Moving a Serving Engine Between GPUs

Live migration left 13.62 s of blackout, 3.10× shorter than a restart, and a snapshot move had none and freed the source GPU in 5.34 s

Pranav Varshney
Nodus Compute

Viswa Kotra
Nodus Compute

Abstract

Consolidating lightly loaded serving engines onto fewer GPUs means moving them, and each way to move one today costs either service or GPU time. A stock restart of Qwen3-8B leaves its users without service for about 42 s, and make-before-break avoids the blackout but holds the source GPU while a replacement boots. In this study, a research prototype from Nodus, the infrastructure execution layer for AI workloads, moved a running stock vLLM 0.30.0 engine in two ways onto another host's GPU that already serves a second engine, neither of them yet in the product. Live migration puts the engine to sleep, which releases its weights, saves its GPU state and process with cuda-checkpoint and CRIU, copies the 3.44 GB image over 8 parallel streams, restores it onto the destination GPU and reloads the weights from the model's own files, while the client holds new requests and resumes the ones in flight. On two GPUs on two hosts with Qwen3-8B, it gave a median blackout of 13.62 s over 6 migrations, against 42.21 s for a stock restart with the compile cache kept and the model in the page cache, 3.10× shorter, with 0 dropped requests. The source GPU was empty 1.33 s after the freeze, and the engine already on the destination kept an SLO attainment of 1.000. An independent verifier measured 13.09 s against 42.21 s on 3 fresh seeds on one host, a ratio of 3.22, and 2.64× against a faster stock restart in eager mode. A zero-blackout move instead makes the replacement in make-before-break a snapshot. A weight-free CRIU and cuda-checkpoint snapshot of the engine's configuration is pre-staged on the destination host and restored onto its GPU beside the engine already serving there, while the original keeps serving. Traffic switches atomically once the replacement answers a probe, and requests in flight finish on the source. On two other hosts with Qwen3-4B, stopping an engine and booting it elsewhere left its users without service for 30.345 s, and live migration for 8.399 s. With the zero-blackout move, the source GPU was empty a median 5.34 s after the move decision over 6 moves on 3 seeds, against 32.57 s for stock make-before-break, 6.10×, with no dropped request, no service gap above 0.201 s and SLO attainment of 1.000 for both engines. Fresh seeds gave 5.81×, and a standby parked with no GPU memory gave 3.74 s, 8.45×. Outputs after a restore were bit-identical with one request at a time, but the pre-registered exact-match guard under live batching failed, so the pre-registered gate failed. An independent verifier measured 5.27× on fresh seeds on one GPU, and found a sleeping stock engine kept on the destination GPU ready 1.52× sooner than the restore, at the cost of 487 MiB of that GPU's memory.

1 Introduction

Many engines in a serving fleet run at light load, each holding its own GPU. Putting two of them on one GPU frees the other, which is how a fleet consolidates work onto fewer GPUs, and it means moving a running engine. A stock vLLM [1] engine with Qwen3-8B [2] took 40.321 to 41.25 s to boot in this study even with its compile cache kept and its model in memory, so a restart on another GPU leaves the engine's users without service for that long, and operators avoid moving engines at all. The standard way to avoid the outage is make-before-break: start a replacement first, switch traffic, then stop the original. With a stock boot the replacement needs half a minute or more to be ready, and the source GPU stays occupied for that whole time.

This report measures two ways to move a running stock vLLM engine onto another host's GPU that already serves a second engine, each in its own section. Live migration puts the engine to sleep, checkpoints its process and GPU state, copies them to the destination and restores them there, so the source GPU is released within seconds while the engine's users wait through a blackout of seconds. For an interactive endpoint, seconds without service are still an outage. A zero-blackout move is make-before-break with a pre-staged snapshot of the engine as the replacement, restored on the destination while the original keeps serving, so there is no blackout and the source GPU stays occupied until traffic has switched and drained. A final section compares what each costs. This report absorbs report 027, Live Migration of a Serving Engine [3], whose measurements are the live-migration section.

Process and GPU checkpointing make both moves possible. CRIU [4] saves and restores a Linux process, cuda-checkpoint [5] moves a process's GPU state to host memory so that CRIU can save it, and GPU checkpointing systems [6] [7] and production snapshot features for cold start and scale-out [8] [9] build on them. Serving systems also move requests or KV caches between engines [10] [11] [12], and sleeping engines kept resident on a GPU wake in seconds but stay bound to that GPU [13]. Our contributions are a measured live move of a whole, unmodified vLLM engine, mid-traffic, onto another host's GPU that is already serving, compared with a restart, a drain and a make-before-break switch on the same hosts, and a measured make-before-break move in which the replacement is a pre-staged snapshot restored onto another host's GPU that already serves an engine, compared with stock make-before-break, a restart and live migration on the same hosts.

Both ways to move are measured by the same two costs. The blackout is how long the moved engine's users are without service, and the source-GPU hold is how long the source GPU stays occupied after the move begins. The two studies define them slightly differently, as each section states.

2 Live migration

2.1 Setup

System. Two hosts with one GPU each and the same driver, stock vLLM 0.30.0 and Qwen3-8B in BF16 with a maximum length of 4096. Engine A is moved from host 1 to host 2, whose GPU already serves engine B. Each engine receives 1 request per second of Poisson traffic with 128 output tokens, for 15 s before the freeze and 60 s after it. A request meets the SLO if its first token arrives within 2 s and it never stalls for more than 2 s.

Nodus live migration. The engine enters vLLM's deepest sleep level, which releases the weights and drops the KV cache. cuda-checkpoint then moves its GPU state to host memory, CRIU dumps the process to memory, the image is copied to the destination over 8 parallel streams on the hosts' private network, CRIU restores it, cuda-checkpoint restores the GPU state onto the destination GPU, and the engine wakes and reloads its weights from the model files. The client holds requests during the blackout and resumes the requests that were in flight.

Baselines. Stock restart stops engine A and starts stock vLLM on the destination with the same flags, a persisted compile cache and the model in the page cache, with the same client hold and resume. Drain then restart first lets in-flight requests finish. Stock make-before-

break boots the replacement on the destination before stopping the original. Arms alternate in ABBA order on seeds 0 to 2.

Metric and gate. The metric is the blackout: seconds from the moment the client stops sending to the engine to the first token the engine serves on its new GPU. The pre-registered gate requires a median blackout at most half the stock restart's and above its repeat spread, no dropped requests, window SLO attainment at least the restart's, engine B's SLO attainment at least 0.9, and a passing output-fidelity check.

2.2 Results

Table 1: Live migration with Qwen3-8B: moving engine A from host 1 to host 2's busy GPU, 6 migrations and 6 restarts on seeds 0 to 2. Blackout is the median, and make-before-break keeps the original engine serving until the switch.

Arm	Blackout s	Range s	Dropped	Source GPU free s
Nodus live migration	13.62	13.06 to 14.88	0	1.33
Stock restart	42.21	41.45 to 42.44	0	0.53
Drain then restart	43.44	42.97 to 43.60	0	2.13
Stock make-before-break	none	n/a	0	53.77
Ablation, one network stream	23.12	23.07 to 23.25	0	1.33
Ablation, no resume	13.36	13.00 to 13.44	5	1.33

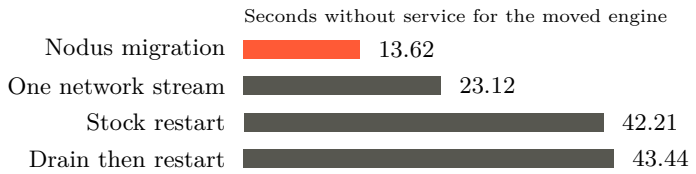


Figure 1: Live migration with Qwen3-8B: median blackout when moving a running engine to another host's GPU, lower is better.

A 13.62 s blackout instead of 42.21 s. The six migrations took 13.34, 14.19, 13.06, 14.88, 13.90 and 13.17 s, a median of 13.62 s, against 42.21 s for a stock restart and 43.44 s for drain then restart, 3.10× and 3.19× shorter. Repeated restarts on the same seed differed by 0.2% to 0.5%. Repeated migrations differed by 5.3% to 13.0%, almost all from the copy, which took between 2.266 and 4.105 s.

The source GPU is freed and nothing is dropped. No request was dropped in any migration. Without resuming in-flight requests, 5 requests were dropped over 3 trials. The source GPU was empty 1.33 s after the freeze. Make-before-break avoids the blackout but holds the source GPU for a median of 53.77 s and needs room on the destination for a booting engine. Over the 60 s after the freeze, engine A met its SLO for 0.788 of requests under migration against 0.306 under a restart, and the two engines delivered 1.720 SLO-met requests per GPU-second against 1.239, 1.39×. Engine B, already on the destination GPU, kept an SLO attainment of 1.000 in every arm.

Where the time goes. Median phase times were 0.04 s to sleep, 0.80 s for cuda-checkpoint, 2.59 s for the CRIU dump of a 3.44 GB image, 2.69 s for the copy, 1.67 s for the CRIU restore, 0.71 s to restore the GPU state onto the new GPU and 4.03 s to wake and reload the weights. The image stays small because the weights are left out and reloaded from the model files. Copying it over a single ssh stream on the public network took 12.24 s and raised the blackout to 23.12 s.

The cost after the move. The two engines share one GPU by time-slicing. After the move, engine B's mean TPOT rose from 13.1 ms to 39.1 ms, against 23.7 ms after a restart, while engine A's was 21.0 ms against 33.3 ms. The SLO in this study has no TPOT bound, and the load is light. The KV cache is not moved, so requests in flight recompute their prompts on the new GPU. Outputs after the move matched a reference as often as those of a freshly booted engine sharing a GPU, 0.506 against 0.497, and no output was degenerate.

What did not count. A faster variant with a streaming weight reload reached a median of 9.98 s, but 95% of its outputs after the move were degenerate, so it was ruled invalid and is not used. The result is 3.10×, below what the pre-registration hypothesized, and it passes the pre-registered gate.

2.3 Independent verification

A separate verifier recomputed every researcher trial from the raw request logs and matched all of them, and confirmed from the raw data that each migration crossed hosts onto a different GPU with every byte of the image received. Its own slot covered host 1 only, so on fresh seeds 11, 12 and 13 it moved the engine onto the same host's busy GPU with a local copy. The median blackout was 13.09 s against 42.21 s for a stock restart, a ratio of 3.22, with restart repeats within 0.2%, 0 dropped requests and engine B at 1.000. An exploratory stock restart in eager mode booted faster, with blackouts of 34.70 and 34.55 s, and against it the migration is 2.64× shorter. The verifier classed the result as a novel variant: cross-host restore of GPU snapshots is published [8] [7], and no source it found measures a whole stock vLLM engine moved live onto another serving GPU against restart, drain and make-before-break.

3 Zero-blackout moves

3.1 Setup

The primary metric is the source-GPU hold, the seconds from the move decision until the source GPU holds less than 1 GiB, measured together with the blackout, the longest time an engine's users wait with a request outstanding and no token served.

System. Two hosts with one GPU each and the same driver, on a private network with an RTT of 0.2-0.4 ms. Every engine is stock vLLM 0.30.0 with Qwen3-4B [2] in BF16, a memory fraction of 0.45, a maximum length of 4096 and sleep mode enabled, which gives each engine a KV cache of 14,480 tokens and lets two engines share one GPU. Engine A is moved from host 1 to host 2, whose GPU already serves engine B. Each engine receives 1.0 request per second of seeded Poisson traffic with 128 output tokens.

Golden snapshot. Once per engine configuration, a stock engine is booted, serves one request, enters vLLM's deepest sleep level, which releases its weights, and is saved with cuda-checkpoint and CRIU into a 3.07 GB image that is copied to the destination's disk ahead of time. A second variant saves the engine at the lighter sleep level that keeps its weights in host memory, restores it on the destination ahead of time and parks it with its GPU state checkpointed, so it holds 0 MiB of GPU memory and 13.9 GB of host memory until it is needed.

Move. At the decision, the golden is restored onto host 2's GPU with a device map, woken and given its weights from host 2's model files. One probe request confirms that it serves, new requests switch to it atomically, requests in flight finish on host 1, and the source engine is stopped.

Baselines. Stock make-before-break boots a stock replacement on host 2 with the same flags, a persisted compile cache and a warm page cache, then switches, drains and stops the source the same way. Stop and cold start stops A and boots a stock engine on host 2. Freeze and move is the live migration of the previous section, from report 027, on this section's engines. Arms alternate in ABBA order on seeds 0 to 2, with same-seed repeats of baseline and candidate, and seeds 3 to 5 replicate the comparison and add the standby variant.

SLO and gate. A request meets its SLO if its first token arrives within 2 s, its mean TPOT is at most 100 ms and it never stalls for more than 2 s. The pre-registered gate requires a hold at least 2× shorter than stock make-before-break and above the repeat spread, a service gap below 2 s with no TTFT miss in every move, no dropped requests, SLO attainment for both engines within 0.02 of the baseline's, and a paired exact-match check under live traffic within 0.10 of an A/A pair.

3.2 Results

Table 2: Zero-blackout moves with Qwen3-4B: moving engine A onto host 2's busy GPU, medians over seeds 0 to 2. Hold is how long the source GPU stays occupied after the decision. TTFT misses and drops are totals over each arm's moves.

Arm	Hold s	Blackout s	TTFT misses	Dropped	Ready s
Nodus snapshot make-before-break	5.34	0.146	0	0	3.96
Stock make-before-break	32.57	0.100	0	0	30.13
Freeze and move	0.33	8.399	16	0	8.26
Stop and cold start	0.33	30.345	101	0	30.24
Ablation, break before make	0.34	4.495	6	0	4.35
Ablation, no drain	4.14	0.134	0	0	3.87
Ablation, no pre-staging	8.14	0.138	0	0	6.21

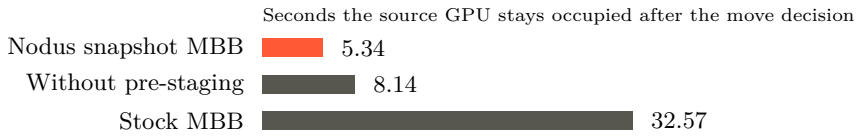


Figure 2: Zero-blackout moves with Qwen3-4B: median source-GPU hold in a move with no blackout, seeds 0 to 2, lower is better.

The source GPU is free in 5.34 s, with no blackout. Over 6 moves on seeds 0 to 2, the source GPU was empty a median 5.34 s after the decision, from 4.53 to 5.94 s, against 32.57 s for stock make-before-break, 6.10×. The replacement was ready in 3.96 s against 30.13 s, 7.6× sooner. No request was dropped or missed its TTFT target, the longest gap in A's service was 0.201 s, and both engines kept an SLO attainment of 1.000. Same-seed repeats differed by 0.1% in both arms. On fresh seeds 3 to 5 the hold was 5.44 s against 31.57 s, 5.81×.

Every other move costs service or GPU time. Freeze and move released the source GPU at once but left A without service for 8.399 s with 16 TTFT misses, and stop and cold start for 30.345 s with 101. Make-before-break with a stock boot has no blackout but holds the source GPU for 32.57 s. Only the snapshot replacement gives both no blackout and a source GPU released within seconds.

What each part does. Stopping A before the same restore cut the hold to 0.34 s but brought back a 4.495 s blackout, so make-before-break is what removes the blackout. Copying the golden at decision time instead of ahead of it raised the hold to 8.14 s, still 4.0× better than stock. Cutting in-flight streams at the switch and resuming them on host 2 from their generated prefix, instead of draining them, lowered the hold to 4.14 s with no blackout. The drain is the largest variable part of the hold, from 0 to 2.4 s at this load, so individual moves ranged from 2.9 to 6.7 s. The restore itself took 0.80 s for CRIU, 0.61 s for the GPU state and 2.23 s to wake and reload the weights.

A standby with no GPU memory. The parked standby only has to bind its GPU state to the destination GPU and wake, so it was ready in 2.56 s against 4.03 s for the restore on the same seeds, and released the source in a median 3.74 s, 8.45× sooner than stock make-before-break. It holds no GPU memory while parked and costs 13.9 GB of host memory. Its pre-registered margin over the plain restore in hold time, 1.455 against a bar of 1.555, failed, because the drain varies more than the gain.

The cost after the move. Once A shares host 2's GPU with B, the two engines time-slice it. B's mean TPOT rose from 19.4-19.9 ms to about 36 ms, the same in every arm once A has moved, and both engines stay within the 100 ms target. The restore did not disturb B, whose longest gap was 0.08-0.12 s. With A at 3 requests per second, a check that was not pre-registered, the hold was 6.64 s against 32.67 s, 4.92×, with no blackout and no drops.

Fidelity and the failed gate. With one request at a time, a restored engine and a parked standby produced per-token logprobs bit-identical to a stock engine's on all 16 probe

prompts. Under live batching, the pre-registered exact-match guard compared each move's outputs with a reference run and failed: 0.600 against 0.797 for the A/A pair, a difference of -0.197 with a 95% CI of -0.259 to -0.130, against a bound of -0.10. A post-hoc split puts the shortfall on requests served on host 2 after the move, 0.539 against 0.738, where the replacement takes traffic at 3.96 s instead of 30.13 s and its requests meet different batch neighbours. Greedy decoding in a batched engine depends on batch composition, so the guard measured batch history, but it was pre-registered and the gate is reported as failed, 4 of 5. No output was degenerate.

3.3 Independent verification

A separate verifier recomputed every researcher trial from the raw files, matched every headline number, and confirmed that every arm saw the same arrivals and prompts. With one GPU on host 1, it re-ran the comparison on fresh seeds 21 to 24 with the replacement started beside the source on the same busy GPU and no engine B. The source engine's memory was released a median 6.28 s after the decision against 33.07 s, 5.27×, with per-seed ratios of 4.04 to 9.99, the replacement ready 7.69× sooner, no service gap above 0.074 s and no drops. Sharing the GPU slowed the source's drain in both arms. The exact-match guard failed again, -0.132 with a CI of -0.184 to -0.078. In an extra test that was not pre-registered, a stock engine sleeping on the destination GPU was ready in 2.49 and 2.60 s, 1.52× sooner than the restore, while parking 487 MiB of that GPU's memory. The verifier classed the result as a novel variant: snapshot restore, resident standbys and make-before-break with a drain each exist [8] [9] [13] [10], and no source it found combines a pre-staged restore onto another host's serving GPU with a make-before-break switch and measures source-GPU hold.

4 Blackout or source-GPU time

The two moves trade one cost for the other. Live migration releases the source GPU almost at once and costs the moved engine's users a blackout, while a zero-blackout move costs them none and holds the source GPU until traffic has switched and drained. A restart pays the blackout of a full boot, 42.21 s with Qwen3-8B and 30.345 s with Qwen3-4B, and stock make-before-break pays the source-GPU time of one, 53.77 s and 32.57 s. The two studies used different GPU types and models, a previous-generation training GPU with Qwen3-8B for live migration and an older inference GPU with Qwen3-4B for the zero-blackout move, so their absolute times differ, but the zero-blackout study also ran live migration on its own hosts.

Table 3: What a move of engine A costs in each study, medians. With Qwen3-8B, the blackout runs from the moment the client stops sending to the first token on the new GPU, so make-before-break has none, and the source-GPU time ends when that GPU is fully empty. With Qwen3-4B, the blackout is the longest wait with a request outstanding and no token served, and the source-GPU time is the hold, which ends when that GPU holds less than 1 GiB.

Move	Model	Blackout s	Source GPU s
Nodus live migration	Qwen3-8B	13.62	1.33
Stock restart	Qwen3-8B	42.21	0.53
Stock make-before-break	Qwen3-8B	none	53.77
Nodus live migration	Qwen3-4B	8.399	0.33
Nodus snapshot make-before-break	Qwen3-4B	0.146	5.34
Stock make-before-break	Qwen3-4B	0.100	32.57
Stop and cold start	Qwen3-4B	30.345	0.33

On the same hosts. With Qwen3-4B, live migration released the source GPU in 0.33 s and left A without service for 8.399 s with 16 TTFT misses, while the snapshot move held the source GPU for 5.34 s, 6.10× less than stock make-before-break, with no TTFT miss and no gap above 0.201 s. The hold is GPU time the fleet pays on every move, 5.34 GPU-s on the source here, and the blackout is service the endpoint's users lose.

What each needs. Live migration needs no snapshot prepared before the move and takes the running process with it, but the KV cache is not moved, so requests in flight recompute their prompts on the new GPU. A zero-blackout move needs a snapshot of the engine's configuration staged on the destination host ahead of time, a 3.07 GB image, and without that pre-staging its hold rose to 8.14 s. An interactive endpoint cannot hide a blackout of seconds, while a fleet that moves engines often pays the hold on every move.

5 Conclusion

On two GPUs on two hosts, a Nodus research prototype moved a running stock vLLM engine onto another host's busy GPU in two ways. Live migration, with Qwen3-8B, gave a median blackout of 13.62 s, 3.10× shorter than a stock restart, with no dropped requests, the source GPU empty in 1.33 s and the destination's own engine unaffected, and it passed its pre-registered gate. A zero-blackout move, with Qwen3-4B, had no blackout and no dropped request, and released the source GPU a median 5.34 s after the decision, 6.10× sooner than stock make-before-break, or 3.74 s with a standby that holds no GPU memory. Its pre-registered gate failed on the exact-match guard under live batching, while outputs with one request at a time were bit-identical. A sleeping engine kept on the destination GPU is readier still when the destination is known ahead of time and can spare the memory. Each result covers one GPU type, one model, light load and engines that share a GPU by time-slicing, the zero-blackout move used a pre-staged snapshot, and the cross-host live migration was measured by the researcher while the verifier reproduced it on one host. For a fleet that consolidates engines, the results suggest that engines could be moved while they serve, and that a move need not choose between an outage and half a minute of a second GPU, although neither live migration nor snapshot make-before-break moves are yet in the product.

References

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

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

[3] Nodus Research. Live Migration of a Serving Engine. Report 027, 2026.

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

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

[6] R. Stoyanov et al. CRIUgpu: Transparent Checkpointing of GPU-Accelerated Workloads. arXiv preprint, 2025.

[7] X. Wei et al. PhoenixOS: Concurrent OS-level GPU Checkpoint and Restore with Validated Speculation. SOSP 2025.

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

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

[10] B. Sun et al. Llumnix: Dynamic Scheduling for Large Language Model Serving. OSDI 2024.

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

[12] X. Miao et al. SpotServe: Serving Generative Large Language Models on Preemptible Instances. ASPLOS 2024.

[13] vLLM project. Sleep mode. vLLM documentation, 2025.