PURETENSOR
INTELLIGENCE DISTILLED

A 475 GB Model Across Two Workstations

First light, a hang, and its forensic

Paper: PT-R-2026-022 v1.0

Author: PureTensor Ltd

Date: 22 September 2026

Status: Published

Abstract

We report first light for DeepSeek-V4.1-Flash, a 475 GiB checkpoint carrying a 189 GiB static embedding table the publisher calls Engram, served across two GPU workstations joined by a 200 GbE RDMA fabric. The model's routed experts ship in MXFP4, a 4-bit block floating-point format; the table is exact FP8. Four workstation Blackwell GPUs (96 GB each, PCIe, no NVLink) supply 384 GB of VRAM, short of the weights alone. A community recipe places the table off-GPU entirely, in a C++ row store that serves rows from NVMe via direct reads with a 64 GiB host-RAM cache, invoked from a CUDA host callback inside CUDA graphs. We adapted the recipe's single-host launcher to two nodes: tensor-parallel 4, expert-parallel 4, one rank per GPU, NCCL over the RDMA interface.

With the bundled draft head enabled (speculative decoding), single-stream decode at 512 input tokens reached 178.3 tok/s; at eight concurrent requests the cluster delivered 628.3 tok/s total decode, a cross-node haircut of roughly 12% against the recipe author's single-box figure of 713.5 tok/s. Prefill throughput matched or exceeded the single-box numbers. With the draft head off, single-stream decode fell to 27–28 tok/s, bound by per-step latency (cross-node synchronisation plus the host callback) rather than compute; batched throughput still scaled, reaching 813 tok/s at 32 concurrent 2,048-input requests.

On our internal frontier benchmark (94 original items, ten dimensions, every verdict produced by code, no language-model judge), the model scored 92.1 agentic, 96.0 scorer, 93.5 flat. These composites beat the previous large seat (a GLM-5.3 full-size EXL3 quantisation: 90.8, 95.4, 92.7) on every dimension; scorer 96.0 is the record for a locally served seat, and flat 93.5 ties the fleet record set by a hosted model. The model became the fleet's large seat the same day.

Three scheduler-watchdog hangs occurred during the draft-on matrix (300 s timeout, full restart required). The signature was first-node GPUs at 100% utilisation, second-node GPUs at 0%, no Xid or fabric-adapter errors. NCCL logs showed all four ranks had enqueued identical collective sequences (70,682 operations each), ruling out rank-divergent work. A discriminating experiment swapped the NVMe row store for a RAM-backed mode (table rows locked in host memory). Ninety-two provoke cases and 30 minutes produced zero hangs; the NVMe path hung at case 26 under the same loop. We infer that the NVMe direct-read path, running inside a CUDA host callback under a 256-way mutex, stalls under variable speculative gather sizes across nodes. RAM mode is now the default. Thirty minutes of clean operation is strong evidence, not proof; relaunch counts remain under watch.

1. The Model and the Fit Problem

DeepSeek-V4.1-Flash ships as a 475 GiB native checkpoint. Its routed experts are stored in MXFP4, and it carries a 189 GiB static embedding table (Engram), a large lookup of exact FP8 rows the model consults per token, distinct from the weights. A bundled draft head for speculative decoding adds about 2 GB per rank.

Our hardware is two GPU workstations, each containing two workstation Blackwell GPUs (96 GB each, PCIe, no NVLink). A 200 GbE RDMA fabric joins the hosts. Four GPUs yield 384 GB of VRAM. The weights alone exceed this; the table on top is out of the question.

Two community repacks exist, both 430 GB, converting the table from FP8 to FP4 (cosine similarity 0.9934, unevaluated). The expert nibbles are bit-identical to the native MXFP4; no serving engine reads the repacked layout. We downloaded one, verified it against both engines' loaders, and discarded it. The benchmark target is the native checkpoint.

2. The Recipe and Its Adaptation

A community-published single-host recipe addresses this model on four workstation Blackwell GPUs. It pins SGLang by digest, applies a patch enabling the model's latent-attention kernel on this GPU class, fixes a GEMM planner issue, and introduces a C++ row store that keeps the 189 GiB table off the GPU entirely. The row store serves exact FP8 rows from NVMe via direct (unbuffered) reads, backed by a 64 GiB host-RAM cache. Invocation happens from a CUDA host callback, allowing the gather to run inside CUDA graphs.

The original launcher asserts four GPUs on one host. We replaced it with a two-node launcher: tensor-parallel 4, expert-parallel 4, one rank per GPU, two ranks per node. The fabric adapter is passed into each container; NCCL runs over the RDMA interface. The engine's custom all-reduce had to be disabled (mandatory across nodes; with it enabled, the first run hung at two concurrent requests).

Fit per rank: 72.6 GB of weights plus 2 GB draft head, leaving 17 GB. At 85% memory fraction with the draft head on, the KV pool holds 260,352 tokens; with the draft off, about 890,000 tokens. Context length is 409,600. A single full-context request would not fit the pool with the draft on; eight 32k requests queue comfortably.

Copying the 476 GB checkpoint to the second node across the fabric took about 3 minutes. Boot time ranged from 150 to 240 s to healthy: 41 s for weight load per rank, roughly 45 s for CUDA-graph capture.

3. Throughput With the Draft Head Enabled

We ran the recipe's own benchmark matrix: 8,192 forced output tokens, ignoring end-of-sequence, varying input length and concurrency. The table below shows our measured throughput alongside the recipe author's single-box figures for comparison.

Input tokens	Concurrent	Prefill tok/s	Total decode tok/s	Per-request tok/s	TTFT (s)	Recipe author single-box decode tok/s
512	1	2,435	178.3	178.3	0.21	200.9
512	2	3,363	261.1	130.5	0.30	347.0
512	4	4,445	449.2	112.3	0.46	517.0
512	8	3,564	628.3	80.0	1.02	713.5

Input tokens	Concurrent	Prefill tok/s	Total decode tok/s	Per-request tok/s	TTFT (s)	Recipe author single-box decode tok/s
2,048	1	6,212	173.4	173.4	0.33	226.8
2,048	2	6,226	315.3	157.7	0.66	376.4
2,048	4	6,394	448.2	111.4	1.12	545.1
2,048	8	6,378	619.5	78.1	1.76	729.1
8,192	1	7,088	151.8	151.8	1.16	196.4
8,192	4	7,270	396.6	99.7	3.22	538.8
8,192	8	7,636	642.3	80.5	5.15	704.5
32,768	1	8,021	189.4	189.4	4.08	211.5
32,768	2	8,105	339.7	169.8	6.23	370.8
32,768	4	8,156	426.0	106.5	10.37	526.6

The cross-node haircut against the recipe author's single-box PCIe numbers is about 10–15% in single-stream decode and 15–25% at four to eight concurrent requests. Prefill throughput is at parity or slightly better in most cells. Draft acceptance rates logged between 0.76 and 0.91.

Smoke tests passed: arithmetic, JSON-schema output, a tool round trip, and a 1,024-token vision request.

4. Throughput With the Draft Head Disabled

Disabling the draft head (k=1) removes speculative decoding. Single-stream decode fell to 27–28 tok/s at 512, 2,048, and 8,192 input tokens. Time to first token was 0.31, 0.34, and 1.13 s respectively; prefill reached 1,627, 6,073, and 7,217 tok/s. The single-stream rate is bound by per-step latency (cross-node synchronisation plus the host callback), not compute.

Per-request throughput rises with batch size. At eight concurrent requests the per-request rate reached 39 tok/s. A concurrency ladder (1,024 output tokens) showed continued scaling:

Input tokens	Concurrent	Total decode tok/s
512	8	312
512	16	509
512	32	777
2,048	16	526
2,048	32	813

Throughput still scaled at 32 concurrent, the launch cap. No hang occurred in ten minutes of operation.

5. Benchmark and Fleet Promotion

We ran our internal frontier benchmark: 94 original items, ten dimensions, every verdict produced by code, no language-model judge. Effort was set to low, sampling to k=1, coverage was 100%, and results are comparable across the fleet.

Composite	DeepSeek-V4.1-Flash	GLM-5.3 EXL3 (previous large seat)
Agentic	92.1	90.8
Scorer	96.0	95.4
Flat	93.5	92.7

The new model beats the previous large seat on every composite. Scorer 96.0 is the record for a locally served seat. Flat 93.5 ties the fleet record set by a hosted model. The model became the fleet's large seat the same day the measurements were taken.

6. The Hang and Its Forensic

Three scheduler-watchdog hangs occurred during the draft-on matrix. The watchdog fires at 300 s, kills the engine's process tree, and a full restart is required. The hangs appeared at 512-input 2-concurrent (before we disabled the custom all-reduce), at the 8,192-input transition from 2 to 4 concurrent, and at 32,768-input 8-concurrent.

With the draft head off, zero hangs occurred across roughly 60 minutes, including the full benchmark run.

The signature was consistent: the first node's GPUs at 100% utilisation, the second node's at 0%, no GPU Xid errors, no fabric-adapter errors.

We reproduced the hang under diagnostics (fabric-debug logging, process-tracing capability in the containers, no automatic relaunch). Twenty-five clean cases of 8,192-input 8-concurrent 1,024-output completed (about 23 s each). The 26th case stalled. The last logged decode batch showed acceptance length 5.04 and rate 0.81. The watchdog fired 5 minutes 24 seconds later on the second node's two ranks.

NCCL logs showed all four tensor-parallel ranks had enqueued identical collective sequences: 70,682 operations each, with identical tails (an all-gather of 32,320 bytes five times, then a broadcast of count 1 on the second communicator). This is not a rank-divergent collective; every rank posted the same work. NCCL's debug log records enqueue, not completion. The GPU asymmetry (first node spinning, second node idle) means the first node's ranks were inside a collective kernel waiting for peers whose streams never reached that point. Something host-side on the second node blocked the stream ahead of the collective.

Two diagnostic paths failed. The Python stack sampler ran after the watchdog's kill, so the container was gone and the dumps were empty. A fix is now in place: sample when decode-batch logging stops for 60 s, before the 300 s watchdog. The framework's collective flight recorder produced no trace because this engine's collectives do not pass through the framework's process group.

The discriminating experiment targeted the host callback. Under speculative batching, gather sizes vary per step. The row store gained a mode that keeps the table rows in locked host RAM (about 95 GiB per node; the nodes have 503 and 251 GB of host memory) instead of NVMe. We ran the same image, same diagnostics, same provoke loop.

Results from 11 September: RAM mode, 92 cases, 30 minutes, 513+ requests, zero hangs. Per-case time dropped to about 20 s versus 23 s in NVMe mode (the RAM gather is faster). Boot time rose to 320 s versus 167 s because the rows are prefaulted. NVMe mode the same evening: hang at case 26 (about 10 minutes). Earlier that day, three hangs had occurred in about 75 minutes of NVMe-mode operation.

The diagnosis: the NVMe path (a direct read inside a CUDA host callback under a 256-way mutex) stalls under variable speculative gather sizes across nodes. RAM mode is now the default for this seat.

7. What These Numbers Will Not Carry

Each cell in the throughput matrix is one pass. The concurrency ladder is one pass per rung. Variance is unknown; the results are point estimates.

The recipe author's single-box numbers come from their hardware, with the table and weights on one host. The haircut we report (10–25% depending on concurrency) is an estimate of the fabric's cost, not a controlled comparison. Hardware, firmware, and driver versions may differ.

The hang forensic rests on two runs under identical provoke loops: one NVMe, one RAM. The mechanism (host-callback stall under a mutex) is inferred from the NCCL enqueue symmetry and the RAM-mode result, not observed directly in a stack sample. The stack sampler fired too late in the NVMe run. Thirty minutes of clean RAM-mode operation is strong evidence, not proof; a rare timing condition could still surface.

Greedy decode on this engine is non-deterministic across batch composition (this is documented upstream). Byte-identity checks between configurations are not meaningful. A 20-prompt greedy set matched 6 to 7 of 20 on rerun in every pairing we tried.

The benchmark scores are single-run, effort low, k=1. Sampling variance is not characterised. The comparison to GLM-5.3 EXL3 is on identical prompts under identical harness settings, but the two models differ in architecture, quantisation, and serving engine.

Sample sizes for the hang forensic: 25 clean NVMe cases before the hang, 92 clean RAM cases. The NVMe hang rate (roughly one per 25 cases under provoke) is estimated from a small sample.

8. What Changed

RAM mode is now the default for this seat. The row store prefaults the 189 GiB table into locked host memory at boot, adding roughly 150 s to startup but eliminating the observed hangs. Relaunch counts are monitored; if the hang reappears, we will revisit.

The stack-sampler trigger has been moved: it now fires when decode-batch logging stops for 60 s, well before the 300 s watchdog. If the hang recurs, we will have live stacks.

The model is in production as the fleet's large seat. The next experiment is a controlled comparison of NVMe-mode and RAM-mode throughput under sustained load, with variance estimates, to determine whether the 3 s per-case difference holds and whether NVMe mode can be rehabilitated with a modified mutex strategy.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. Measurements were taken on 10 and 11 September 2026; raw artefacts are retained. This paper is PT-R-2026-022.