PURETENSOR
INTELLIGENCE DISTILLED

Past the VRAM Wall

Serving a 397B model that does not fit three workstation GPUs

Paper: PT-TN-2026-010 v1.0

Author: PureTensor Ltd

Date: 3 May 2026

Status: Published

Abstract

A 397-billion-parameter mixture-of-experts model with a 378.23 GiB FP8 checkpoint was served across three 96 GB-class Blackwell workstation GPUs totalling roughly 294 GiB of usable VRAM. The checkpoint exceeds aggregate GPU memory by a factor of 1.29. The model ran.

The configuration that worked: pipeline parallelism across three GPUs spanning two nodes over a 200G Ethernet/RoCE fabric, with 64 GB of CPU memory per GPU reserved for expert-weight offload, context capped at 2,048 tokens, and a maximum of four concurrent sequences. Warm single-stream throughput reached 8.80 tokens per second; four concurrent 128-token streams achieved 17.77 tokens per second aggregate, reproducible to within 1.7% across two independent passes. Cold start was brutal: 24.88 seconds to first token, 1.86 tokens per second, rising to 1.20 seconds and 8.80 tokens per second once warm. Batching rescued the configuration. A single request through an empty pipeline is the worst case; keeping the pipe full delivered most of the gain.

GPU utilisation during the best runs averaged 44–50% on the two-GPU node and 71% on the single-GPU node, with power draw well below board limits. The fabric, measured at 151 Gbps for bulk object transfer, was never the constraint. The bottleneck was memory capacity and the overhead of shuttling expert weights between host and device, not network bandwidth, not thermal headroom. There is idle silicon in every measured second of this benchmark.

This is a capability proof, not a performance result. Context was short, concurrency was low, eager execution was forced, no optimised MoE kernel existed for this GPU and weight shape, and the enabling fix was a local patch to a vLLM assertion rather than an upstream change. The demonstration shows that the VRAM wall is not a hard stop; it also shows that crossing it extracts a price, paid immediately and in full.

The gap, and the plan to cross it

The model is the official FP8 checkpoint of a 397-billion-parameter mixture-of-experts architecture with 17 billion active parameters per token. Checkpoint size as recorded in the safetensors index: 406.13 GB decimal, 378.23 GiB. The available hardware: two GPUs in a Threadripper PRO workstation (Node A), one Max-Q variant in a second node (Node B), each reporting about 97.9 GiB usable VRAM. Aggregate: roughly 294 GiB. The checkpoint is 84 GiB larger than the sum of all GPU memory in the building.

No quantisation step below FP8 was attempted. The question was whether the model could be served at all without shrinking the weights, using only the machinery already deployed.

The inter-node path is a 200G Ethernet/RoCE fabric. Ray 2.52.1 spans the nodes; vLLM uses Ray as its distributed executor backend. A third cluster member, a 64-core EPYC node with no GPU, contributes CPU and memory but no inference capacity. At full strength the cluster reports 236 CPUs, 3 GPUs, 612.26 GiB memory, and 600 GB object store. The two GPUs inside Node A share a PCIe topology at NODE class. There is no NVLink or NVSwitch anywhere.

The plan, written before any measurement, followed vLLM's own distributed guidance: for a model too large for one node, combine tensor parallelism with pipeline parallelism; where the GPU count does not divide cleanly, run tensor-parallel size 1 with pipeline-parallel size equal to the GPU count. CPU offload extends each GPU with host RAM over unified addressing, but the documentation warns that it demands a fast CPU-to-GPU interconnect and should be minimised. The plan targeted the expert weights specifically for offload, keeping the always-used dense and attention weights GPU-resident.

The trial order: stage the weights on both nodes, quiet the other GPU services, run a two-GPU smoke test with CPU expert offload, then bring up the three-GPU Ray configuration at PP=3 and TP=1. Pipeline parallelism was chosen specifically to minimise cross-node traffic compared to TP=3, which would have required every layer's activations to traverse the fabric. A fallback path, a CPU-GPU heterogeneous MoE kernel under a different serving stack, was documented but never needed.

Clearing the path

Three categories of obstacle had to be removed before the model would start.

The Ray cluster itself was the first. Ray refuses to join workers across Python patch-level mismatches: a 3.12.13 worker could not join a 3.12.3 head node, and the worker's virtual environment had to be rebuilt to match exactly. The object store must be sized against /dev/shm, which had to be remounted larger before the Ray services started. Stale Ray nodes from failed start attempts pollute the autoscaler and must be force-stopped before a clean launch. These are not interesting problems, but they gate any reproduction.

The fabric was the second. A 1 GB Ray object transfer completed in 0.06 seconds during bring-up, roughly 151 Gbps. A separate iperf3 measurement after a physical card reseat on Node B held at 118 Gbit/s; the NIC was stuck at PCIe width x8, downgraded from its nominal width, and the reseat did not fix it. The lane count, not the cabling, is the limit. For this trial the 118–151 Gbps range was more than sufficient.

The vLLM runtime was the third, and the only one that required a code change. When CPU weight offload attempted to reinitialise the input batch under this model's hybrid KV-cache behaviour, vLLM raised an assertion. Without a fix, the CPU-offload path could not start at all. A reversible local patch to the GPU model-runner was applied on Node A, guarded behind an explicit environment flag so the relaxed assertion is opt-in. On Node B the site-packages path was not writable by the serving user, so the same change was injected at runtime as an import-time monkeypatch, propagated to Ray workers through the job runtime environment. Original files were backed up before modification and the patch was reverted at cleanup.

Two further environment-level fixes were required. The first three-GPU Ray attempt failed on NCCL transport selection; pinning the NCCL socket interface to the 200G path on both nodes and disabling the InfiniBand transport resolved the startup failure. Separately, vLLM logged that no optimised MoE kernel configuration existed for this GPU class at this exact FP8 MoE shape. That warning represents an unquantified amount of performance left on the table.

What the measurements show

The serving configuration for the best-performing run: tensor-parallel size 1, pipeline-parallel size 3, GPU memory utilisation 0.92, 64 GB CPU offload per GPU with offload restricted to expert parameters, maximum model length 2,048 tokens, maximum sequences 4, maximum batched tokens 8,192, FP8 KV cache, language-model-only mode with reasoning disabled, eager execution enforced.

Every row in the table below is a direct measurement from the trial's benchmark logs.

Configuration	Workload	TTFT	Wall time	Output tokens	Throughput	Note
2 GPU, TP=2, 128 GB offload/GPU	1×64 cold	14.95 s	19.61 s	64	3.26 tok/s	First working path
3 GPU Ray, PP=3, 64 GB offload/GPU	1×64 cold	24.88 s	34.40 s	64	1.86 tok/s	Pipeline bubble, cold
3 GPU Ray, PP=3, batch-4	4×64 concurrent	10.15 s median	22.79 s	256	11.23 tok/s	Early batch run
3 GPU Ray, PP=3, batch-4	4×128 concurrent	1.65 s median	29.29 s	512	17.48 tok/s	Best aggregate
3 GPU Ray, PP=3, batch-4	1×128 warm	1.20 s	14.55 s	128	8.80 tok/s	Best single stream
3 GPU Ray, PP=3, batch-4	1×64 warm	1.20 s	7.85 s	64	8.15 tok/s	Warm single check
3 GPU Ray, PP=3, batch-4, telemetry	4×128 concurrent	1.67 s median	28.82 s	512	17.77 tok/s	Repeat with telemetry

The cold-to-warm delta is the largest single effect. Time to first token falls from 24.88 seconds cold to 1.20 seconds warm. Single-stream throughput rises from 1.86 to 8.80 tokens per second, a 4.7× gain on the identical configuration, purely from being warm.

The aggregate throughput at four concurrent streams (17.48 and 17.77 tokens per second across two passes) is real concurrency, not one fast stream carrying the average. In the best 4×128 run the four streams returned 4.40, 4.37, 4.37, and 4.37 tokens per second individually. The two independent passes of the same configuration agree to within about 1.7%, so the ceiling is reproducible at least at this sample size.

Batching is what rescues this configuration. Going from one to four concurrent 128-token streams doubles aggregate throughput while lowering median time to first token (1.65 seconds at batch-4 versus 24.88 seconds cold single). Pipeline parallelism with an empty pipe is the worst case; keeping the pipe full is most of the win.

Where the time went

During the repeated 4×128 run, GPU utilisation averaged approximately 44–50% on the two Node A GPUs and about 71% on the Node B GPU. Power draw stayed well below board limits throughout. The fabric, measured at 151 Gbps for bulk object transfer, was not saturated.

The conclusion the numbers force: the run is bounded by CPU offload latency, partitioning overhead, unoptimised kernels, and pipeline scheduling, not by raw board power or thermal headroom. More aggregate VRAM would help first; more GPUs to split cleanly would help second; faster east-west fabric would help third; runtime and partitioning quality would help

fourth. NVLink or NVSwitch comes last for inference, and only where tensor-parallel collectives dominate.

For training the ordering inverts. Training is communication-heavy: forward pass, backward pass, gradient synchronisation, optimiser state, reduce-scatter, all-gather, all-reduce. There NVLink or NVSwitch inside a node plus very fast inter-node fabric move close to essential. This trial says nothing empirical about training; the inference conclusion should not be read as one.

What was not measured, and why it matters

The research plan carried a fabric-extrapolation table for a hypothetical 200 GB model that fits entirely in 3×96 GB VRAM and delivers 50 tokens per second all-local. The table estimated throughput at various fabric speeds: 1G Ethernet roughly 5 tokens per second (range 2–10); 10G roughly 25 (15–35); 25G roughly 33 (25–40); 200G Ethernet/RoCE roughly 44 (40–47); 400G roughly 46–47 (43–49); 800G roughly 48–49 (47–50); local NVLink/NVSwitch roughly 60 (55–70). These are explicitly labelled informed engineering estimates, not direct measurements. The only measured point in the trial is the 200G run of the non-fitting model. The estimates assume a fitting model, clean pipeline parallelism, decent batching, and no CPU weight offload. They are the plan's predictions, not results.

TP=3 was never benchmarked. The choice of PP=3 over TP=3 was a design decision based on vLLM's documentation and the desire to minimise cross-node traffic, not a measurement-driven comparison. Whether TP=3 would have performed better or worse on this fabric is unknown.

Context was capped at 2,048 tokens and sequences at 4, which puts both long context and high concurrency outside what these numbers can speak to. Eager execution was forced, so no CUDA-graph benefit was captured. The enabling vLLM patch relaxes a safety assertion behind a flag and is not an upstream fix; anyone reproducing this must apply the same patch or wait for an upstream change that may never arrive.

The trial is a single run across two nodes, unrepeated except for the one 4×128 telemetry pass. The 1.7% agreement between those two passes is encouraging but not a substitute for a larger sample. The claims that survive: the model loaded, the model ran, the throughput numbers are what they are. The claims that do not survive without further work: any statement about how this configuration would perform at longer context, higher concurrency, or under production load patterns.

Cleanup and residues

The trial was fully unwound the following morning. The serving session was stopped and the endpoint confirmed dead. The vLLM patch was reverted. The staged weights were deleted from both nodes. All previously quieted GPU services (speech-to-text, three text-to-speech services, an image service and web UI, a consumer service, and the resident model server on Node B) were restarted and individually health-checked, including a smoke completion against the restored resident model.

Two residues were recorded honestly rather than papered over. A 64 MB root-owned download-cache fragment on Node B was left in place, because the standing rule is to report a permission error rather than force ownership changes. The operator noted afterwards that archiving the weights before deletion would have been preferable to deleting them outright; this became a policy change for future large-model cleanups.

The run directory retains the serve scripts, per-request benchmark logs, per-second GPU telemetry, and the pre-patch backups. The weights themselves were deleted at cleanup. Anyone reproducing this re-downloads 406.13 GB first.

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-2026-010. Measurements taken 29–30 April 2026 (serving trial) and 5–15 April 2026 (Ray cluster bring-up). Raw artefacts retained.