PURETENSOR
INTELLIGENCE DISTILLED

The Layered Ceiling

Every stage between wire speed and a finished file copy

Paper: PT-TN-2026-001 v1.0

Author: PureTensor Ltd

Date: 28 February 2026

Status: Published

Abstract

A 200 GbE RoCE v2 fabric connecting three nodes was measured across five weeks. Raw RDMA achieved 24.4 GB/s, 98% of the 25 GB/s theoretical line rate. Eight-stream TCP with zero-copy sendfile reached 24.2 GB/s average, 97% of line. A real 2.5 TB restore over rsync and SSH, run on the same fabric in the same window, sustained 1.98 GB/s: 8% of line.

The gap is not one bottleneck but a stack of them, each with its own ceiling. On this path the measured ladder ran: raw TCP 24.8 GB/s, RDMA 24.4 GB/s, NFS-over-RDMA writes 14.5 GB/s, eight-stream rsync daemon 12.9 GB/s, single-NVMe sequential read 10.4 GB/s, rsync over SSH 1.98 GB/s. Synthetic line-rate results had been green for five weeks while every production transfer left 92% of the available capacity unused.

Eleven distinct bottleneck shapes were found and classified. Three of them (TCP socket buffers against the bandwidth-delay product, single-stream congestion-window limits, and serial erasure-coded writes) presented initially as hardware faults and were resolved by measurement rather than repair. Two (userspace copy defeating sendfile, and SSH's single-threaded framing) required policy changes. One (CPU-bound transmit asymmetry on a lower-clocked processor) was left in place because real workloads naturally parallelise past it.

Caveats survive. The 8-stream transmit throughput of one node disagreed between two measurement dates (194 Gbps on 1 February, 149 Gbps on 28 February) and was not reconciled within the window. A mid-window storage rebuild changed the binding constraint on one endpoint, so pre- and post-rebuild numbers are not comparable. Several projected throughput figures for methods not yet deployed remain projections, not measurements.

Publishing only the top number is the standard dishonesty of fabric benchmarking. This note publishes the whole ladder.

1. The instrument under test

Three nodes share a dedicated 200 GbE RoCE v2 fabric. Each carries a Mellanox/NVIDIA ConnectX-6 200G NIC on an isolated private subnet, MTU 9000 (jumbo frames). Line speed is 200 Gbps, which is 25 GB/s theoretical maximum.

Node A is a Threadripper PRO workstation (TR 7970X, Zen 4, 5.3 GHz boost) with two Blackwell-generation GPUs, 512 GB DDR5, and the ConnectX-6 on PCIe Gen5 x16. Node B is a compute node with 256 GB DDR4 and the ConnectX-6 on PCIe Gen4 x16. Node C is a compute

node built around an EPYC 7443 (Zen 3, 4.0 GHz) with 512 GB DDR4 and the same NIC generation on PCIe Gen4 x16.

Storage evolved across the measurement window. Node A began with a single Samsung 9100 PRO (PCIe5 x4, ext4) and was rebuilt on 28 February as a two-drive Samsung 9100 PRO 4 TB RAID0 array (mdadm, 512K chunk, ext4, 7.3 TiB usable). Node B ran a three- or four-drive Samsung 990 PRO RAID0 plus a later two-drive 9100 PRO Gen5 array. Node C ran four Crucial P310 drives in a ZFS stripe.

A separate 25 GbE tier (ConnectX-4 Lx, MTU 1500) carries a four-node Ceph cluster. A 10 GbE tier carries general compute traffic; a 1 GbE tier carries management. Jumbo frames are confined to the 200G fabric. A 9000-byte probe across the 25G tier fails, as designed.

Instruments: iperf3 for TCP (synthetic and zero-copy sendfile from file), ib_write_bw for RDMA (10-second runs, 64 KB messages), socat, dd, fio, rsync in both SSH and daemon modes, RADOS bench, and NFS over RDMA.

2. Wire and RDMA: the ceiling everyone quotes

RDMA write bandwidth from Node A to Node B on 24 January measured 196.03 Gbps at 374,000 messages per second, RoCE v2, MTU 4096. A follow-up measurement on 13 February confirmed 24.4 GB/s (195 Gbps), 98% of line rate. This is the number that appears on the invoice. It is also the number that matters least, because nothing in the production stack touches it.

Latency on 28 February:

Path	Min (ms)	Avg (ms)	Max (ms)	Jitter (ms)	Loss
A → B	0.097	0.282	0.493	0.141	0%
A → C	0.056	0.077	0.114	0.016	0%
B → C	0.102	0.123	0.170	0.019	0%
Into 25G tier	n/a	0.269–0.297	n/a	n/a	0%

The A-to-B path shows higher jitter than the others. The cause was not identified, but throughput was unaffected.

3. TCP and the parallelism law

Synthetic iperf3 on 24 January: a single TCP stream from Node A to Node B reached 69.7 Gbps with zero retransmits. Eight parallel streams reached 197 Gbps (A to B), 198 Gbps (A to C), and 180 Gbps (B to C, with 446 retransmits). The single-stream ceiling is real. Each stream caps around 25–50 Gbps depending on congestion-window dynamics; eight streams recover the link.

On 1 February, a 100 GB file on a RAM disk was transferred using iperf3 with zero-copy sendfile (-F -Z):

Direction	4 streams	8 streams	Retransmits (8)
A → B	165 Gbps (82.5%)	192 Gbps (96.0%)	475
A → C	173 Gbps (86.5%)	196 Gbps (98.0%)	1,384
B → C	110 Gbps (55.0%)	190 Gbps (95.0%)	1,143
C → B	134 Gbps (67.0%)	194 Gbps (97.0%)	0
B → A	118 Gbps (59.0%)	196 Gbps (98.0%)	0
C → A	135 Gbps (67.5%)	195 Gbps (97.5%)	0

Average at eight streams: 193.8 Gbps, 24.2 GB/s, 96.9% of line speed.

The retransmit counts (up to 1,384 on A to C) coexist with 95–98% throughput. Retransmits alone are not a health signal; the throughput beside them is.

Single-stream tools on the same RAM disks performed worse. Socat from A to B reached 14.6 Gbps (7.3% of line). Socat with 256 MB buffers reached 18.8 Gbps. A tar | socat pipeline over 100 one-gigabyte files reached 17.9 Gbps. Four parallel socat instances reached 65.4 Gbps (32.7%). Zero-copy roughly doubles per-stream throughput: socat's userspace copy at around 18 Gbps against iperf3 sendfile at around 45 Gbps per stream.

4. The socket-buffer discovery

Before tuning, Node A carried 208 KB default TCP socket buffers. At 200 Gbps with a 0.4 ms round-trip time, the bandwidth-delay product (the amount of data that must be in flight to fill the pipe) is approximately 10 MB. With 208 KB buffers, Node A would have been limited to roughly 4 Gbps regardless of the NIC.

All three nodes were set to net.core.rmem_max and wmem_max of 256 MB, net.ipv4.tcp_rmem of 4096 87380 268435456, net.ipv4.tcp_wmem of 4096 65536 268435456, and net.core.netdev_max_backlog of 250000. Every figure above 190 Gbps in this note is post-tuning. The pre-tuning state was never measured at scale, only diagnosed from the default values; the 4 Gbps estimate is arithmetic, not observation.

5. A transmit asymmetry that looked like a fault

A full fabric reassessment on 28 February, eight-stream iperf3:

Path	Throughput	% line	Retransmits
A → B	198 Gbps	99.0%	409
B → A	193 Gbps	96.5%	9
A → C	191 Gbps	95.5%	6,126
C → A	147 Gbps	73.5%	4
B → C	191 Gbps	95.5%	2,246
C → B	149 Gbps	74.5%	0

Node C shows a consistent 25% transmit deficit regardless of destination: 147–149 Gbps transmitting against 191 Gbps or better receiving. The link reported 200G full duplex. PCIe Gen4 x16 provides 256 Gbps theoretical bandwidth, so the slot is not the constraint. TX pause frames on Node C numbered 3,558, against 15 on Node B and 10 on Node A: 240 times higher. One tx_pci_signal_integrity event appeared on Node C.

The attributed cause is per-core TCP processing throughput. The EPYC 7443 runs Zen 3 at 4.0 GHz; Node A runs Zen 4 at 5.3 GHz. Prior testing confirmed that 48 streams recover Node C to 195 Gbps. The ceiling is CPU-bound, not NIC-bound, and real many-connection workloads (Ceph, Kubernetes) saturate naturally. Classified moderate severity, no action taken; the only fix is a CPU with higher boost clocks.

A disagreement must be recorded. On 1 February, at eight streams on RAM disks, Node C transmitted at 194–195 Gbps. On 28 February it transmitted at 147–149 Gbps. Both measurements are presented with their dates. The 48-stream recovery supports the CPU-bound reading, but the two 8-stream results were not reconciled within the measurement window. The earlier report attributed the four-stream asymmetry (Node A initiating at 165–173 Gbps versus

Nodes B or C initiating at 110–135 Gbps) to DDR5 versus DDR4 sender memory bandwidth and observed that it vanished at eight streams.

6. Storage feeds the wire, then stops feeding it

Fio baselines on 1 February: Node A's single 9100 PRO read at 9.6 GB/s and wrote at 10.3 GB/s. Node B's three-drive 990 PRO RAID0 read at 18.5 GB/s and wrote at 15.7 GB/s. Node C's four-drive ZFS stripe read at 2.7 GB/s and wrote at 6.3 GB/s.

Node C's ZFS performance required explicit tuning. The default configuration (zstd compression, direct I/O) wrote at 0.6 GB/s. Disabling compression, setting recordsize=1M, sync=disabled, and atime=off reached 6.3 GB/s. Even tuned, Node C remained the weakest storage in the cluster and unsuitable as a bulk receive target.

Disk-to-disk transfers of 200 GB files on 1 February, using iperf3 with the -F flag (disk read plus network, receiver discards):

Path	Throughput	% line
A → B	198 Gbps	99%
A → C	181 Gbps	90%
B → C	180 Gbps	90%
C → B	153 Gbps	76%
B → A	198 Gbps	99%
C → A	149 Gbps	75%

The same paths with eight-stream dd | socat, actually writing to the destination disk:

Path	Throughput	Gbps	% line
A → B	6.72 GB/s	57.7	28.8%
A → C	2.18 GB/s	18.8	9.4%
B → C	1.82 GB/s	15.6	7.8%
C → B	4.51 GB/s	38.8	19.4%
B → A	5.95 GB/s	51.1	25.5%
C → A	4.30 GB/s	36.9	18.4%

The best socat result reached 27% of what the same path sustained under sendfile. Three causes: dd's pipe-to-socat copy defeats sendfile; eight parallel dd processes with skip and count turn a sequential read into a random-like pattern; socat copies through userspace.

7. The application protocol: where most of the capacity went

An NFS-over-RDMA backup pipeline from Node A to Node B on 13 February actually transferred 3,180 GB (3.1 TiB):

Method	Throughput	Gbps	% link
rsync (single-threaded, buffered)	2.7 GB/s	21	11%
tar pipe	2.7 GB/s	21	11%

Method	Throughput	Gbps	% link
parallel cp, 16 jobs, no direct I/O	3.7 GB/s	29	15%
chunk-dd 16×, single large file, cold read	4.1 GB/s	33	16%
chunk-dd 8×, multi-file	10.0 GB/s	79	40%
dd 8× from /dev/zero → NFS	13.0 GB/s	104	52%
fio 8 jobs over NFS-RDMA	14.5 GB/s	116	58%
dd 16× from /dev/zero → single array	21.9 GB/s	175	87%
raw RDMA	24.4 GB/s	195	98%

The layered bottleneck at that date: 200G RDMA wire at 24.4 GB/s, NFS-RDMA write ceiling at 14.5 GB/s, single-NVMe read at 10.4 GB/s (the binding constraint), chunk-dd on real files at 4–10 GB/s. Direct I/O gave a two- to three-times speedup over buffered cp and rsync. Multi-file beat single-file (10 GB/s versus 4.1 GB/s) by avoiding read-offset contention inside the filesystem. NFS-RDMA capped rsize and wsize at 1 MB regardless of the requested 4 MB.

8. Encryption on a trusted path

A restore of 2.5 TB (33,642 files) over the fabric on 28 February sustained 1.98 GB/s and took 21 minutes. Post-mortem measurements on the same path, same hour:

Layer	Throughput	% of 200G
Raw fabric, iperf3 8 streams	24.8 GB/s (198 Gbps)	99%
Single TCP stream, iperf3	8.1 GB/s (64.9 Gbps)	32%
SSH pipe, AES-128-GCM (VAES hardware)	1.94 GB/s (15.5 Gbps)	7.8%
SSH pipe, ChaCha20 (no hardware accel)	0.90 GB/s (7.2 Gbps)	3.6%
rsync over SSH (actual restore)	1.98 GB/s (15.8 Gbps)	7.9%

SSH is single-threaded. Protocol framing, MAC computation, and sequence processing cap one SSH stream near 1.9 GB/s even with hardware crypto. The transfer that should have taken about two minutes took 21. Ninety-two percent of the available capacity was wasted.

The decision recorded: on a private point-to-point fabric with no untrusted hops, transport encryption defends against an adversary that cannot exist on that path, so it is pure overhead. This is an argument about where the trust boundary sits, not a general recommendation to disable encryption. The same policy explicitly does not extend to any path crossing a shared or routed network.

An unencrypted rsync daemon was deployed on both endpoints, bound to fabric addresses only, restricted to the fabric subnet, with use chroot = no and max connections = 8. Measured on 5 GB model shards, Node B to Node A:

Method	Streams	Throughput	Gbps	% of 200G	vs SSH
rsync over SSH	1	1.98 GB/s	15.8	8%	baseline
rsync daemon	1	3.05 GB/s	24.4	12%	1.5×
rsync daemon	4	7.8 GB/s	62.4	31%	3.9×

Method	Streams	Throughput	Gbps	% of 200G	vs SSH
rsync daemon	8	12.9 GB/s	103.2	52%	6.5×

A single daemon-mode rsync stream caps near 3 GB/s on rsync's per-file protocol overhead, against 8.1 GB/s for one raw TCP stream. Eight streams reach 12.9 GB/s, 52% of line, still roughly half the 24.8 GB/s that raw TCP achieves on the identical path. The gap is rsync's own accounting: checksumming, delta encoding, file metadata exchange.

9. After the rebuild: storage stops being the ceiling

Node A was rebuilt on 28 February as a two-drive Samsung 9100 PRO RAID0 array. Fio on the new array: sequential read 29.2 GB/s (27,880 IOPS, 1M blocks, 4 jobs, iodepth 64); sequential write 26.5 GB/s (25,330 IOPS); random 4K read 18.1 GB/s (4,412,787 IOPS, 16 jobs, iodepth 256); random 4K write 22.8 GB/s (5,576,295 IOPS). Node B's arrays are rated at approximately 24 GB/s (four-drive 990 PRO Gen4) and 28 GB/s (two-drive 9100 PRO Gen5).

Both endpoints now exceed the 25 GB/s fabric. The fabric became the sole bottleneck. Pre- and post-rebuild storage numbers for Node A are not comparable: the single 9100 PRO read at 9.6 GB/s and wrote at 10.3 GB/s; the array reads at 29.2 GB/s and writes at 26.5 GB/s. Any claim that storage is no longer the bottleneck is only true after 28 February.

Several throughput figures for methods not yet deployed (parallel tar | nc at an estimated 20–25 GB/s, NFS-RDMA at an estimated 22+ GB/s) remain projections. The only method actually measured after the policy change is the rsync daemon, at 12.9 GB/s.

10. The other tiers and what contention looks like

Node A driving both destinations simultaneously reached 106 Gbps plus 91 Gbps, for 197 Gbps aggregate (98.5%). The ConnectX-6 handles dual destinations at near line rate.

Two 200G senders into one 25G Ceph node reached 12.8 plus 12.4 Gbps, for 25.2 Gbps aggregate: perfect fair-share. The bottleneck is the receiver's 25G NIC, not the switch.

The 25G tier measured 23.5 Gbps (94% of line) on every path. Approximately 15,400 retransmits per 10-second test appeared on 200G-to-25G paths (speed-mismatch buffer overrun), but zero retransmits on the switch-to-switch path. Priority flow control and explicit congestion notification were held in reserve but not deployed.

Earlier tier baselines from 24 January: 25G mesh 22.2–23.5 Gbps; 10G paths 9.36–9.41 Gbps; 1G management path 0.93 Gbps.

11. Erasure-coded storage and the same lesson

The Ceph cluster on 24 January: HEALTH_OK, three monitors, 16 OSDs up and in, 170 TiB total, 167 TiB available. RADOS bench on the SSD erasure-coded pool: 1,103 MB/s write (275 IOPS, 57.9 ms average latency) and 1,230 MB/s random read (307 IOPS, 49.3 ms). The HDD erasure-coded pool: approximately 300 MB/s write (74 IOPS, 213 ms).

A filesystem client from Node A over the 25G tier initially measured 12 MB/s write and was reported as a defect. Fio with 1 MB blocks resolved it: one job at iodepth 1 reached 13 MB/s (13 IOPS); four jobs at iodepth 16 reached 143 MB/s (ten times higher); read at four jobs and iodepth 16 reached 1,680 MB/s (1,682 IOPS).

The cause: the data pool is erasure-coded k=3, m=1 (jerasure, host failure domain) on spinning HDDs. Each 1 MB write becomes four chunks of approximately 333 KB to four OSDs, all of which

must acknowledge. Average IO latency is around 75 ms, and 1000 ms divided by 75 ms multiplied by 1 MB equals 13 MB/s. The observation matched the arithmetic exactly.

Recorded verdict: not a bug. The inherent cost of erasure coding plus HDD seek latency plus strong durability. Remedies listed were parallelism, pinning hot directories to the SSD pool, client mount tuning (rsize and wsize 4 MB, rasize 8 MB), or block volumes on the SSD pool.

A small disagreement in the 24 January reports: the baseline records CephFS single-thread write as 12 MB/s and the analysis as 13 MB/s; the baseline reports the HDD pool read as unmeasured while the summary report gives approximately 300 MB/s.

12. The eleven bottleneck shapes

1. Untuned socket buffers against the bandwidth-delay product. 208 KB buffers against a 10 MB BDP would have capped a 200 Gbps link at approximately 4 Gbps. Fixed by 256 MB buffer maxima.
2. Single-stream TCP ceiling. One stream reaches 8.1–18 Gbps depending on tool; the congestion-window and per-core processing limit is real. Fixed by standardising on eight streams.
3. Userspace copy defeating zero-copy. dd | socat pipes cannot use sendfile and achieved 27% of the same path's sendfile throughput. Fixed by mandating sendfile or direct-I/O chunked tools.
4. Buffered I/O on the read side. Direct I/O gave two to three times the throughput of buffered cp and rsync. Fixed in the production pipeline (16 streams, 4 MB blocks, O_DIRECT).
5. Single-file read-offset contention. Chunked parallel reads of one large file reached 4.1 GB/s; the same parallelism across separate files reached 10.0 GB/s.
6. Filesystem defaults on the weakest node. Default ZFS compression plus direct I/O wrote at 0.6 GB/s; disabling compression and sync and setting a 1 MB record size reached 6.3 GB/s.
7. Encryption on a trusted path. SSH's single-threaded framing capped bulk transfer at 7.8–7.9% of link, hardware crypto notwithstanding. Fixed by an unencrypted, subnet-restricted rsync daemon.
8. Application-protocol overhead. Even unencrypted, rsync's per-file protocol accounting caps one stream near 3 GB/s and eight streams at 52% of line.
9. CPU-bound transmit asymmetry. An older-generation, lower-clocked server CPU cost approximately 25% of transmit throughput at eight streams, visible as 240 times the TX pause frames; recoverable at 48 streams.
10. Cross-tier speed mismatch. Approximately 15,400 retransmits per 10 seconds when a 200G sender targets a 25G receiver; zero on same-speed paths. Classified expected, not fixed.
11. Latency-bound erasure-coded writes misread as a fault. A 12–13 MB/s single-thread figure was a correct 75 ms-per-IO consequence of k=3, m=1 on HDDs, not a defect.

13. What survives and what does not

The RDMA and eight-stream TCP numbers are solid: multiple runs, multiple dates, multiple paths, consistent within a few percent. The storage numbers after the 28 February rebuild are solid for Node A; the pre-rebuild numbers are solid for their date but no longer describe the system. The rsync-daemon numbers are from a single measurement session and should be

confirmed before being used as capacity planning inputs.

The Node C transmit disagreement (194 Gbps on 1 February, 149 Gbps on 28 February, both at eight streams) is unresolved. Both figures are real measurements. The 48-stream recovery to 195 Gbps supports the CPU-bound hypothesis, but something changed between the two dates that was not identified.

The projected throughput figures for parallel tar | nc and NFS-RDMA remain projections. They are plausible extrapolations from the measured ceilings, but they have not been measured.

The single most valuable measurement in the whole window is the cheapest: a real 2.5 TB restore that ran at 8% of link and would have gone unnoticed had nobody instrumented each layer beneath it.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction.

PT-TN-2026-001. Measurements taken 24 January 2026 to 28 February 2026. Raw artefacts retained.