OAK RIDGE National Laboratory

Impact of ZFS Direct I/O on OSD Performance

Christopher Brumgard

brumgardcd@ornl.gov

LUG 2021

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Objectives

- Present our findings on the performance and implications of utilizing ZFS's new direct I/O pipeline for Lustre's OSD layer.
- Identify any performance or scaling pitfalls within the storage stack.
  - Benchmark the storage stack at multiple layers to find any substantially drops in performance.
- Then we compare the OSD performance characteristics using direct I/O vs. traditional buffered I/O on a variable-sized population of 1 to 24 NVMes in a striped configuration.

Methodology

- Tested Direct I/O at multiple layers
  - From 1 to 24 NVMe devices
  - Single raw block device
  - Striped raw block device
  - Striped ZFS (using the ZPL)
  - OBDFilter-survey to test the ZFS-OSD over striped NVMes
- For ZFS testing used a combination of parameters settings:
  - Lustre suggested settings for ZFS
  - 32 KB and 1 MB record size, no compression, and no checksum.

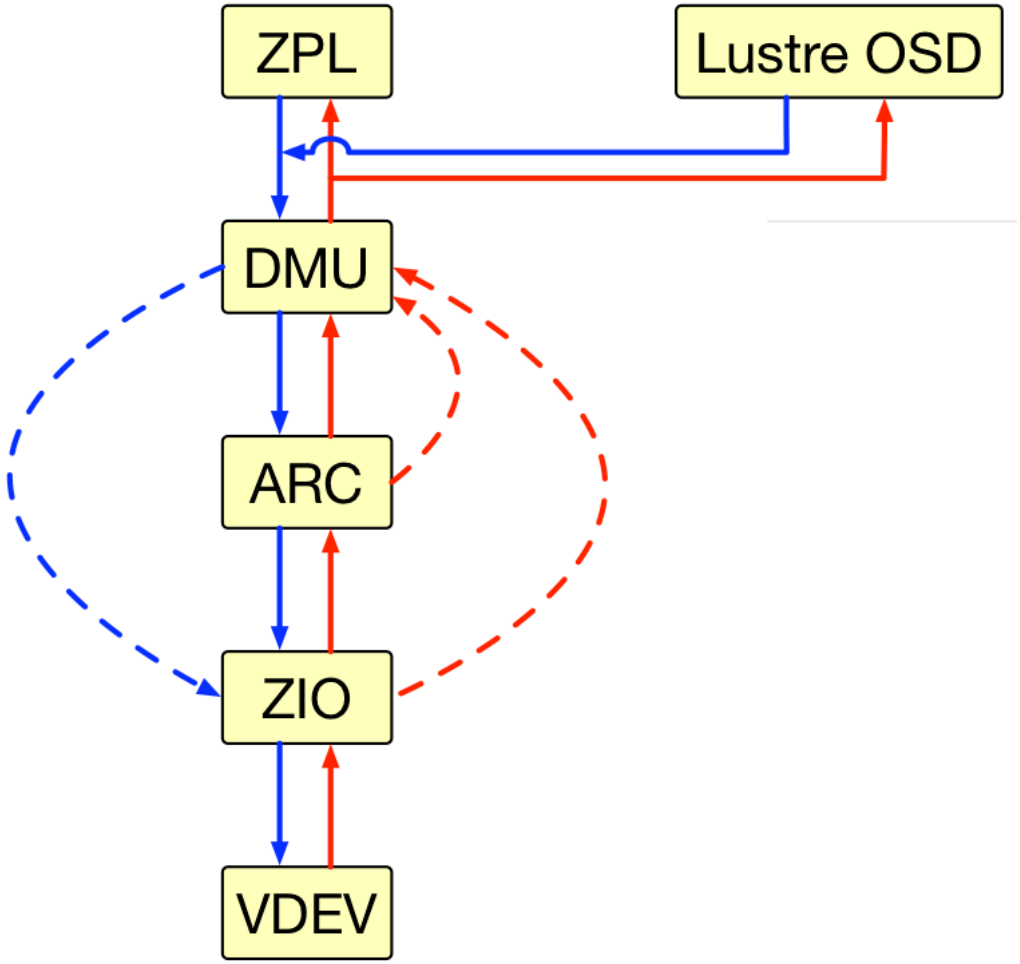
Hardware

- Dell PowerEdge R7515
  - AMD EPYC 7702P
    - 64 Core Rome
    - 2 GHz
    - Single NUMA
  - 512 GB RAM
    - 16 x 32 GB @ 3200 MT/s
  - PCI Express Gen 3
    - 2 Host controllers
      - 1 per 12 Drives
      - 16x aggregate lanes @<~15.75 GB/s
- 24 Samsung PM1725A/B NVMe's
  - 1725A – 8x
    - r/w: 6,400 MB/s and 3,000 MB/s
  - 1725B – 4x
    - r/w: 3,500 MB/s and 2,000 MB/s
  - 1.6 TB Capacity
  - Each drive is in a 4x slot

Software

- RHEL 7.9 with 3.10.0-1160.15.2 kernel
- FIO 3.25
- Custom ZFS 2.x with Direct I/O
- Lustre 2.12.2 with patch #41689

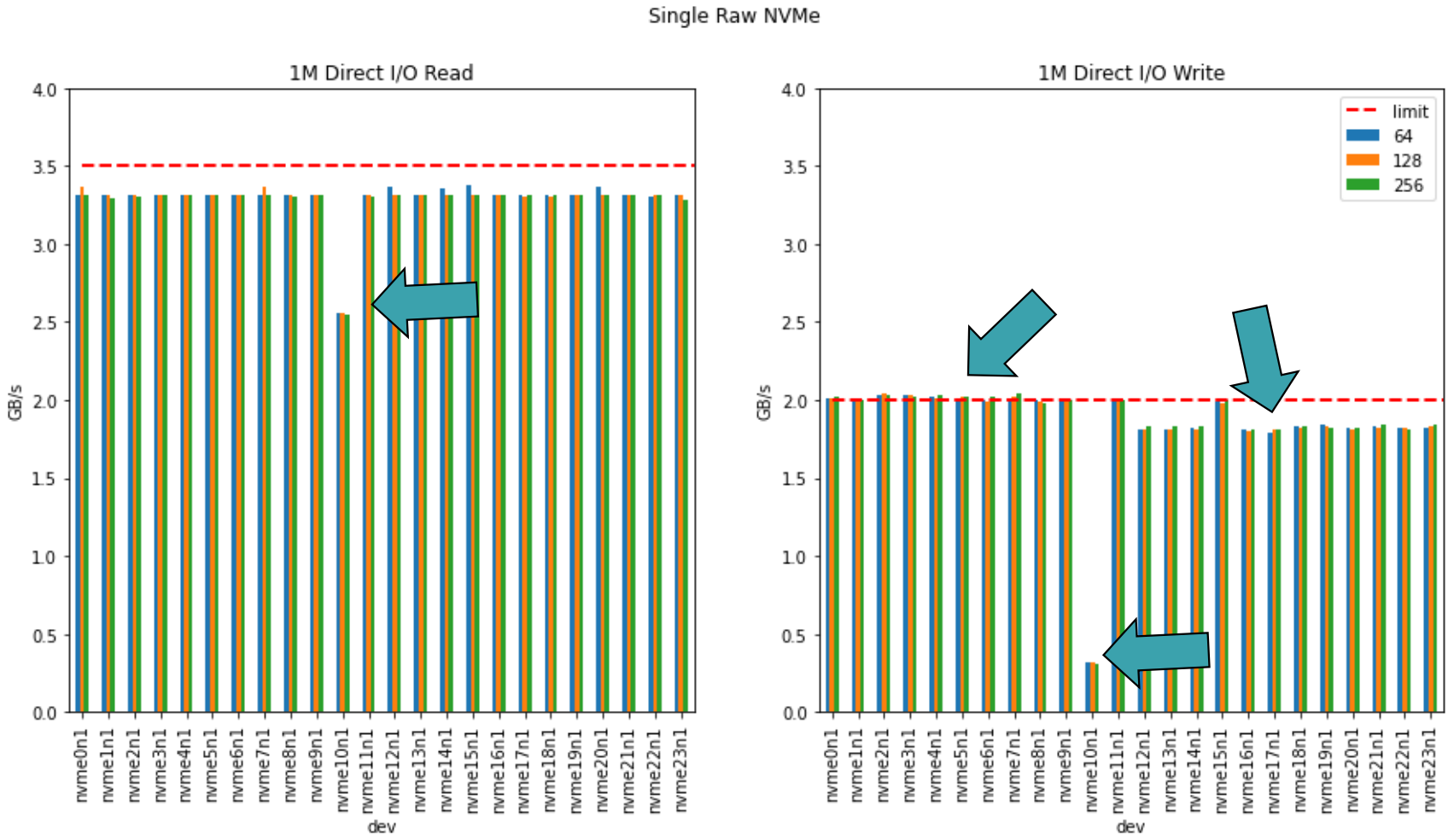
Simplified ZFS Stack



- The red and blue lines represent the read and write path, respectively.
- The solid lines are the traditional buffered paths.
- The dashed lines are the changes introduced to support Direct I/O.
  - Allows for bypassing *the ARC.*
  - *Requires page and ZFS record size alignment.*
- LU-14407
- Credit to Rick Mohr

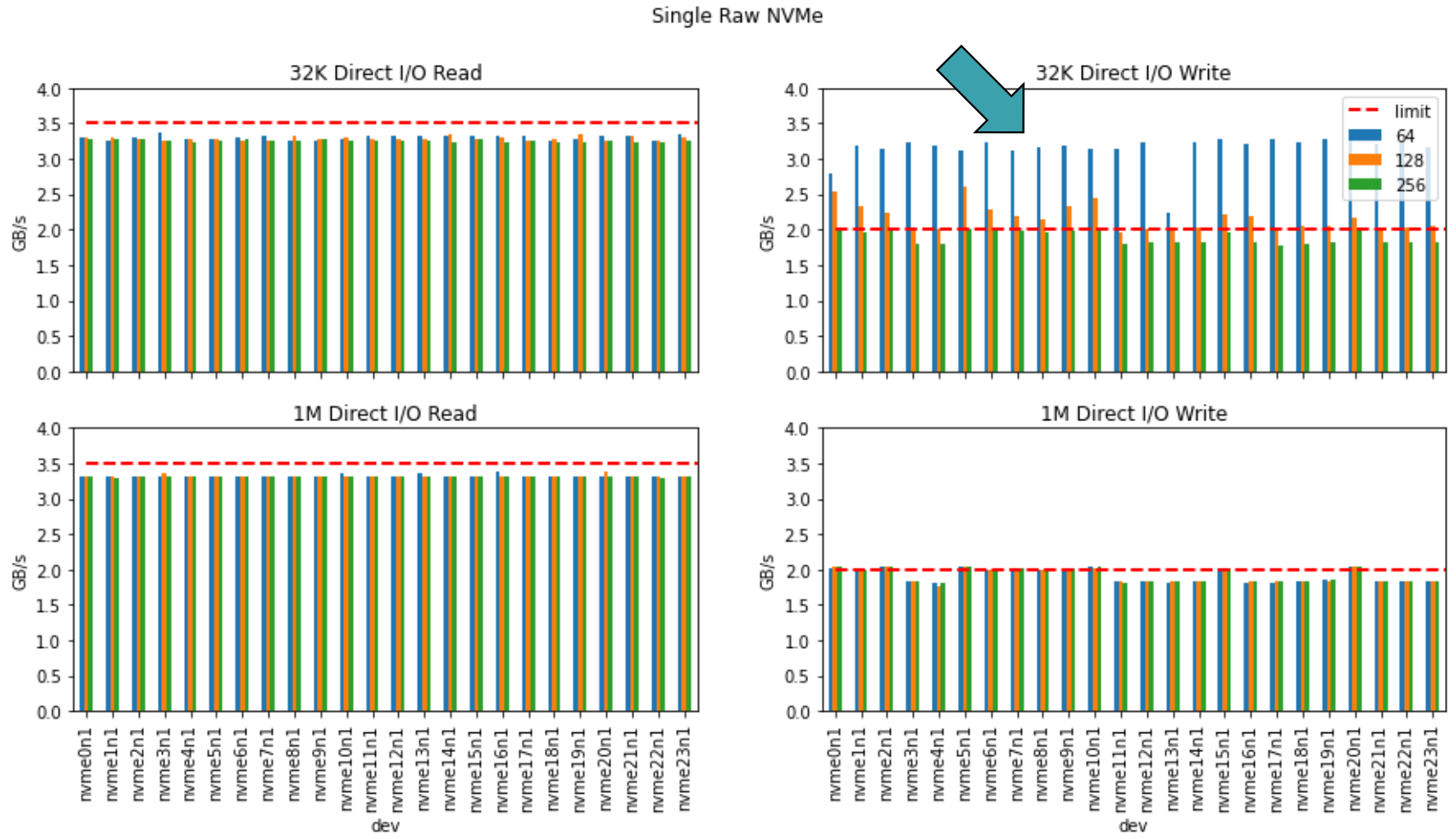
Single Block Device

- We began by running tests on each NVMe.
- FIO on the block devices using synchronous I/O.
- Block sizes 32K and 1M
- Jobs sizes: 64, 128 and 256
- /dev/nvme10n1 was defective.
- 2 different NVMe models were present in the population.



Single Block Device

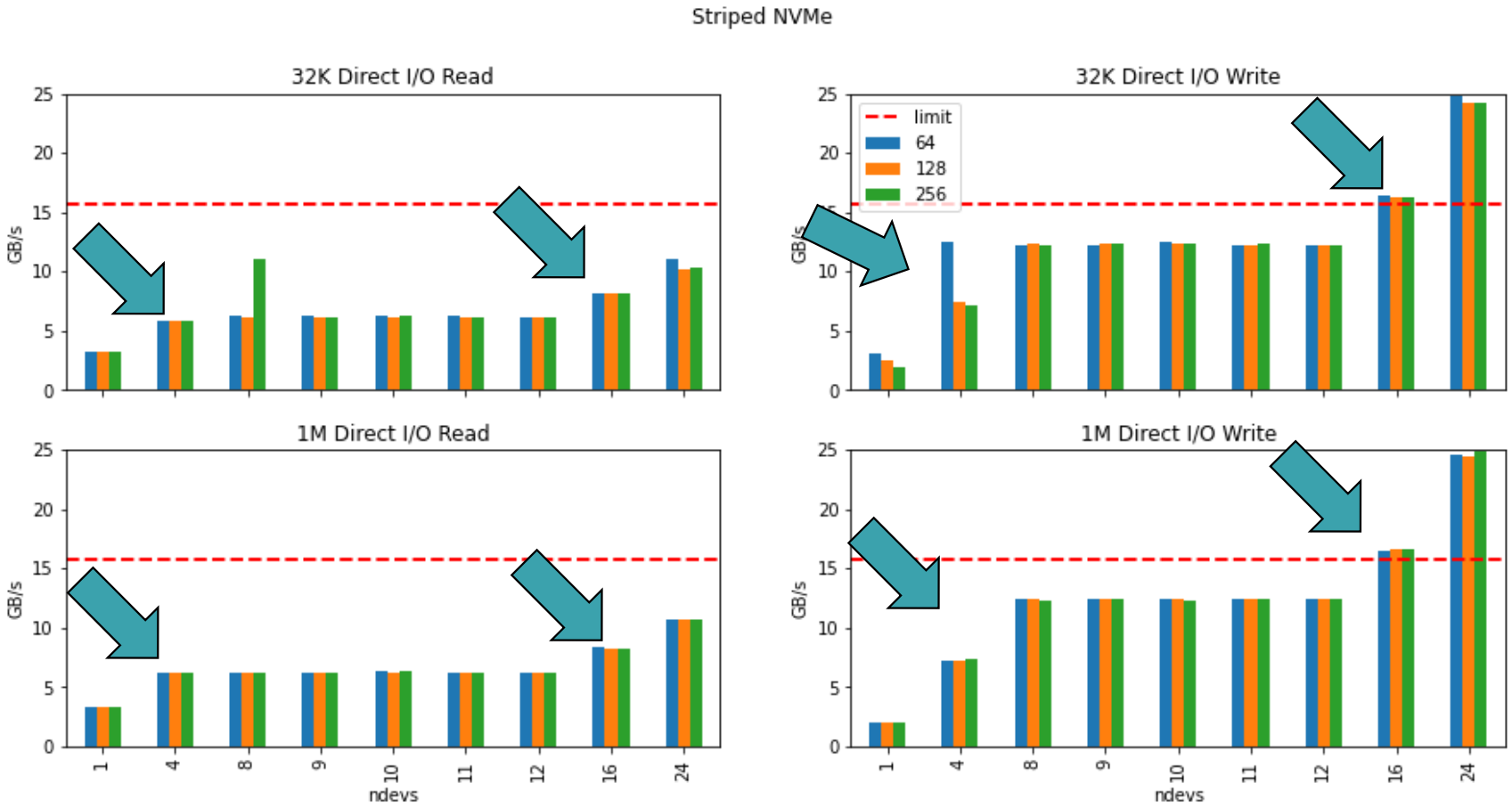
- Fixed some of the problems.
- Results are roughly where we expected them.
- Job sizes don't impact much.
  - except for 32 KB writes.
- 32 KB Direct I/O does show greater than expected performance.



Striped Block Device

(naïve)

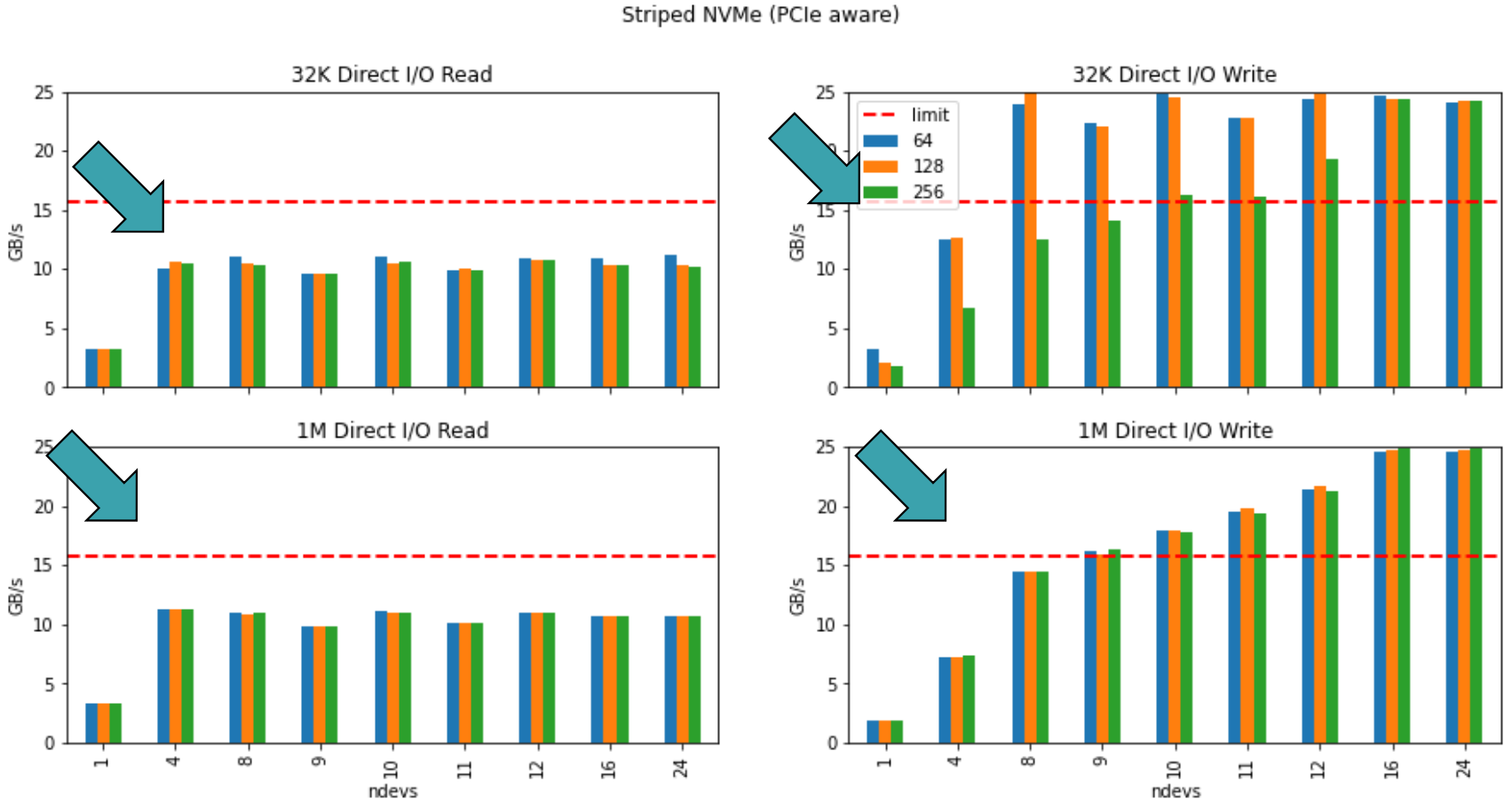
- FIO striping across block devices.
- Block sizes 32K and 1M
- Jobs sizes: 64, 128 and 256
- Not a significant amount of difference based on jobs count.
- Direct I/O reads are significantly slower than writes.
- Both reads and writes show an early plateau starting around 4 to 8 devices and continuing until 16 devices.



Striped Block Device

(PCI-Aware)

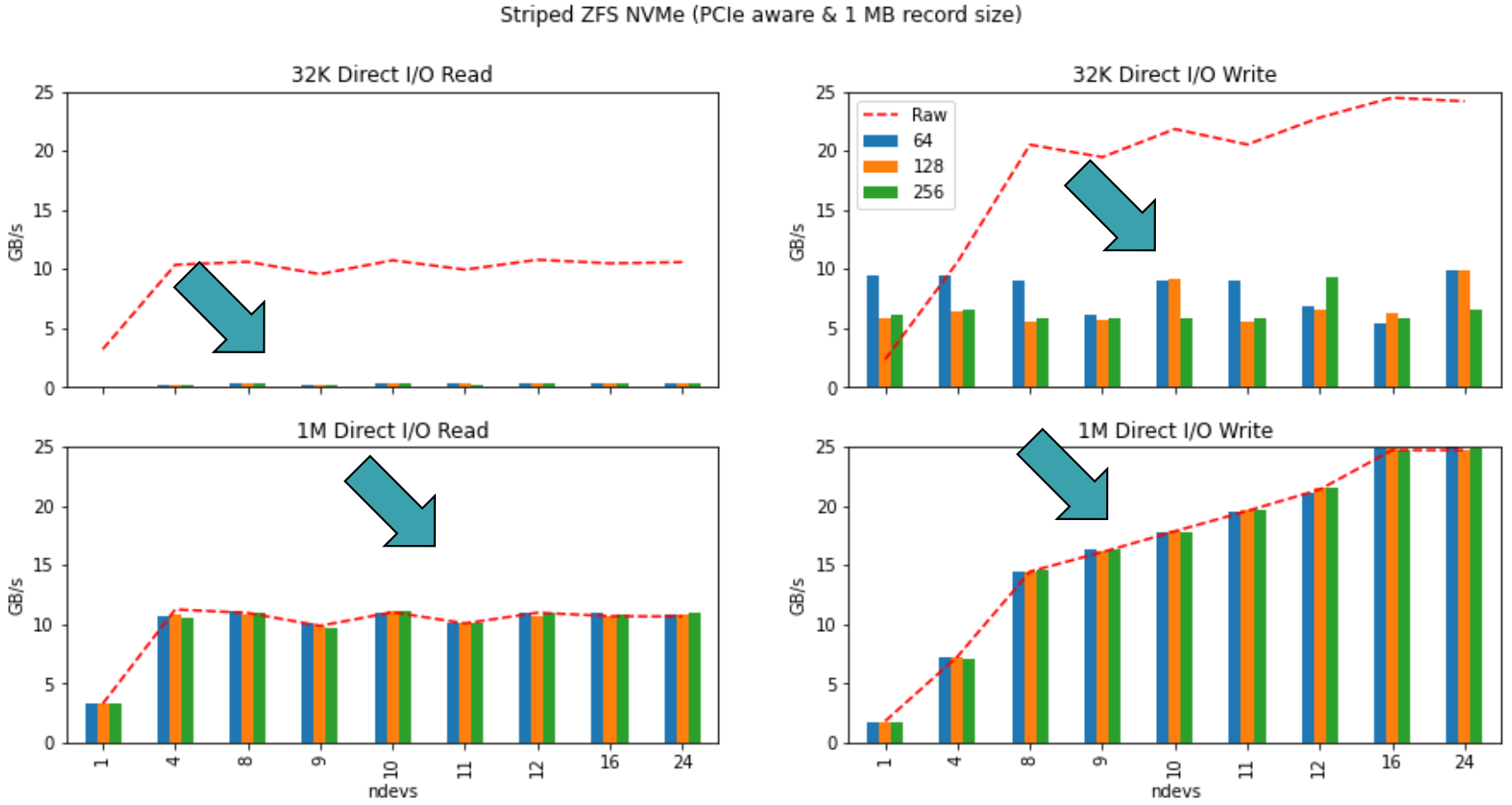
- Assigned devices round robin across the 2 PCIe host controllers.
- For all the workloads, performance scales much faster
- Not a significant amount of difference based on number of jobs except for 32K writes with 256 jobs.
- Direct I/O reads are significantly slower than writes and a limit is hit at 4 NVMes and then they plateau.



Striped ZFS(ZPL)

(PCI-Aware)

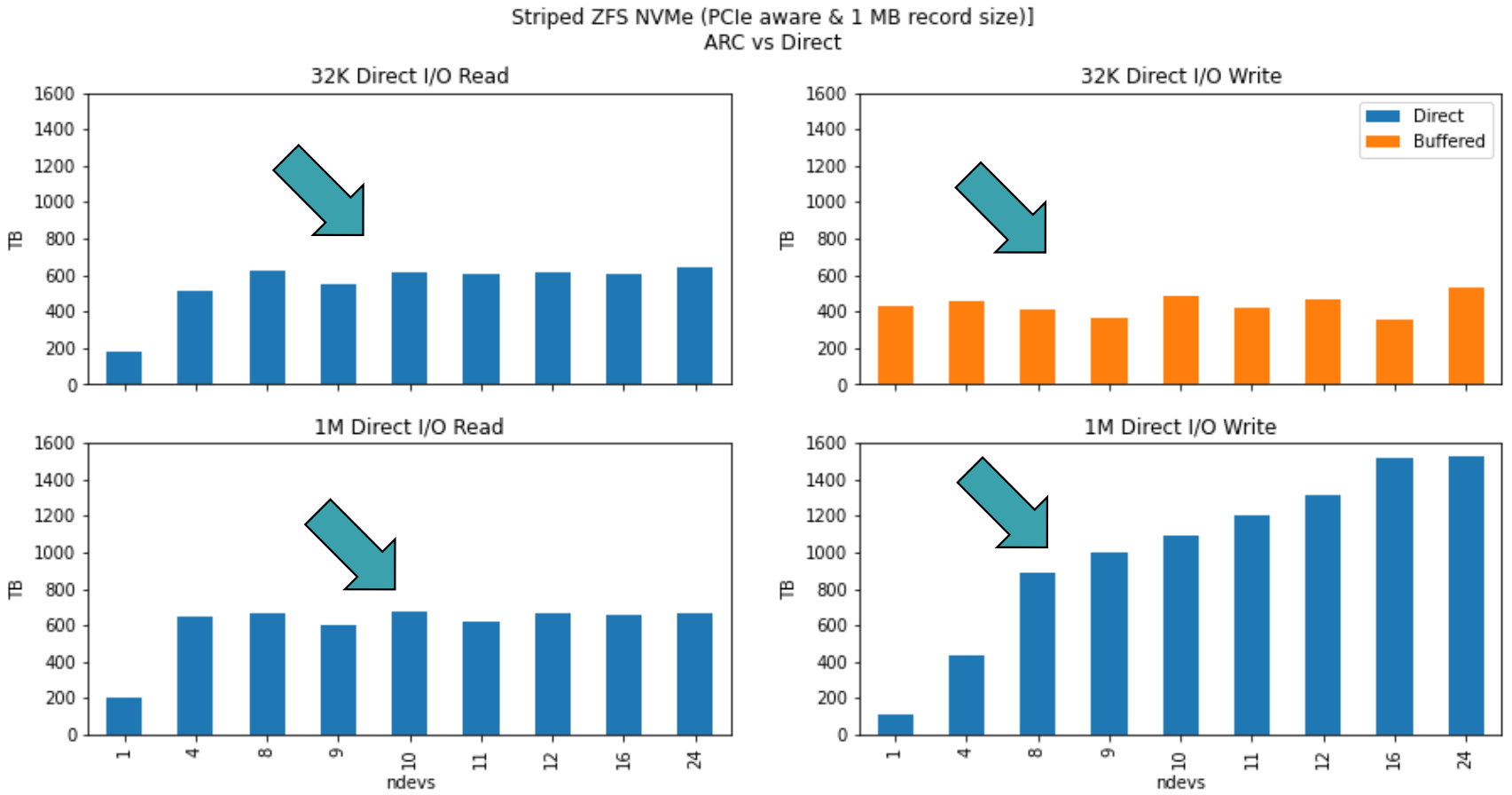
- Standard ORNL settings with 1 MB record size.
- Good News
  - 1 MB reads and writes don't drop performance from the raw block access.
- Bad News
  - 32 KB reads and writes.
  - Direct I/O requires reading and writing whole records.
  - Every read translates to a 1 MB read.
  - Every write falls back to the buffered path.



Striped ZFS(ZPL)

(PCI-Aware)

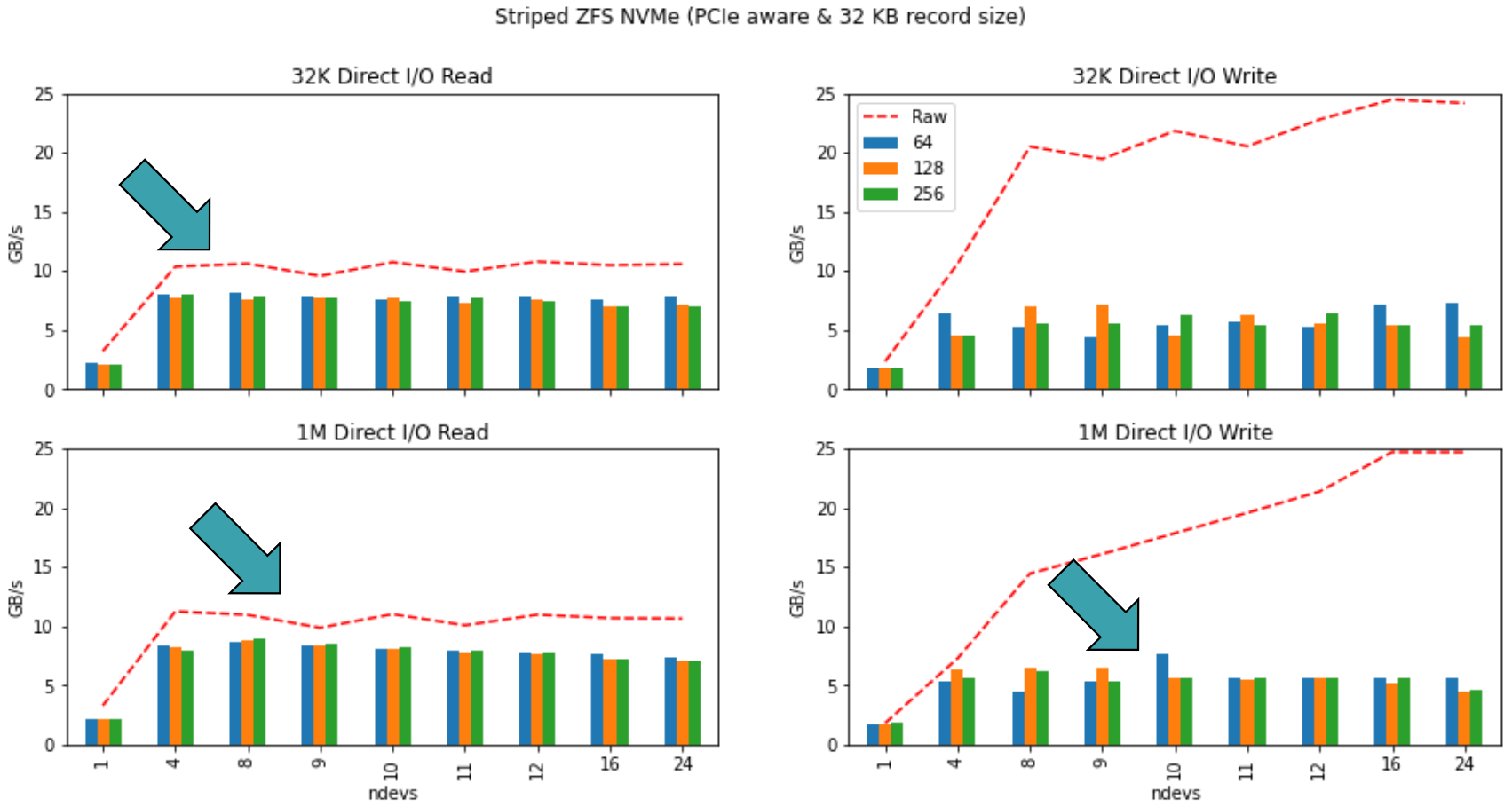
- For 1 MB reads and writes, all the I/O is direct.
- Explains the poor performance for 32 KB.
- 32 KB reads, the quantity read is the same as for the 1 MB reads.
  - 32 KB Direct I/O causes full 1 MB reads
- 32 KB writes is converted to the buffered path.
  - Direct I/O causes 1MB rewrites.



Striped ZFS(ZPL)

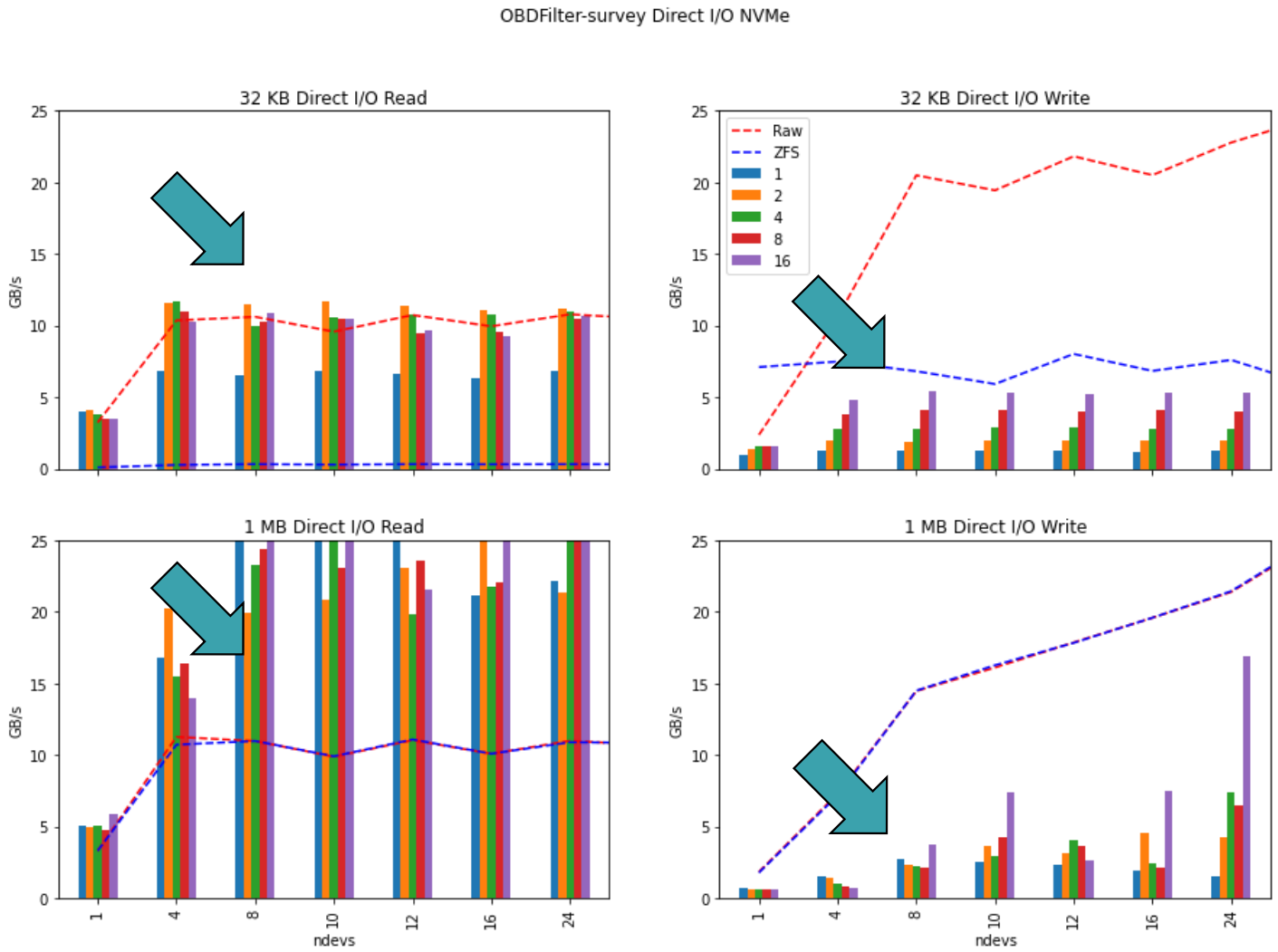
(PCI-Aware)

- Changing the record size to 32 KB resulted in the I/O following the direct path.
- It improved the performance of the 32KB read.
- But drastically diminished the 1MB read/write.
- The performance of both 32 KB and 1 MB are nearly identical.
  - The 32 KB record binning becomes the bottleneck.



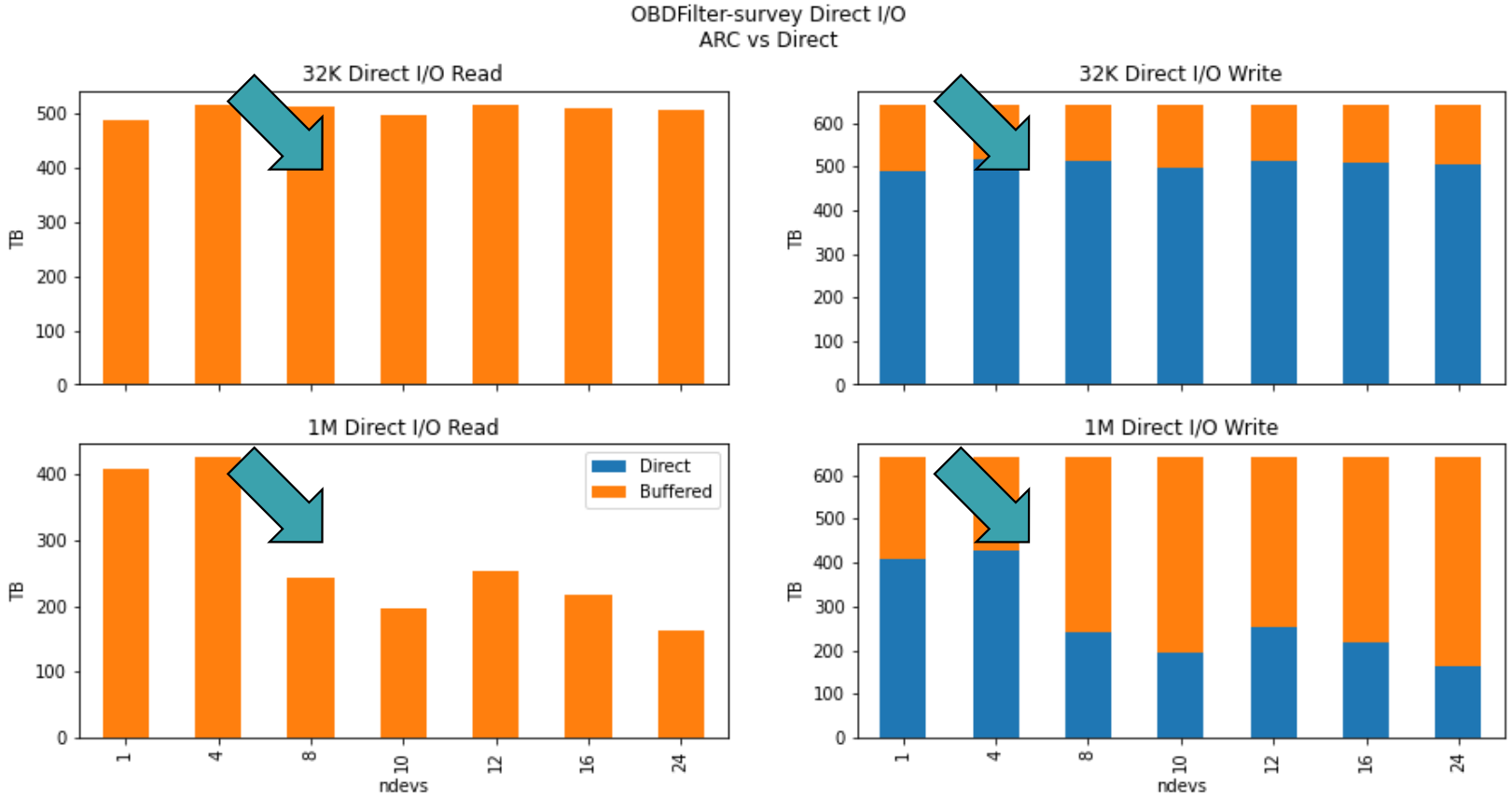
OBDFilter Striped ZFS OSD
(PCI-Aware)

- Ran OBDFilter survery on an OST with these settings.
  - 1 to 16 objs
  - 2048 threads
  - 512 MB/obj
- Some interesting read results
  - Both 32KB and 1MB get substantial better performance.
- Write performance continues to decrease.



Direct I/O vs ARC usage

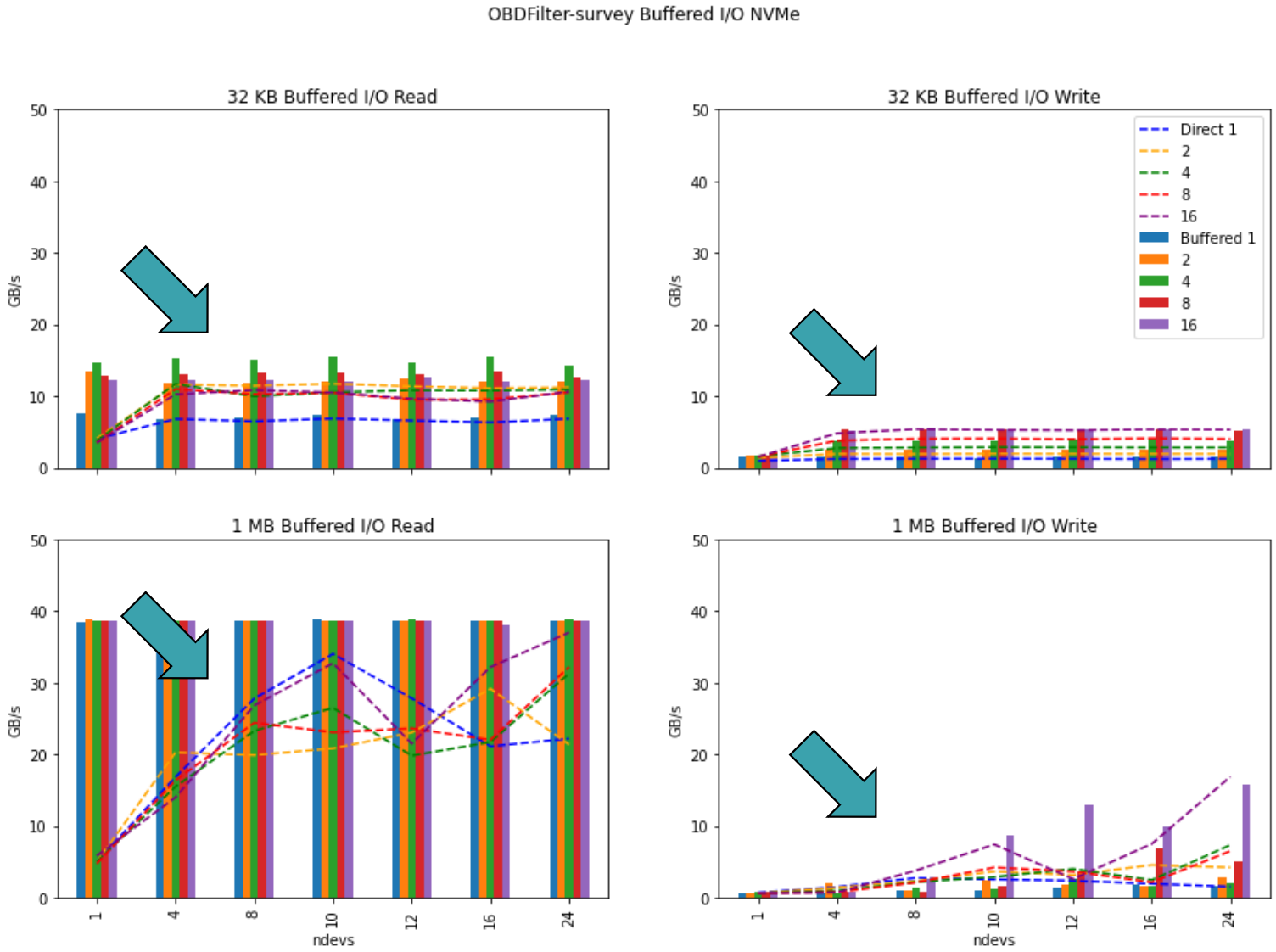
- Reads are better because they are all coming from the ARC and Lustre.
  - All buffered.
- Writes are a mixture of direct and buffered I/O.
- The ARC usage illustrates an implementation issue with the Lustre patch.
  - It is not properly aligning reads and writes to utilize Direct I/O.



Comparison to Buffered

(PCI-Aware)

- Buffered performance is given by the bars and the dashed lines are the direct I/O from before.
- The reads show performance improvements.
  - Probably due to the intentional usage of the ARC.
  - Before there was latency, due to failing over to the buffered I/O path.
- Writes are almost all the same.
  - Indicates performance problems are elsewhere.



Summary

- Is it worth it to use ZFS Direct I/O in the OSD layer?
  - Taking full advantage of this new feature for Lustre's OSD layer has proven challenging as direct I/O places size and alignment constraints on data buffers as well as additional ZFS configuration concerns.
    - Delicate balance of record size.
    - Lustre patch needs work to ensure properly page and record alignment.
  - For our setup, buffered I/O has shown equal or better performance.
  - There has been substantial performance improvements in the rest of ZFS that may not make Direct I/O necessary.
  - Of course, Direct I/O is still in the early stages and could potentially improve in the future.

Acknowledgments

- Thanks To
  - Brian Behlendorf (LLNL)
  - Jeff Niles (ORNL)
  - James Simmons (ORNL)
- This research used resources of the Oak Ridge Leadership Computing at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.