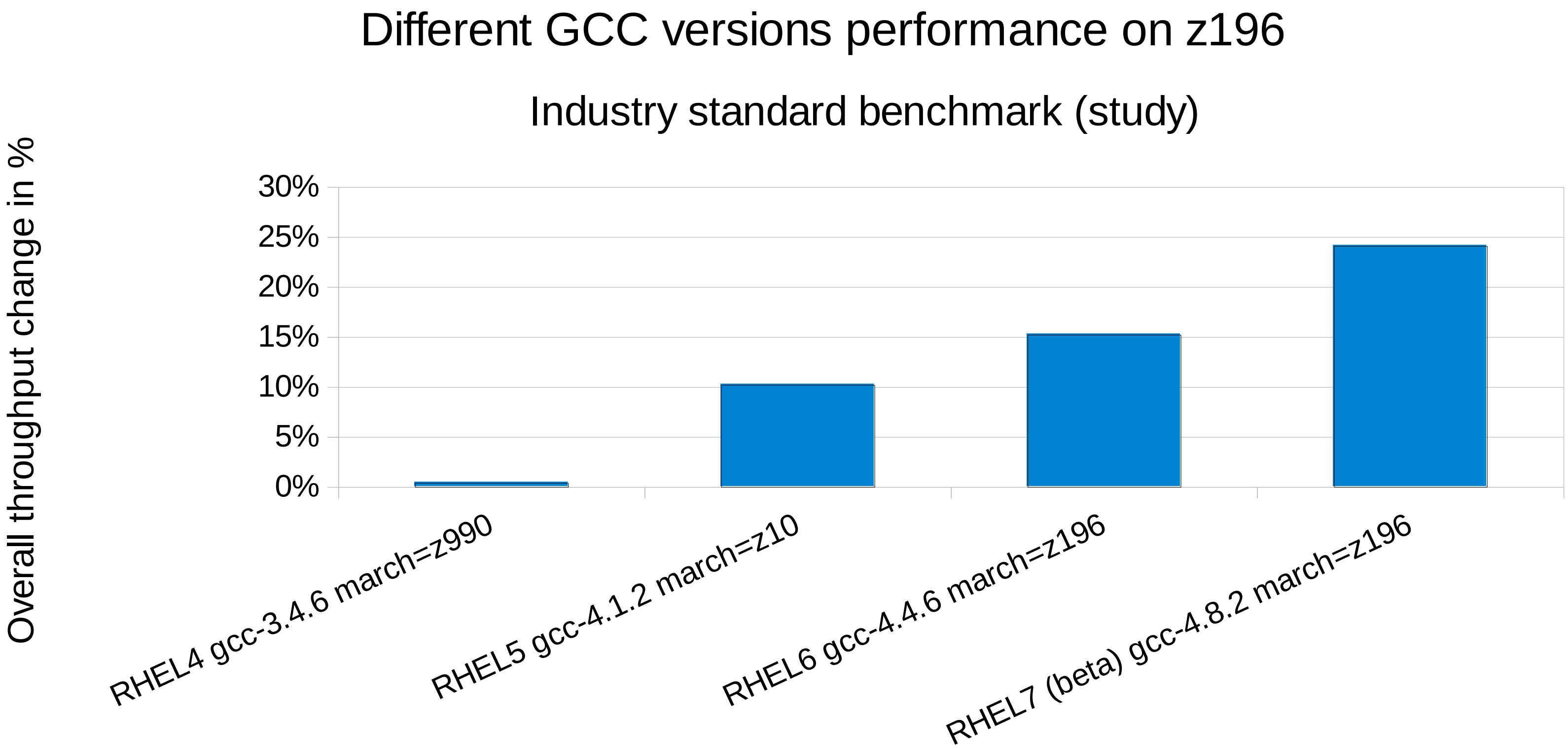
- Most network cards support some kind of hardware offloads
- Those shift work from the CPU to the network card itself
- The two most prominent here are TCP segmentation offload (TSO) and generic receive offload (GRO)
- It is advisable to enable those
  - Caveat: TSO only works for physical adapters in layer 3 mode
- Another relevant one would be TX and RX checksumming



Agenda

▪ Disk performance	approximately 40% of external support requests
▪ Network performance	approximately 25% of external support requests
▪ **Compiler**	ISVs and RYO C/C++ applications
▪ Huge pages	beneficial in almost every huge installation
▪ Java	without basic tuning always a trouble maker

GCC evolution



- Advantages of using current compilers are significant
  - Improved machine support is introduced with newer GCC versions
    - Distributors often back-port patches
  - Applications of different characteristics will show different throughput changes when using a newer compiler

GCC versions in Linux on System z supported distributions

GCC stream	x.y.0 release	Included in SUSE distribution	Included in Red Hat distribution
GCC-3.3	05/2003	SLES9 (z990 backport)	n/a
GCC-3.4	04/2004	n/a	RHEL4 (z990 support)
GCC-4.0	04/2005	n/a	n/a
GCC-4.1	02/2006	SLES10 (z9-109 support)	RHEL5 (z9-109 support)
GCC-4.2	05/2007	n/a	n/a
GCC-4.3	05/2008	SLES11 (z10 backport)	n/a
GCC-4.4	04/2009	n/a	RHEL6.1 / 5.6** (z196 backport)
GCC-4.5	04/2010	SLES11 SP1	n/a
GCC-4.6	03/2011	SLES11 SP2 (z196 support)*	n/a
GCC-4.7	03/2012	SLES11 SP3 (z196 support)*	n/a
GCC-4.8	03/2013	SLES12 (zEC12 support)	RHEL7 (zEC12 support)
GCC-4.9	04/2014	n/a	n/a

* included in SDK, optional, not fully supported
** fully supported add-on compiler

Optimizing C and C++ code

- Produce optimized code
  - Options -O3 is a good starting point and are used in most frequently in our performance measurements
  - Optimize GCC instruction scheduling with the performance critical target machine in mind using -mtune parameter
    - -mtune=values <z900, z990 with all supported GCC versions>
    - <z9-109 with gcc-4.1>
    - <z10 with SLES11 gcc-4.3 or gcc-4.4>
    - <z196 with RHEL6 gcc-4.4, optional SLES11 SP1 gcc-4.5*, or GNU gcc-4.6>
    - <zEC12 with GNU gcc-4.8>
  - Exploit also improved machine instruction set and new hardware capabilities using the -march parameter
    - -march=values <z900, z990, z9-109, z10, z196, zEC12> available with the same compilers as mentioned above
    - Includes implicitly -mtune optimization if not otherwise specified
    - -march compiled code will only run on the target machine or newer machines

* not fully supported version

GCC compile options

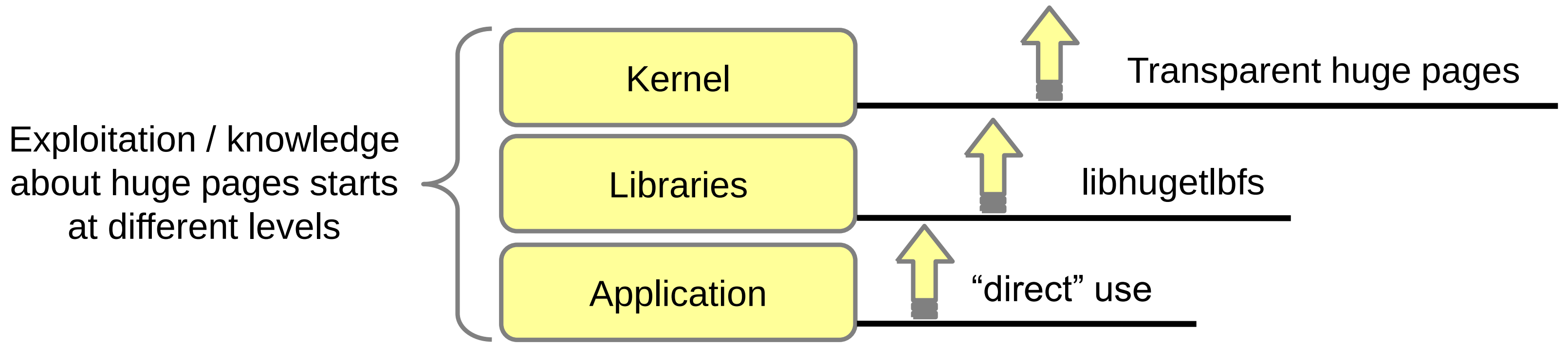
- Fine Tuning: additional general options on a file by file basis
  - `-funroll-loops` often has advantages on System z
    - Unrolling is internal delimited to a reasonable value by default
  - Use of inline assembler for performance critical functions may have advantages
  - `-ffast-math` speeds up calculations (if not exact implementation of IEEE or ISO rules/specifications for math functions is needed)
  - `-fno-strict-aliasing` helps to overcome code flaws detected with newer compiler versions
  - profile directed feedback is very benefitial if you have an applicable training workload: `-fprofile-generate, -fprofile-use`

Agenda

▪ Disk performance	approximately 40% of external support requests
▪ Network performance	approximately 25% of external support requests
▪ Compiler	ISVs and RYO C/C++ applications
▪ **Huge pages**	beneficial in almost every huge installation
▪ Java	without basic tuning always a trouble maker

Huge pages – three kinds of exploitations

- Huge Pages exploited directly by applications
  - Common exploiters using this approach are Java, Databases and other common huge memory consumers
- Huge pages exploited via libhugetlbfs
  - Common exploiters using this approach are administrators who force an application to use huge pages without change to the application itself
- Huge Pages exploited via transparent huge pages
  - Common exploiters are full system environments starting with the given releases



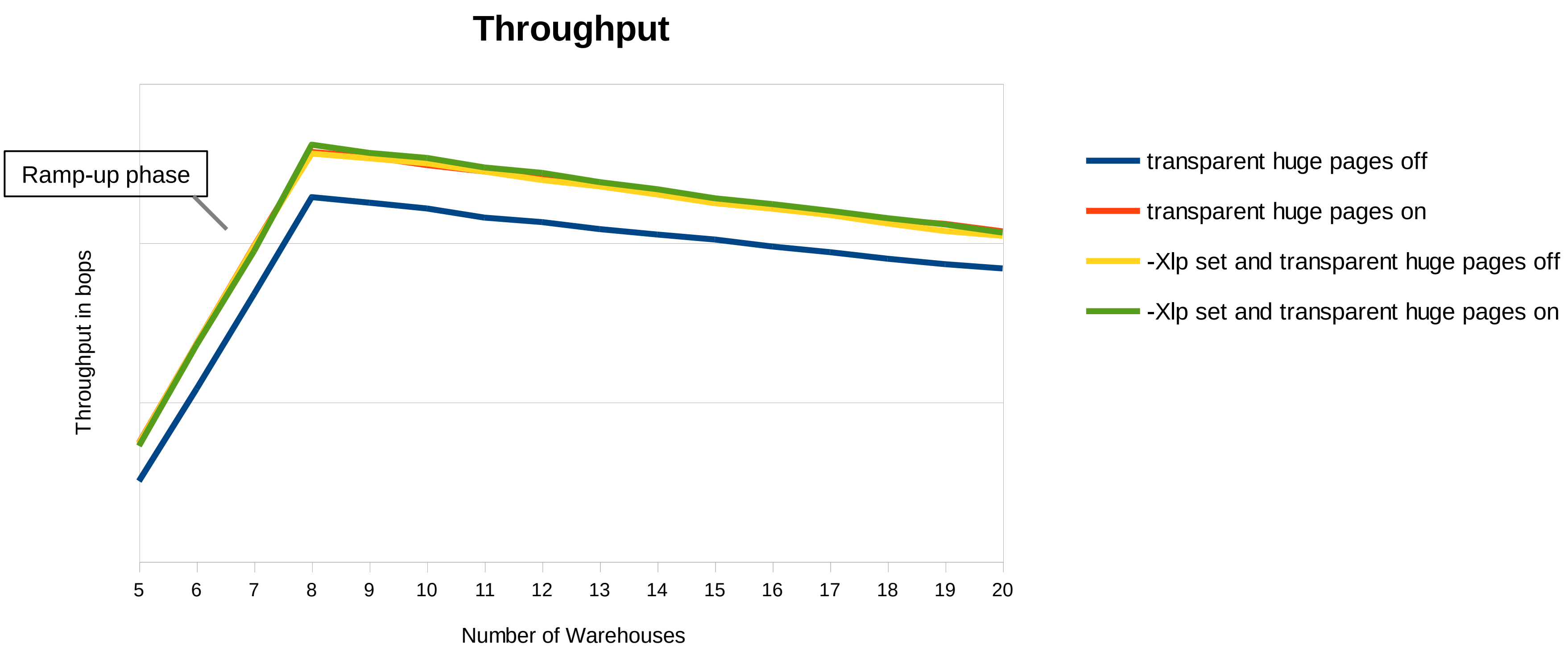
Huge pages – three kinds of availability

- Huge Pages exploited directly by applications
  - hugetlbfs support available from kernel 2.6.26 on (SLES 11, RHEL 6)

- Huge pages exploited via libhugetlbfs
  - For libhugetlbfs System z support started with version 2.15 (SLES11-SP3, RHEL7*)

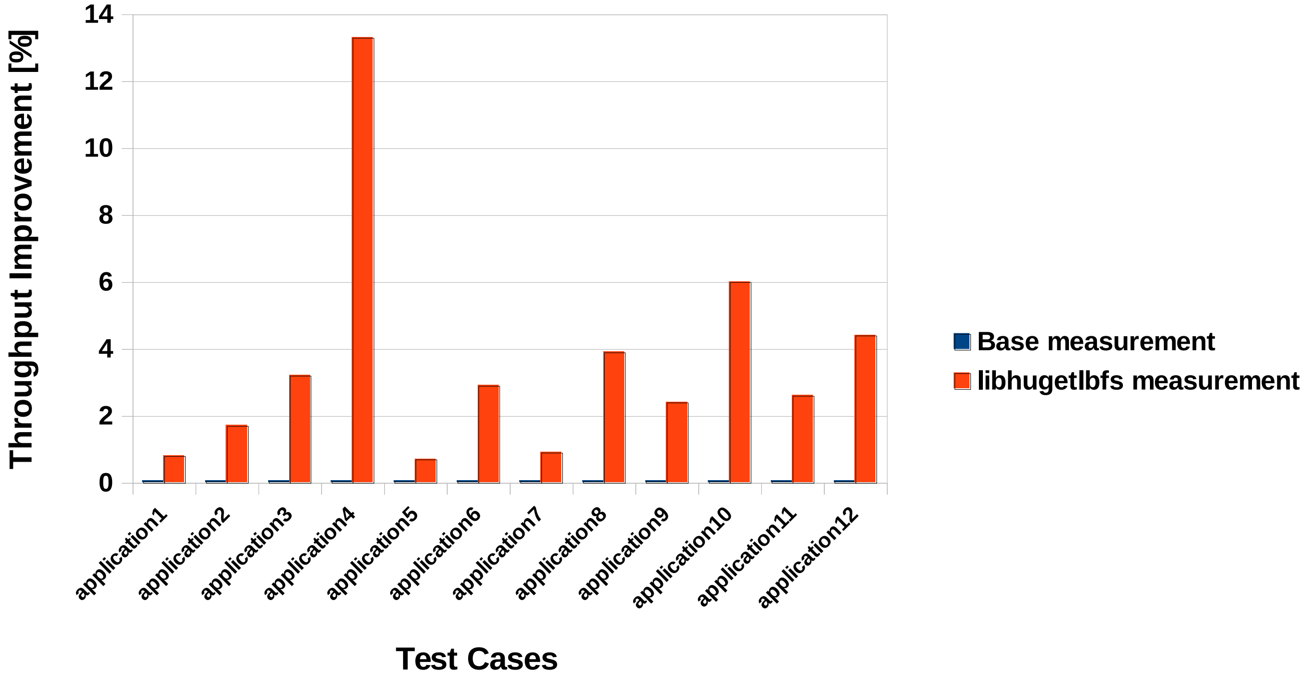
- Huge Pages exploited via transparent huge pages
  - Allows transparent access to huge pages for any application
    - Kernel tries to back with huge pages
    - Splits for swapping on demand
    - Danger of fragmentation, scanning demon in place to reassemble
  - Linux on System z support starting with kernel 3.7
    - recommended usage starting with kernel 3.8
    - available with RHEL 7 and SLES 12
  - Check `/sys/kernel/mm/transparent_hugepage/*` in your live system
    - here you can also disable them

Huge pages for Java standard benchmark



- Usage of transparent huge pages doesn't conflict with direct usage of huge pages
  - Processor savings are comparable for all cases using huge pages (~ 5.5 %)
  - Usage of transparent huge pages yields ~ 5 % performance gain
  - direct usage of huge pages (-Xlp) results in approximately the same: ~ 5% performance gain
  - recent java versions benefit even more at the potential price of setup hazzles

libhugetlbfs for compute intense integer benchmark



- Application had no native huge page code
- Usage of libhugetlbfs yields ~ 4 % overall performance gain
- All measured real life applications show a performance improvement
  - The degree of the performance improvement depends heavily on the characteristic and quantity of memory accesses
  - No tested application suffered from the usage of libhugetlbfs

Huge Pages also can save Page Table Space e.g. with Oracle

- Oracle Database uses many processes in parallel
- In general **10-15% throughput** can be gained by the reduction in processor usage as well as having **a lot more memory** for applications that would be consumed in Linux Page Tables
- The screen-shot shows that approximately 91GiB of memory were used for page tables without defined huge pages
  - At the same time system started slightly swapping
- Page tables were below 3G after switching to huge pages

```
procs -----------memory---------- ---swap-- -----io---- -system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
338  8 1766820 1096980   1200 158901132    1  467 11419   721 2140 2724  1 93  0  0  7
125 13 1767088 1096700   1316 158896948    8  135  7199  1092 2227 4262  2 91  0  0  7
420  4 1767396 1073704   1416 158891792   17  137 18407 25048 5875 11215  6 80  4  5  1
302  5 1767588 1089200   1424 158876220    3  172  1256   329 1705 1483  0 93  0  0  6
227  7 1767652 1088700   1448 158870652    9   97  4889   361 1987 1926  1 92  0  0  7
165 16 1767796 1093696   1444 158858216    0  129  3617   605 2205 2874  2 91  0  0  7
452 16 1768980 1074352   1480 158858772   35  453 11801 14244 4667 8128  5 85  2  2  6
257 14 1769204 1096292   1276 158828368    5   84  1320   505 2066 2657  2 91  0  0  7
177  6 1769172 1098028   1320 158821092    0   20  1647   447 1761 1984  2 91  0  0  7
217 16 1769600 1095124   1364 158816144   19  224  2167  1055 2029 2703  2 91  0  0  7
144 17 1770068 1088160   1256 158814320   12  239  1760   659 1884 2295  2 91  0  0  7
122 11 1771576 1082412   1276 158810608   11  561  1817   868 1862 2049  2 92  0  0  7
219 10 1772768 1073684   1260 158807908   29  408  2385   863 2200 2916  2 91  0  0  7
315  3 2033292 1076748   1152 158561024  100 86901 21179 87940 45540 33283  0 93  0  0
```

```
SReclaimable:      386028 kB
SUnreclaim:        222484 kB
KernelStack:        16880 kB
PageTables:      91964268 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:    173377556 kB
Committed_AS:   214527304 kB
VmallocTotal:   134217728 kB
VmallocUsed:      2629972 kB
VmallocChunk:   131453796 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        1024 kB
```

Huge pages – usage considerations

- In Linux the terms “huge pages” and “large pages” are used synonymously

- Due to the fact that “normal” huge pages are not swappable they may increase pressure on memory management
  - If the system starts swapping frequently usage of huge pages may consume more processor cycles than saved by huge pages in the first place

- In LPAR
  - Decreased page table overhead by using hardware feature “Enhanced DAT”

- Under z/VM
  - z/VM does not support huge pages for its guests (EDAT)
  - Still Linux can “emulate” huge pages which still drops the page table sizes
    - Can be useful for applications with a memory footprint > 10GB
    - Trade-off “cpu cycles for huge page emulation” for “page table size savings”

- Transparent huge pages only carry the TLB speedup, not the page table saving
  - Due to pre-allocated page table entries that couldn't be allocated under pressure
  - Watch out how „persistent“ they are in your case!

Huge Pages - comparison and conclusion

	direct usage of huge pages (provided by application code)	**usage of huge pages via libhugetlbfs**	**Transparent huge pages**
Administration	Proper application configuration is administration effort	Properly setting LD_PRELOAD is administration effort	No extra effort
Certainty	Usage of huge pages guaranteed, once allocated	Usage of huge pages guaranteed, once allocated	Usage of huge pages if resources are available
Overhead	None	None	Defragmentation, not getting the Page Table saving
Swap	Not swappable	Not swappable	Swappable

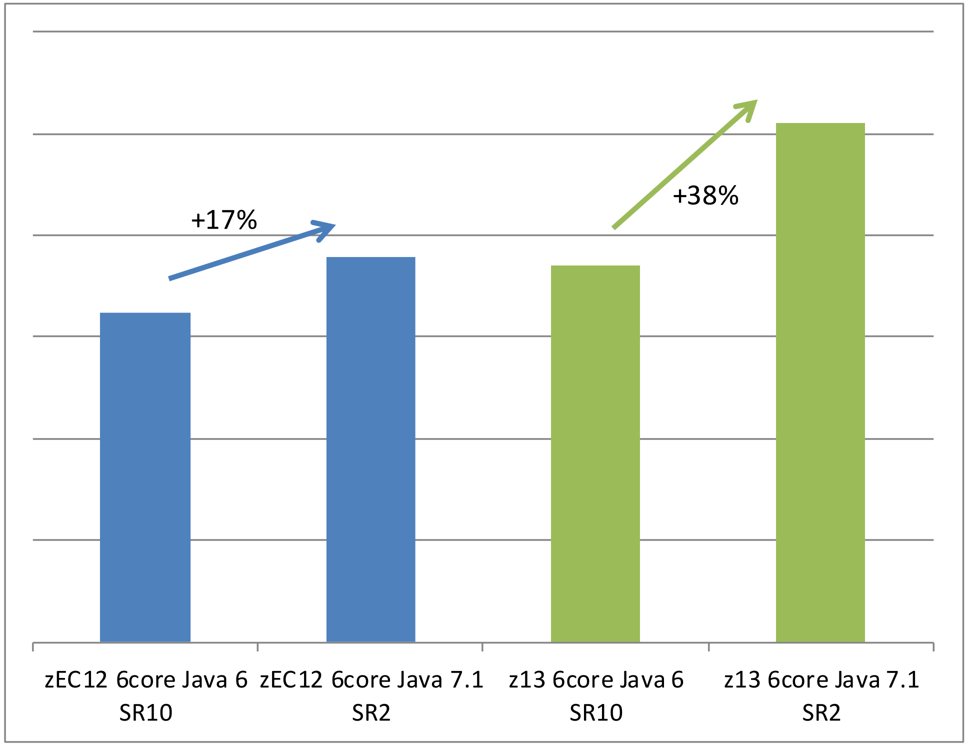
- Use whatever you can in your environmenzt
  - prio 1: direct usage e.g. Oracle, Java
  - prio 2: libhugetlbfs if applicable
  - prio 2: THP with new distributions – watch defragmentation overhead and efficiency!

Agenda

▪ Disk performance	approximately 40% of external support requests
▪ Network performance	approximately 25% of external support requests
▪ Compiler	ISVs and RYO C/C++ applications
▪ Huge pages	beneficial in almost every huge installation
▪ **Java**	without basic tuning always a trouble maker

Java

- Update frequently to get the performance benefits



Java6 SR10 is missing the z13 toleration fix
Use SR16 FP3.

- Optimize garbage collection (at least one optimization cycle)

- Use large pages (-Xlp), compressed refs (-Xcompressedrefs)

Questions ?

- Further information is available at
  - Linux on System z – Tuning hints and tips
    http://www.ibm.com/developerworks/linux/linux390/perf/index.html
  - Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/

- Blog: http://linuxmain.blogspot.com

- Contact me by mail: epasch@de.ibm.com