# Isolated Engines on One Weight Copy

3 customers' isolated serving engines per GPU within SLO instead of 2, and 4 under MPS, as compute becomes the limit

Pranav Varshney
Nodus Compute

Viswa Kotra
Nodus Compute

## Abstract

Customers who need process isolation get a serving engine of their own, and each such engine loads its own copy of the base model's weights, so a GPU holds only as many isolated engines as fit beside those copies. Nodus, the infrastructure execution layer for AI workloads, measured what happens when isolated engines share one copy instead. On one GPU with Qwen3-4B, stock vLLM 0.30.0 engines, each in its own container with its own scheduler and KV cache, mapped one GPU-resident copy of the weights through vLLM's zero-copy CUDA IPC loader, held by an owner process that refuses to release it and checksums it after every window. Under default time-slicing, 3 such engines met every engine's own SLO on replayed production traffic in every seed window, against 2 stock engines, since a third stock engine runs out of memory at startup. That is 1.5× the isolated engines per GPU, below the pre-registered gate of 2×, so the gate failed. Run as clients of one MPS server, a variant chosen after the data, 4 engines met SLO, 2.0×. An independent verifier reproduced both on fresh seeds. Memory stopped being the limit: the shared copy leaves room for about 7 engines, but GPU compute held the count to 3 or 4, and one engine serving the same tenants without isolation ran at a worst TPOT p95 of 22.36 ms.

## 1 Introduction

Some customers cannot share a serving engine. They need their own process, flags, adapters, scheduler and KV cache, and today each such engine loads the base model's weights again. On the GPU in this study, two Qwen3-4B engines with their own 7.56 GiB weight copies fill the GPU, so it holds two isolated customers however lightly they use it.

Sharing weights across engines is supported upstream. vLLM [1] ships a zero-copy CUDA IPC weight loader documented for fast restarts [2], NVIDIA Dynamo's GPU Memory Service lets a second engine map an active engine's weights [3], and sharing tensors across colocated instances to raise density is older still [4] [5]. What is missing is a measurement of active-active density: how many isolated engines, all serving at once from one weight copy, meet each engine's own SLO on production traffic, with checks that the shared copy stays intact and that one engine's crash leaves the others serving.

The metric is isolated engines per GPU, the largest number of engines on one GPU at which every engine meets its own SLO in every seed window.

## 2 Experimental setup

System. One GPU, stock vLLM 0.30.0 and Qwen3-4B [6] in BF16. Every engine runs in its own container with a maximum length of 4096, at most 32 sequences, a 2048-token step budget and an explicit 1 GiB KV cache, so the arms differ only in how engines get their weights.

Shared weights. An owner process loads the weights once and serves them through vLLM's weight cache daemon. Each engine starts with vLLM's zero-copy IPC load format and maps the owner's copy instead of loading its own. The owner refuses the release request that a copying engine would send, and checksums the weights after every window.

Traffic. Each engine serves one tenant, which replays its own segment of the public 2023 LLM inference conversation trace released with Splitwise [7] at 0.15 requests per second, with prompts up to 2048 tokens and answers up to 512, for 180 s per window on seeds 1 to 3. An engine meets its SLO in a window if its p95 TTFT is at most 2000 ms, its p95 TPOT at most 100 ms and at least 99% of its requests complete.

Gate and guards. The pre-registered gate requires 2× the stock engines per GPU, every engine passing on every seed at that count, a goodput gain above the A/A spread, and three guards: the weight checksum unchanged after every window, greedy outputs identical to a stock engine's, and a killed engine leaving the others serving.

## 3 Results

Table 1: Engines on one GPU, and whether every engine met its own SLO in each seed window. Worst TPOT p95 is the highest across engines and seeds.

| Arm | Engines | Seed 1 | Seed 2 | Seed 3 | Worst TPOT p95 ms |
|---|---|---|---|---|---|
| Stock, own weights | 2 | pass | pass | pass | 46.7 |
| Stock, own weights | 3 | out of memory | n/a | n/a | n/a |
| Nodus shared weights | 3 | pass | pass | pass | 91.71 |
| Nodus shared weights | 4 | pass | pass | fail | 123.27 |
| Nodus shared weights | 5 | fail | fail | fail | 135.08 |
| Shared weights with MPS | 4 | pass | pass | pass | 79.54 |

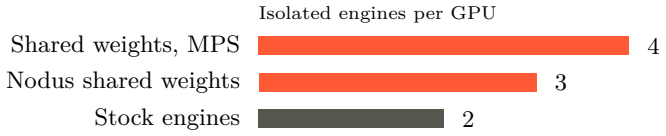



Figure 1: Isolated engines on one GPU with every engine within its own SLO on every seed.

3 isolated engines instead of 2. With shared weights, 3 engines met every engine's own SLO in all 3 seed windows of two separate runs, against 2 stock engines, whose third engine ran out of memory at startup. That is 1.5× the isolated engines per GPU. At 4 engines one engine missed its TPOT target on seed 3, at 123.27 ms, and at 5 every seed failed. SLO-met requests per GPU-second rose from 0.2926 to 0.4648, 1.589×, but with a fixed request rate per tenant this restates the number of engines rather than adding efficiency. The pre-registered gate of 2× failed.

MPS adds a fourth engine. Run as clients of one MPS server, a variant chosen after the time-sliced 4-engine run failed, 4 engines met SLO on every seed in two runs, 2.0×. Under contention MPS cut the worst TPOT p95 from 123.3 to 79.5 ms and the worst TTFT p95 from 950 to 351 ms. The greedy-identity and crash checks were not repeated under MPS, and MPS puts every tenant in one GPU fault domain.

Memory is no longer the limit, compute is. The owner took 8775 MiB and each attached engine 1614 to 1682 MiB including its 1 GiB KV cache, against 9249 MiB for a stock engine.

The verifier counted about 220 MiB of growth per engine while serving, so about 7 engines would fit. Only 3 or 4 met SLO, and with 5 engines the GPU was 95% to 99% busy at its power limit. One stock engine serving the same 3 tenants without isolation ran at a worst TPOT p95 of 22.36 ms, so isolation, not memory, now costs most of the GPU's batching efficiency.

Guards. The weight checksum was identical at the owner's start and after every checksummed window, including after a crash. All 80 greedy sequences from 5 shared engines matched a stock engine token for token, a 95% lower bound of 0.955. When one of 3 engines was killed at 59.7 s, the other two completed 26 of 26 and 30 of 30 requests within their SLO. Starting an engine against resident weights took 72.5 to 73.1 s against 74.1 s for a normal load, because process and server start dominate on this host.

Isolation boundaries. Each engine is its own process with its own scheduler and KV cache, but the containers shared the host's IPC, process and network namespaces, the CUDA IPC mapping is writable, and the integrity of the shared copy rests on the checksum rather than on a read-only mapping. An owner crash, which would stop every attached engine, was not tested.

Noise. Two stock runs on the same seeds gave identical goodput and worst TPOT p95 within 0.14 ms, and two shared runs within 0.29 ms. Margins at the limit are thin: the 3-engine shared run peaked at 91.71 ms against the 100 ms target on seed 3.

## 4 Independent verification

A separate verifier recomputed all 26 researcher windows from the per-request logs, matched every verdict, and reproduced the memory readings to the MiB. On the same GPU, with fresh seeds 11 and 12 chosen blindly and a stress seed 15 chosen by a rule fixed before any shared run, shared weights again held 3 engines against 2 stock engines, 1.74× and 1.66× in SLO-met requests per GPU-second, and 4 engines under MPS, with worst TPOT p95 of 53.86 and 73.84 ms. Time-sliced, 4 engines passed seed 15 at 98.94 ms. The verifier classed the result as a narrow novel variant: the mechanism is upstream [2] [3] and weight sharing for density is published [4] [5], and what is new is the SLO-judged active-active density of unmodified isolated engines on one copy, its safety checks and the MPS combination.

## 5 Conclusion

On one GPU, isolated stock vLLM engines sharing one copy of Qwen3-4B's weights held 3 customers per GPU within each customer's SLO instead of 2, 1.5× and below the preregistered 2× gate, and 4 under MPS, 2.0×, with the shared weights unchanged and, without MPS, a killed engine leaving the others serving. Shared weights move the limit from memory to compute, and isolation itself now costs most of the GPU. The result covers one GPU, one model, one light tenant rate and process-level isolation. For customers who require an engine of their own, a GPU can hold more of them than its memory suggests.

## References


[1] W. Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP 2023.

[2] vLLM project. Weight preloading with the ipc_cache load format. vLLM documentation, 2026.

[3] NVIDIA. Dynamo GPU Memory Service. NVIDIA Dynamo documentation, 2026.

[4] J. Li et al. Tetris: Memory-efficient Serverless Inference through Tensor Sharing. USENIX ATC 2022.

[5] Zhu et al. Tangram: Accelerating Serverless LLM Loading through GPU Memory Reuse and Affinity. arXiv preprint, 2025.

[6] A. Yang et al. Qwen3 Technical Report. arXiv preprint, 2025.

[7] P. Patel et al. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. ISCA 2024.