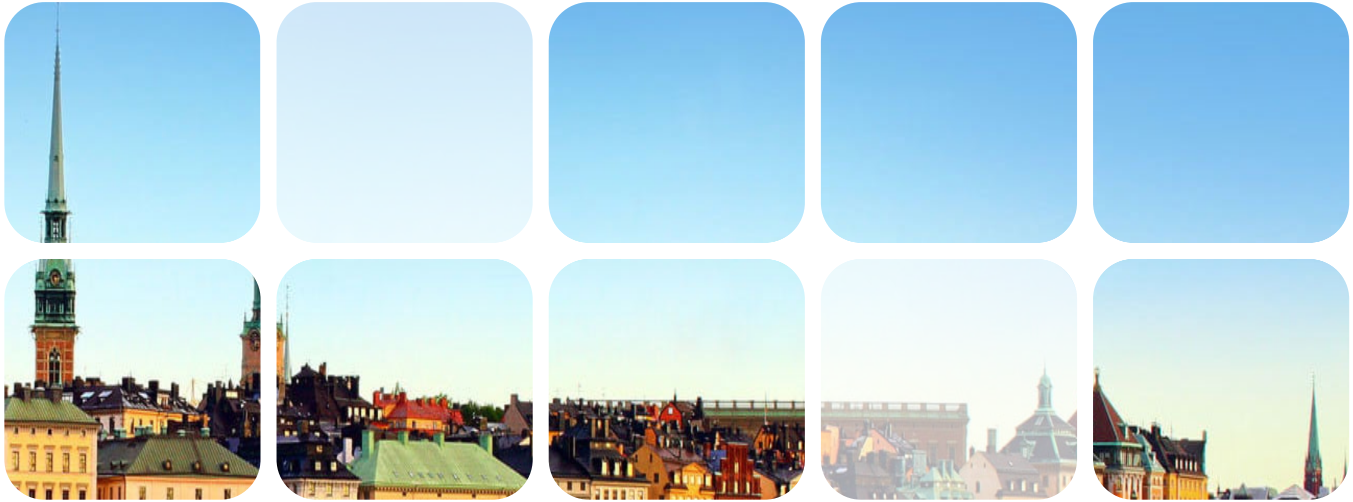


The Role of SSD Block Caches in a World with Networked Burst Buffers

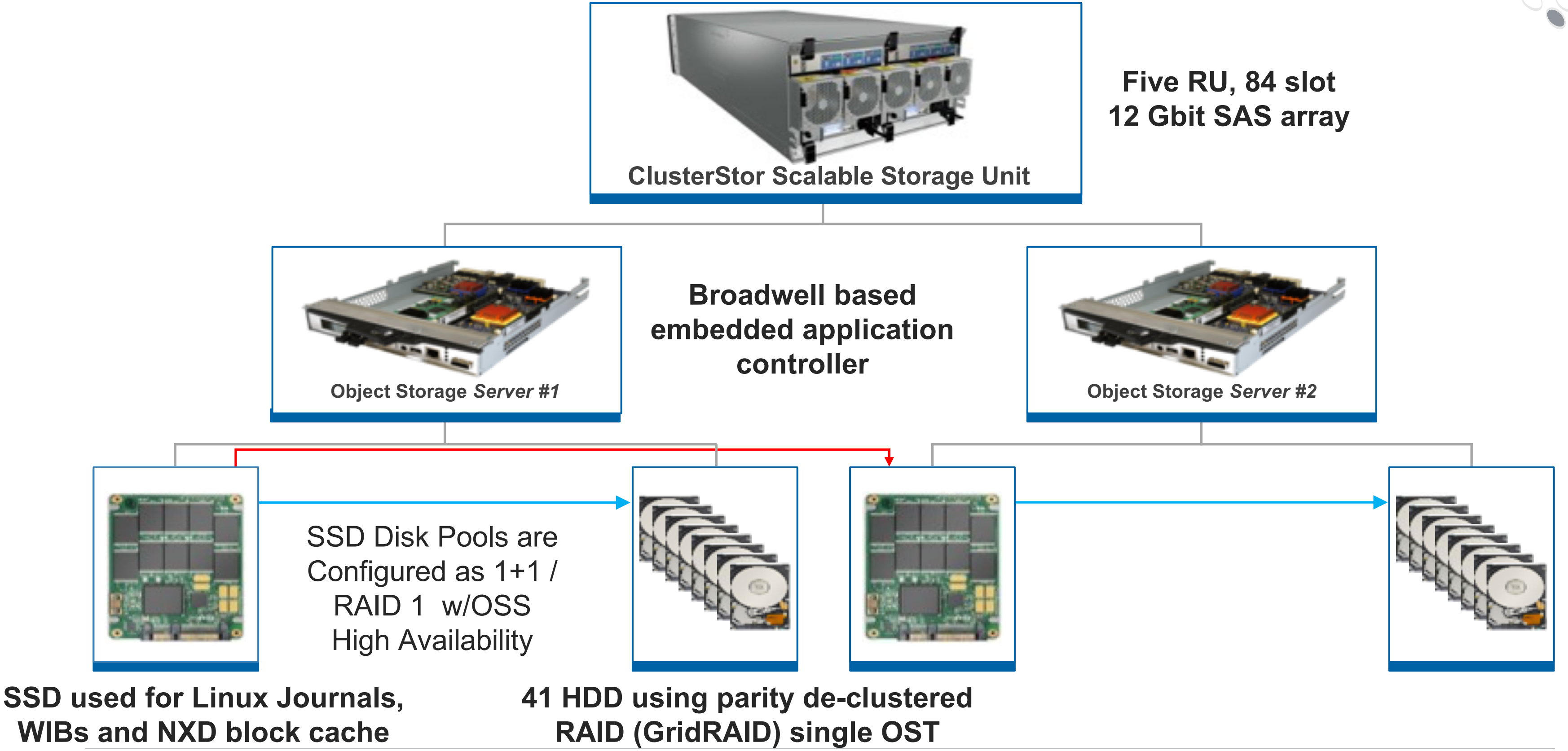
CUG 2018

Torben Kling Petersen & Bill Loewe, Cray Inc.

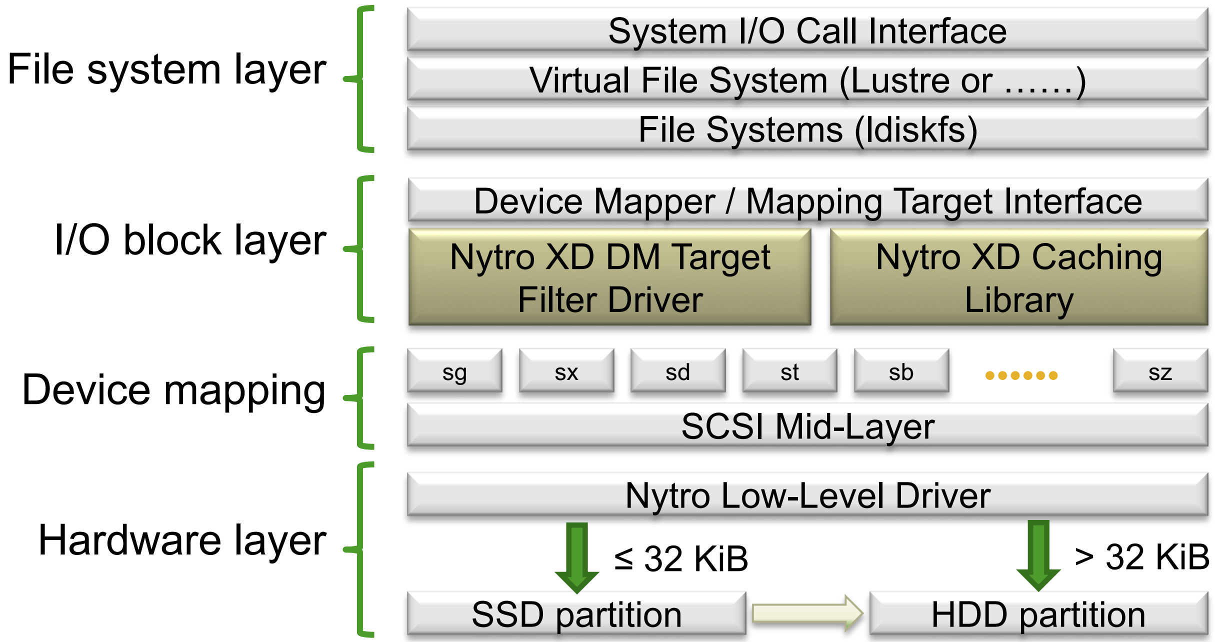
Agenda

- **Purpose**
  - HPC storage performance is under constant strain to deliver required performance
  - Modern I/O bound applications tend to be very complex with a mix of large sequential and smaller IOPS bound I/O
  - Examine the ability of NXD to improve application performance and manage a mixture of large streaming I/O in conjunction with small, random I/O.
- **Benefit**
  - Users with complex I/O workloads are always looking for solutions to the problem. NXD is potentially one of these solutions.
- **Experimental setup**
- **Results**
- **Summary**
- **Q&A**

NXD Hardware Components



NXD Software Architecture



NXD

- Filter driver implemented as device mapper target driver
- Core library compiled as a Linux kernel module with well defined APIs
- Work at the block layer be **transparent** to file system and applications
- Core caching function is an OS agnostic portable library with well defined interfaces
- Filter Driver intercept's IO and routes through Cache Management Library for Caching functions

Benchmarking Setup

- **Single ClusterStor L300N**
  - 82x Seagate 6TB Enterprise SAS HDDs
  - 2x Seagate (ST3200FM0033) SSDs
  - High speed interconnect (EDR IB or OmniPath)
- **16 Compute clients on IB or OPA**
- **Storage Software**
  - Lustre 2.7.19.12.x8-51 (user app tests run on Lustre 2.5.1)
    - End user app tests used Lustre 2.5.1
  - NXD Version 3.1.0.3 (2017.12.20)
    - End user app tests used NXD Version 3.0.10.3

Experimental Setup

1. **IOR sequential**

   IOR -b 32g -t 1m -F -k -m -e -v -v -C -i 5 -o $OUTFILE

2. **Mixed IOR**

   Sequential IOR: IOR -b 32g -t 1m -F -k -m -e -v -v -C -i 10 -o $OUTFILE

   Random IOR: IOR -b 280m -t 4k -F -k -m -e -v -v -C -i 10 -z -o $OUTFILE

3. **Concurrent IOR, increasing load of random**

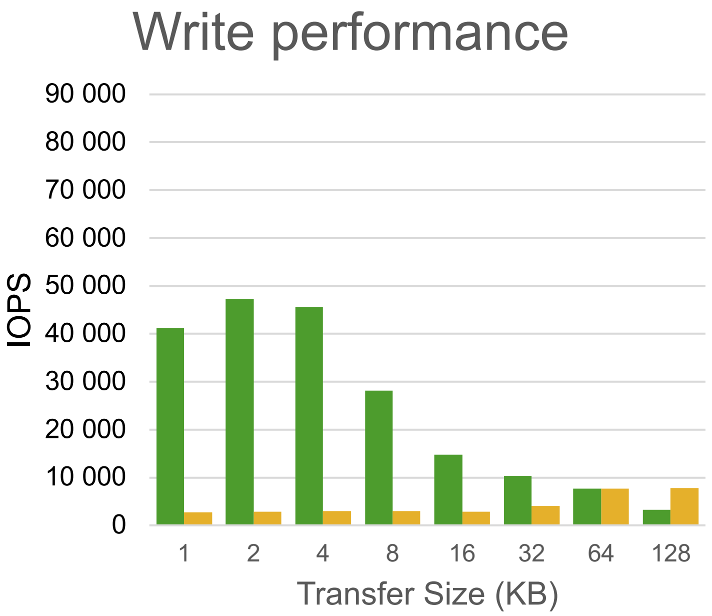
   Mixed IOR (above) with 32, 96, 160 and 224 I/O threads respectively

4. **SWMR**

   swmr write --niter 2500 --testdatafile $IN_FILE $OUT_FILE

NXD / ClusterStor L300N – Write/Rewrite, Aligned I/O

16 nodes, 64 processes total, 1x SSU, (2x SSDs vs. 82x HDDs) running Lustre 2.5.1

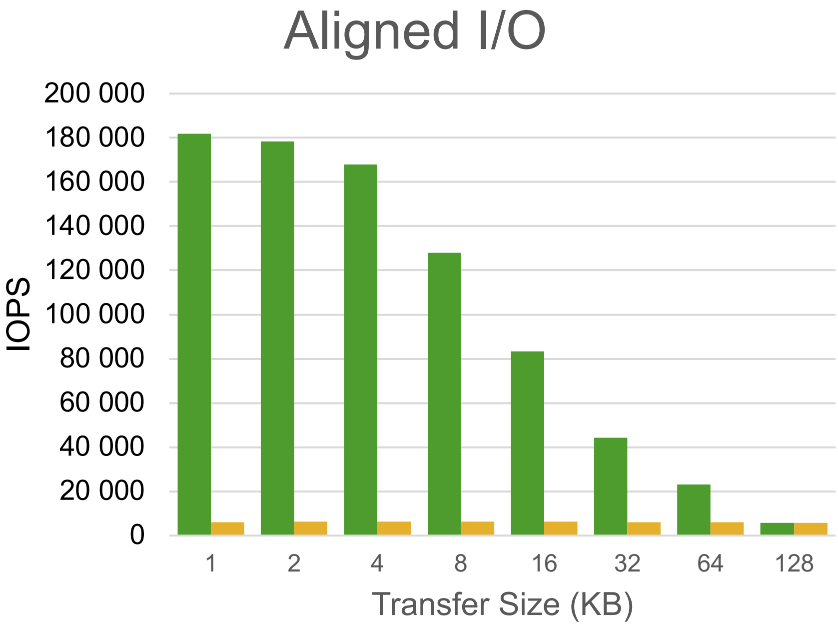


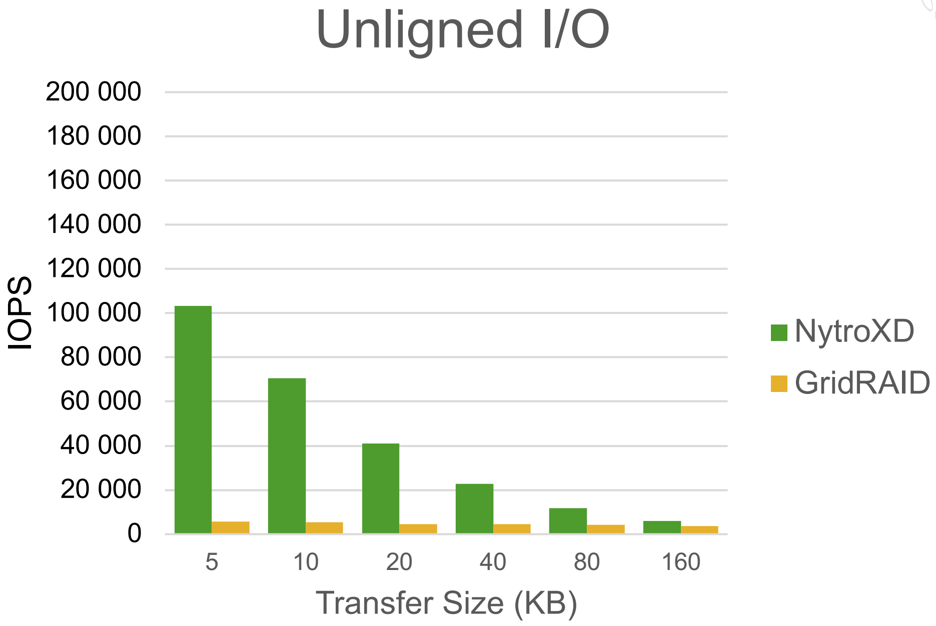
NXD / ClusterStor L300N – Write/Rewrite, Unaligned I/O

16 nodes, 64 processes total, 1x SSU, (2x SSDs vs. 82x HDDs) running Lustre 2.5.1

NXD / ClusterStor L300N – Read Performance

16 nodes, 64 processes total, 1x SSU, (2x SSDs vs. 82x HDDs) running Lustre 2.5.1



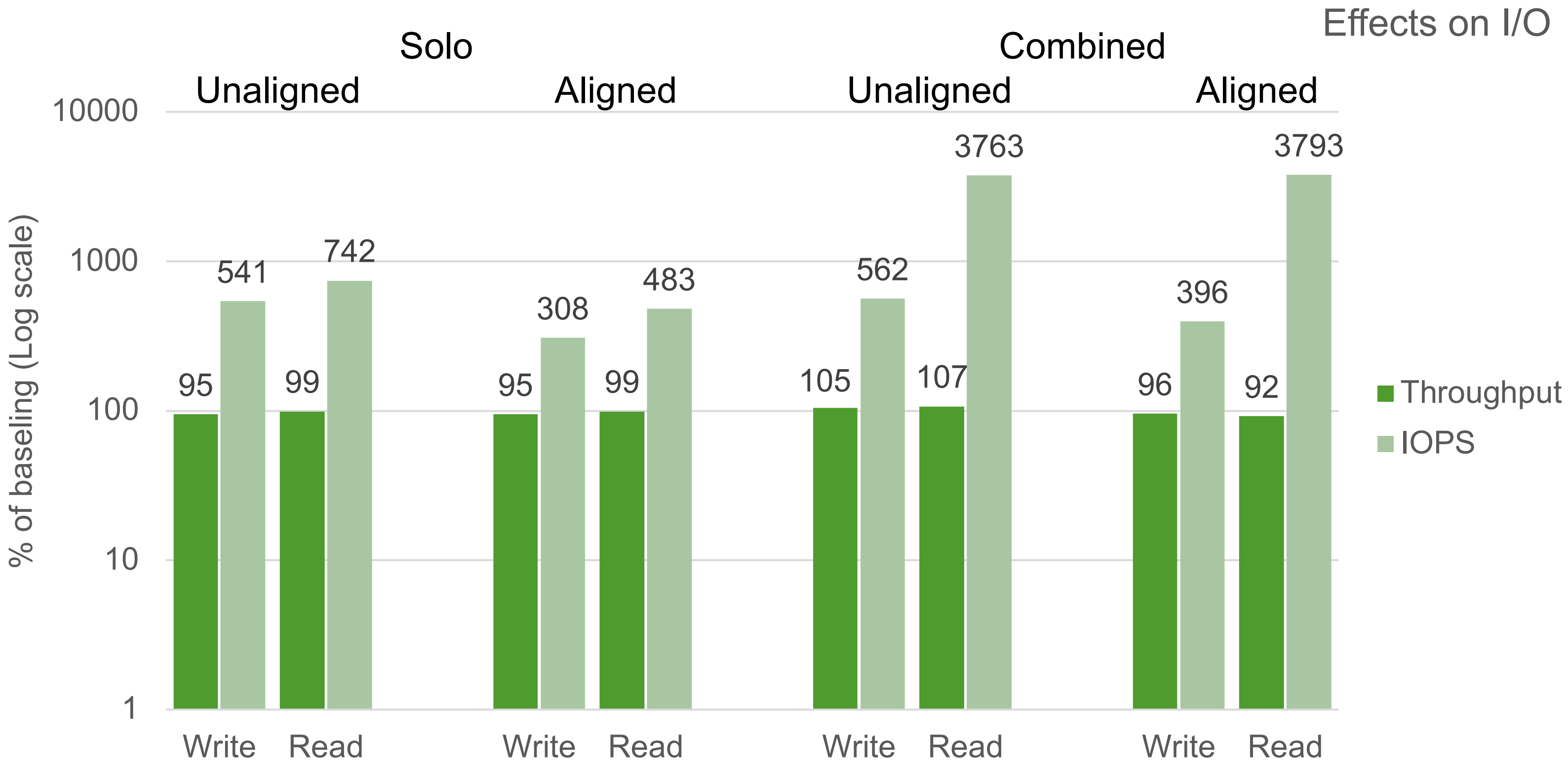


Experiment 1 – IOR Sequential

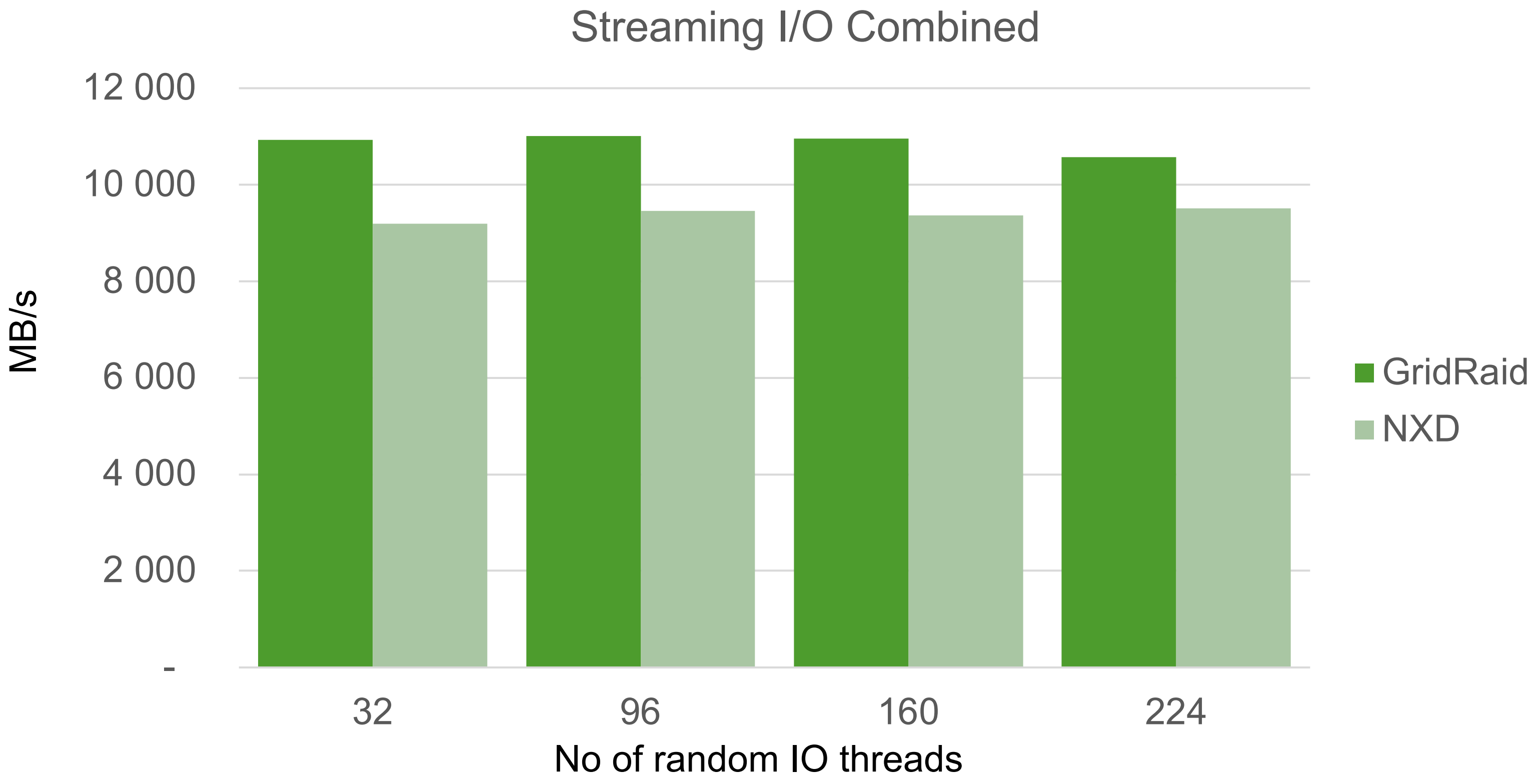
Pattern	NXD	Nodes	PPN	Procs	File Size (MB)	Transfer (KB)	Iterations	Write (MB/s)	Read (MB/s)	Write (MiB/s)	Read (MiB/s)
Sequential	**Disabled**	16	2	32	32,768	1024	5	**13,143**	**11,182**	**12,535**	**10,664**
Sequential	**Enabled**	16	2	32	32,768	1024	5	**13,014**	**11,154**	**12,411**	**10,637**

- No significant decrease in write or read performance
  - Small trend of slightly slower performance with NXD
- Mean impact over 5 iterations < 1%

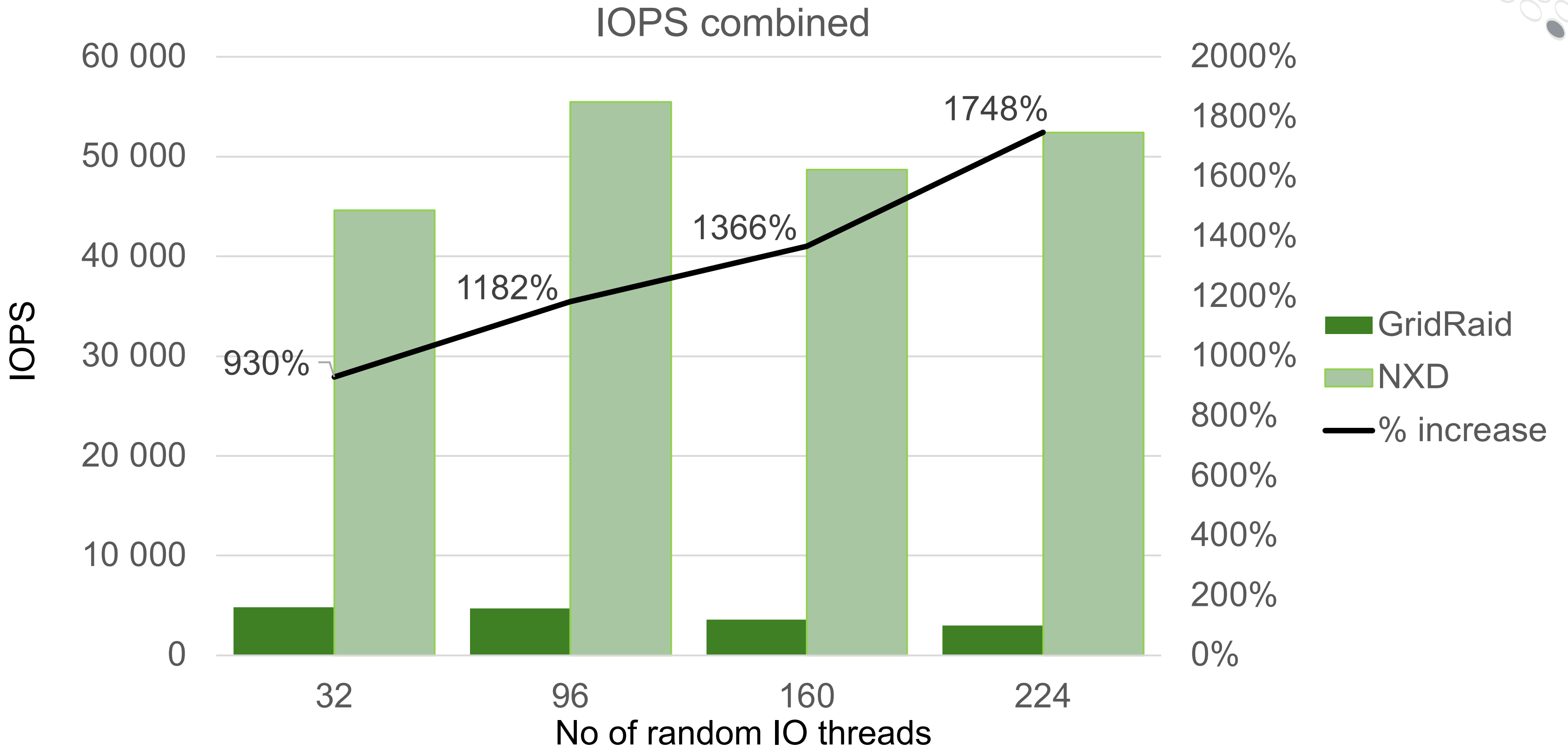
Experiment 2 – Mixed IOR



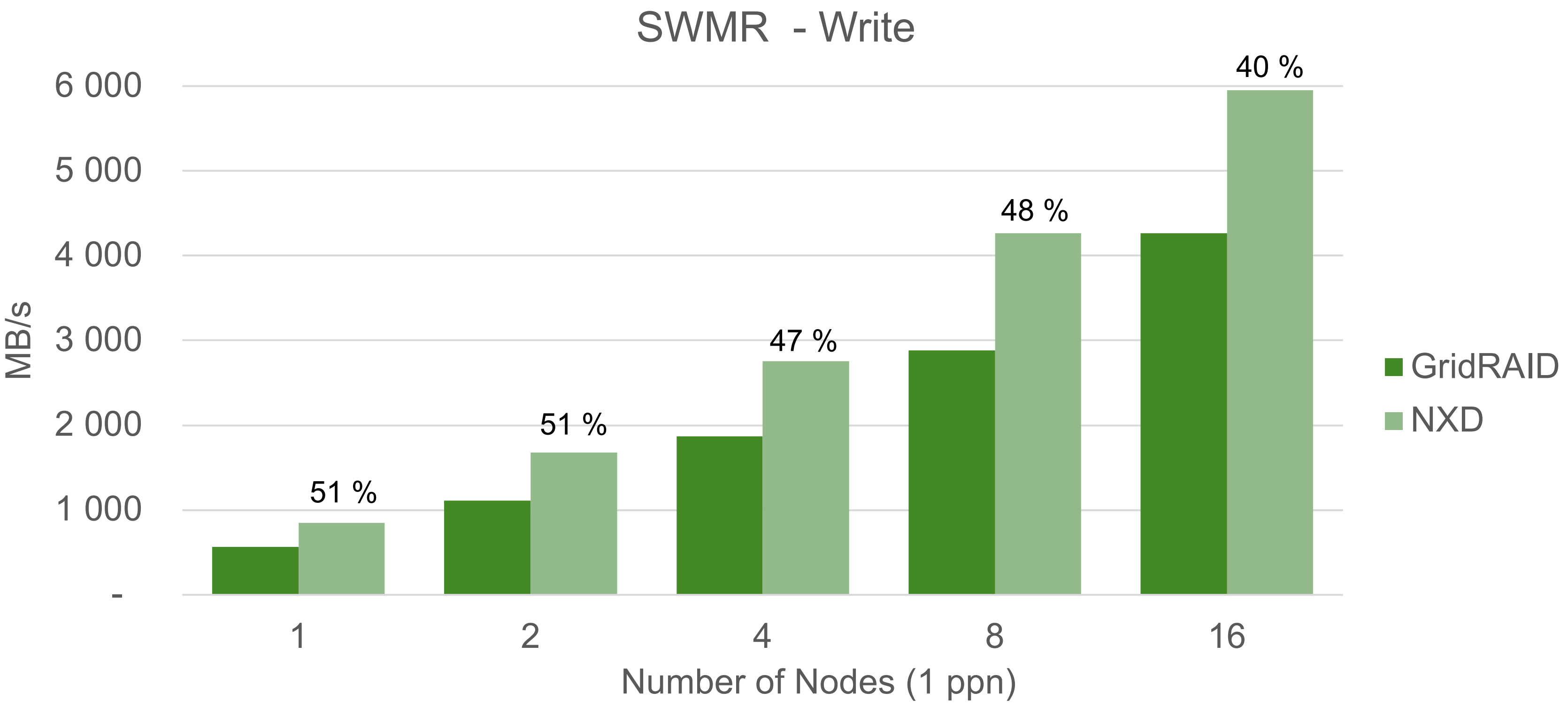
Experiment 3 – Concurrent IOR, Increasing Load of Random



Experiment 3 – Concurrent IOR, Increasing Load of Random



Experiment 4 – SWMR



Customer Test Runs – Production Codes and Data

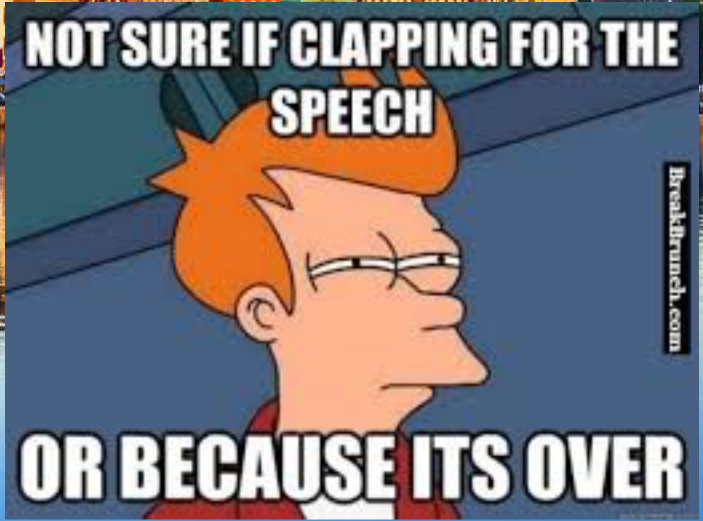
Application	Work Load Type	Large Block Range	Small Block Range	NXD Impact	Overall CS Performance
Hydra (open source)	Write intensive Large block	4M-8M – 99.9%	< 4K – 0.1%	Minimal	No degradation
Hydra (open source)	Write intensive Large block	4M-8M – 99.9%	< 4K – 0.1%	Perf gain 2.6% to 6%	Tuning down StripeSize from 4M to 1M and the gains started showing up.
CDI-PIO (DKRZ)	Write intensive Mixed mode	1M – 90%	< 512K – 10%	Perf gain of 10%	Performance improvement with NXD enabled / Histogram OFF and StripeSize – 1M / RPC 512/7
FESOM (DKRZ)	Read intensive Small block	10% large reads	90% Small Read	Perf gain of 6%	Performance improvement with NXD enabled. No tuning, no pre-load
FESOM (DKRZ)	Read intensive Small block	10% random large block	90% small Read (mostly seq.)	Perf gain of 10% over a run of continuous 100 iterations.	CS performs around 10% Sample size is only 9GiB
IOR (open source)	2 seqential runs 2 random runs	4 MB for seq	4K for random	Read: 10x increase Write: 6x increase	No degradation
COSMO (open source)	Write intensive large blocks	1 MB	Mostly 4K	1% with stripe count 1 and stripe size 4m	No degradation over 7 runs

Conclusions

- **NXD is transparent to 100% streaming I/O**
- **NXD can preferentially enhance small block I/O by orders of magnitude with only a modest impact on sequential I/O during mixed I/O workloads**
- **Performance improvement is very application dependent**
  - Careful I/O pattern analysis is required for optimal tuning
  - Example: NXD improved `tar -c` on a large directory by 43% whereas `tar -x` was unaffected
- **Still limited field experience of NXD**
  - But the number of customers implementing NXD is growing rapidly

Q&A

Torben Kling Petersen, PhD
tpetersen@cray.com



Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publicly announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: CHAPEL, CLUSTER CONNECT, CLUSTERSTOR, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used on this website are the property of their respective owners.