# Cold Start in Seconds

A stock vLLM engine goes from a cold host to its first token in 7.158 s, 7.66× faster than the best stock boot

Pranav Varshney
Nodus Compute

Viswa Kotra
Nodus Compute

## Abstract

GPUs that serve inference are often kept reserved and idle because bringing a model back takes minutes. Nodus, the infrastructure execution layer for AI workloads, restores a fully warmed stock vLLM engine from a snapshot instead of booting it. The snapshot holds the engine process and its GPU state after compilation and graph capture, but not the model weights, so it is 4.04 GB for Qwen3-8B, and the weights are reloaded from the model's own files while the process is restored. On one GPU with the page cache dropped and verified empty before every trial, the restored engine serves its first token in a median of 7.158 s across 5 trials, against 54.857 s for stock vLLM with a persisted compile cache and 122.968 s for stock defaults, a 7.66× and a 17.2× gain. A streaming reloader brings the restore to 6.141 s. An independent verifier measured 7.194 s against 55.011 s on fresh trials, a ratio of 7.65. Throughput after restore is 4172 tokens per second against 4174 for a normal start, and every restored engine produced exactly the same greedy outputs as a normally started engine of the same configuration.

## 1 Introduction

An inference endpoint that sees no traffic for a while could give its GPU back, but only if it can return quickly when the next request arrives. A stock vLLM [1] engine with Qwen3-8B takes about a minute to boot on a cold host even with its compile caches kept, and about two minutes without them. Because of that, fleets keep GPUs reserved through idle periods, and that reserved idle time is one of the largest sources of wasted GPU-hours.

Process and GPU checkpointing make it possible to skip the boot. CRIU [2] saves and restores a process, and cuda-checkpoint [3] moves a process's GPU state to host memory so that CRIU can save it. Snapshots of inference engines built on these tools are already used in production systems [4] [5] and in serverless research [6]. Our contribution is a measured cold-host restore of an unmodified vLLM engine. It uses vLLM's own sleep mode to leave the weights out of the snapshot and its own weight reload to bring them back from the model's files, so the snapshot stays small and no separate weight artifact is needed.

The metric is the cold start to first token: seconds from the launch or restore command to the first streamed token, with the page cache dropped before every trial and verified to hold none of the snapshot or model files.

## 2 Experimental setup

System. One host, stock vLLM 0.30.0, Qwen3-8B [7] in BF16 on one GPU. Every arm uses the same vLLM installation and model files. Before every trial the page cache is dropped and checked to hold 0 bytes of the snapshot and model files.

Arms. Stock default starts vLLM with empty caches. Stock best starts vLLM with its compile caches persisted from an earlier boot, the fastest stock start we found. Nodus restores a snapshot taken after the engine was fully warmed and put into deep sleep, which releases the weights, and reloads the weights in parallel with the process restore. A second Nodus variant uses a streaming reloader for the weights. Two ablations put the weights inside the snapshot and remove the parallel reload. A warm-host reference wakes an engine that never left memory.

Gate and guards. The pre-registered gate requires at least 2.0× the speed of stock best, a gain beyond the spread of repeated trials, throughput within 3%, and outputs matching a reference engine. Each arm runs 5 cold trials, and the Nodus arm is repeated in a second block of 5.

## 3 Results

Table 1: Cold start to first token on one GPU with Qwen3-8B, median of 5 cold trials.

| Arm | Median seconds | Stock best divided by arm |
|---|---|---|
| Nodus, streaming reloader | 6.141 | 8.93 |
| Nodus, snapshot restore | 7.158 | 7.66 |
| Ablation, weights inside the snapshot | 17.552 | n/a |
| Ablation, no parallel reload | 19.557 | n/a |
| Stock best, compile caches kept | 54.857 | 1.0 |
| Stock default, empty caches | 122.968 | n/a |

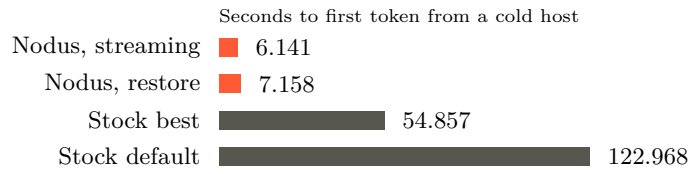



Figure 1: Cold start to first token, median seconds, lower is better.

From minutes to seconds. The restored engine serves its first token in 7.158 s, against 54.857 s for the best stock start and 122.968 s for stock defaults. The repeat block gave a median of 7.141 s. With the streaming reloader the median is 6.141 s, and 6.098 s in its repeat block. The pinned container image with its compile cache kept took 83.323 s on the same host.

Keeping the weights out of the snapshot is what makes it fast. A snapshot that includes the weights is 26.7 GB and restores in 17.552 s, against 4.04 GB for the snapshot without them. Restoring without reloading the weights in parallel takes 19.557 s. The gain comes from a small snapshot plus a weight reload that overlaps with the process restore.

The baseline is not slowed by the cold disk alone. Even with the page cache fully warm, a stock start took 44.112 s and 44.392 s in the verifier's runs, about 6 times the cold restore, so most of a stock boot is engine start-up work that the snapshot skips. A warm-host wake of an engine that never left memory takes 1.727 s, but it requires keeping the process and its memory resident.

Nothing is lost. Throughput after restore is 4172 tokens per second against 4174 for a normal start. Every restored engine produced the same 100 greedy outputs, identical to normally started engines of the same configuration. The pre-registered output check compared against

a reference engine whose compiled kernels differed, and two normal starts can differ in the same way, so it matched 0.86 for the restored engines and failed as worded.

## 4 Independent verification

A separate verifier reran the restore and the best stock start on fresh trials with the same code. The restore took a median of 7.194 s and the stock start 55.011 s, a ratio of 7.65, and the streaming reloader took between 6.034 s and 6.094 s. The verifier confirmed the cold-cache checks, found the baseline not handicapped, and classed the result as an incremental variant of published snapshot systems [5] [4]. The restore was measured on one host, one GPU type and one model, and was not tested on a different host or GPU.

## 5 Conclusion

On one GPU, Nodus restores a fully warmed stock vLLM engine from a 4.04 GB snapshot and serves its first token in 7.158 s from a cold host, 7.66× faster than the best stock boot and 17.2× faster than stock defaults, with throughput and outputs unchanged. Leaving the weights out of the snapshot and reloading them in parallel is what makes it fast. The result covers one model and one GPU type on one host. For fleets that keep GPUs reserved through idle periods, an endpoint that can return in seconds can give its GPU back during gaps that are far shorter than a stock boot.

## References


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

[2] CRIU project. Checkpoint/Restore In Userspace. criu.org.

[3] NVIDIA. cuda-checkpoint: checkpoint and restore of CUDA process state. github.com/NVIDIA/cuda-checkpoint, 2024.

[4] A serverless GPU platform's engineering blog. GPU memory snapshots for faster cold starts. 2025.

[5] NVIDIA. Dynamo Snapshot and GPU Memory Service. NVIDIA Dynamo documentation, 2026.

[6] Y. Fu et al. ServerlessLLM: Low-Latency Serverless Inference for Large Language Models. OSDI 2024.

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