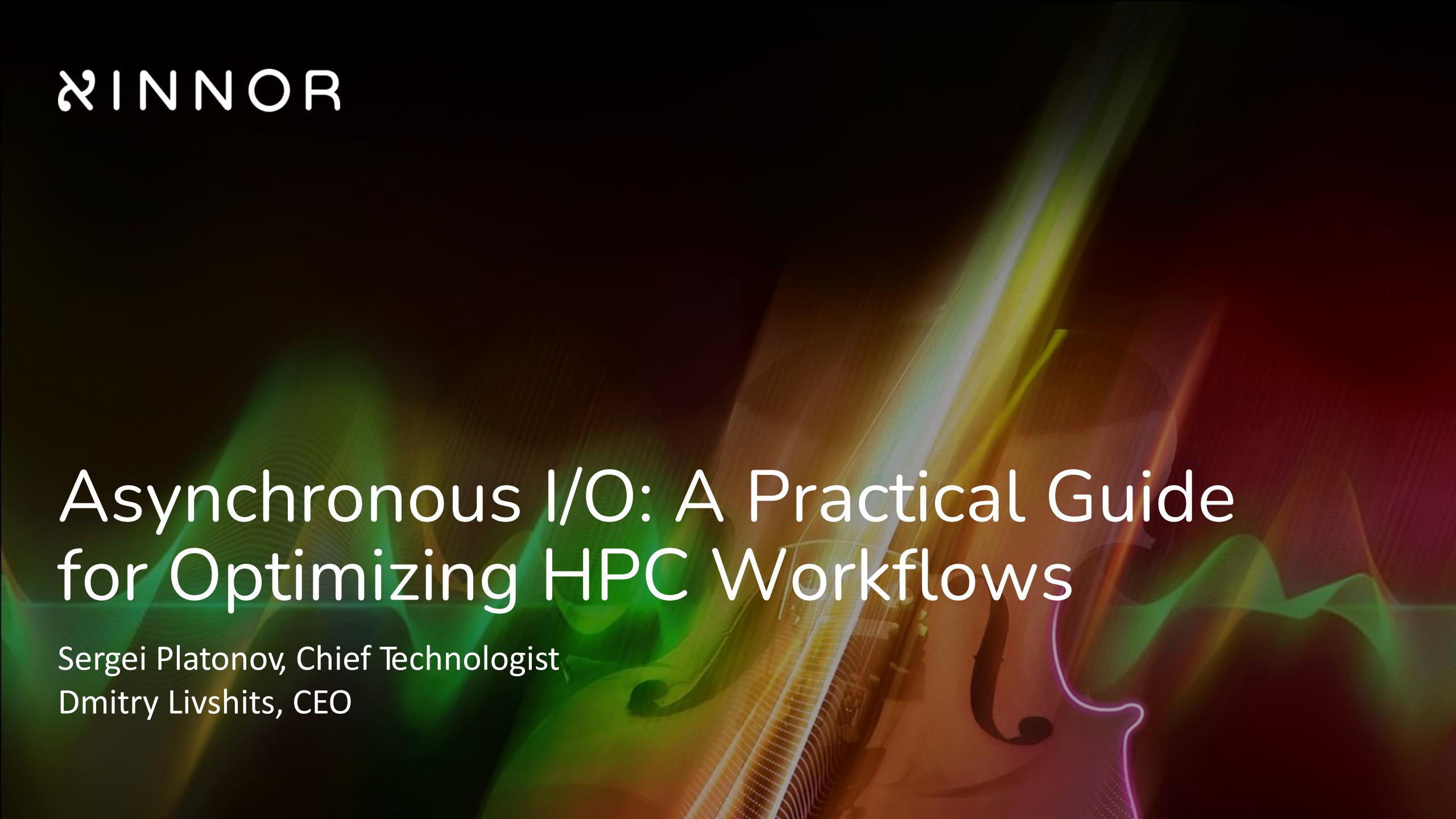


Asynchronous I/O: A Practical Guide for Optimizing HPC Workflows

Sergei Platonov, Chief Technologist
Dmitry Livshits, CEO

ABOUT XINNOR

Most Innovative Flash Memory Customer Implementation

- Founded in Haifa, Israel, May 2022
- Background: 10+ years of experience with software RAID design and mathematical research
- Mission: to be the fastest RAID Engine
- Team: Around 50 people; >35 are accomplished mathematicians and industry talents from Global Storage OEMs
- 20+ selling partners worldwide
- 100PB+ of end-customers data

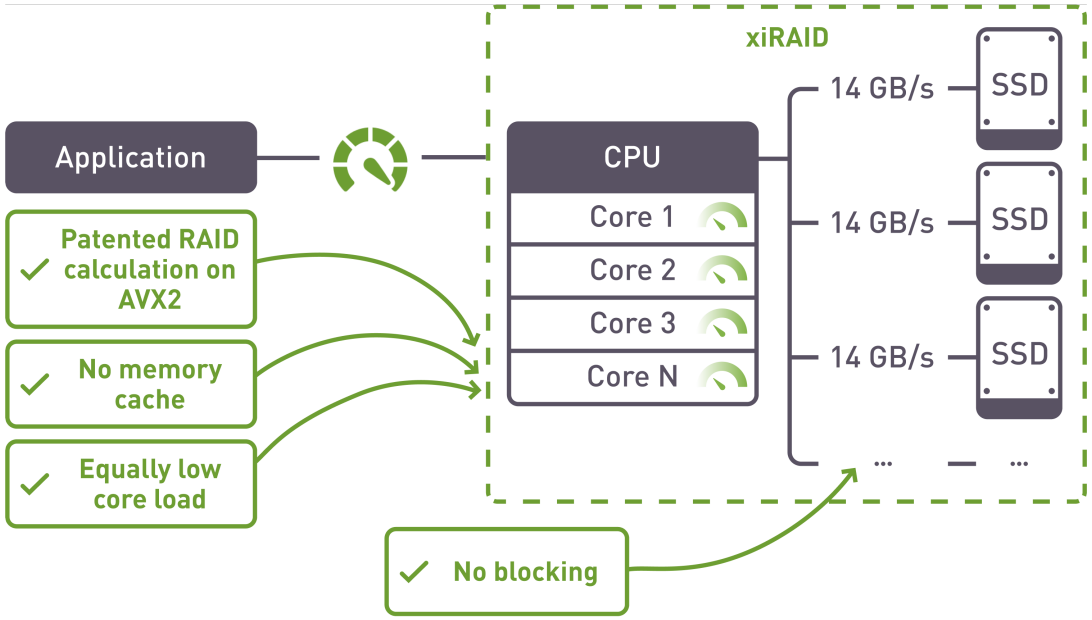
Technology partners

ATTO, Western Digital, ScaleFlux, KIOXIA, BeeGFS, ingrasys, TUXERA, LINBIT, NVIDIA, DapuStor

WHAT WE DO

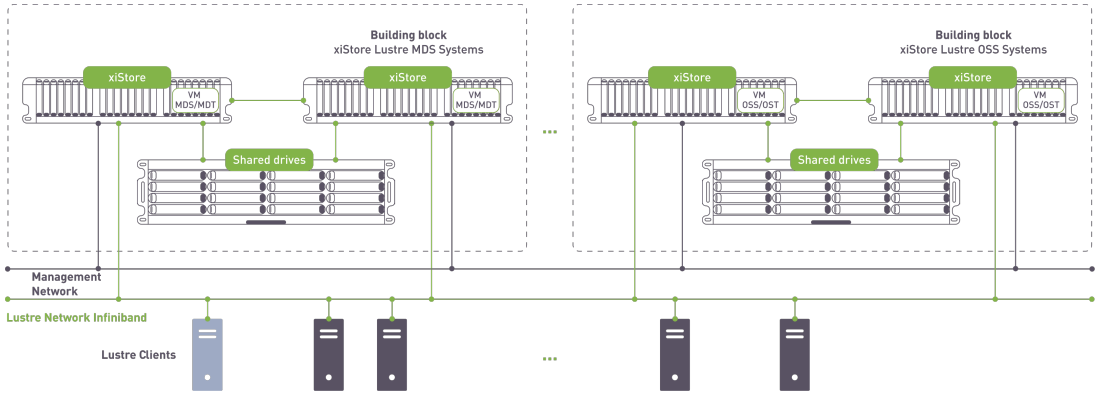
xiRAID

The fastest and most reliable software RAID for NVMe



xiSTORE

A software Defined Storage (SDS) solution tailored to HPC/AI workloads for both HDD and SSD infrastructures



ABOUT OUR PARTNER: **CORE MICRO SYSTEMS**

- Founded in Japan, 1992
- Development, manufacturing and maintenance of server and storage solutions
- Business results
  - Industrial measurement market – 100PB+ per year
  - Academic super computer market – 10-80PB per year
- Certification
  - Quality Management System: ISO 9001:2015
  - Environmental Management System: ISO 14001:2015
  - Information Security Management System: ISO/IEC 27001:2013
  - And many more certifications
- Meet Mr. Toan Nguyen at LUG24

WHY WE RELY ON LUSTRE?

- We make high-performance block volume and we have customers from a variety of areas
- Recently, the demand for storage for HPC and AI tasks has been growing
- And our partners want to build shared file storage
  - For small installations with 1-2 DGXs/HGXs
  - And for cloud providers

The requirements for such solutions are

- For small installations
  - To saturate 400Gbit Interface from the client and 800Gbit in the near future
- Cloud providers want to get 20GBps for each host in virtualized environment
- To get as much small block IOps as possible for both solutions

Modern RAID performance capabilities

Performance with x86 Host

	Measured single drive performance	2x RAID5 theoretical performance	xiRAID 2x RAID5 performance	Efficiency
4K Random Read (Millions of IOPS)	2,7	65	65	**100%**
4K Random Write (Millions of IOPS)	0,7	8	8	**100%**
Sequential Read (GB/s)	14	336	310	**92%**
Sequential Write (GB/s)	6,75	149	144	**97%**

In joint testing with KIOXIA with 24 PCIe Gen5 drives, xiRAID scored the highest performance in the market, in both RAID5 and RAID6 configurations, using minimum CPU load for RAID calculations (3-9%)

WHAT CAN WE DO FOR THE LUSTRE COMMUNITY?

1. We will present Lustre performance in non-standard installations like Cluster in the Box
2. We will look at Lustre as an alternative to SAN
3. We will show how different parameters affect performance

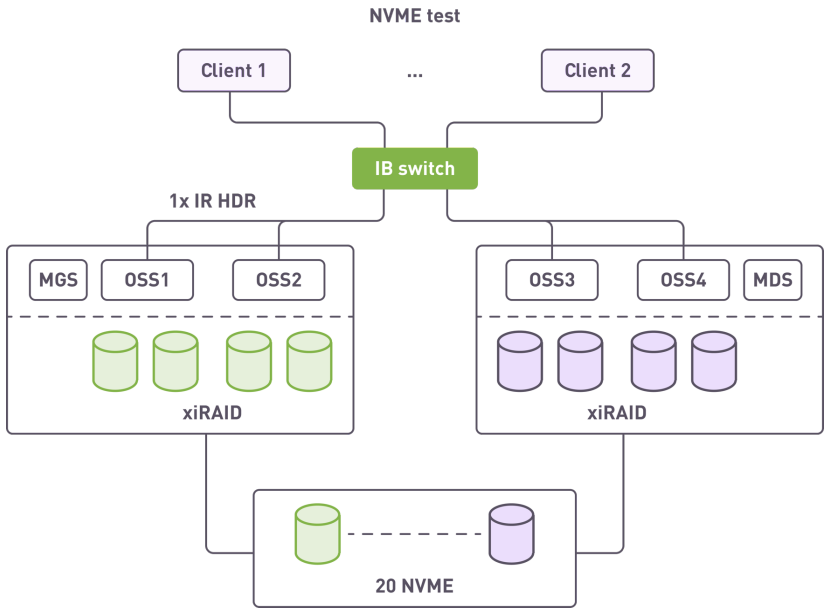
We will show how using AIO will allow us to achieve results

WHAT DO WE COMPARE?

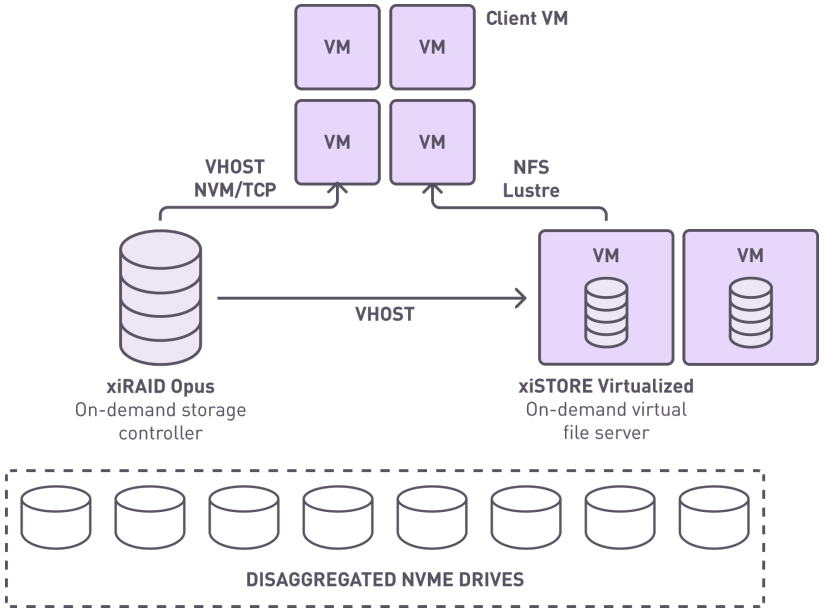
- Lustre 2.15.4 over ldiskfs
- Lustre 2.15.4 over zfs
- NFSoRDMA (v3 and v4.2)

ARCHITECTURES TESTED

1. Cluster-in-a-box solution



2. Virtualized solution



IOR
READ 1M

89 GB/s

IOR
WRITE 1M

63 GB/s

10 clients DIO TEST

How to improve single client small block IO performance?

TEST STAND CONFIGURATION

- **CPU**
  64-Core Processor per node (AMD 7702P)
- **Memory**
  256 GB RAM per Node
- **Networking**
  1 x MT28908 Family [ConnectX-6] per node
- **Drives**
  24x KIOXIA CM6-R 3.84TB: 1.6TB namespace per node

The clients are based on the same hardware and Rocky Linux 9

4 x OSS + MDS+MGS

SW Configuration

Rocky Linux 8 with Lustre 2.15.4.
RAID: 4 x RAID 6: 10 drives(8d+2p), ss=64k for OSS
2x RAID1 for MGS and MDS

IOR SINGLE CLIENT TESTS RESULTS

IOR DIO
WRITE 64M
13053 MiB

IOR DIO
READ 64M
12288 MiB

```
  Main   I/O
    PID USER      PRI  NI  VIRT   RES   SHR S  CPU%▽MEM%   TIME+  Command
 396587 root       20   0  391M  147M 21144 R  39.2  0.1  0:08.24 ./src/ior -w -r -t 64M -b 64G -e -o /lustre/iorfile --posix.odirect
```

IOR BUFFERED IO
WRITE 64M
3874 MiB

IOR BUFFERED IO
READ 64M
12757 MiB

```
  Main   I/O
    PID USER      PRI  NI  VIRT   RES   SHR S  CPU%▽MEM%   TIME+  Command
 396598 root       20   0  391M  144M 21244 R  99.4  0.1  0:12.98 ./src/ior -w -r -t 64M -b 64G -e -o /lustre/iorfile
```

IOR DIO
WRITE 4k
6542 IOps

IOR DIO
READ 4k
6742 IOps

IOR BUFFERED IO
WRITE 4k
7359 IOps

IOR BUFFERED IO
READ 4k
556629 IOps

Buffered IO write results vary from 2k to 28k iops during test
CPU load 6-100% during test

EXISTING IO BENCHMARK ISSUES

- Buffered IOs are demanding on CPU
- DIOs create uneven load on Storage, which is bad for both HDD and NVME
- We can scale performance by adding more IO threads but HDDs and Read-Intensive SSD don’t like such a approach
- The second option is to increase IO size, but that doesn't always work either and can create new problems
- Using libaio and io_uring theoretically will improve CPU utilization and overall performance

Today we want to demonstrate how the move to AIO will enable us to achieve our goals.

OUTCOMES 1

1. We can achieve the required performance on large sequential IOs with the existing approach.
2. A single thread limit is 13GBps
3. DIO gives more stable performance
4. We are far from block device performance on small random IOs

Let’s look at async IOs

TESTING METODOLOGY

- 4k random reads and writes with fixed numjobs=1 and variable iodepth
- 4k random reads and writes with fixed numjobs=32 and variable iodepth
- 4k random reads and writes with fixed iodepth=1 and variable numjobs
- 1M sequential reads and writes with fixed numjobs=1 and variable iodepth
- 1M sequential reads and writes with fixed numjobs=32 and variable iodepth
- ioengines=libaio, io_uring, sync
- Variable Lustre client settings and Lustre OSS settings

Not all the numbers will be shown during presentation

CONFIGURATIONS DESCRITION

OPT_OSS – CHANGED OSS/OST SETTINGS, CLIENTS WITHOUT SETTINGS.

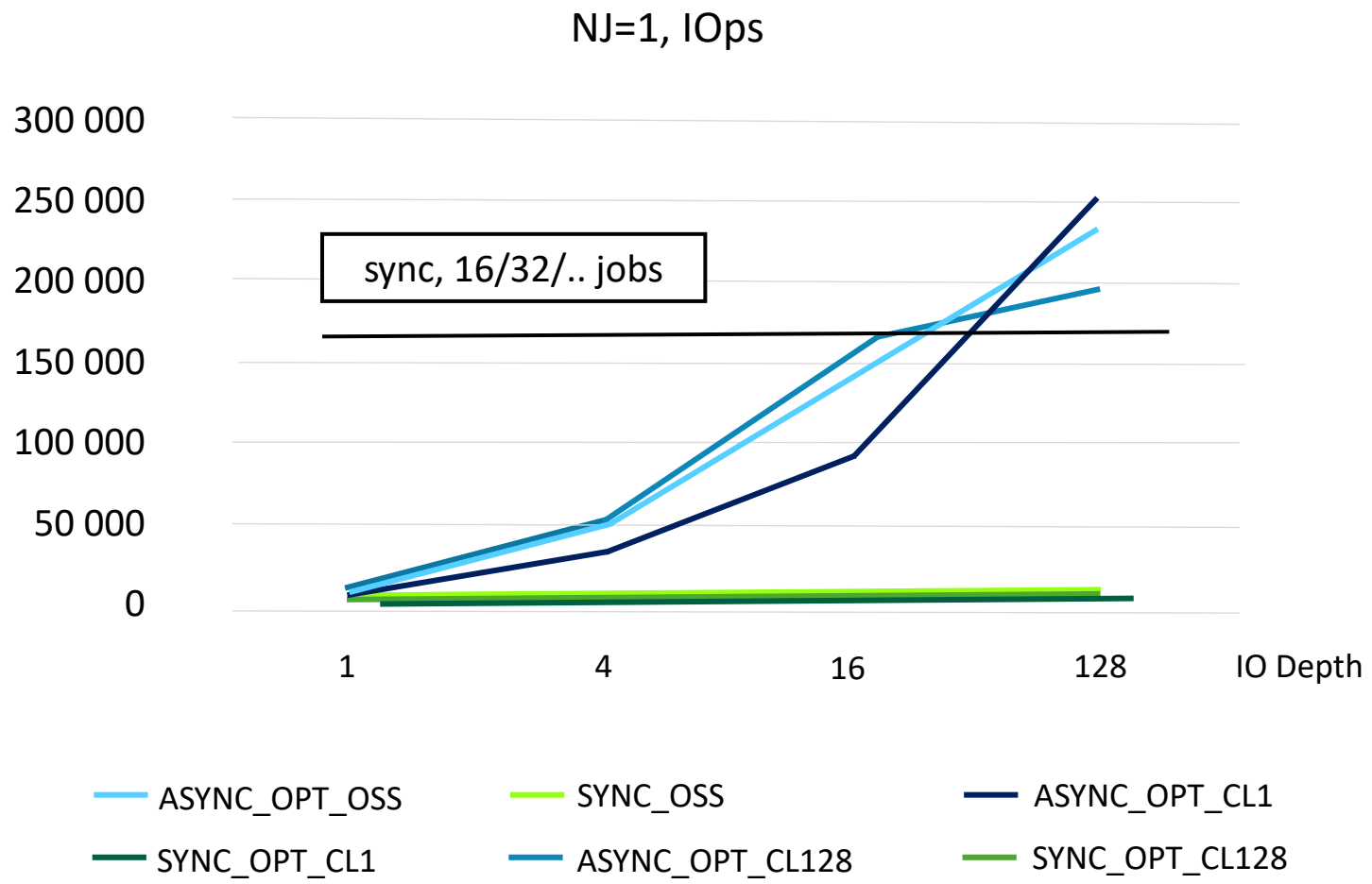
OPT_CL1 – lctl set_param osc.*.max_pages_per_rpc=**4096** osc.*.checksums=**0** osc.*.max_rpcs_in_flight=1

OPT_CL128 – lctl set_param osc.*.max_pages_per_rpc=**4096** osc.*.checksums=**0** osc.*.max_rpcs_in_flight=128

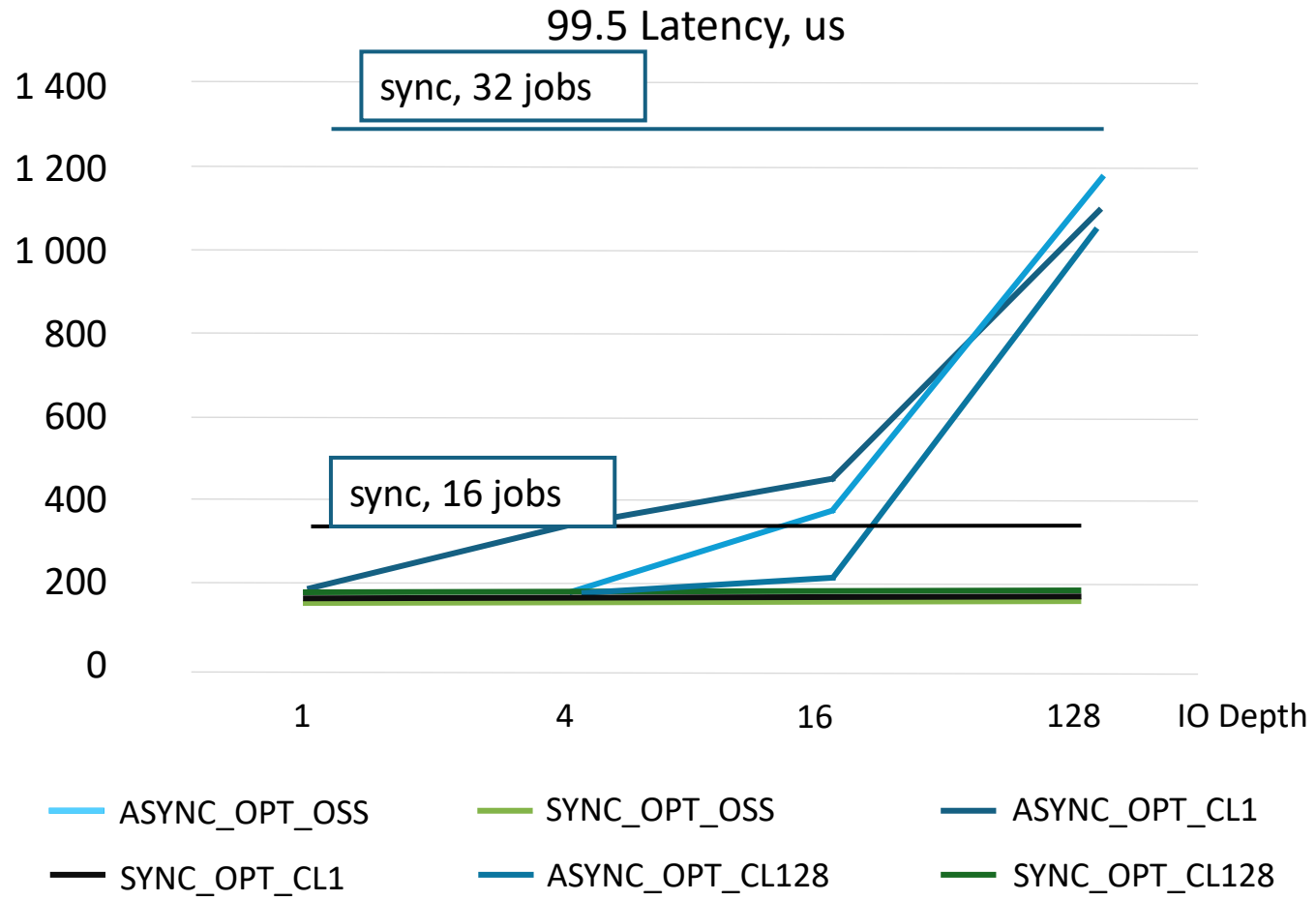
ASYNC – ioengine=libaio/io_uring

SYNC – ioengine=sync

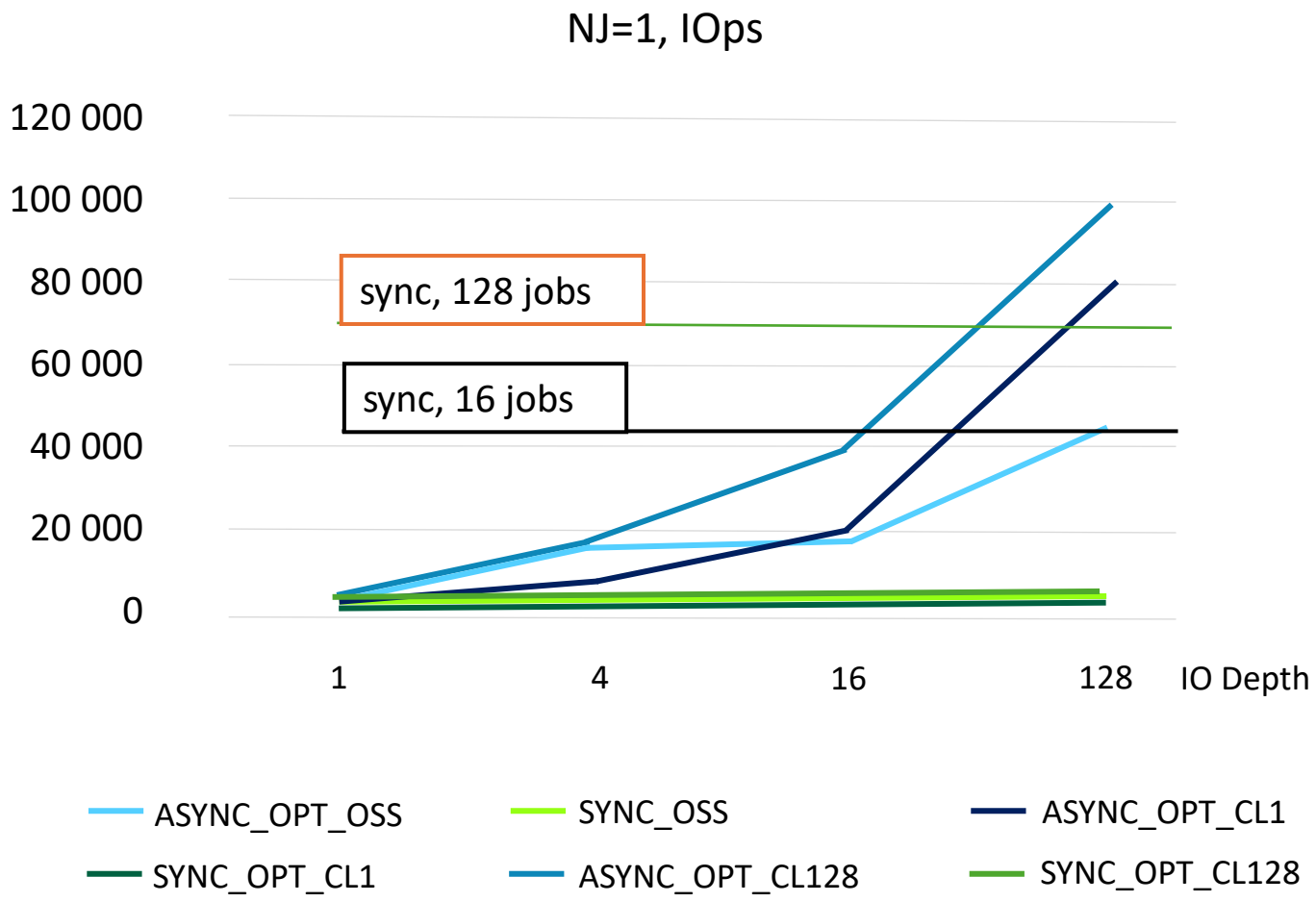
FIO RESULTS WITH DIFFERENT PARAMETERS. 4k RANDOM READS



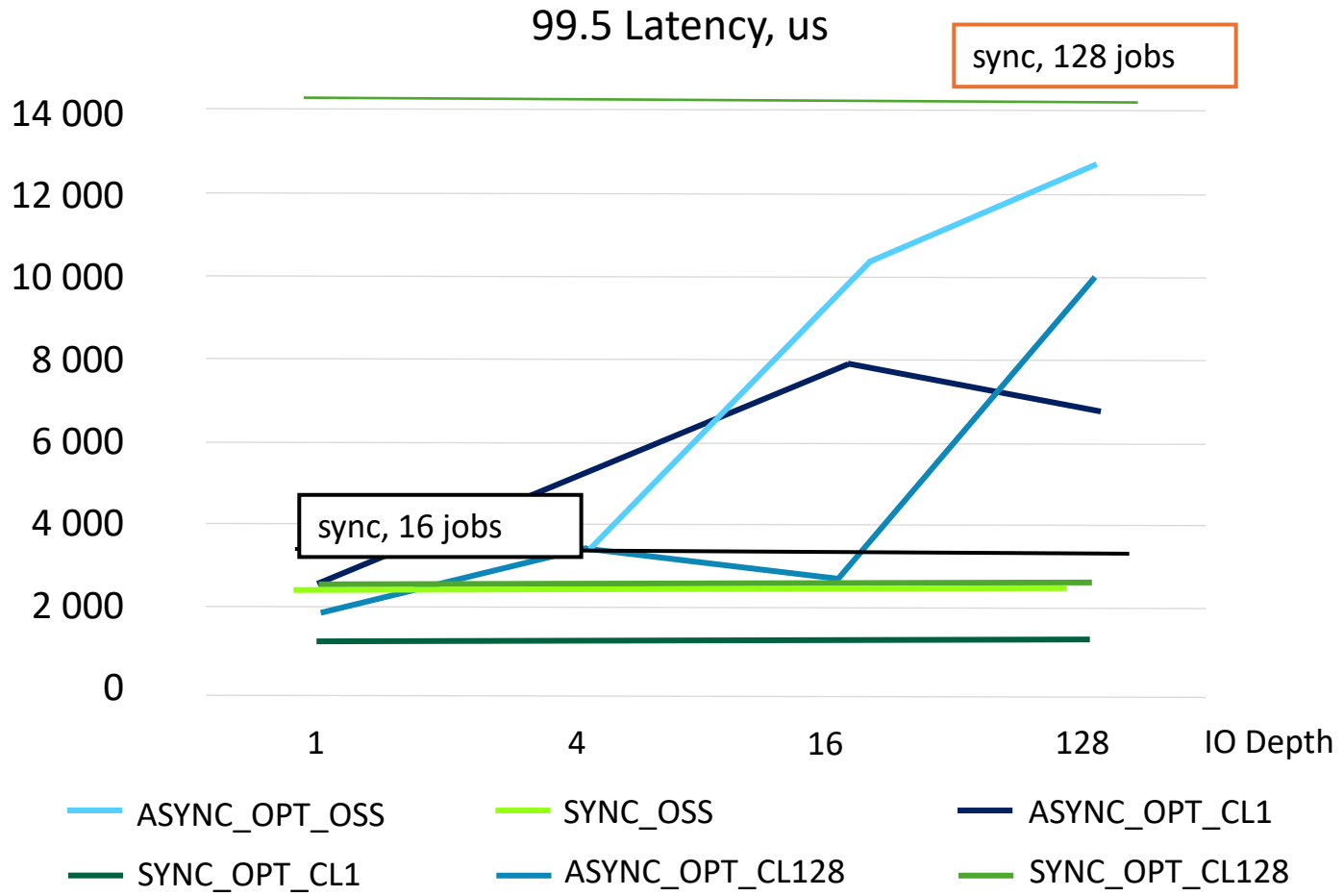
FIO RESULTS WITH DIFFERENT PARAMETERS. 4k RANDOM READS



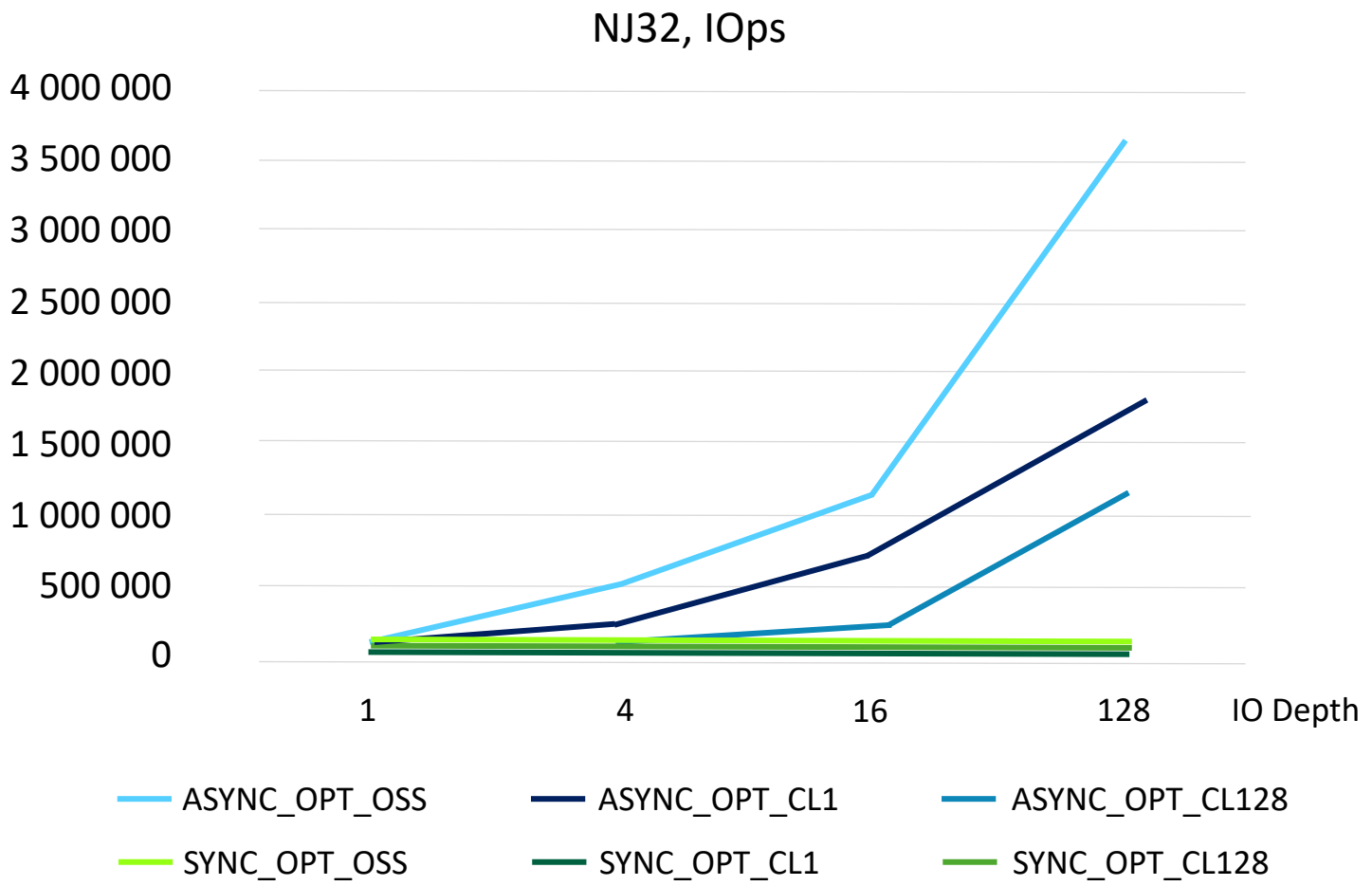
FIO RESULTS WITH DIFFERENT PARAMETERS. 4k RANDOM WRITES



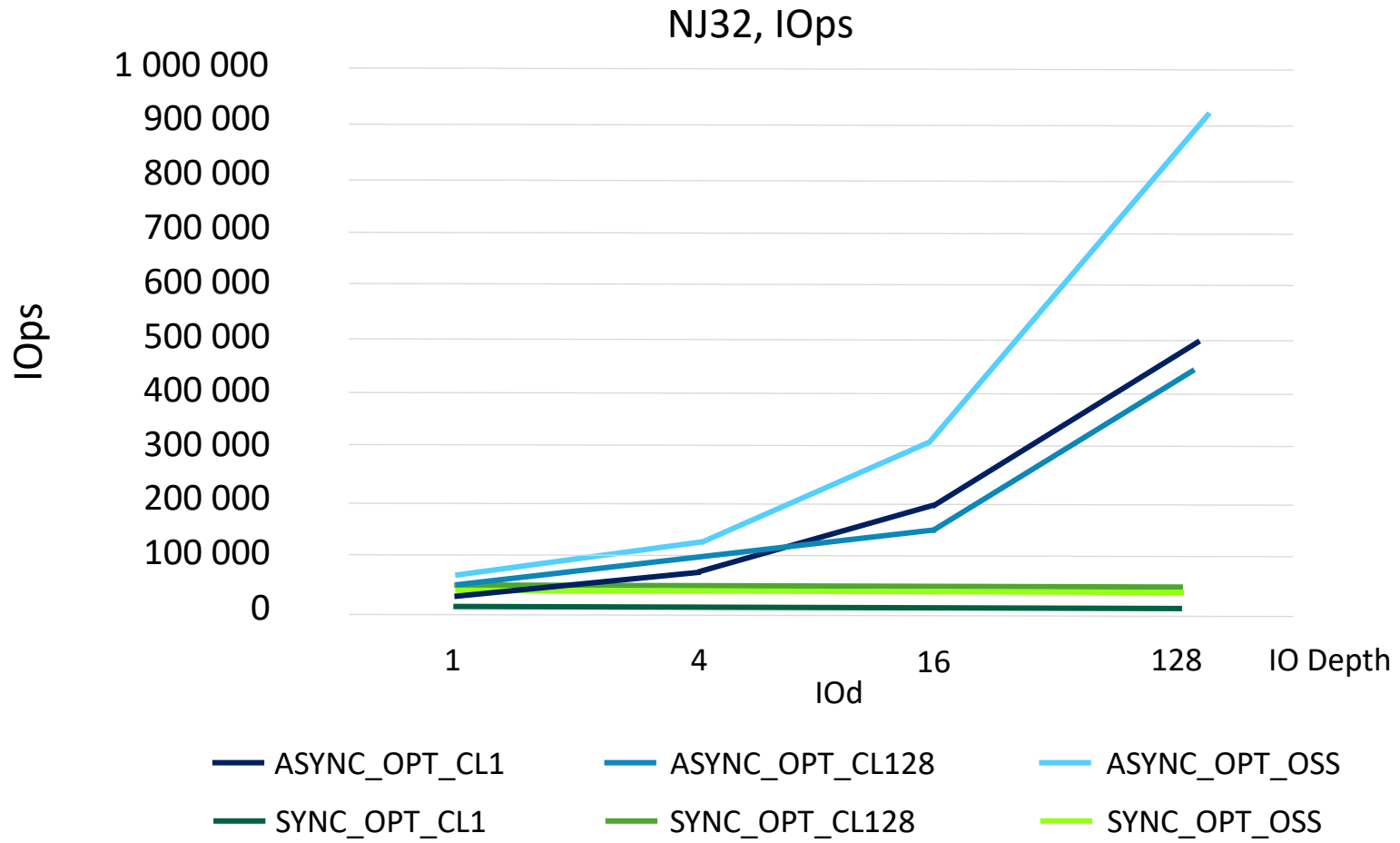
FIO RESULTS WITH DIFFERENT PARAMETERS. 4k RANDOM WRITES



FIO RESULTS WITH DIFFERENT PARAMETERS. 4k RANDOM READS



FIO RESULTS WITH DIFFERENT PARAMETERS. 4k RANDOM WRITES



OUTCOMES 2

1. The difference between DIO and AIO is not significant for large IOs (Not demonstrated on the chart)
2. On small block IOs the difference reaches several times. SYNC engine scales well up to 16 jobs.
3. We achieved 49% of maximum for random writes (limited by drives performance)
4. And 46% of maximum for random reads (limited by 2x200Gbit HCA)
5. Performance is maximally affected by the client parameter

   max_rpc_in_flight: 8-24 show the best results

IO_URING OPTIONS

4k random reads, 1 JOB, QD=128, IOps

Settings	Lustre	XFS over NVMf
Ioengine=io_uring	235k	440k
ioengine=io_uring fixedbufs=1 registerfiles=1 sqthread_poll=1	220k	800k
Previous config + hipri=1	ERROR REPORTED	798k

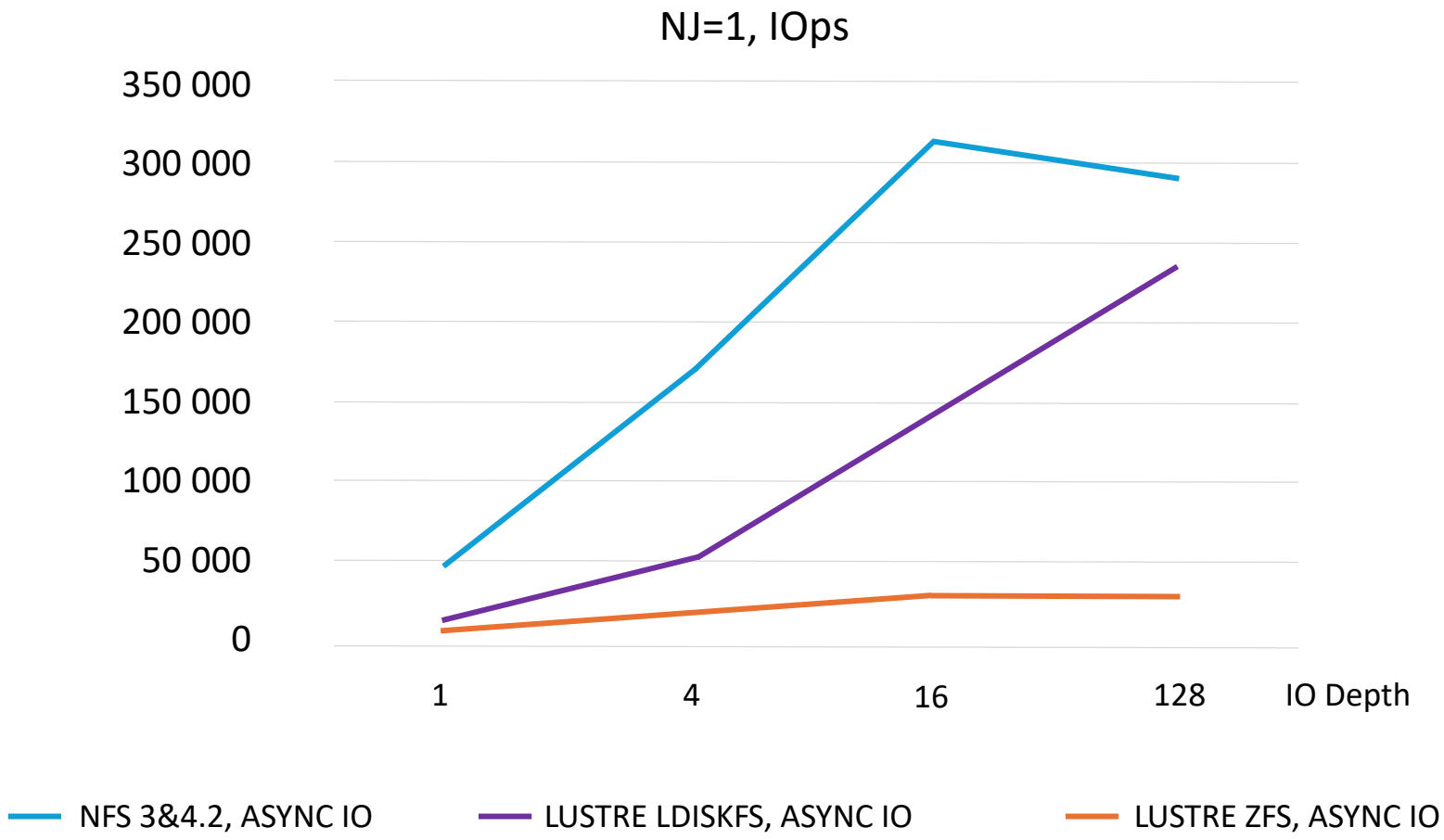
OUTCOMES 3

1. IO_uring with additional options provides significant performance improvement for XFS over NVMf devices
2. And does not work for Lustre
3. The is no difference between io_uring and libaio performance for Lustre

LUSTRE VS NFSoRDMA TESTING

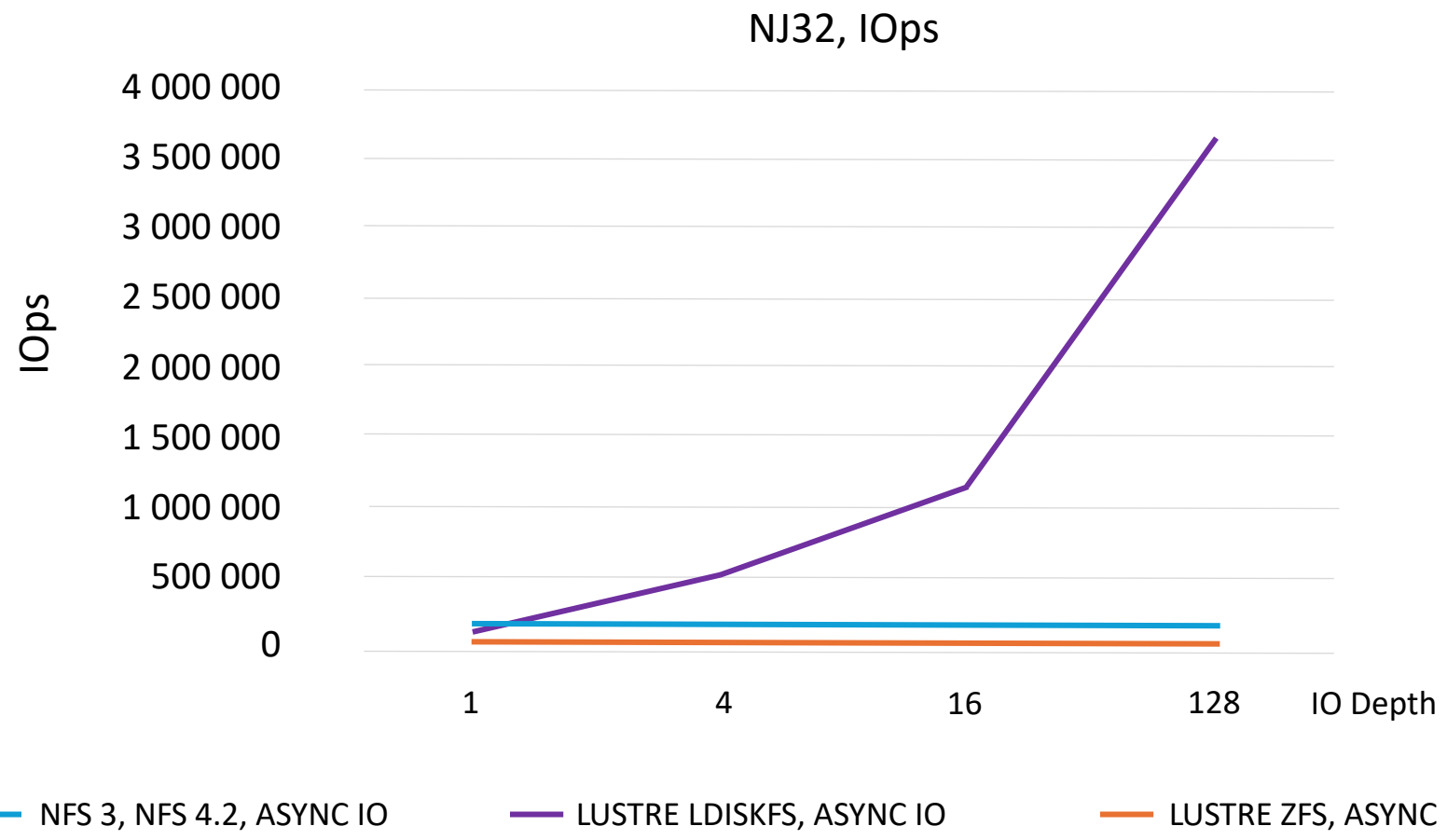
- We compared NFSoRDMA and Lustre 2.15.4 over lfiskfs and zfs.
- We used the same testing approach.
- We mounted NFS with sync and async options.
- We changed the NFS server and client settings.
- We observed no difference between sync and async mount options for reads (which is expected).
- We observed no difference between NFS3 and NFS4.2 in most cases.

LUSTRE VS NFS3 VS NFS 4.2, 4k READ IOs



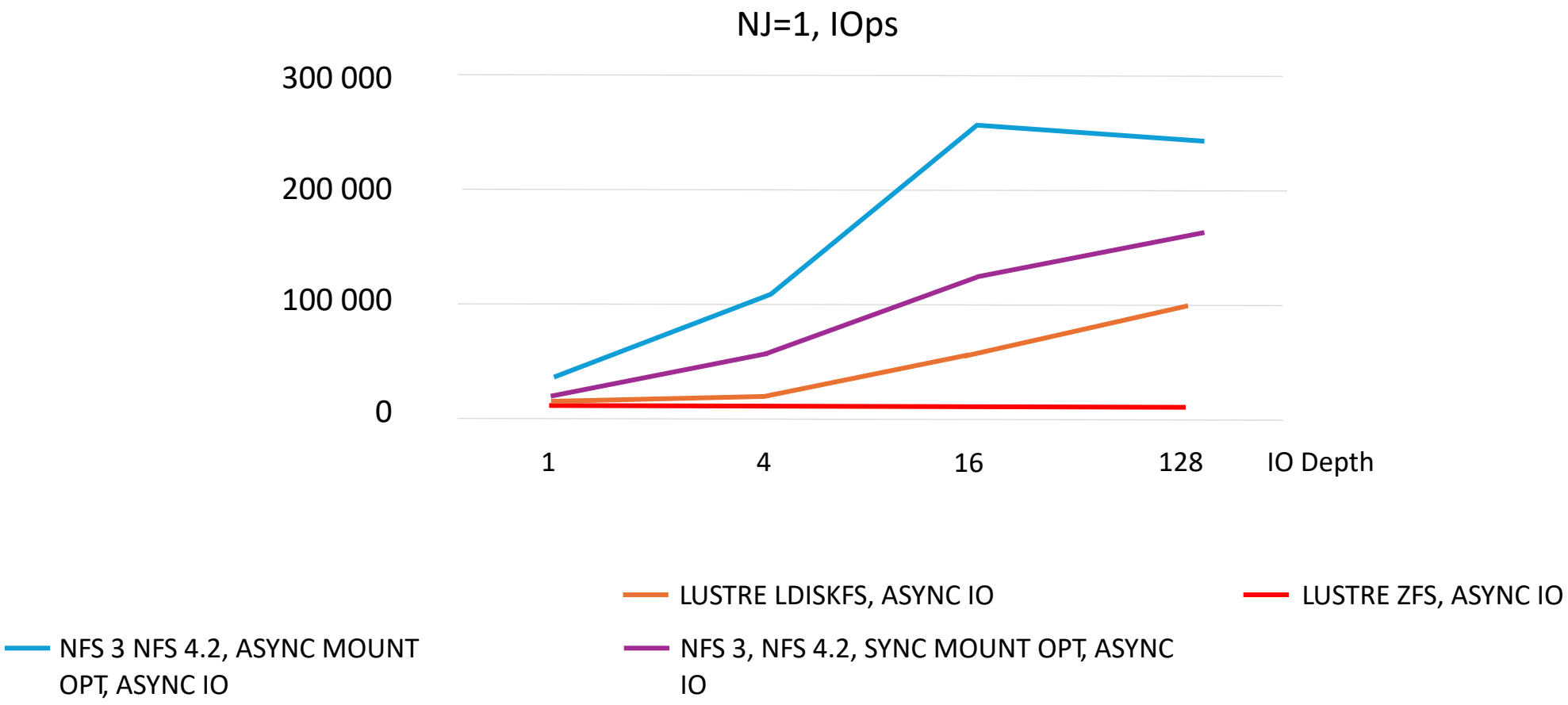
FOR AIO READS WE SEE NO DIFFERENCE BETWEEN LIBAIO and IO_URING, NFS VERSIONS AND NFS MOUNT OPTIONS

LUSTRE VS NFS3 VS NFS 4.2, 4k READ IOs

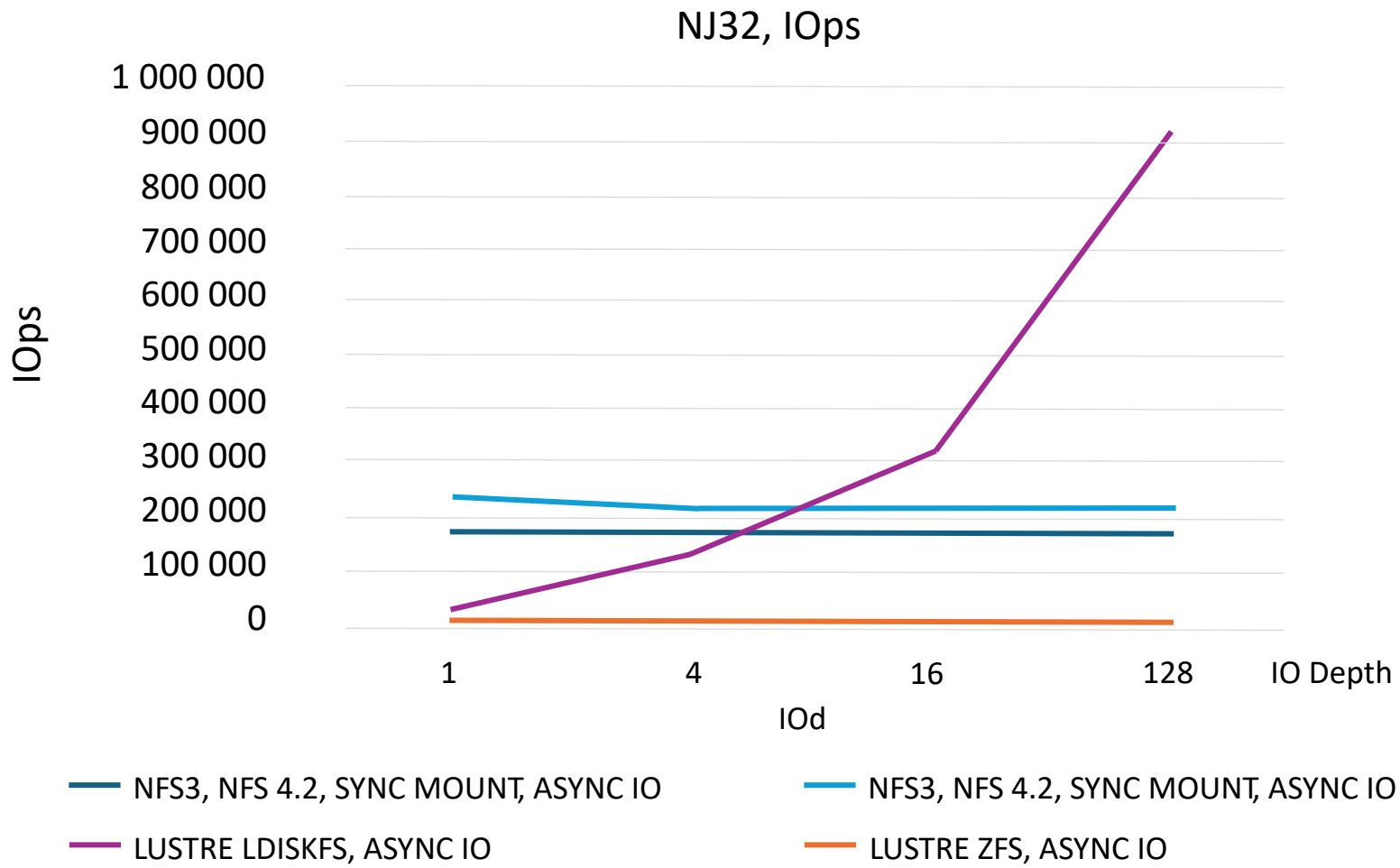


FOR AIO READS WE SEE NO DIFFERENCE BETWEEN LIBAIO and IO_URING, NFS VERSIONS AND NFS MOUNT OPTIONS

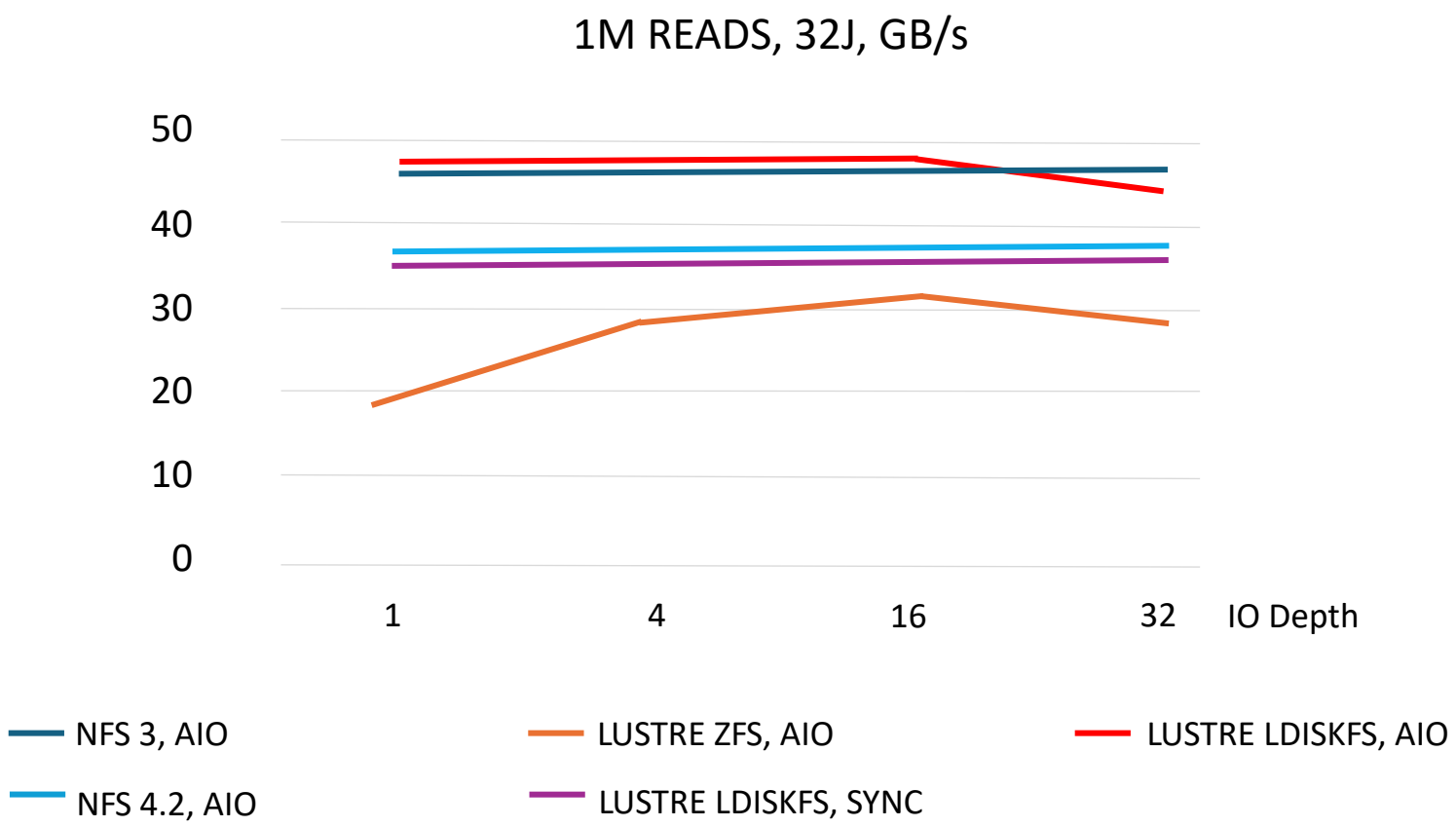
LUSTRE VS NFS3 VS NFS 4.2, 4k WRITE IOs



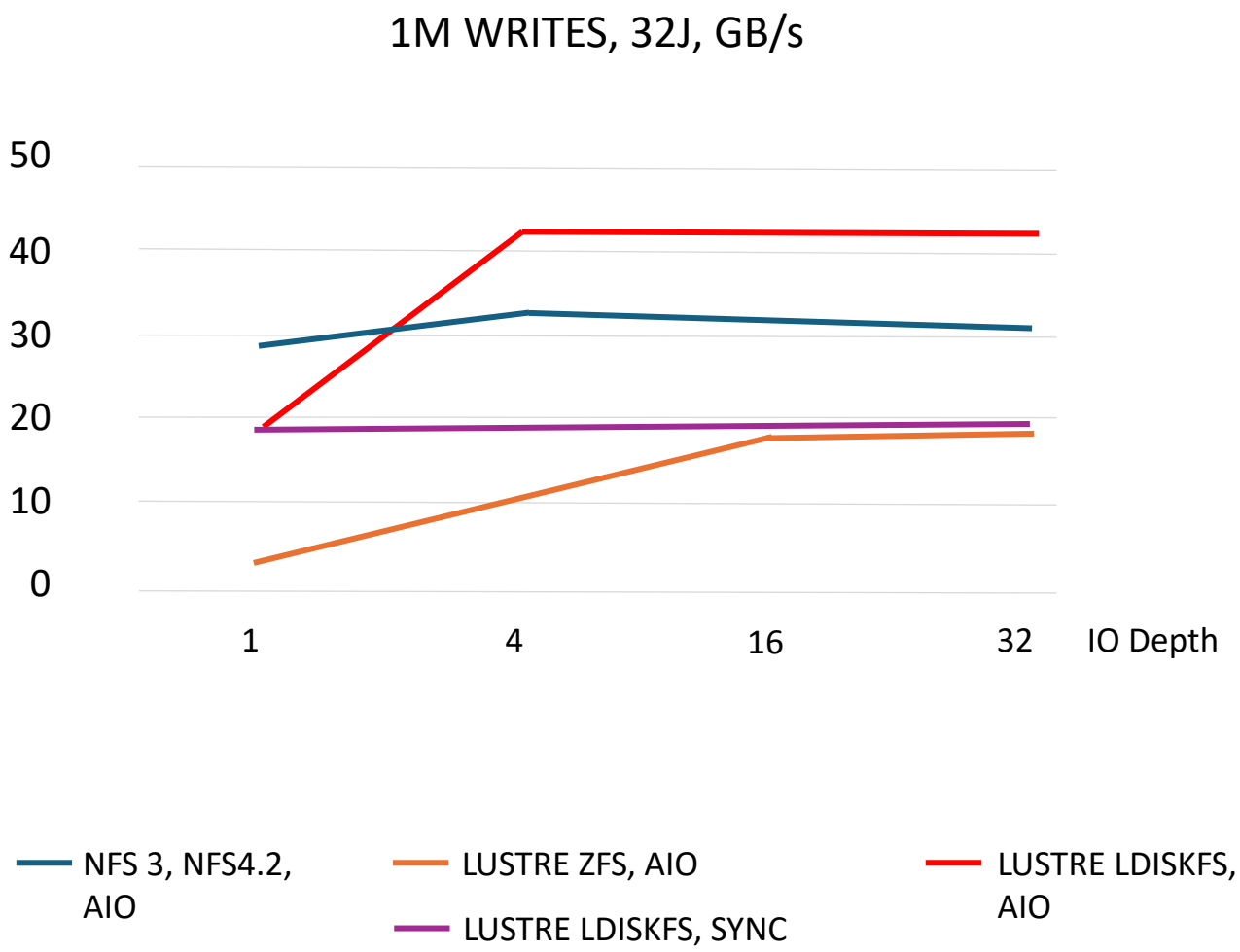
LUSTRE VS NFS3 VS NFS 4.2, 4k WRITE IOs



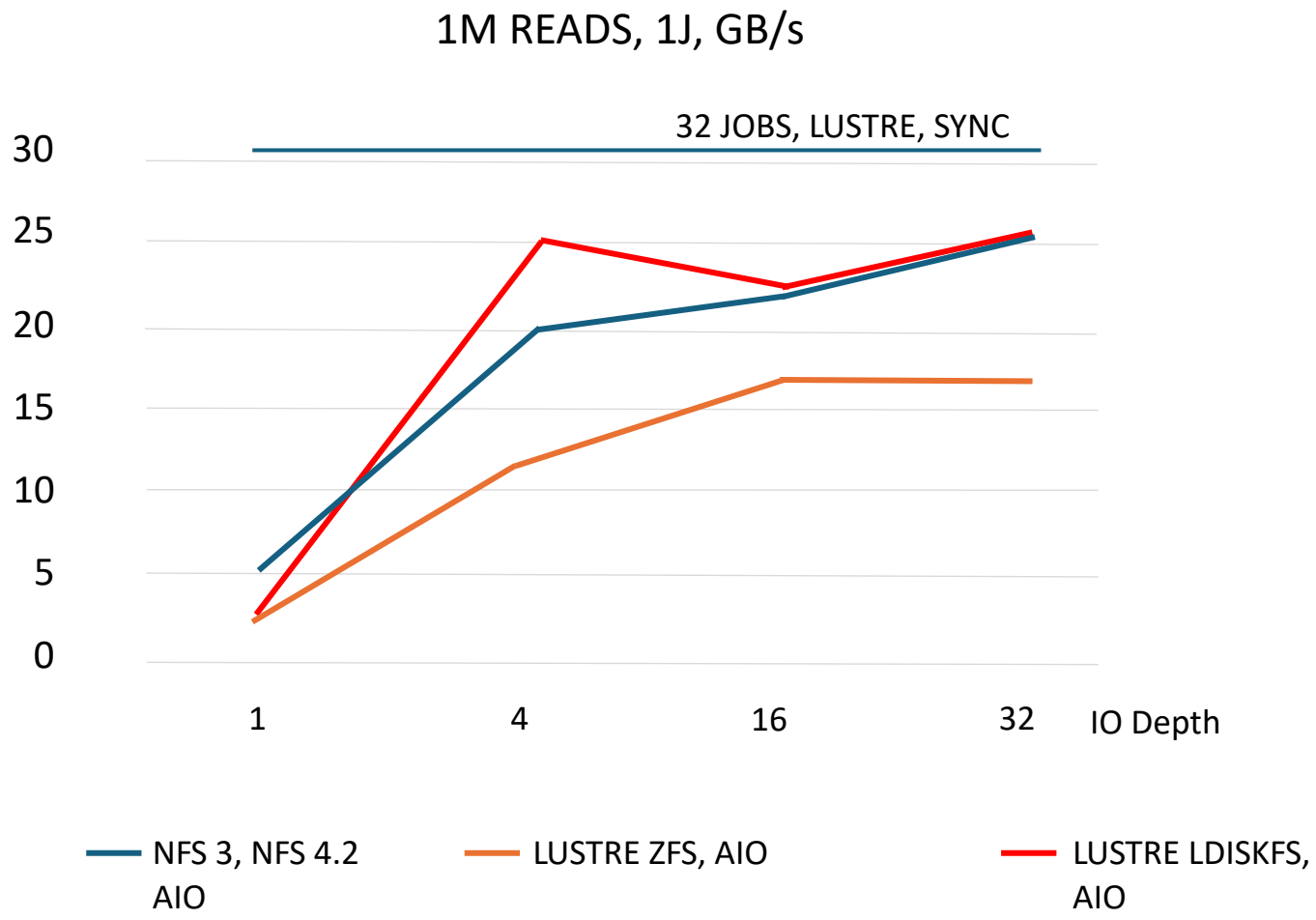
LUSTRE VS NFS, 1M SEQUENTIAL READS



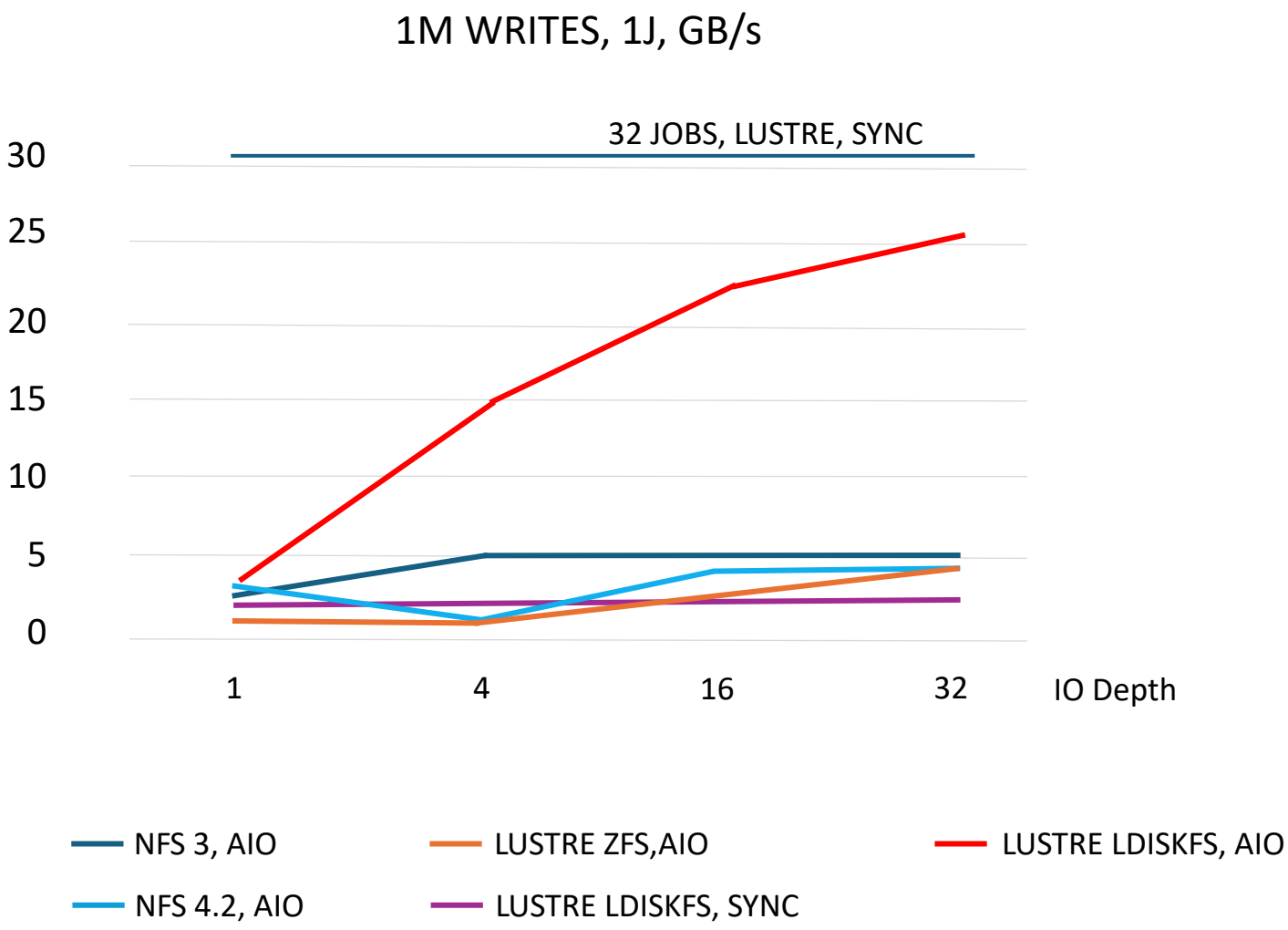
LUSTRE VS NFS, 1M SEQUENTIAL WRITES



LUSTRE VS NFS, 1M SEQUENTIAL READS



LUSTRE VS NFS, 1M SEQUENTIAL WRITES



OUTCOMES 4

1

ZFS does not allow to achieve the required performance numbers on small block IOs.

2

The DIRECTIO patch does not improve the situation for small IO

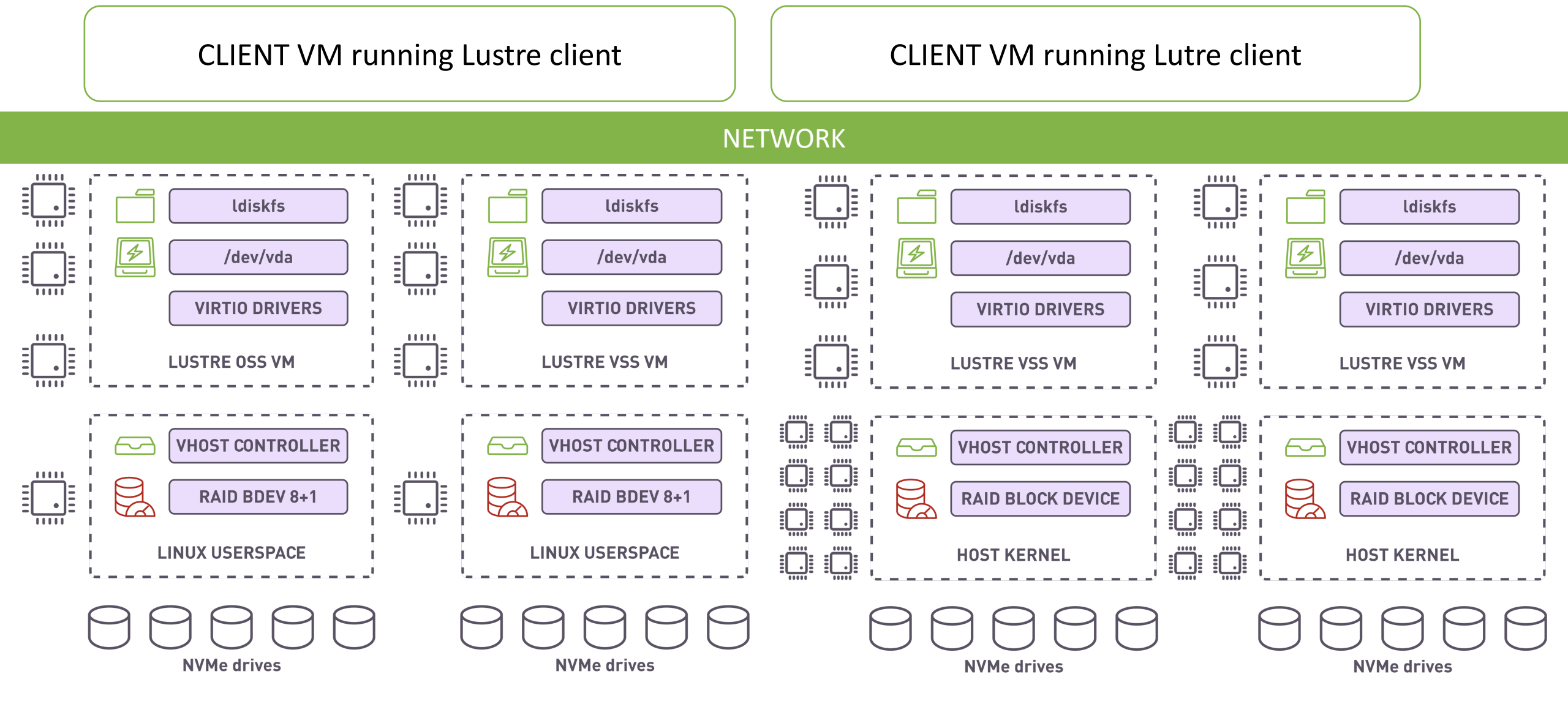
3

NFS over RDMA performs better for NJ=1 and for iodepth=1

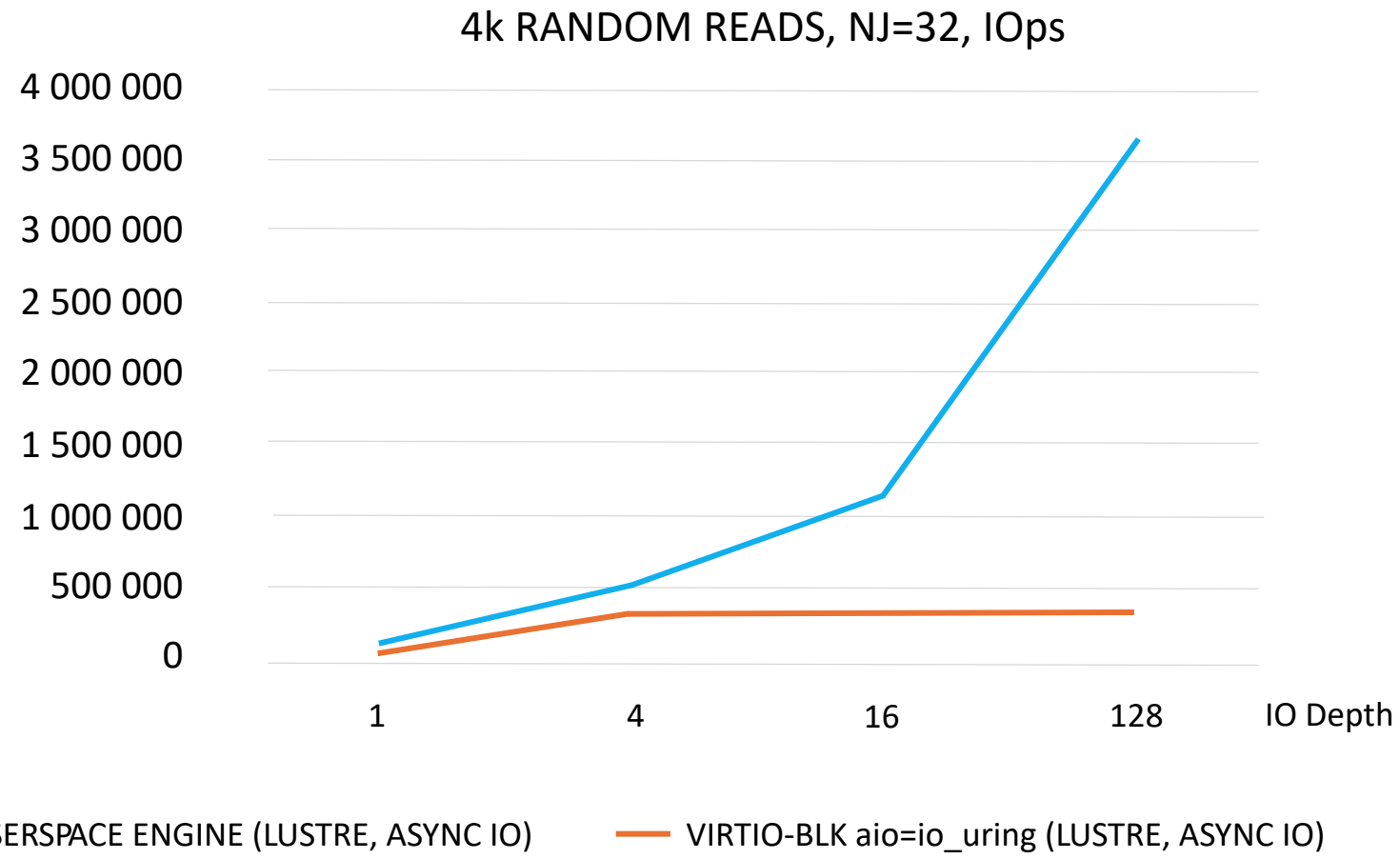
4

But NFS over RDMA does not scale well. Lustre significantly better as workload is increasing.

LUSTRE IN THE CLOUD ENVORIENMENT



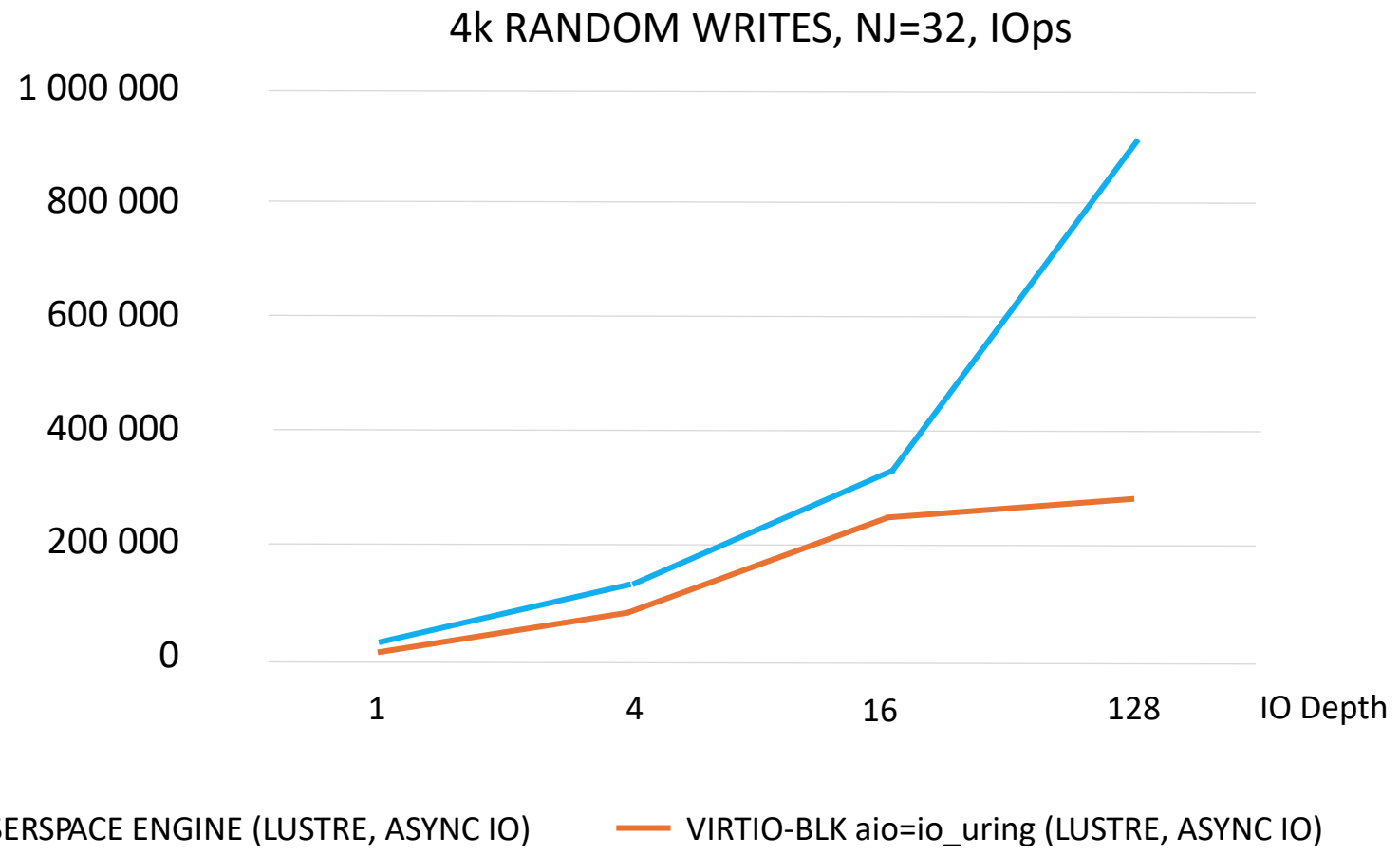
LUSTRE IN THE CLOUD ENVORIENMENT



USERSPACE ENGINE
READ 1M, 32J
44 GB/s

VIRTIO-BLK
READ 1M, 32J
44 GB/s

LUSTRE IN THE CLOUD ENVORIENMENT



USERSPACE ENGINE
WRITE 1M, 32J
43 GB/s

VIRTIO-BLK
WRITE 1M, 32J
43 GB/s

OUTCOMES 5

1. Virtio-blk helps achieve good performance numbers on large IOs

2. But kernel block device exposed to VM does not provide good performance on small block IOs.

3. Solution: running block device and vhost controller in user space.

   xiRAID OPUS solves the problem

FINAL THOUTGHTS

1. Lustre is a good solution for standalone storage and compete with SAN and NFS very well
2. AIO greatly improves performance
3. Many io_uring features do not work well and should be implemented to get better results
4. ZFS does not work well for small block AIO, ldiskfs should be used
5. For virtual environments, the userspace storage engine with vhost controller is the best solution

NEXT STEPS FOR THE COMMUNITY

1. To publish Ansible Playbooks to easily install our RAID engines, Lustre components, and benchmarking tools in minutes

2. To publish more detailed reports for different workloads and Lustre settings

3. To provide detailed analysis of io_uring performance

Thank you!

APPENDIX 1 Lustre OST/OSS settings

```
options lnet networks="o2ib(ib0)"

options ko2iblnd peer_credits=32 peer_credits_hiw=16
credits=256 concurrent_sends=64 nscheds=8

options libcfs cpu_npartitions=1

options ost oss_num_threads=128
```

```
lctl set_param *.*.brw_size=16
```

APPENDIX 2 Lustre OST/OSS settings

options lnet networks="o2ib(ib0)"

options ko2iblnd peer_credits=32 peer_credits_hiw=16 credits=256 concurrent_sends=64

lctl set_param osc.*.max_pages_per_rpc=4096 osc.*.checksums=0 osc.*.max_rpcs_in_flight=1/8/16/32/128