SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

Choosing Your AI Storage Protocol: A Deep Dive into SMB and NFS Performance, Tuning, and Overhead

Sergei Platonov
VP of Strategy, Xinnor

www.sniadeveloper.org

Our motivations

Our partners demand **high-performance storage** but run **small-to-mid clusters** where Lustre / IBM Storage Scale / BeeGFS would be an overkill to deploy and operate.

Where NAS fits:

- For **small installations:** a tuned NAS on top of fast local RAID/NVMe delivers the required throughput and simplicity.
- For **large installations:** modular NAS can act as a **component** (e.g., pNFS data servers) inside broader architectures.
- For NAS-on-Demand solution for GPU cloud installation

Our approach

Presenting a high-performance RAID (local or composable NVMe-oF), format correctly (XFS/EXT4, aligned), and export via **NFSv4.2 /** with either **TCP + nconnect** or **RDMA** to hit both streaming bandwidth and low tail latency.

Why do we need high performance NAS?

We need to keep GPU busy!

The most expensive part of modern Datacenter is GPU time

Different workloads require different storage performance characteristics

- Training
- Checkpointing
- RAG

Why NFS for AI

- **Ubiquity & simplicity**
  - Ships with every Linux distribution; one mount command and you're done
- **POSIX semantics**
- **Great fit for some AI I/O patterns**
- **Performance features, when needed**
  - NFSv4.1/4.2 sessions & delegations; server-side copy (v4.2); TCP multistreaming; NFSoRDMA, NFS LOCAL_IO
- **Operational efficiency**
  - Mature observability (nfsstat, mountstats, /proc/fs/nfsd), straightforward tuning (nfsd threads).
- **Security options**
  - From fast sec=sys to Kerberos (krb5/krb5i/krb5p) when compliance requires it.

Why also evaluate SMB

- Easy **AD/ACL** integration for teams and labs.

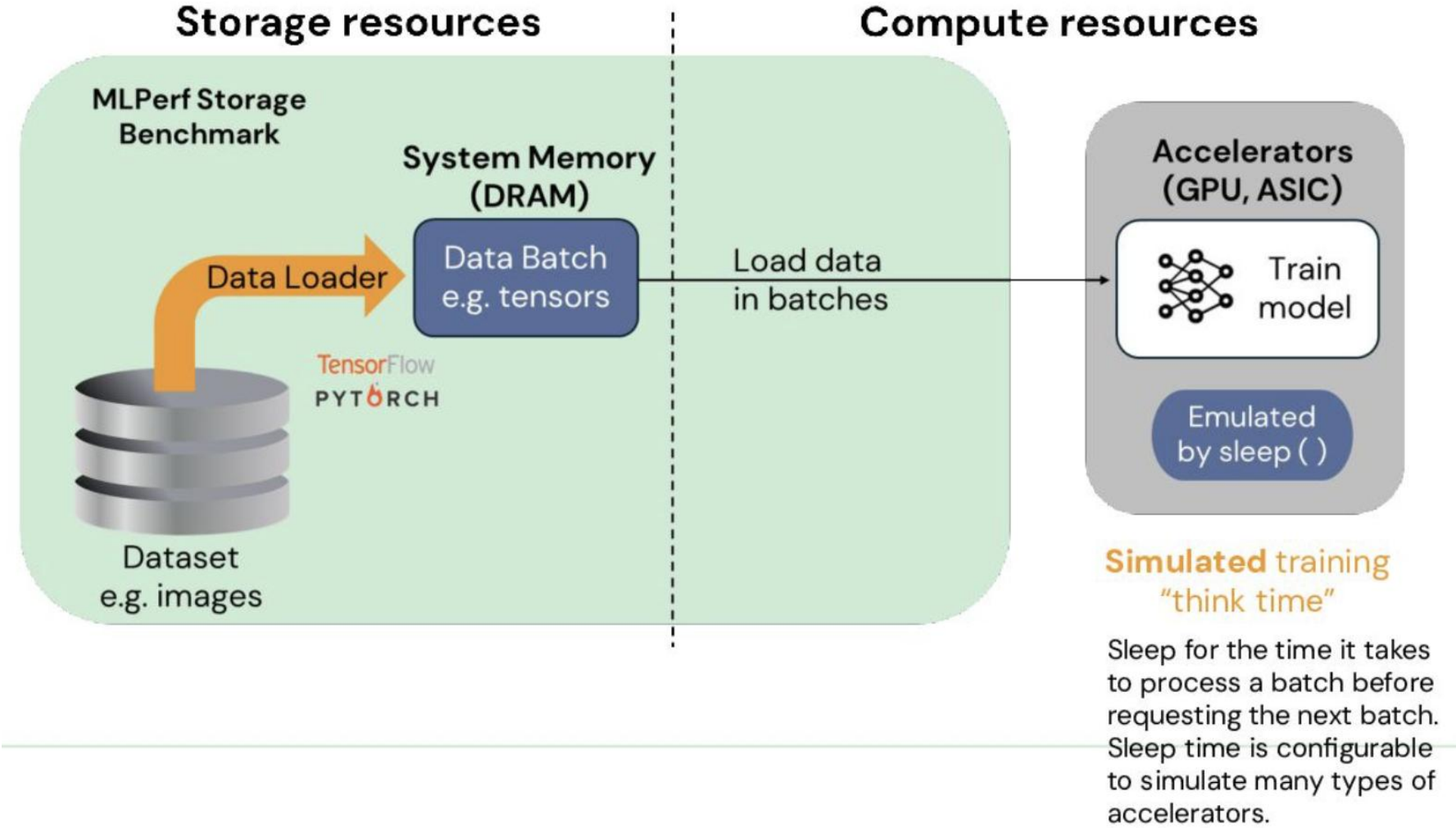
Performance features:

- **SMB3.1.1 Multichannel** ⇒ multiple TCP lanes per client (better NIC/CPU utilization).
- **SMB Direct (RDMA)** ⇒ low CPU, low tail latency (where supported).
- **Leasing/oplocks & durable handles** reduce network overhead; **Continuous Availability** for HA.

Architecture options:

- Standalone high-perf NAS.
- Scale-out gateways (e.g., IBM Storage Scale SMB).

MLPerf Storage Benchmark



Source: ML Commons, IT Press Tour 60

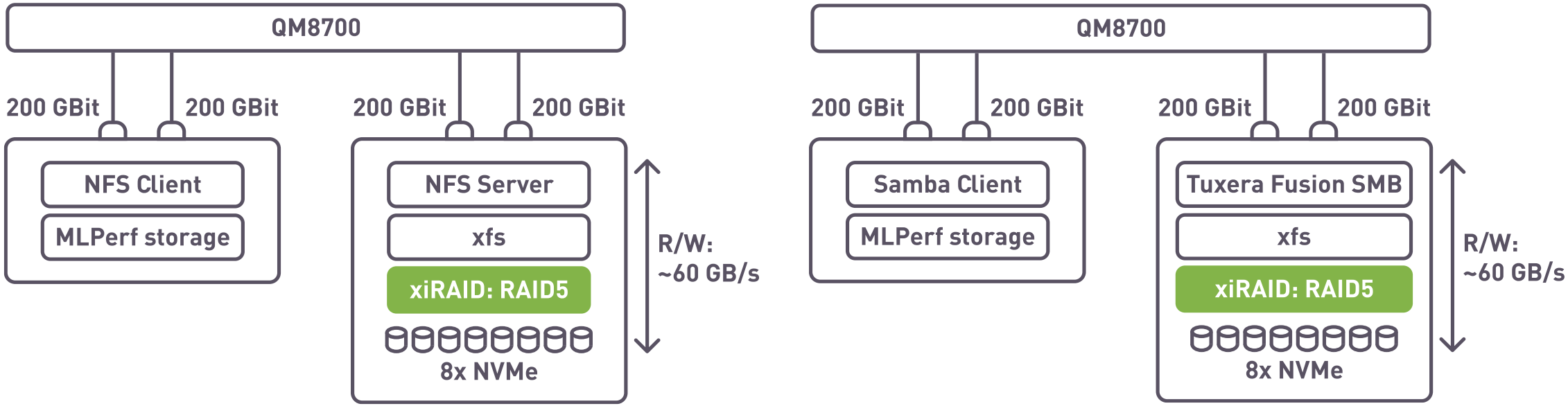
Workloads simulated by MLPerf Storage

Workload		Reference Network	Sample size	Framework	Reference Quality
Image segmentation (medical)	Synthetic – from KiTS19	3D-Unet	146 MB	PyTorch	maximize MB/s, and # of accelerators with >90% accelerator utilization
Image classification	Synthetic – from ImageNet	ResNet50	150 KB	Tensorflow	maximize MB/s, and # of accelerators with >90% accelerator utilization
Scientific (cosmology)	Synthetic – from CosmoFlow N-body simulation	Parameter prediction	2 MB	Tensorflow	maximize MB/s, and # of accelerators with >70% accelerator utilization

Workloads can be run with simulated NVIDIA A100 or H100 accelerators

Now MLPerf Storage 2.0 has added support for Checkpointing workloads.

Test bed description

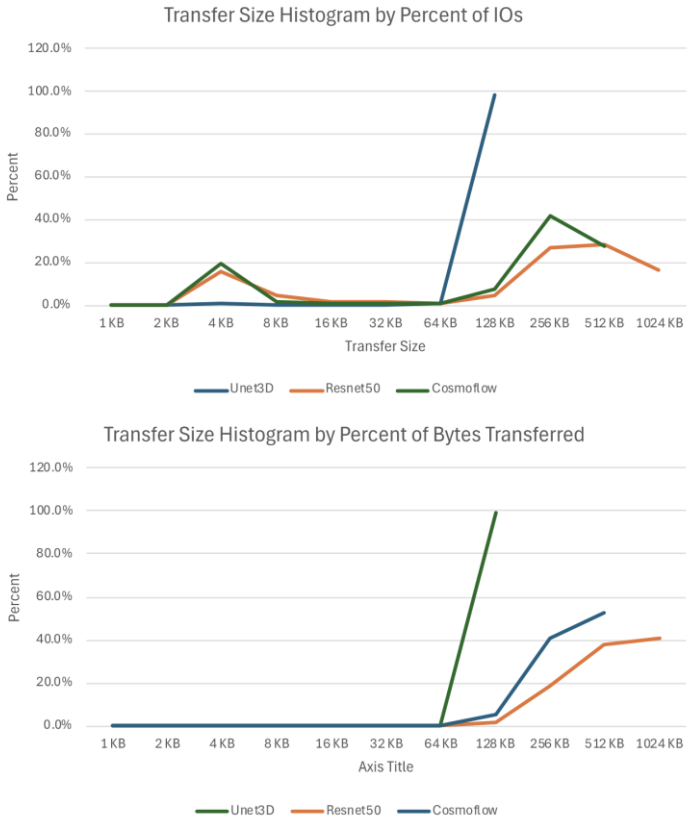


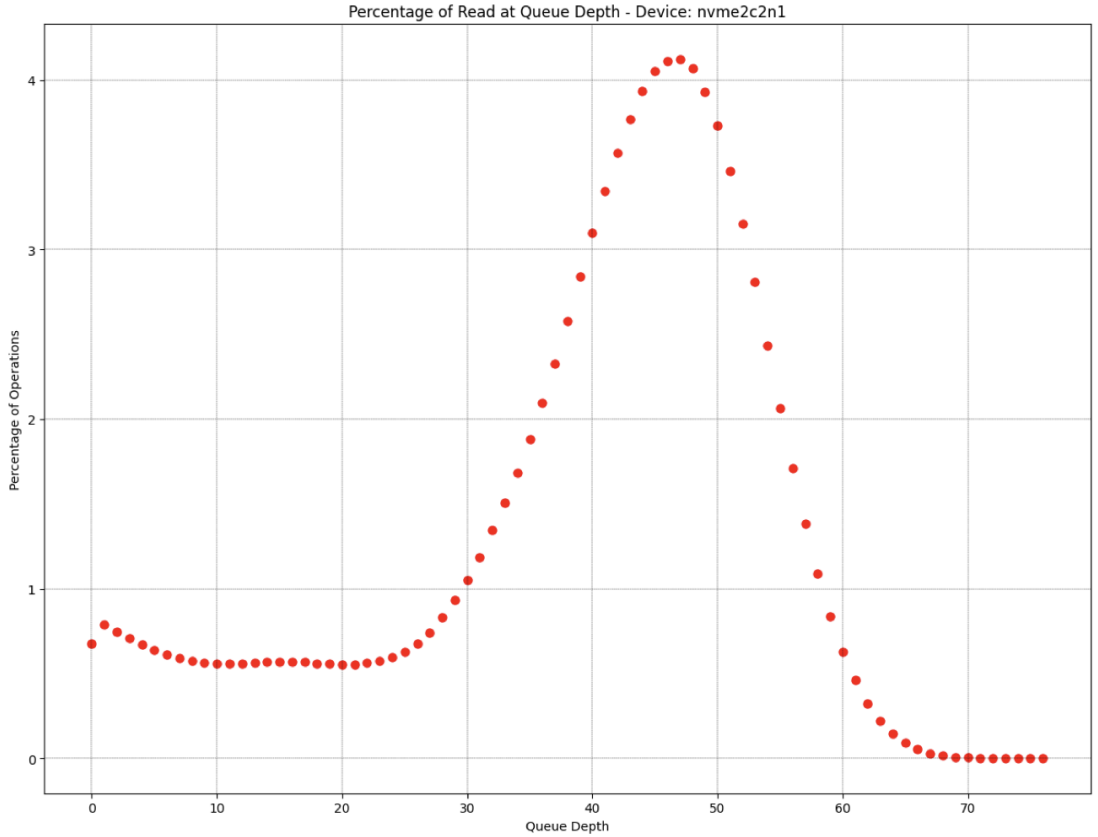
The node configuration: 48 CPU cores, 512 GB RAM, 8xPCIe 4.0 NVMe drives
Ubuntu 24.04 with a customized 6.16 kernel.

Limitations

- **Hardware.** The nodes used were neither the fastest nor the most modern, so their performance was limited.
- **Configuration.** Tests were conducted in the simplest setup – single server and single client.
- **Workloads.** Evaluation was restricted to two workloads:
  - Checkpointing – selected for its distinct read-write balance.
  - UNET3D – the only workload supporting O_DIRECT, and the most storage-intensive among MLPerf workloads.

Workload analysis





Source: https://files.futurememorystorage.com/proceedings/2024/20240808_AIML-303-1_Vaske.pdf

Benchmark parameters

Workflow: Calculate minimum dataset size → Generate the dataset → Run the benchmark → Generate report

What is "success": throughput (samples/sec) while keeping **AU ≥ 90%** (the benchmark's "passing" utilization threshold; results pages describe throughput at ≥ 90% AU).

We will focus on UNET3D model training and checkpointing as the most storage-intensive workloads.

- `mlpstorage training run --hosts 127.0.0.1 --num-client-hosts 1 --client-host-memory 512 --num-accelerators {variable} --accelerator-type h100 --model unet3d --data-dir unet3d_data --results-dir unet3d_results --param dataset.num_files_train=65000 reader.odirect=true reader.read_threads=8 reader.prefetch_size=4 --allow-run-as-root`
- `mlpstorage checkpointing run -rd ch_r3 -m llama3-405b --client-host-memory-in-gb 512 -np 36 -cf CP --allow-run-as-root --param parameters.checkpoint.fsync=true parameters.framework=pytorch parameters.model.parallelism.pipeline=32 parameters.model.parallelism.tensor=16`

How easy it is to do **badly**

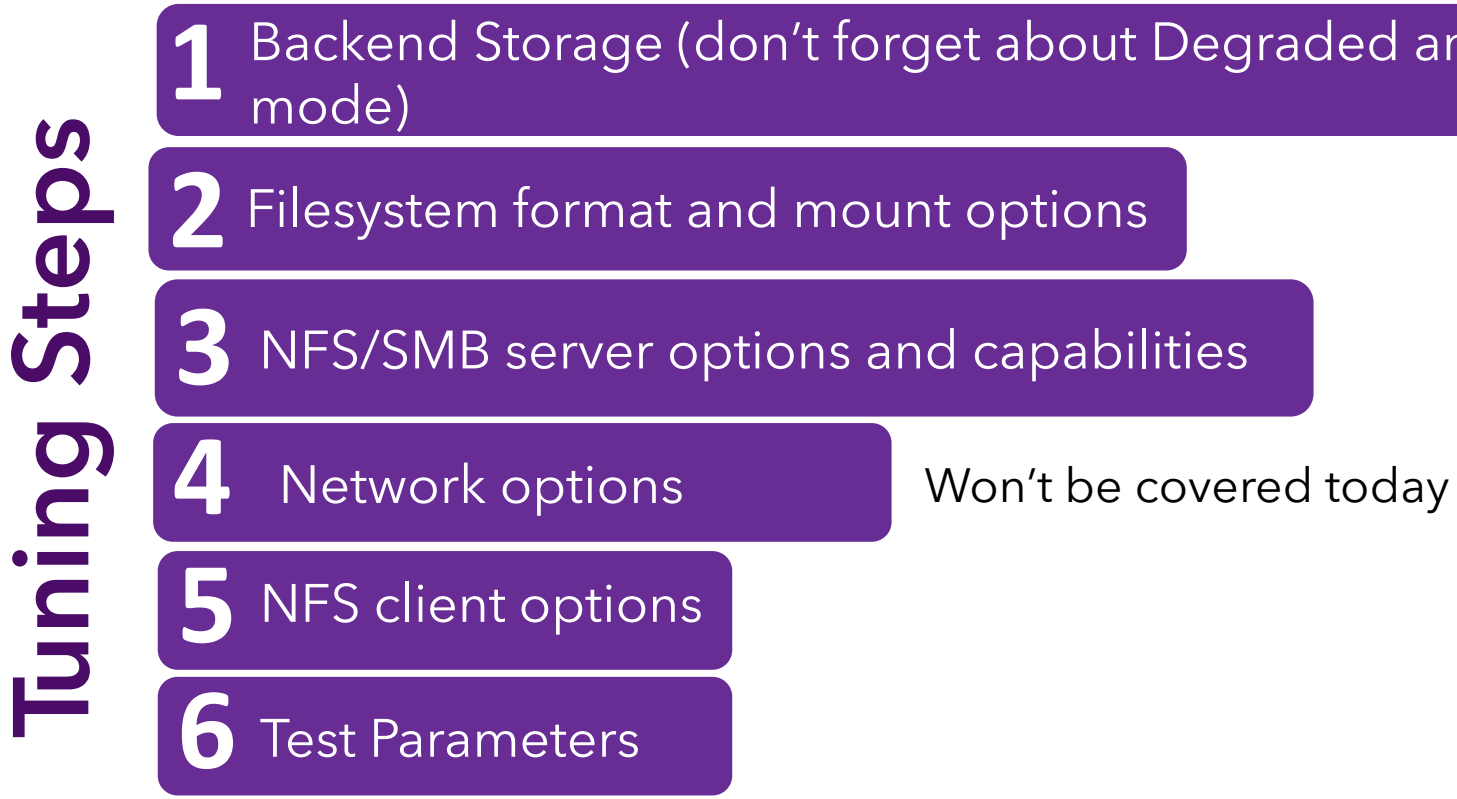
```
W0000 00:00:1756809703.126950   88089 computation_placer.cc:177] computation placer already registered. Please check linkage and avoid linking the same targe
[OUTPUT] 2025-09-02T12:41:48.109123 Running DLIO [Training & Checkpointing] with 8 process(es)
[OUTPUT] 2025-09-02T12:41:49.433245 Model size: 0.000010 GB
[OUTPUT] 2025-09-02T12:41:49.485327 Total checkpoint size: 0.000010 GB
[OUTPUT] 2025-09-02T12:41:49.545071 Max steps per epoch: 178 = 1 * 10000 / 7 / 8 (samples per file * num files / batch size / comm size)
[OUTPUT] 2025-09-02T12:41:49.634427 Starting epoch 1: 178 steps expected
[OUTPUT] 2025-09-02T12:41:49.692760 Starting block 1
[OUTPUT] 2025-09-02T13:00:50.200696 Ending block 1 - 178 steps completed in 1140.51 s
[OUTPUT] 2025-09-02T13:00:50.205989 Epoch 1 - Block 1 [Training] Accelerator Utilization [AU] (%): 5.3253
[OUTPUT] 2025-09-02T13:00:50.206400 Epoch 1 - Block 1 [Training] Throughput (samples/second): 9.1782
[OUTPUT] 2025-09-02T13:00:50.206727 Epoch 1 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T13:00:50.214275 Ending epoch 1 - 178 steps completed in 1140.58 s
[OUTPUT] 2025-09-02T13:00:50.229324 Starting epoch 2: 178 steps expected
[OUTPUT] 2025-09-02T13:00:50.230466 Starting block 1
[OUTPUT] 2025-09-02T13:19:35.228000 Ending block 1 - 178 steps completed in 1125.00 s
[OUTPUT] 2025-09-02T13:19:35.230258 Epoch 2 - Block 1 [Training] Accelerator Utilization [AU] (%): 5.3205
[OUTPUT] 2025-09-02T13:19:35.230608 Epoch 2 - Block 1 [Training] Throughput (samples/second): 9.1699
[OUTPUT] 2025-09-02T13:19:35.230905 Epoch 2 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T13:19:35.234168 Ending epoch 2 - 178 steps completed in 1125.00 s
[OUTPUT] 2025-09-02T13:19:35.250804 Starting epoch 3: 178 steps expected
[OUTPUT] 2025-09-02T13:19:35.251325 Starting block 1
[OUTPUT] 2025-09-02T13:38:21.594688 Ending block 1 - 178 steps completed in 1126.34 s
[OUTPUT] 2025-09-02T13:38:21.596913 Epoch 3 - Block 1 [Training] Accelerator Utilization [AU] (%): 5.3165
[OUTPUT] 2025-09-02T13:38:21.597288 Epoch 3 - Block 1 [Training] Throughput (samples/sec
[OUTPUT] 2025-09-02T13:38:21.597581 Epoch 3 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T13:38:21.601204 Ending epoch 3 - 178 steps completed in 1126.35 s
[OUTPUT] 2025-09-02T13:38:21.616529 Starting epoch 4: 178 steps expected
[OUTPUT] 2025-09-02T13:38:21.617093 Starting block 1
[OUTPUT] 2025-09-02T13:57:05.304699 Ending block 1 - 178 steps completed in 1123.69 s
[OUTPUT] 2025-09-02T13:57:05.306801 Epoch 4 - Block 1 [Training] Accelerator Utilization [AU] (%): 5.3294
[OUTPUT] 2025-09-02T13:57:05.307202 Epoch 4 - Block 1 [Training] Throughput (samples/second): 9.1853
[OUTPUT] 2025-09-02T13:57:05.307527 Epoch 4 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T13:57:05.311171 Ending epoch 4 - 178 steps completed in 1123.69 s
[OUTPUT] 2025-09-02T13:57:05.328231 Starting epoch 5: 178 steps expected
[OUTPUT] 2025-09-02T13:57:05.328809 Starting block 1
```


Let’s enable RDMA

```
[OUTPUT] 2025-09-02T14:09:46.048071 Running DLIO [Training & Checkpointing] with 8 process(es)
[OUTPUT] 2025-09-02T14:09:46.210299 Model size: 0.000010 GB
[OUTPUT] 2025-09-02T14:09:46.210747 Total checkpoint size: 0.000010 GB
[OUTPUT] 2025-09-02T14:09:46.211081 Max steps per epoch: 178 = 1 * 10000 / 7 / 8 (samples per file * num files / batch size / comm size)
[OUTPUT] 2025-09-02T14:09:46.223124 Starting epoch 1: 178 steps expected
[OUTPUT] 2025-09-02T14:09:46.223653 Starting block 1
[OUTPUT] 2025-09-02T14:11:18.432076 Ending block 1 - 178 steps completed in 92.21 s
[OUTPUT] 2025-09-02T14:11:18.445925 Epoch 1 - Block 1 [Training] Accelerator Utilization [AU] (%): 65.4031
[OUTPUT] 2025-09-02T14:11:18.446412 Epoch 1 - Block 1 [Training] Throughput (samples/second): 112.7204
[OUTPUT] 2025-09-02T14:11:18.446744 Epoch 1 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T14:11:18.450108 Ending epoch 1 - 178 steps completed in 92.23 s
[OUTPUT] 2025-09-02T14:11:18.463413 Starting epoch 2: 178 steps expected
[OUTPUT] 2025-09-02T14:11:18.463965 Starting block 1
[OUTPUT] 2025-09-02T14:12:56.980259 Ending block 1 - 178 steps completed in 98.52 s
[OUTPUT] 2025-09-02T14:12:56.986036 Epoch 2 - Block 1 [Training] Accelerator Utilization [AU] (%): 60.9304
[OUTPUT] 2025-09-02T14:12:56.986544 Epoch 2 - Block 1 [Training] Throughput (samples/second): 105.0121
[OUTPUT] 2025-09-02T14:12:56.986916 Epoch 2 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T14:12:56.990154 Ending epoch 2 - 178 steps completed in 98.53 s
[OUTPUT] 2025-09-02T14:12:57.006394 Starting epoch 3: 178 steps expected
[OUTPUT] 2025-09-02T14:12:57.006930 Starting block 1
[OUTPUT] 2025-09-02T14:14:34.697217 Ending block 1 - 178 steps completed in 97.69 s
[OUTPUT] 2025-09-02T14:14:34.702836 Epoch 3 - Block 1 [Training] Accelerator Utilization [AU] (%): 61.3886
[OUTPUT] 2025-09-02T14:14:34.703318 Epoch 3 - Block 1 [Training] Throughput (samples/second): 105.8023
[OUTPUT] 2025-09-02T14:14:34.703807 Epoch 3 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T14:14:34.706827 Ending epoch 3 - 178 steps completed in 97.70 s
[OUTPUT] 2025-09-02T14:14:34.722971 Starting epoch 4: 178 steps expected
[OUTPUT] 2025-09-02T14:14:34.723502 Starting block 1
[OUTPUT] 2025-09-02T14:16:12.229962 Ending block 1 - 178 steps completed in 97.51 s
[OUTPUT] 2025-09-02T14:16:12.235625 Epoch 4 - Block 1 [Training] Accelerator Utilization [AU] (%): 61.6071
[OUTPUT] 2025-09-02T14:16:12.236132 Epoch 4 - Block 1 [Training] Throughput (samples/second): 106.1787
[OUTPUT] 2025-09-02T14:16:12.236502 Epoch 4 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T14:16:12.239622 Ending epoch 4 - 178 steps completed in 97.52 s
[OUTPUT] 2025-09-02T14:16:12.255987 Starting epoch 5: 178 steps expected
[OUTPUT] 2025-09-02T14:16:12.256526 Starting block 1
[OUTPUT] 2025-09-02T14:17:49.287498 Ending block 1 - 178 steps completed in 97.03 s
[OUTPUT] 2025-09-02T14:17:49.294245 Epoch 5 - Block 1 [Training] Accelerator Utilization [AU] (%): 62.0539
[OUTPUT] 2025-09-02T14:17:49.294766 Epoch 5 - Block 1 [Training] Throughput (samples/second): 106.9475
[OUTPUT] 2025-09-02T14:17:49.295174 Epoch 5 - Block 1 [Training] Computation time per step (second): 0.3231+/-0.0000 (set value: {'mean': 0.323})
[OUTPUT] 2025-09-02T14:17:49.295937 Starting saving checkpoint 1 after total step 178 for epoch 5
[OUTPUT] 2025-09-02T14:17:49.306866 Finished saving checkpoint 1 for epoch 5 in 0.0109 s; Throughput: 0.0009 GB/s
[OUTPUT] 2025-09-02T14:17:49.310264 Ending epoch 5 - 178 steps completed in 97.05 s
[OUTPUT] 2025-09-02T14:17:49.322256 Saved outputs in /mnt/nfstest/unet3d_results/training/unet3d/run/20250902_140940
[OUTPUT] Averaged metric over all steps/epochs
[METRIC] Number of Simulated Accelerators: 8
[METRIC] Training Accelerator Utilization [AU] (%): 62.1268 (1.6111)
[METRIC] Training Throughput (samples/second): 107.0730 (2.7767)
[METRIC] Training I/O Throughput (MB/second): 14969.7905 (388.2063)
[METRIC] train_au_meet_expectation: fail
[METRIC] ==========
```

We need to do some tuning to achieve **reasonable performance**

What affects performance?



Won't be covered today

NFS Perf Test Toolbox

- **Client-side NFS**
  - nfsstat -m – negotiated vers/proto/rsize/wsize.
  - nfsiostat -h 1 – per-mount ops/s, kB/s, avg RTT/queue.
  - mountstats -S /mnt/nfs – per-op latencies (READ/WRITE/COMMIT).
  - grep /mnt/nfs /proc/self/mountstats – raw counters for diffs.
- **Server-side NFS**
  - nfsstat -s – server op mix & retrans.
  - watch -n1 cat /proc/net/rpc/nfsd – RPC queues/threads.
  - cat /proc/fs/nfsd/{threads,versions,portlist,max_block_size} – live params (6.16+ max_block_size).
  - rpcinfo -p | egrep '2049|20049' – TCP(2049) & RDMA(20049) services.
  - ss -lntp | egrep ':2049|:20049' – listeners & bound IPs.

NFS Perf Test Toolbox

CPU & scheduler

- mpstat -P ALL 1 – per-CPU utilization.
- pidstat -t -C nfsd 1 – per-thread nfsd usage.
- perf top / perf record -g (optional deep dive).

Storage / FS backend

- iostat -x 1 – device util/await/avgqu-sz.
- xfs_info /mnt/fs – stripe/alignment sanity (XFS).

How to read results

Workloads	Parameters			
	Threads=1	Threads=CPU core count	Threads=1/2 CPU core count	Threads=2 CPU core count
Unet3D	1@98%	15@93%	8@91%	10@93%
CPU load	49%@1	70%@48	78%@24	95%@48
Checkpointing	2.1 GBps / 10 GBps	3.2 GBps / 10 GBps	3.2 GBps / 10 GBps	3.2 GBps / 10 GBps
CPU Load	95%@1	15%@48	20%@24	15%@48
Fio Seq Writes/Reads	2.5 GBps / 5 GBps			

1@98%

Number of GPU **AU**

3.2 GBps / 10 GBps

Checkpoint Write / Read performance

20%@24

Average CPU load / number of CPU utilized

The results have been rounded for ease of understanding.

Server options: filesystem format and mount options

	XFS default	XFS OPT	EXT4 DEF	EXT4 OPT
Unet3D	18@91%	22@94%	16@95%	18@91%
Checkpointing Write / Read	17.2GBps / 20.3 GBps	28.1GBps / 26.4 GBps	18.5 GBps / 16.4 GBps	21.8 GBps / 20.6 GBps

- `sudo mkfs.xfs -f -b size=4096 -d su=128k,sw=7,agcount=128 logdev=/dev/xi_raid10 sectsize=4096,size=1024m /dev/xi_raid6`
- `sudo mount -t xfs -o noatime,nodiratime,logbsize=256k,logbufs=8,allocsize=1M,largeio,inode64,logdev=/dev/xi_raid10 /dev/xi_raid6 /srv/nfs/`

Outcomes 1

- **End-to-end alignment reduces wasted stripes.** Format with correct RAID hints (e.g., `mkfs.xfs -d su=<stripe>,sw=<width>`) so writes land on **full stripes**.
- **External log reduces checkpoint stalls.** Place the XFS log on a fast NVMe (`-l logdev=/dev/…,sectsize=4096,size=2-4G`) to cut metadata/journal contention during `rename()+fsync()` heavy checkpoints.
- **Parallelism from AGs, not just CPUs.** Use sensible **AG count** (e.g., `-d agcount=64-128` for multi-core servers) to enable parallel allocators without excessive fragmentation.
- **Mount options deliver real performance gains.** Prefer `noatime,inode64,logdev=/dev/…` (and keep default delayed logging). Avoid client-side sync for streaming writes; let the app call fsync() at checkpoints.
- **Feed the pipeline.** Increase device readahead for scans (`blockdev --setra 16384-65536`), and smooth write-back with `vm.dirty_bytes / vm.dirty_background_bytes`.
- **Expected impact.** Typically **+20–30%** sustained BW and **smoother tails** on sequential I/O vs. default format/mount; CPU per GB written often drops as well.

NFS server options

```
[nfsd]
debug=0
threads=64
host=10.10.10.1
,30.30.30.1
port=2049
grace-time=45
lease-time=45
udp=n
tcp=y
vers4.1=y
vers4.2=y
rdma=y
rdma-port=20049
```

`nfsd threads` — practical recommendations

- **Start point:** threads ≈ number of effective cores servicing the NFS NIC (think physical cores feeding that NIC's RX/TX queues; don't count SMT unless you've verified wins).
- **Rule-of-thumb bands:**
  - Small/medium fleets: **32–64** threads.
  - Large fan-in (100s of clients) or heavy small-IO metadata: **64–96.**
  - Going **>128** rarely helps and often increases lock contention/context switches.
- **Turn up when:** RPC backlog > 0 under load, nfsd worker CPU < 70% but requests queue; ta
- **Turn down when:** run-queue per core > 2, system time spikes
- **Validate:** watch `/proc/net/sunrpc/nfsd (queue/threads)`, `nfsstat -s`, and `mpstat -P ALL 1` during load.

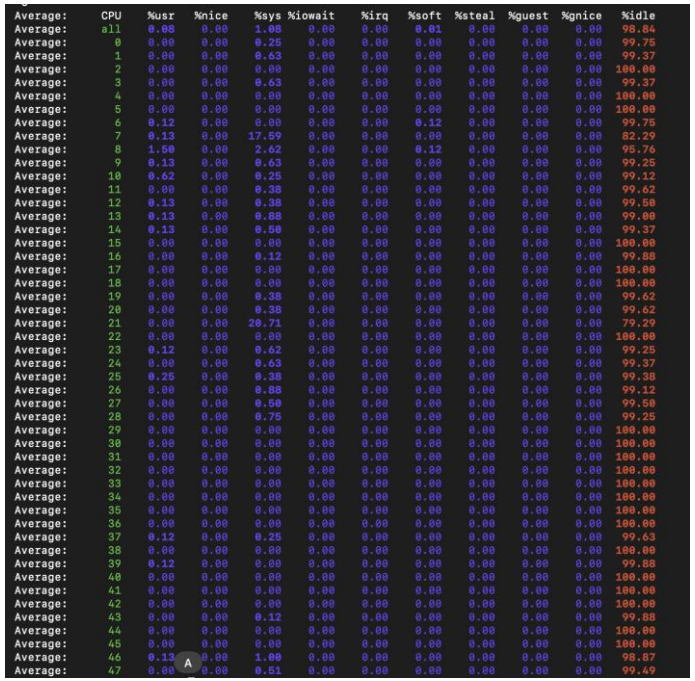
Server options: number of nfsd threads

	Threads=1	Threads=CPU core count	Threads=1/2 CPU core count	Threads=2 CPU core count
Unet3D	1@98%	14@93%	8@91%	10@93%
CPU load	49%@1	60%@48	78%@24	90%@48
Checkpointing	2.1 GBps / 4.2 GBps	7.3 GBps / 17.5 GBps	7.3 GBps / 17.3 GBps	7.6 GBps / 17.2 GBps
CPU Load	95%@1	15%@48	20%@24	15%@48
Fio Seq Writes/Reads	2.5 GBps / 5.1 GBps	22.4 GBps / 41.1 GBps	18.1GBps / 25.8GBps	24.7GBps / 26.7 GBps

```
mount -t nfs -o vers=4.2, proto=rdma, port=20049, rsize=1048576, wsize=1048576,
max_connect=16, nconnect=8, sync, trunkdiscovery
nfsserv:/srv/nfs/ /mnt/nfstest
```

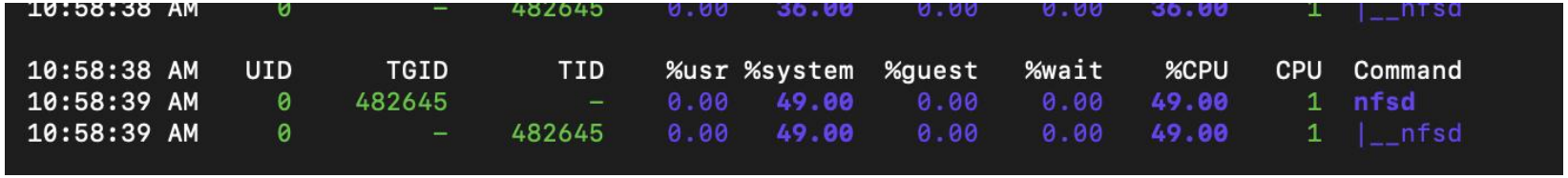

NFS threads=1

mpstat -P ALL 1



Average:	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
Average:	all	0.08	0.00	1.08	0.00	0.00	0.01	0.00	0.00	0.00	98.84
Average:	0	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	99.75
Average:	1	0.00	0.00	0.63	0.00	0.00	0.00	0.00	0.00	0.00	99.37
Average:	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	3	0.00	0.00	0.63	0.00	0.00	0.00	0.00	0.00	0.00	99.37
Average:	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	6	0.12	0.00	0.00	0.00	0.00	0.12	0.00	0.00	0.00	99.75
Average:	7	0.13	0.00	17.59	0.00	0.00	0.00	0.00	0.00	0.00	82.29
Average:	8	1.50	0.00	2.62	0.00	0.00	0.12	0.00	0.00	0.00	95.76
Average:	9	0.13	0.00	0.63	0.00	0.00	0.00	0.00	0.00	0.00	99.25
Average:	10	0.62	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	99.12
Average:	11	0.00	0.00	0.38	0.00	0.00	0.00	0.00	0.00	0.00	99.62
Average:	12	0.13	0.00	0.38	0.00	0.00	0.00	0.00	0.00	0.00	99.50
Average:	13	0.13	0.00	0.88	0.00	0.00	0.00	0.00	0.00	0.00	99.00
Average:	14	0.13	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	99.37
Average:	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	16	0.00	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.00	99.88
Average:	17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	19	0.00	0.00	0.38	0.00	0.00	0.00	0.00	0.00	0.00	99.62
Average:	20	0.00	0.00	0.38	0.00	0.00	0.00	0.00	0.00	0.00	99.62
Average:	21	0.00	0.00	20.71	0.00	0.00	0.00	0.00	0.00	0.00	79.29
Average:	22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	23	0.12	0.00	0.62	0.00	0.00	0.00	0.00	0.00	0.00	99.25
Average:	24	0.00	0.00	0.63	0.00	0.00	0.00	0.00	0.00	0.00	99.37
Average:	25	0.25	0.00	0.38	0.00	0.00	0.00	0.00	0.00	0.00	99.38
Average:	26	0.00	0.00	0.88	0.00	0.00	0.00	0.00	0.00	0.00	99.12
Average:	27	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	99.50
Average:	28	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	99.25
Average:	29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	34	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	37	0.12	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	99.63
Average:	38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	39	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	99.88
Average:	40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	41	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	43	0.00	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.00	99.88
Average:	44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00
Average:	46	0.13	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	98.87
Average:	47	0.00	0.00	0.51	0.00	0.00	0.00	0.00	0.00	0.00	99.49

pidstat -t -C nfsd 1



10:58:38 AM	0	–	482645	0.00	38.00	0.00	0.00	38.00	1	|__nfsd
10:58:38 AM	UID	TGID	TID	%usr	%system	%guest	%wait	%CPU	CPU	Command
10:58:39 AM	0	482645	–	0.00	49.00	0.00	0.00	49.00	1	nfsd
10:58:39 AM	0	–	482645	0.00	49.00	0.00	0.00	49.00	1	|__nfsd

NFS threads=96

`mpstat -P ALL 5`

```
11:59:34 AM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
11:59:39 AM  all    0.03    0.00   95.25    2.33    0.00    0.00    0.00    0.00    0.00    2.39
11:59:39 AM    0    0.00    0.00   95.16    2.42    0.00    0.00    0.00    0.00    0.00    2.42
11:59:39 AM    1    0.00    0.00   97.39    1.20    0.00    0.00    0.00    0.00    0.00    1.41
11:59:39 AM    2    0.00    0.00   96.98    1.01    0.00    0.00    0.00    0.00    0.00    2.01
11:59:39 AM    3    0.00    0.00   95.18    2.41    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM    4    0.00    0.00   95.77    2.41    0.00    0.00    0.00    0.00    0.00    1.81
11:59:39 AM    5    0.00    0.00   94.98    2.21    0.00    0.00    0.00    0.00    0.00    2.81
11:59:39 AM    6    0.00    0.00   99.60    0.20    0.00    0.00    0.00    0.00    0.00    0.20
11:59:39 AM    7    0.00    0.00   95.38    2.41    0.00    0.00    0.00    0.00    0.00    2.21
11:59:39 AM    8    0.00    0.00   94.96    2.82    0.00    0.00    0.00    0.00    0.00    2.22
11:59:39 AM    9    0.00    0.00   95.57    2.01    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   10    0.00    0.00   95.36    2.42    0.00    0.00    0.00    0.00    0.00    2.22
11:59:39 AM   11    0.00    0.00   94.97    2.62    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   12    0.00    0.00   95.17    2.41    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   13    0.00    0.00   95.38    2.21    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   14    0.00    0.00   94.98    2.61    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   15    0.00    0.00   95.77    1.61    0.00    0.00    0.00    0.00    0.00    2.62
11:59:39 AM   16    0.00    0.00   95.57    2.21    0.00    0.00    0.00    0.00    0.00    2.21
11:59:39 AM   17    0.00    0.00   94.75    2.63    0.00    0.00    0.00    0.00    0.00    2.63
11:59:39 AM   18    0.00    0.00   94.59    2.61    0.00    0.00    0.00    0.00    0.00    2.81
11:59:39 AM   19    0.00    0.00   94.77    2.41    0.00    0.00    0.00    0.00    0.00    2.82
11:59:39 AM   20    0.00    0.00   94.98    3.01    0.00    0.00    0.00    0.00    0.00    2.01
11:59:39 AM   21    0.00    0.00   95.18    2.41    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   22    0.00    0.00   94.58    2.81    0.00    0.00    0.00    0.00    0.00    2.61
11:59:39 AM   23    0.20    0.00   94.77    2.62    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   24    0.00    0.00   94.35    2.42    0.00    0.00    0.00    0.00    0.00    3.23
11:59:39 AM   25    0.00    0.00   94.98    2.21    0.00    0.00    0.00    0.00    0.00    2.81
11:59:39 AM   26    0.40    0.00   94.37    2.62    0.00    0.00    0.00    0.00    0.00    2.62
11:59:39 AM   27    0.00    0.00   94.38    2.81    0.00    0.00    0.00    0.00    0.00    2.81
11:59:39 AM   28    0.00    0.00   94.75    2.63    0.00    0.00    0.00    0.00    0.00    2.63
11:59:39 AM   29    0.00    0.00   96.98    1.01    0.00    0.00    0.00    0.00    0.00    2.02
11:59:39 AM   30    0.00    0.00   96.58    1.21    0.00    0.00    0.00    0.00    0.00    2.21
11:59:39 AM   31    0.00    0.00   94.76    2.62    0.00    0.00    0.00    0.00    0.00    2.62
11:59:39 AM   32    0.40    0.00   93.98    3.01    0.00    0.00    0.00    0.00    0.00    2.61
11:59:39 AM   33    0.00    0.00   94.96    2.62    0.00    0.00    0.00    0.00    0.00    2.42
11:59:39 AM   34    0.00    0.00   94.16    3.42    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   35    0.00    0.00   99.80    0.00    0.00    0.00    0.00    0.00    0.00    0.20
11:59:39 AM   36    0.00    0.00   94.77    2.62    0.00    0.00    0.00    0.00    0.00    2.62
11:59:39 AM   37    0.40    0.00   94.78    2.41    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   38    0.00    0.00   94.77    2.41    0.00    0.00    0.00    0.00    0.00    2.82
11:59:39 AM   39    0.00    0.00   93.37    3.61    0.00    0.00    0.00    0.00    0.00    3.01
11:59:39 AM   40    0.00    0.00   95.57    2.21    0.00    0.00    0.00    0.00    0.00    2.21
11:59:39 AM   41    0.00    0.00   94.37    3.02    0.00    0.00    0.00    0.00    0.00    2.62
11:59:39 AM   42    0.00    0.00   94.97    2.21    0.00    0.00    0.00    0.00    0.00    2.82
11:59:39 AM   43    0.20    0.00   94.78    2.21    0.00    0.00    0.00    0.00    0.00    2.81
11:59:39 AM   44    0.00    0.00   95.57    2.01    0.00    0.00    0.00    0.00    0.00    2.41
11:59:39 AM   45    0.00    0.00   94.57    3.22    0.00    0.00    0.00    0.00    0.00    2.21
11:59:39 AM   46    0.00    0.00   93.95    3.43    0.00    0.00    0.00    0.00    0.00    2.62
11:59:39 AM   47    0.00    0.00   94.78    2.01    0.00    0.00    0.00    0.00    0.00    3.21
```

`pidstat -t -C nfsd 5`

```
12:00:14 PM   0    506750         -    0.00   88.40    0.00    5.40   88.40    47  nfsd
12:00:14 PM   0         -    506750    0.00   88.40    0.00    5.40   88.40    47  |__nfsd
12:00:14 PM   0    506751         -    0.00   89.40    0.00    4.40   89.40    23  nfsd
12:00:14 PM   0         -    506751    0.00   89.40    0.00    4.40   89.40    23  |__nfsd
12:00:14 PM   0    506752         -    0.00   90.20    0.00    4.40   90.20    29  nfsd
12:00:14 PM   0         -    506752    0.00   90.20    0.00    4.40   90.20    29  |__nfsd
12:00:14 PM   0    506753         -    0.00   88.80    0.00    4.80   88.80    37  nfsd
12:00:14 PM   0         -    506753    0.00   88.80    0.00    4.80   88.80    37  |__nfsd
12:00:14 PM   0    506754         -    0.00   88.60    0.00    4.60   88.60    32  nfsd
12:00:14 PM   0         -    506754    0.00   88.60    0.00    4.60   88.60    32  |__nfsd
12:00:14 PM   0    506755         -    0.00   88.00    0.00    5.40   88.00    14  nfsd
12:00:14 PM   0         -    506755    0.00   88.00    0.00    5.40   88.00    14  |__nfsd
12:00:14 PM   0    506756         -    0.00   89.40    0.00    4.40   89.40    41  nfsd
12:00:14 PM   0         -    506756    0.00   89.40    0.00    4.40   89.40    41  |__nfsd
12:00:14 PM   0    506757         -    0.00   89.00    0.00    4.40   89.00    22  nfsd
12:00:14 PM   0         -    506757    0.00   89.00    0.00    4.40   89.00    22  |__nfsd
12:00:14 PM   0    506758         -    0.00   88.40    0.00    5.20   88.40    42  nfsd
12:00:14 PM   0         -    506758    0.00   88.40    0.00    5.20   88.40    42  |__nfsd
12:00:14 PM   0    506759         -    0.00   89.80    0.00    4.00   89.80     1  nfsd
12:00:14 PM   0         -    506759    0.00   89.80    0.00    4.00   89.80     1  |__nfsd
12:00:14 PM   0    506760         -    0.00   87.60    0.00    5.20   87.60    10  nfsd
12:00:14 PM   0         -    506760    0.00   87.60    0.00    5.20   87.60    10  |__nfsd
12:00:14 PM   0    506761         -    0.00   88.80    0.00    4.40   88.80    12  nfsd
12:00:14 PM   0         -    506761    0.00   88.80    0.00    4.40   88.80    12  |__nfsd
12:00:14 PM   0    506762         -    0.00   88.60    0.00    5.80   88.60    38  nfsd
12:00:14 PM   0         -    506762    0.00   88.60    0.00    5.80   88.60    38  |__nfsd
12:00:14 PM   0    506763         -    0.00   88.80    0.00    5.20   88.80    30  nfsd
12:00:14 PM   0         -    506763    0.00   88.80    0.00    5.20   88.80    30  |__nfsd
12:00:14 PM   0    506764         -    0.00   89.00    0.00    4.40   89.00    43  nfsd
12:00:14 PM   0         -    506764    0.00   89.00    0.00    4.40   89.00    43  |__nfsd
12:00:14 PM   0    506765         -    0.00   88.00    0.00    4.60   88.00     3  nfsd
12:00:14 PM   0         -    506765    0.00   88.00    0.00    4.60   88.00     3  |__nfsd
12:00:14 PM   0    506766         -    0.00   89.60    0.00    4.60   89.60    35  nfsd
12:00:14 PM   0         -    506766    0.00   89.60    0.00    4.60   89.60    35  |__nfsd
12:00:14 PM   0    506767         -    0.00   88.20    0.00    5.40   88.20    20  nfsd
12:00:14 PM   0         -    506767    0.00   88.20    0.00    5.40   88.20    20  |__nfsd
12:00:14 PM   0    506768         -    0.00   90.00    0.00    3.80   90.00     4  nfsd
12:00:14 PM   0         -    506768    0.00   90.00    0.00    3.80   90.00     4  |__nfsd
12:00:14 PM   0    506769         -    0.00   88.20    0.00    4.40   88.20    16  nfsd
12:00:14 PM   0         -    506769    0.00   88.20    0.00    4.40   88.20    16  |__nfsd
12:00:14 PM   0    506770         -    0.00   90.00    0.00    4.40   90.00    29  nfsd
12:00:14 PM   0         -    506770    0.00   90.00    0.00    4.40   90.00    29  |__nfsd
12:00:14 PM   0    506771         -    0.00   88.20    0.00    4.80   88.20    17  nfsd
12:00:14 PM   0         -    506771    0.00   88.20    0.00    4.80   88.20    17  |__nfsd
12:00:14 PM   0    506772         -    0.00   88.20    0.00    5.60   88.20     8  nfsd
12:00:14 PM   0         -    506772    0.00   88.20    0.00    5.60   88.20     8  |__nfsd
12:00:14 PM   0    506773         -    0.00   88.00    0.00    5.40   88.00    44  nfsd
12:00:14 PM   0         -    506773    0.00   88.00    0.00    5.40   88.00    44  |__nfsd
12:00:14 PM   0    506774         -    0.00   88.80    0.00    4.40   88.80    28  nfsd
12:00:14 PM   0         -    506774    0.00   88.80    0.00    4.40   88.80    28  |__nfsd
12:00:14 PM   0    506775         -    0.00   89.00    0.00    4.60   89.00    24  nfsd
12:00:14 PM   0         -    506775    0.00   89.00    0.00    4.60   89.00    24  |__nfsd
12:00:14 PM   0    506776         -    0.00   87.80    0.00    5.00   87.80    40  nfsd
12:00:14 PM   0         -    506776    0.00   87.80    0.00    5.00   87.80    40  |__nfsd
12:00:14 PM   0    506777         -    0.00   88.20    0.00    4.60   88.20     9  nfsd
12:00:14 PM   0         -    506777    0.00   88.20    0.00    4.60   88.20     9  |__nfsd
12:00:14 PM   0    506778         -    0.00   88.60    0.00    4.40   88.60     4  nfsd
12:00:14 PM   0         -    506778    0.00   88.60    0.00    4.40   88.60     4  |__nfsd
12:00:14 PM   0    527167         -    0.00    4.60    0.00    8.40    4.60    33  kworker/u194:12-nfsd4
12:00:14 PM   0         -    527167    0.00    4.60    0.00    8.40    4.60    33  |__kworker/u194:12-nfsd4
```


Outcomes 2

- **Defaults aren't enough.** Out-of-the-box NFS/NFSD settings limit throughput and inflate tail latency; they don't deliver acceptable performance for modern ML/AI or checkpointing workloads.
- **Set threads ≈ effective cores.** Configure nfsd threads roughly equal to the number of effective *CPU cores* servicing the NFS NIC(s). Adjust with awareness of the **storage backend's CPU demand** (RAID/erasure coding/SPDK/encryption/checksumming) so you don't starve it.
- **More threads ≠ better performance.** Increasing nfsd threads **beyond core count** typically adds context switches and lock contention.
- **Threads ≈ cores ⇒ NIC-limited performance.** With proper IRQ/NUMA locality and no storage bottleneck, threads near core count achieves **maximum practical NIC throughput** (approaches line-rate).
- **Checkpoint is different.** For large sequential Checkpoint operations, NFSD thread count has **negligible effect**; observed checkpoint performance remains **unsatisfactory** under current settings.
- **Implication.** Improving Checkpoint requires **further system-level tuning** (filesystem/journal, write-back policy, I/O path, and data layout)—not just NFSD thread adjustments.

/etc/exports

- **async vs sync:** async wins for throughput (checkpoints/streaming); sync protects against crash-loss — pair it with fast journals/logs if mandated.
- **sec=sys vs Kerberos:** sec=sys keeps CPU free for I/O; krb5i/p can cost **tens of %** of BW — use only where required and cache NSS/ID mapping well.
- **v4 pseudo-root done right:** fsid=0 + crossmnt = one clean namespace; fewer mount storms, smoother directory traversals.
- **wdelay** lets the NFS server briefly **coalesce small writes** before committing (better throughput); no_wdelay **forces immediate commits** (lower single-op latency, more IOPS). They only matter when the export is **sync**. With **async**, they're effectively irrelevant.

Server options: wdelay vs. no_wdelay

	wdelay	**no_wdelay**
Checkpointing	7.3 GBps /17.5 GBs	12.8 GBps /17.5 GBps
FIO Sequential write	22.6 GBps	23.5 GBps

Since the checkpointing workload is highly synchronous and latency-sensitive, enabling the no_wdelay parameter significantly improves performance.

Server options: sync vs. async

	sync	async
Checkpointing	12.8 GBps /17.5 GBps	13.5 GBps /18.5 GBps
FIO Sequential write/read	23.5 GBps / 41.2 GBps	25.3 GBps / 41.2 GBps

Async mode shows slightly better performance, but on practice it tends to be unstable on more powerful systems.

SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

NFS Client tuning

SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

Let’s start from NFS kernel module

Quick Tuning Profiles

Param	Training / Checkpoint (Throughput)	Dataloaders / Feature store (IOPS/Latency)	Mixed (RAG + Train)
max_session_slots	128–256	64–128	128–192
max_session_cb_slots	32–64	16–32	24–48
callback_nr_threads	8–12 (pNFS/delegations)	8–16	8–12
nfs4_disable_idmapping	1 if sec=sys & unified UID/GID; else 0	1 if unified UID/GID	per env
nfs_idmap_cache_timeout	600–1200s	300–600s	600–900s
delay_retrans	-1 (default backoff)	0–1 (fast EAGAIN)	0–1
nfs_access_max_cachesize	1M	64k–256k	128k–256k
enable_ino64	1	1	1

max_session_slots (Parallelism = Bandwidth)

- **What.** Maximum number of outstanding NFSv4.1 requests negotiated by the client.
- **Why for AI.** High concurrency is crucial for saturating fast NICs during large tensor/checkpoint I/O.
- **Recommend.** 128–256 for bandwidth-bound training; keep closer to 64–128 for pure low-latency small I/O.
- **Set:**

```
# Temporary
```

- `echo 256 | sudo tee /sys/module/nfs/parameters/max_session_slots`

```
# Persistent (/etc/modprobe.d/nfs.conf)
```

- `options nfs max_session_slots=256`

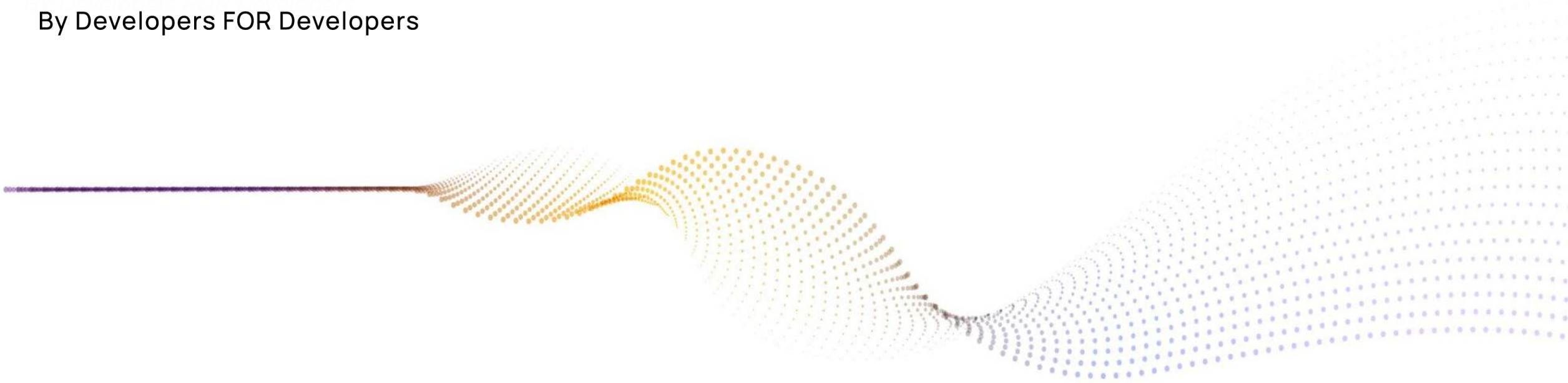
- **Watch-outs**

Benefits depend on server slot limits; too high can increase queuing delay.

Server options: Defaults vs Optimal kernel module parameters

	Defaults	Optimal (training preset)
Unet3D	12@92%	14@93%
Checkpointing	12.8GBps / 17.5GBps	13.5 GBps / 18.5 GBps
FIO Sequential write/read	23.6 GBps / 41GBps	29.2 GBps / 43.2 GBps

Hyatt Regency Santa Clara, CA
September 15-17, 2025



Client mount options

Parallelism: nconnect vs max_connect

- **nconnect=<1..16>:** Multiple TCP/RDMA connections to **one server IP** for a given mount; boosts throughput and mitigates head-of-line blocking.
- **max_connect=<1..16>:** For **NFSv4.1+ session trunking** across **multiple server IPs** that belong to the same server; improves bandwidth & resiliency. Mount via each IP (or rely on trunking discovery where supported).
- **Rules of thumb:** Start with nconnect=8 (TCP) or 4 (RDMA). If server exposes multiple IPs, prefer max_connect and spread traffic across them.

Other options are described in the Appendix

End-to-end Examples

A) TCP – Training throughput (1 IP)

```
mount -t nfs -o vers=4.2,proto=tcp,hard,\
nconnect=16,rsize=1048576,wsize=1048576,\
lookupcache=all,acregmin=60,acregmax=600,acdirmin=60,acdirmax=600 \
nfs-srv:/export /mnt/train
```

B) TCP – Session trunking across 2 IPs (same server):

```
# Mount via IP A
aaa.bbb.ccc.ddd:/export /mnt/trainA nfs vers=4.1,hard,max_connect=16,
nconnect=8,rsize=1048576,wsize=1048576,trunkdiscovery
# Mount via IP B
eee.fff.ggg.hhh:/export /mnt/trainB nfs vers=4.1,hard,max_connect=16, nconnect=8,
rsize=1048576,wsize=1048576,trunkdiscovery
# Bind a single workload path (symlink or union mount) if needed
```

C) RDMA

```
mount -t nfs -o vers=4.2,proto=rdma,port=20049,hard,\
max_connec=16,rsize=1048576,wsize=1048576,lookupcache=positive,actimeo=120 \
nfs-srv:/dataset /mnt/data
```


RDMA vs TCP Session Trunking

	RDMA Defaults 1 port	TCP nconnect=4 1 port	TCP nconnect=8 1 port	TCP nconnect=16 1 port
Unet3D	8@93%	1@91%	1@93%	3@93%
CPU load	11%@48	10%@48	15%@48	15%@48
Checkpointing	15.7 GBps / 16.1 GBps	6.2 GBps / 3.1 GBps	11.7 GBps / 4.2 GBps	13.7 GBps / 9.6 GBps
CPU load	33% @ 48	24% @ 48	44% @ 48	60% @ 48
Fio Sequential Read / Write	17.2 GBps / 22.5 GBps	5.6 GBps / 3.3 GBps	13.2 GBps / 4.1 GBps	13.2 GBps / 10.1 GBps

	RDMA Defaults max_connect=16 2 ports	TCP max_connect=16 2 port
3DUnet	14@93%	6@96%
CPU load	25%@48	18%@48
Checkpointing	16.2 GBps / 18.5GBps	14.2 GBps / 12.9 GBps
CPU load	65% @ 48	74% @ 48
Fio Sequential Read / Write	29.2 GBps / 43.2 GBps	17.2 GBps / 17.8 GBps

Backend storage health Impact

	xiRAID Normal	xiRAID Degraded	MDRAID Normal	MDRAID Degraded
Unet3D	14@93%	14@91%	8@90%	1@56%
CPU load	25%@48	32%@48	40%@48	68%@48

SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

SMB 3.1.1 as AI Storage Protocol

Why you might want SMB

- **Identity & ACLs that match the enterprise**
  - Native **Active Directory/Kerberos** integration and **rich NTFS-style ACLs** across millions of SIDs.
  - Avoids classic NFS **AUTH_SYS group-list limits** (historically small—often 16 to 64 groups depending on stack), which break users in many AD groups.
- **Performance features that fully utilize modern network bandwidth of up to 800Gb per port**
  - **SMB Multichannel:** up to 16 TCP or RDMA channels per session, auto-striped across NICs/queues for higher throughput and resiliency.
  - **SMB Direct (RDMA):** CPU-offloaded, low tail latency – great for latency-sensitive I/O and saturating fast links.

Why you might want SMB

- **Built-in HA & maintenance friendliness**
  - **Transparent Failover** (Continuously Available shares) with **persistent handles**—long-running AI jobs survive node failovers and rolling updates, along with Direct IO support for data integrity and cache bypasses.
- **Data-movement for AI workflows**
  - **Server-side copy offload** (COPYCHUNK / reflink when supported): rapid dataset staging/clone without host-side bandwidth.
- **POSIX where you need it**
  - **SMB3 POSIX extensions** (e.g., Samba) bring Linux-friendly semantics (case sensitivity, chmod/chown, fast rename) so AI toolchains "just work".
- **When SMB shines vs NFS**
  - Mixed Windows/macOS/Linux estates with strict **AD/ACL** requirements. Windows support is only NFS2/3.
  - Hitting NFS limits (e.g., **small group lists with AUTH_SYS**, idmap complexity) or wanting **built-in multichannel/RDMA** without per-client nconnect tuning.

Quick AI Profiles (mount options)

Profile	Transport	Parallelism	Caching/Coherency
Throughput (Training/CKPT, TCP)	vers=3.1.1,proto=tcp	multichannel,max_channels=16 max_credits=4096–8192	cache=none, actimeo=0, serverino(or cache=loose only on read-only)
Ultra-low-latency (RDMA)	vers=3.1.1,rdma	max_credits=4096–8192 (credits drive I/O depth)	cache=strict, actimeo=120
Dataloaders / Small-files	vers=3.1.1,proto=tcp	multichannel,max_channels=>4	cache=strict, actimeo=30–120, serverino

Version & Transport

- **vers=3.1.1:** most secure & feature-complete (preauth integrity, better perf semantics).

- **TCP vs RDMA:**
  - proto=tcp + **SMB3 Multichannel** → parallel TCP lanes across NICs.
  - rdma (SMB Direct) → lower CPU/latency when both client & server support it.
- **Client source IP:** clientaddr=<ip> to pin egress path on multi-homed nodes.

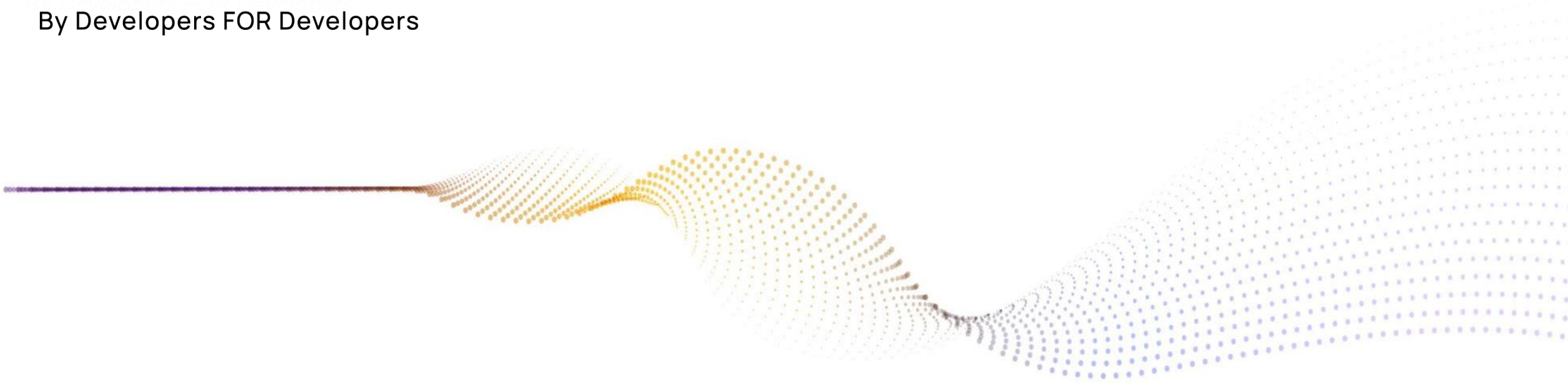
Tuxera Fusion SMB Server Capabilities

- Multi-threaded user & kernel mode server for Linux X64, ARM, PPC with low CPU & memory footprint for bare metal, VM, containers
- Validated 100GbE throughput 11.4GB/s Windows & 8.5GB/s macOS
- Scale-out transparent failover on up to 32-node cluster
- Mature, production-proven support of RDMA, multichannel, persistent handles, Direct IO, compression (and soon QUIC)
- Active Directory, Open LDAP, & Apple Open Directory support
- Used in HPC, cloud, media & entertainment, medical research, energy, & enterprise storage solutions

Tuxera TSMB Server tuning options for AI workloads

- **tsmb-cfg global add**
  - `connections_max, open_files_max, sess_open_files_max`
  - `transport_rx_threads, transport_tx_threads`
  - `vfs_metadata_threads, vfs_data_threads`
  - `vfs_zerocopy_write, vfs_zerocopy_read`
  - `vfs:force_sync,nodirect,lock`
  - `oplock:all`
  - `smb2_credits_max`
  - `smb2_read_max, smb2_write_max, smb2_trans_max`
  - `ANY,0.0.0.0,RDMA_IPv4,445,SMBD`

Hyatt Regency Santa Clara, CA
September 15-17, 2025



CIFS Client options

Parallelism: Multichannel & Credits

- **multichannel,max_channels=N:** enable SMB3 Multichannel and set the channel count per session. Use when server exposes one or more NICs; aligns with AI training flows.
- **max_credits=:** raise SMB3 credit window (I/O queue depth). Start **4096,** scale to **8192** if server permits.
- **Don't mix** too many channels with tiny credits; find a good product of channels x credits to match BDP.

Caching & Metadata

cache=:

- strict (default): safest; respects oplocks/leases.
- loose: faster, but risk with concurrent writers; use only on read-only/immutable datasets.
- none: bypass page cache (useful for specific pipelines, test carefully).

Attributes TTL: actimeo=<sec> (single knob).

- Training/read-mostly: actimeo=600.
- Dataloaders: 30–120.

Inodes: serverino ensures stable inode numbers from server (recommended for big trees).

POSIX semantics (if server supports): add posix to reduce chattiness for create/unlink/stat (Samba/KSMBD with SMB3 POSIX extensions).

End-to-end Examples

A) TCP – Training throughput (Multichannel + credits):

```
mount -t cifs //smb-srv/train /mnt/train \
-o vers=3.1.1,username=ai,credentials=/root/.smbcred,\
proto=tcp,multichannel,max_channels=4,max_credits=8192,\
cache=none,actimeo=0,serverino
```

B) RDMA – Low-latency dataloaders:

```
mount -t cifs //smb-srv/data /mnt/data \
-o vers=3.1.1,rdma,username=ai,credentials=/root/.smbcred,\
max_credits=4096,cache=strict,actimeo=120,serverino
```


Compare. First Run

	Defaults	Multichannel, MC=4	Multichanel, MC=16 Max_credits=8k	SMB Direct
Unet3D	2@91%	3@92%	7@92%	8@90%
CPU load	20%@8	24%@16	32%@48	8%@48
Checkpointing	2.6 GBps / 5.4 GBps	3.2 GBps / 7.2 GBps	4.3 GBps / 14.6 GBps	14.5 GBps / 16.2 GBps
CPU load	24%@12	17%@24	38%@24	20%@48
FIO Sequential WRITE/READ	2.6 GBps / 5.3 GBps	4.6 GBps / 9.4 GBps	19.2 GBps / 20.1 GBps	22.5 GBps / 24.6 GBps

```
mount -t cifs //smb-srv/data /mnt/data \
-o vers=3.1.1,{rdma},username=ai,credentials=/root/.smbcred,\
max_credits=4096/8192,cache=strict,actimeo=120,serverino
```


Let's improve mount options

- **Before:** `cache=strict, actimeo=1, … soft, mapposix, reparse=nfs, nativesocket, symlink=native, rsize/wsize/bsize set`
  - Client kept **page cache** for file data → fsync() forced cache flush **plus** server commit.
  - Extra coherency work (attribute cache + validations) and more round-trips under strict semantics.
  - More code paths toggled (reparse/mapping options), increasing overhead.
- **After (faster):** `cache=none, actimeo=0, serverino, multichannel, max_channels=16, echo_interval=60`
  - **No client data cache** → writes go **directly** to the server; fsync() doesn't pay a "double flush."
  - **actimeo=0**: no attribute caching → fewer stale checks around sync points, simpler correctness.
  - **serverino**: consistent inode numbers from server (cleaner metadata behavior).

Optimized mount options

	Defaults, Multichanel, MC=16 Max_credits=8k	Optimized, Multichanel, MC=16 Max_credits=8k	Defaults SMB Direct
Unet3D	7@92%	7@92%	8@92%
CPU load	32%@48	36%@48	8%@48
Checkpointing	4.3 GBps / 14.6 GBps	16.5 GBps / 19.1 GBps	16.5 GBps / 18.2 GBps
CPU load	38%@24	24%@48	24%@48
FIO Sequential WRITE/READ	19.2 GBps / 20.1 GBps	20.2 GBps / 21.2 GBps	22.5 GBps / 24.6 GBps

	Multichanel, MC=16, Dual Port	SMB Direct, Dual Port
Unet3D	9@91%	15@93%
CPU load	41%@48	62%@48
Checkpointing	20.4 GBps / 21.7 GBps	23.5 GBps / 28.2 GBps
CPU load	46%@48	51%@48
FIO Sequential WRITE/READ	25.5 GBps / 27.3 GBps	48.4 GBps / 48.8 GBps

NFS vs SMB Head-to-Head Compare, TCP

	NFS	SMB	NFS, Dual Port	SMB, Dual Port
Unet3D	3@93%	7@92%	6@96%	9@91%
CPU load	15%@48	36%@48	18%@48	41%@48
Checkpointing	13.7 GBps / 9.6 GBps	16.5 GBps / 19.1 GBps	14.2 GBps / 12.9 GBps	20.4 GBps / 21.7 GBps
CPU load	60% @ 48	24%@48	74% @ 48	46%@48
FIO Sequential WRITE/READ	13.2 GBps / 10.1 GBps	20.2 GBps / 21.2 GBps	17,2 GBps / 17,8 GBps	25.5 GBps / 27.3 GBps

NFS vs SMB Head-to-Head Compare, RDMA

	NFS	SMB	NFS, Dual Port	SMB, Dual Port
Unet3D	8@93%	8@92%	14@93%	15@93%
CPU load	11%@48	8%@48	25%@48	62%@48
Checkpointing	15.7 GBps / 16 GBps	16.5 GBps / 18.2 GBps	16.2 GBps / 18.5GBps	23.5 GBps / 28.2 GBps
CPU load	33% @ 48	24%@48	65% @ 48	51%@48
FIO Sequential WRITE/READ	17.2 GBps / 22.5 GBps	22.5 GBps / 24.6 GBps	29.2 GBps / 42.2 GBps	48.4 GBps / 48.8 GBps

SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

NFS Local IO

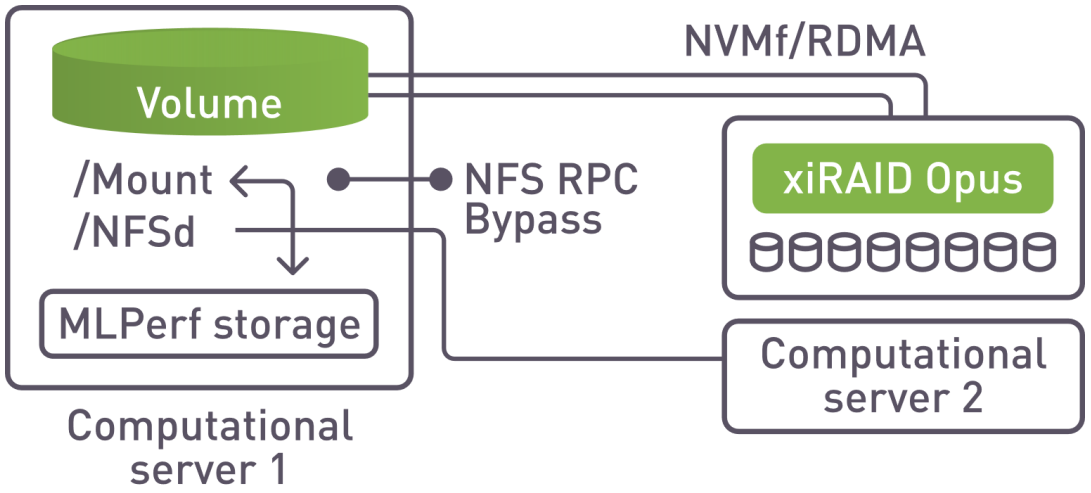
NFS LOCAL IO

- **What is it:** Local fast path that preserves NFSv4 semantics while bypassing the network stack when client and nfsd are on the **same machine.**

- Lower **P95/P99 latency,** fewer context switches/IRQs, lower CPU overhead, higher sustained BW for big sequential I/O.

- **Caveat:** Results do **not** reflect multi-node behavior (no NIC queues, no nconnect, no RDMA link effects).

Storage Disaggregation and NFS LocalIO



Present storage with Opus

Expose volumes to compute nodes via **NVMe-oF (RDMA or TCP)** – line-rate, low-latency block access right on the node.

Format & mount locally

On each compute node, create an aligned local filesystem (e.g., XFS with proper su/swidth,) and mount it for the workloads.

Run an NFS server on the node

Export that filesystem and build a **single, unified namespace.** Remote nodes consume it over TCP (nconnect) or RDMA as appropriate.

Local I/O fast path

On the hosting node, apps hit the **NFS Local I/O** path (kernel short-circuit, no TCP/RDMA), avoiding extra **RPC** overhead and minimizing tail latency/CPU.

Why this works:

- **Flexibility:** Opus composes and places capacity exactly where compute needs it.
- **Performance:** direct NVMe-oF locally; NFS provides high-BW multi-reader/writer semantics to the rest of the cluster.
- **Operational simplicit:** one POSIX view, standard tools (nfsd, nfsstat, mountstats), easy policy (quotas, auth).

NFS LOCAL IO + NVMf

	NFS Dual Port, TCP	NFS Dual Port, RDMA	SMB Dual Port, TCP	SMB Dual Port, RDMA	NVMf/RDMA	NVMf/TCP
Unet3D	6@96%	14@93%	9@93%	15@93%	16@93%	14@91%
Checkpointing	14.2 GBps / 12.9 GBps	16.2 GBps / 18.5 GBps	20.4 GBps / 21.7 GBps	23.5 GBps / 28.2 GBps	24GBps / 28 GBps	24 GBps / 26 GBps

What's next

Objective

- Prove performance and stability on a bigger, realistic topology and quantify gains from server/client tuning at 400 Gb/s.

Topology under test:

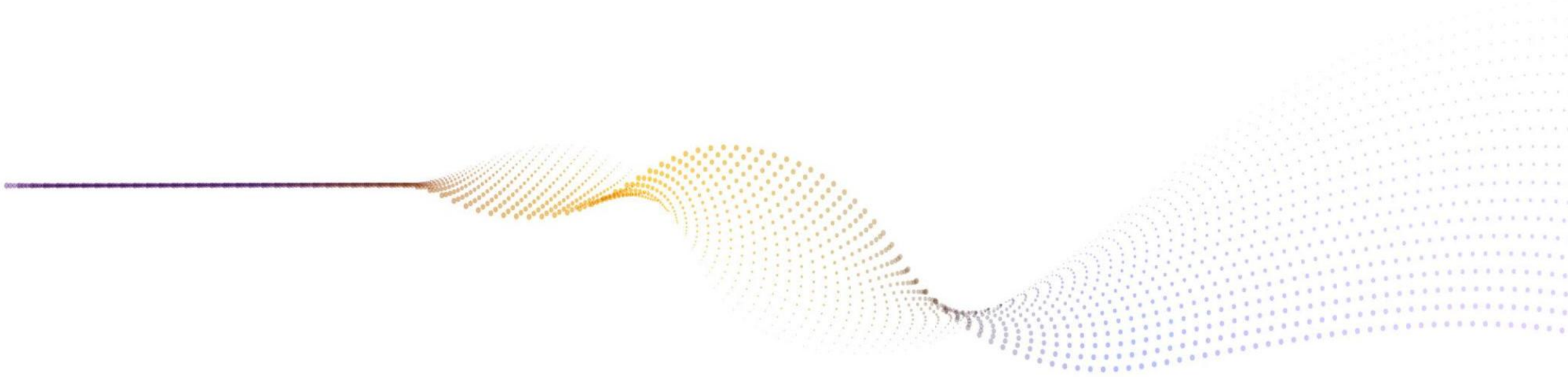
- 2× storage nodes: NVMe PCIe Gen5 arrays, dual 400 Gb links each.
- 4× clients: 1x400GBit Each, 64 CPU Cores Each

Test matrix (A/B comparisons):

- NFSv4.2 RDMA vs TCP + nconnect (multichannel).
- SMB Direct vs SMB Multichannel

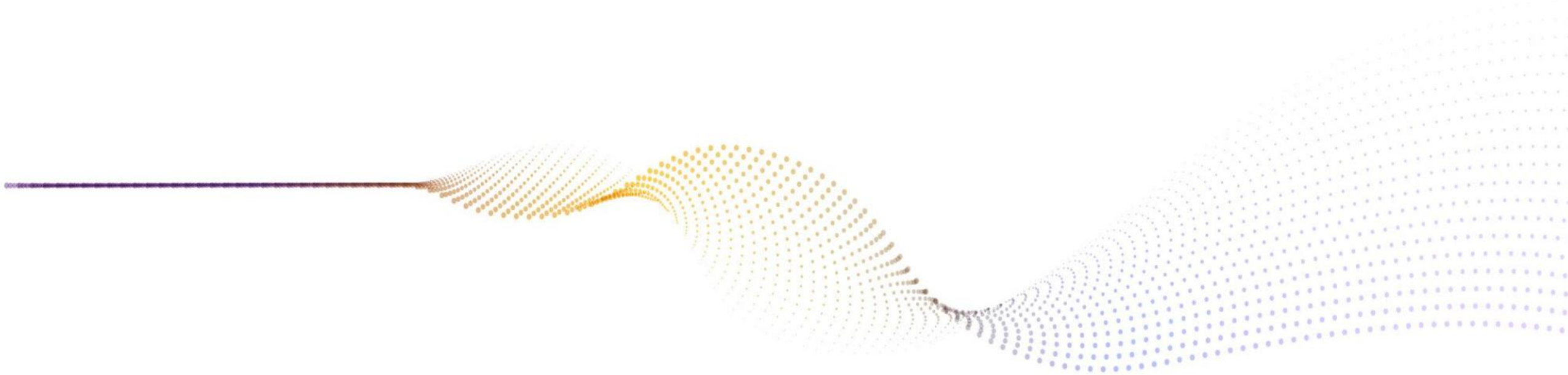
Workloads to run:

- MLPerf Storage "training" (datasize → datagen → run).
- Checkpoint streaming (1–4 MiB writes, multi-writer).
- GPU/Accelerator Utilization ≥ 90% where applicable.
- Target ≥ 90% of link rate sustained without tail blow-ups.



Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference



Appendix

NFS server options

Server-side max payload per READ/WRITE can be raised to 4 MiB in 6.16 kernel.

```
# check current limit
cat /proc/fs/nfsd/max_block_size
# raise to 4 MiB (4194304) and apply
echo 4194304 | sudo tee /proc/fs/nfsd/max_block_size
sudo systemctl restart nfs-server    # or nfs-kernel-server
```

Client: check negotiated sizes:

```
nfsstat -m
grep -E 'rsize|wsize' /proc/self/mountstats
```


Server Options

vm.dirty_bytes = 1073741824 (1 GiB)

Absolute cap (bytes) at which a process doing writes must start **writeback itself**. Smooths large write bursts and prevents massive "all-at-once" flushes. Too high ⇒ long stalls during flush; too low ⇒ over-eager flushing.

vm.dirty_background_bytes = 268435456 (256 MiB)

Absolute threshold that wakes the kernel's **background flusher threads** to start draining dirty pages. Keeps a steady writeback pipeline so foreground I/O isn't jolted by sudden flushes.

vm.swappiness =

Biases the kernel to **avoid swapping** anonymous memory unless truly necessary, preserving page cache for filesystem I/O. Good for storage servers with ample RAM (reduces cache churn).

net.core.rmem_max = 268435456

Upper bound for **per-socket receive buffers**. Allows TCP/UDP autotuning (and RDMA ULPs using sockets) to grow windows on high-BDP paths. Doesn't force buffers by itself; it raises the ceiling.

net.core.wmem_max = 268435456

Upper bound for **per-socket send buffers**. Lets autotuning open bigger send windows for long, fat links (useful with multi-stream TCP NFS).

net.core.netdev_max_backlog = 250000

Maximum packets queued on the **ingress backlog** when the NIC delivers faster than the stack can process. Higher values absorb short bursts and reduce drops; if set too high on an overloaded CPU, it can add queuing latency.

max_session_cb_slots + callback_nr_threads

- **What:**
- `max_session_cb_slots` – parallel callbacks (delegations, pNFS recalls) the client can process from a server.
- `callback_nr_threads` – number of kernel threads handling those callbacks.
- **Why for AI:** With pNFS/flexfiles or heavy parallel opens/closes, responsive callback handling prevents stalls and delegation recalls from becoming a bottleneck.
- **Recommend:** `max_session_cb_slots=32–64`, `callback_nr_threads=8–12` (up to 16 for metadata-intensive loaders).
- **Set:**

```
echo 64 | sudo tee /sys/module/nfs/parameters/max_session_cb_slots
echo 12 | sudo tee /sys/module/nfs/parameters/callback_nr_threads
# persistent
options nfs max_session_cb_slots=64 callback_nr_threads=12
```


nfs4_disable_idmapping & nfs_idmap_cache_timeout

- **What:**
- `nfs4_disable_idmapping=1` (with `sec=sys`) skips v4 idmapping and uses numeric UID/GID directly.
- `nfs_idmap_cache_timeout` controls TTL of idmap cache.
- **Why for AI:** Reduces metadata RPC churn during massive parallel file access by many workers; keeps `stat()/open()` paths light.
- **Recommend:**
- If all nodes share **identical numeric UID/GID,** set nfs4_disable_idmapping=1.
- `nfs_idmap_cache_timeout=600–1200s` (throughput) or `300–600s` (latency-sensitive small-file workloads).
- **Set:** `options nfs nfs4_disable_idmapping=1 nfs_idmap_cache_timeout=900`
- **Watch-outs:** Only use `nfs4_disable_idmapping=1` when UID/GID spaces are truly aligned.

delay_retrans (Fast Fail for Small-IO Paths)

- **What:** After server replies NFS4ERR_DELAY, limit retries before returning EAGAIN.
- **Why for AI:** Dataloaders and micro-services often prefer quick retry over long stalls.
- **Recommend:** 0–1 for latency-sensitive small I/O; keep -1 (default) for pure bulk-throughput training.
- **Set:**
- `echo 1 | sudo tee /sys/module/nfs/parameters/delay_retrans`
- `# persistent`
- `options nfs delay_retrans=1`

nfs_access_max_cachesize (Access Cache Budget)

- **What:** Global budget for caching ACCESS results (permission checks).
- **Why for AI:** Many processes (workers) touching vast directory trees benefit from a larger ACCESS cache, cutting metadata round-trips.
- **Recommend:** 128k–512k for large training sets; 64k–256k for small-file/latency paths to control memory.
- **Set:**
- `echo 262144 | sudo tee /sys/module/nfs/parameters/nfs_access_max_cachesize`
- `options nfs nfs_access_max_cachesize=262144`
- **Watch-outs:** Too small ⇒ excess RPC; too large ⇒ client RAM overhead.

I/O Sizes: rsize / wsize

I/O Sizes: rsize / wsize

- **Set to 1048576 (1 MiB)** – current Linux client cap per RPC. Verify with nfsstat -m and /proc/self/mountstats.
- Kernel 6.16 supports for 4M for the storage side.

Reliability & Timeouts

- **hard** *(default for v4)*: Required for training/checkpoints to avoid silent corruption.
- **timeo= / retrans=:** Use defaults for bulk; for latency-sensitive small-IO consider slightly lower timeo and verify behavior under loss.
- **retrans:** Don’t set too low; allow the client to ride out transient blips during epochs.

Caching & Coherency (metadata)

- **lookupcache=:**
  - all (aggressive): fastest for read-mostly, may delay visibility of new files created by others.
  - positive: good balance for dataloaders (cache hits for existing entries, fewer negatives).
  - none: strongest coherency; avoid unless required (metadata RPC storm).

- **Attribute cache:** acregmin/max, acdirmin/max, or coarse actimeo=<sec> to set all four.
- **Training/checkpoints (read-mostly):** longer timers (e.g., actimeo=600).
- **Dataloaders:** shorter timers (e.g., acregmax=60,acdirmax=60).
- **nocto:** disables close-to-open consistency; choose **only** on strictly read-only datasets staged once.

