PURETENSOR
INTELLIGENCE DISTILLED

Zero Percent or Nothing

The performance cliff at the VRAM boundary

Paper: PT-TN-2026-005 v1.0

Author: PureTensor Ltd

Date: 18 February 2026

Status: Published

Abstract

This note documents a performance cliff at the boundary between full GPU residency and partial CPU offload during local mixture-of-experts inference. The measurements were taken on a two-GPU node (RTX PRO 6000 Blackwell, 192 GB total VRAM, no NVLink) running several large language models across multiple runtimes.

The cliff is steep. On MiniMax M2.5, the transition from 0% to 3% CPU offload costs 40% of throughput: 121 t/s falls to 72 t/s. The relationship is binary rather than proportional. Inside the fully-resident band, throughput is nearly flat across an eightfold range of context length (2048 to 16384 tokens). Outside it, every percentage point of offload extracts a toll that no other tuning axis can repay.

The practical consequence is counterintuitive: the smaller, nominally lower-quality quantisation wins. Q5_K_XL at 151 GB on disk runs 68% faster than Q6_K at 175 GB, because the former fits and the latter does not. The same pattern recurs in a head-to-head between Qwen3.5-397B and the incumbent Qwen3-235B. The larger model, architecturally newer and more capable on paper, cannot achieve full residency on this hardware. It runs at 27.7 t/s with four CPU layers and the audio services stopped. The smaller model runs at 65.8 t/s with everything resident and the services running. The smaller model stays.

A second cliff appears at the concurrency axis, but only after residency is secured. A batching runtime (vLLM) scales near-linearly to eight-way concurrency, reaching 821.9 aggregate t/s at 32-way. A serialising runtime (Ollama on an architecture it could not parallelise) holds flat at 115 t/s while median time-to-first-token rises from 0.12 s to 30.87 s. In production, an 18-way parallel run completed in 27.3 seconds what a sequential run completed in 304.2 seconds: 11.1x improvement at 62% parallel efficiency. These gains are available only to deployments that have already solved the residency problem.

The sample sizes here are small, the comparisons are confounded, and several results reflect software defects with a shelf life. The central claim survives these caveats: on a fixed local VRAM budget, model residency dominates every other tuning axis available.

1. The test environment

The hardware under test was a single node: two RTX PRO 6000 Blackwell GPUs (96 GB GDDR7 ECC each, 192 GB total) on a Threadripper PRO host with 503 GiB DDR5. The GPUs

communicate over PCIe Gen 5 x16; there is no NVLink. Quoted memory bandwidth is 1,597 GB/s per GPU, with peak FP4 throughput of 4 PFLOPS per card.

The absence of NVLink shapes the stack's standing preferences. Tensor parallelism, which splits individual matrix multiplications across devices, is sensitive to interconnect latency. Pipeline parallelism, which assigns whole layers to devices, tolerates PCIe better. Mixture-of-experts architectures, which activate only a fraction of their parameters per token, tolerate it better still. These preferences are not incidental to the measurements; they are part of what the measurements are about.

Persistent resident services occupied roughly 9–10 GB of GPU 0 throughout: a Whisper speech-recognition server (2.9 GB), XTTS v2 (2.4 GB), and Qwen3-TTS (4.8–5.0 GB). Their presence is deliberate. They are part of the production workload, and their footprint is part of the VRAM budget that determines whether a model fits.

Runtimes tested included Ollama (versions 0.15.2, 0.15.4, 0.15.5-rc2, and 0.16.1), vLLM 0.12.0, and llama.cpp built from source at commit b8070.

2. The cliff at the VRAM boundary

Two quantisations of MiniMax M2.5 were swept across context sizes using an identical six-test benchmark battery (short answer, code, long prose, mathematical reasoning, summarisation, long-context prompt). The runtime reported its own CPU/GPU split; memory is total footprint.

MiniMax M2.5 Q6_K (175 GB on disk)

Context	CPU/GPU split	Memory	Avg gen speed
196608	47% / 53%	337 GB	25.5 t/s
8192	7% / 93%	193 GB	62.5 t/s
4096	5% / 95%	190 GB	65.4 t/s
2048	3% / 97%	188 GB	72 t/s
512	3% / 97%	187 GB	72 t/s

Q6_K never reaches full residency. Its floor is 3% CPU at any context size, including 512. Shrinking context cannot buy back what the weights alone overrun.

MiniMax M2.5 Q5_K_XL (151 GB on disk)

Context	CPU/GPU split	Memory	Avg gen speed
196608	42% / 58%	311 GB	29.0 t/s
16384	0% / 100%	173 GB	120 t/s
8192	0% / 100%	167 GB	121.5 t/s
4096	0% / 100%	164 GB	121.6 t/s
2048	0% / 100%	162 GB	121.3 t/s

The 0% to 3% CPU-offload boundary is a cliff, not a slope. The best fully-resident configuration runs at 121 t/s; the best partially-offloaded one runs at 72 t/s. That is a 68% speedup for crossing a threshold that sounds, on paper, like a rounding error.

Inside the fully-resident band, the curve is flat. Generation speed varies by less than 2% across an eightfold range of context (2048 to 16384 tokens). Context size is nearly free until it crosses the boundary, at which point it is catastrophic.

At context 32768, even Q5_K_XL spills to 3% CPU and drops to 75 t/s. The deployed default was set at 16384: 100% GPU, 120 t/s, sufficient for the large majority of the operator's agent tasks.

The practical envelope derived from these measurements: with 192 GB of VRAM and the resident audio services in place, the maximum model size for 100% GPU residency at 16K context is approximately 155–160 GB. Q5_K_XL at 151 GB fits with roughly 25 GB of headroom for Whisper, XTTS, and TTS.

The quantisation that is smaller and nominally lower quality is unambiguously the better deployment. The 24 GB saved between Q6_K and Q5_K_XL buys a 1.68x speedup.

3. The same cliff, seen from the layer-offload side

Qwen3.5-397B-A17B arrived during the measurement window: a hybrid SSM-Transformer architecture (Gated DeltaNet plus attention at 3:1), 512 experts, 17B active parameters, 262K context. It was fetched as a Q4_K_XL GGUF in six shards (214 GB) and merged to a single 200 GB file.

Ollama 0.16.1 had no support for the qwen35moe architecture. The ollama create command succeeded, but ollama run failed with "unknown model architecture: 'qwen35moe'". Serving was done through llama.cpp b8070 directly.

The model is 200 GB against a 192 GB budget. It cannot fit. Layer-mode splitting places about 3.3 GB per layer, capping at roughly 29 layers per GPU. Sweeping GPU layer count from 45 to 59 of 60:

GPU layers	CPU layers	Resident GPU services	Gen speed
45/60	15	running	14.7 t/s
52/60	8	running	21.0 t/s
53+/60	n/a	running	OOM
56/60	4	stopped	28.2 t/s
57+/60	n/a	stopped	OOM

Speed rises monotonically with layers on the GPU (14.7 to 21.0 to 28.2 t/s), and the ceiling is set not by compute but by where OOM lands. Killing the audio services buys exactly four more layers and 7 t/s, then OOM again. Full residency is unreachable at any setting.

MoE layer granularity forbids fine tuning. Layers are indivisible 3.3 GB chunks, so the offload fraction is quantised too. There is no smooth knob between 52 and 56.

4. The head-to-head: 397B versus the incumbent

An eight-test battery (mathematical reasoning, logic puzzle, coding, instruction following, knowledge retrieval, text analysis, constrained creative writing, multi-step reasoning) was run identically against both seats. Qwen3.5-397B ran at 56 GPU layers with services stopped; Qwen3-235B-A22B Q4_K_M ran at 100% GPU with all services running.

Metric	Qwen3.5-397B	Qwen3-235B	Winner
Gen speed	27.7 t/s	65.8 t/s	Qwen3 (2.4x)
Accuracy	6/8	8/8	Qwen3
VRAM fit	needs services off + CPU offload	100% GPU	Qwen3
Wall-clock (math)	67 s	7 s	Qwen3 (9.6x)
Response quality	more detailed explanations	concise, correct	tie
Thinking depth	extensive CoT	standard CoT	Qwen3.5 (when it completes)

The two Qwen3.5 failures were 120-second timeouts, not wrong answers. Thinking mode emitted 3–12x more tokens per answer than the incumbent. The verbosity penalty compounds the speed penalty: a 2.4x slower seat producing 3–12x more tokens yields the 9.6x wall-clock gap on the mathematical item.

Thinking mode had no local disable (no /nothink equivalent), so the token cost could not be traded away.

Verdict recorded: the smaller model stays as daily driver. The larger, architecturally newer model is rejected on fit, not on capability.

5. The runtime axis: offload dominates engine choice

A single-request comparison between vLLM and Ollama on the same model class, Qwen3-235B-A22B:

Metric	vLLM 0.12.0, FP8	Ollama, Q8_0
Model size	239 GB	233 GB (merged GGUF)
CPU offload	80 GB explicit	~60 GB automatic
VRAM used	180.2 GB	191.5 GB
Startup time	~9.5 min	~18 s
Avg tok/s	5.44	18.63
Avg latency	95.71 s	27.48 s

With CPU offload, the production-grade serving engine is 3.4x slower than the hobbyist one. This is the cliff wearing a different costume: vLLM's 80 GB explicit offload over PCIe is the binding constraint, not its scheduler.

The same engine on the same model class with the weights made to fit (Qwen3-235B-A22B GPTQ-Int4, 124.5 GB across 32 safetensors shards, no CPU offload) serves single requests at 35.8 t/s. That is 6.6x the offloaded FP8 figure and nearly 2x Ollama's 18.63 t/s. The engine did not change; the residency did.

A caveat on the vLLM numbers: --enforce-eager was required on vLLM 0.12.0 with GPTQ-Int4 on this hardware. CUDA-graph capture OOM'd during the sampler dummy run on the large vocabulary. A tuned Blackwell MoE kernel config (E=128,N=768) was absent from vLLM. These numbers are a floor for that engine, not its ceiling.

The comparison must not be read as "Ollama beats vLLM" or the reverse. It is the same finding again: the 5.44 versus 18.63 t/s result is an offload result, and the 35.8 versus 18.63 result is the same engine with the weights made to fit.

6. The second cliff: batching

Engine choice matters at the second cliff, where a batching runtime can exploit concurrency and a serialising runtime cannot.

vLLM GPTQ-Int4, no CPU offload, 256 max tokens across 32 diverse prompts:

Concurrent	Aggregate t/s	Per-request t/s	Avg latency
1	35.8	35.8	7.2 s
2	74.0	37.0	6.9 s
4	148.0	37.0	6.9 s

Concurrent	Aggregate t/s	Per-request t/s	Avg latency
8	288.3	36.1	7.1 s
16	497.8	31.1	8.2 s
32	821.9	25.7	10.0 s

Scaling is near-linear to eight-way (35.8 to 288.3 t/s aggregate at essentially unchanged per-request speed). At 32-way, aggregate is 821.9 t/s for a 40% latency increase.

The control: the same hardware under a runtime that serialises. Qwen3-Coder-Next Q8_0 (84 GB, 79.7B params, 262K context, qwen3next architecture) on Ollama segfaulted its runner at OLLAMA_NUM_PARALLEL of 8, 4, and 2, and could only run at 1. Single-stream: 119.7 t/s server-reported, 114.4 t/s wall-clock, 700–830 t/s prompt processing, TTFT 0.12 s unloaded. Under queued load (34 requests total, all successful):

Queue depth	Requests	Wall (s)	Aggregate t/s	Server t/s	Median TTFT	Median total
1 (serial)	4	17.90	114.4	119.9	0.12 s	4.48 s
2	6	26.52	115.8	119.7	4.42 s	8.81 s
4	8	35.39	115.7	119.7	13.25 s	17.65 s
8	16	70.65	116.0	119.7	30.87 s	35.29 s

Aggregate throughput is flat at approximately 115 t/s from queue depth 1 to 8 while median TTFT rises 257x (0.12 s to 30.87 s). Adding work to a serialising runtime buys nothing and costs everything in latency.

A note on the control's validity: the segfault is a software defect in a pre-release Ollama build, not a hardware property. The serialisation result is genuine as a control, but it should not be quoted as a statement about that runtime in general or in perpetuity. Ollama also loaded the entire 78.7 GiB model onto one GPU and left the second (98 GB) idle: no tensor parallelism for that architecture. Half the fleet's most expensive resource was unused during the flat-throughput result.

7. Batching in production

The dictated-notes enrichment pipeline, Qwen3-32B BF16 at tensor-parallel size 2, run sequentially and then at 18-way concurrency over identical work:

Metric	Sequential	18-way parallel	Improvement
Wall-clock	304.2 s	27.3 s	11.1x
Aggregate throughput	46.3 t/s	516.0 t/s	11.1x
Total tokens	14,085	14,085	identical

11.1x at 18-way is 62% parallel efficiency against a theoretical 18x. Per-call latency rose only about 20% under full load (16–27 s versus approximately 17 s solo). The shortfall is attributed to KV-cache contention (the memory used to store attention state for in-flight sequences) and memory-bandwidth saturation, not to compute.

Token output was byte-for-byte identical in volume between the two runs. Batching changed the schedule, not the answers. Extrapolated to the full 294-entry corpus: roughly 30–40 seconds against roughly 80 minutes sequential.

Thermally, the parallel run is cheap. Inference here is memory-bandwidth-bound rather than compute-bound. Expected GPU range is 45 °C idle to 65–75 °C under sustained parallel inference, nothing like a training profile. The visible thermal effect was on the host CPU, which

peaked at 93.2 °C (Tctl) bursting during prefill-heavy phases, within spec against a 95 °C throttle. DDR5 DIMMs 49–57 °C, NVMe 40–48 °C, 200G NIC 55 °C.

This production figure was measured on a different model (Qwen3-32B BF16) and a different workload from the synthetic sweep, and was recorded in a follow-on analysis session. The 11.1x and the 32-way 821.9 t/s are not points on one curve.

8. Where the numbers sit

A parallel survey of public sources compiled during the same period offers perceptual bands for generation speed: under 10 t/s is described as frustrating; 10–20 as sluggish; 20–40 as acceptable (the usable minimum for interactive chat); 40–70 as good; 70+ as excellent. Time-to-first-token targets from the same survey: under 200 ms is golden, under 500 ms acceptable, over 2 s frustrating. Human reading crossover is placed at roughly 6–8 t/s, with 30 t/s exceeding the fastest human readers.

None of these bands were measured by us. They are used for orientation only, and no claim in this note rests on them.

Placing our measurements against those bands: the fully-resident Q5_K_XL seat at 121 t/s sits in "excellent"; the 3%-offloaded Q6_K at 72 t/s just clears it; the 397B seat at 27.7 t/s sits in "acceptable"; the offloaded vLLM FP8 seat at 5.44 t/s sits below the "frustrating" line and beneath human reading speed.

9. What survives the caveats

Sample sizes are small. These are operational benchmarks, not controlled experiments. The MiniMax battery has six tests; the Qwen3.5 head-to-head has eight; the vLLM-versus-Ollama single-request comparison has three prompts; the queued-load control has 34 requests; the concurrency sweep has 32 prompts. No repeated trials or confidence intervals were computed. The single-request comparison is the weakest measurement in the set.

The comparisons are not clean A/B. The two Qwen3.5 configurations differ in whether the resident audio services were running. The head-to-head compares different quantisations (Q4_K_XL versus Q4_K_M) as well as different models. The vLLM-versus-Ollama single-request row compares FP8 against Q8_0 with different offload amounts. Each confound is stated in the tables and must be carried forward rather than smoothed over.

Two of the results are software-defect artefacts with a shelf life: the qwen3next parallel segfault in a pre-release Ollama build and the absent qwen35moe architecture support. The serialisation control is genuine as a control, but it does not generalise.

The 155–160 GB residency ceiling is specific to this hardware, this context length, and this set of co-resident services. It is an engineering rule of thumb for one machine, not a general constant.

What survives: the shape of the cliff. Across every model, every runtime, and every quantisation tested, the transition from full GPU residency to partial CPU offload extracts a cost that no other axis can repay. The 3% offload costs 40% of throughput. The 47% offload costs 79%. The relationship is not linear, not graceful, not recoverable by batching. The system has two states, and the space between them is not a gradient.

The corollary that matters operationally: choose the largest model that fits, never the largest model that runs. In every one of the three pairs examined (Q5_K_XL versus Q6_K, Qwen3-235B versus Qwen3.5-397B, GPTQ-Int4 versus FP8), the smaller, nominally weaker artefact wins, and wins by a wide margin.

Two cliffs, in order of magnitude: residency first (up to 6.6x on these measurements), concurrency second (up to 11.1x in production, 23x aggregate in synthetic sweep). A deployment that gets residency wrong cannot recover it by batching, because the PCIe path is already the bottleneck.

Resident non-LLM services are part of the model-size budget, not overhead to be ignored. The approximately 10 GB of speech services is what makes the practical residency ceiling 155–160 GB rather than 192 GB, and it is directly responsible for four of the Qwen3.5 layers and 7 t/s.

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-2026-005. Measurements taken 2 February – 16 February 2026; follow-on analysis session 25 February 2026. Raw artefacts are retained.