TECHNICAL REPORT · NOVEMBER 2026

gpt-oss-120b on one H100: vLLM vs SGLang vs TensorRT-LLM

Isaac Levinsky · MCMLV1 LLC · www.linkedin.com/in/isaaclevinsky/ · mcmlv1.com · github.com/IsaacLevinsky

One H100 SXM5 80GB, one model (gpt-oss-120b), three serving engines at their defaults and with FP8 KV cache and deterministic modes. Accuracy with the MLPerf v6.0 gpt-oss-120b method, speed from 1 to 64 concurrent users, a repeat test of identical requests under load, Nsight Systems profiles, and a Kubernetes deployment test. Total cloud spend $92.79 under a hard $100 cap.

Engine	Version
vLLM	0.33.0
SGLang	0.5.23
TensorRT-LLM	1.4.0rc2

Findings at a glance

1. **No engine and no FP8 KV setting changed accuracy beyond noise.** Six configurations, 75.4–79.8%; one vLLM pass alone lands in 75.0–80.7% (95%).
2. **FP8 KV cache cut cost per token by up to 37%.** It doubles the KV pool, so twice the users pass the KV-fit rule: vLLM 32 → 64 users, $0.72 → $0.45 per million output tokens; SGLang 16 → 32, $1.04 → $0.68; TensorRT-LLM −8% (it already fit 64 users).
3. **At defaults, SGLang returned 3 distinct outputs for 1,000 identical requests under load from 24 concurrent clients; vLLM returned 82** (110 with its batch-invariant mode). 512-token outputs, not final answers.
4. **One flag made vLLM's batch-invariant mode exact on gpt-oss-120b/H100:** `--attention-backend TRITON_ATTN` — 1 distinct output, 200/200 equal to the solo run. Prefix caching, CUDA graphs and FlashInfer autotuning were not the cause.
5. **Determinism costs speed:** per decode step at 16 users, vLLM +44%, SGLang +20%; per token, +30% and +16%.
6. **A warmed-up, "ready" vLLM server compiled kernels under load:** 8 idle gaps of ~1 s in a 12 s window, each ending in a GPU module load.
7. **About half of the wrong answers never finished:** 10–14% of answers hit the 32,768-token limit; they are 40–55% of the wrong answers, the same hard questions on every engine.
8. **SGLang responded fastest under load:** time to first token at 16 users 0.49 s vs 1.14 s (TensorRT-LLM) and 1.54 s (vLLM).

1. Method

Machine. One cloud VM per session with 1× NVIDIA H100 SXM5 80GB; every session got the same physical GPU (same UUID), and engines ran one at a time from container images pinned by digest. The vLLM default speed sweep is from an earlier session than the FP8 sweeps. The model weights were downloaded once per session and verified against published SHA-256 sums before use.

Accuracy. The MLPerf Inference v6.0 gpt-oss-120b accuracy method (AIME 2025 and GPQA Diamond, scored by the official evaluator), on the 228 distinct questions. vLLM ran the full 1,230-row set — every question repeated 8× (AIME) or 5× (GPQA) — which gives a per-question solve rate and therefore a noise floor. Every other configuration answered each question once and is compared question by question with vLLM's solve rates (paired difference with a z score whose variance includes the uncertainty of the reference) and with each other (exact McNemar test).

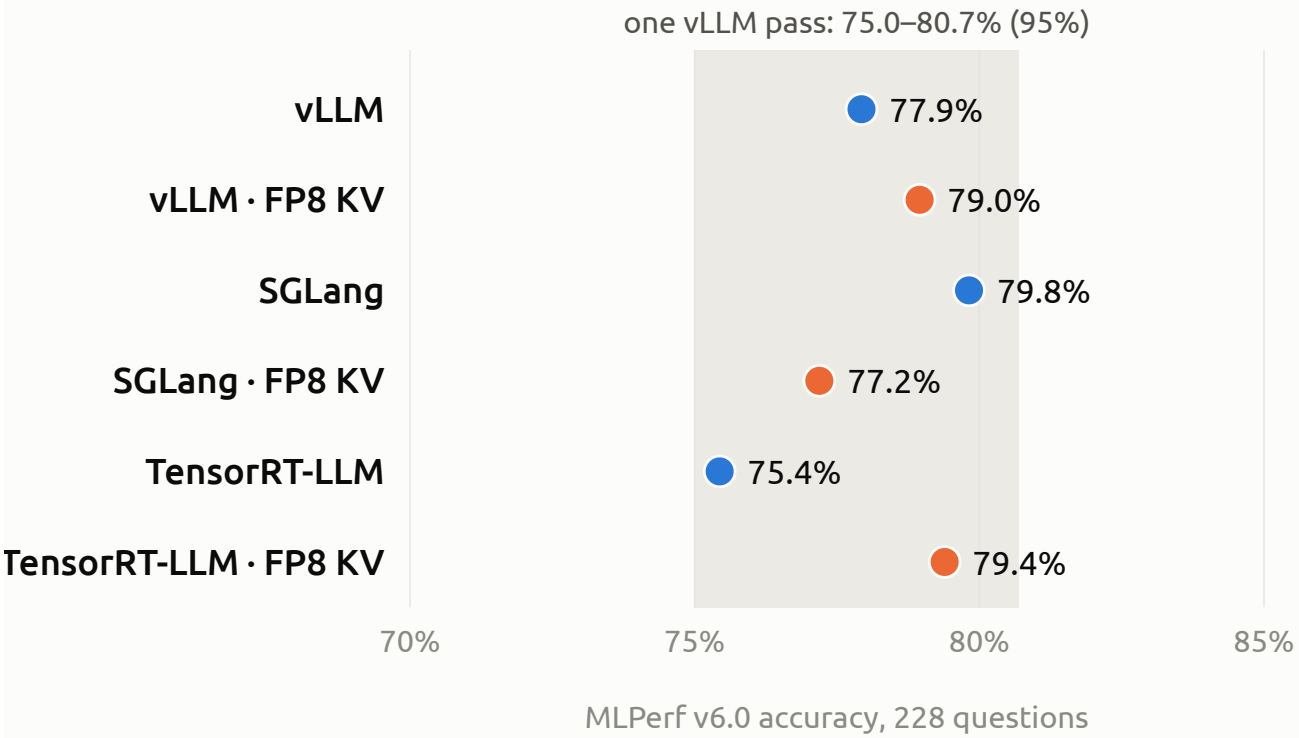
Speed. Synthetic chat load (~1k-token prompts, 1,024 output tokens), 1–64 concurrent users, three repeats per level. Servers were configured with a 2,304-token limit per request. A level runs only if users × 2,304 tokens fits the engine's reported KV pool (the “KV-fit” rule), so no engine is measured while silently truncating. (Accuracy runs used a 36,864-token limit.) Cost is GPU rental only at $5.241/h; energy from the GPU's own power telemetry.

Repeatability. The repeat test from Thinking Machines' work on nondeterminism: the same temperature-0 request, 512 tokens, sent 1,000 times while background traffic changes the batch around it. Load: 24 concurrent clients — 8 sending the identical 181-token probe and 16 sending other prompts with 64–1,024-token limits. Distinct outputs are counted over generated token IDs; every one of the 1,000-request probes stopped at 512 tokens, before a final answer, so these are distinct outputs, not distinct final answers.

Profiles. Nsight Systems, capturing only the window between the server's own start/stop-profile calls: 16 users, 32 requests × 256 tokens, after two warm-up requests.

Discipline. Every run is sealed with checksums and re-verified after copying off the VM; every table here regenerates from the sealed data with two Python scripts. Spending was capped at $100 by prepaid credit, a per-session stop line, and two independent watchdogs on two machines that delete the VM at the line; the session runner is tested against a fake cloud API with a local GPU, and the final session was rehearsed end to end that way before it ran.

2. Accuracy

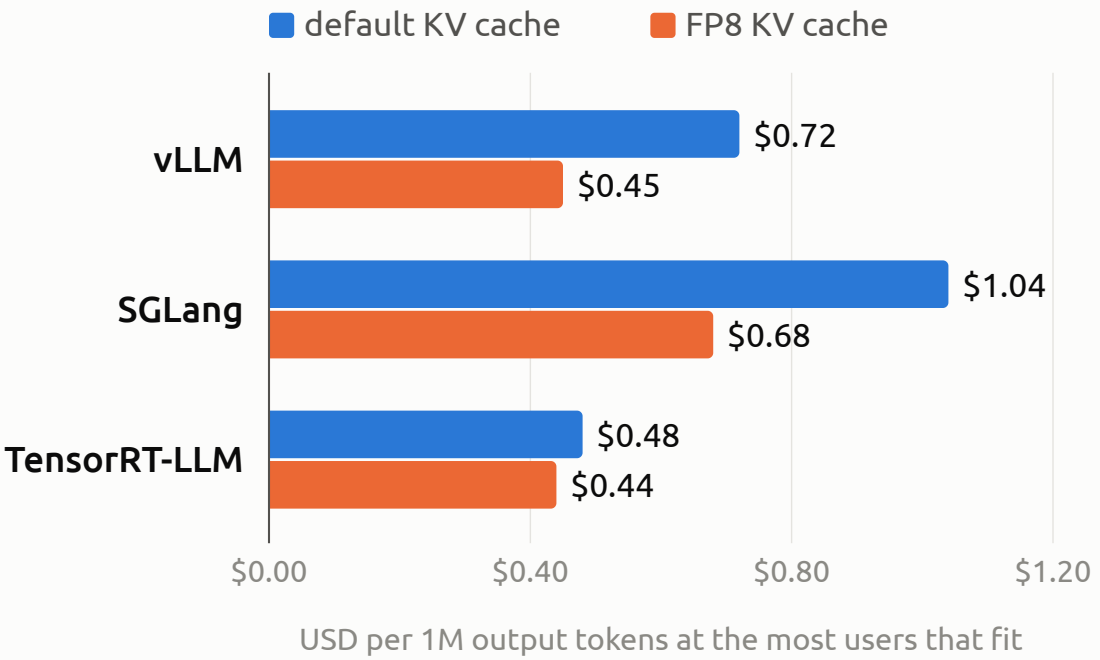


Blue = default KV cache, orange = FP8 KV cache. The grey band is where a single vLLM pass falls 95% of the time.

Configuration	Accuracy	Paired difference vs vLLM	p
vLLM (mean of 1,230 rows)	77.9%	reference	
TensorRT-LLM	75.4% (172/228)	-2.5 pts	0.12
SGLang	79.8% (182/228)	+1.9 pts	0.24
vLLM · FP8 KV	79.0% (180/228)	+1.0 pts	0.52
SGLang · FP8 KV	77.2% (176/228)	-0.7 pts	0.65
TensorRT-LLM · FP8 KV	79.4% (181/228)	+1.5 pts	0.36

At 228 questions, differences under about 3 points cannot be detected. FP8 KV cache moved accuracy between −2.6 and +4.0 points against the same engine's default, with no consistent sign. This supports the weak form of the project's starting thesis: for accuracy, precision choices matter more than the runtime — and here, neither mattered at this resolution.

3. Cost, energy and latency

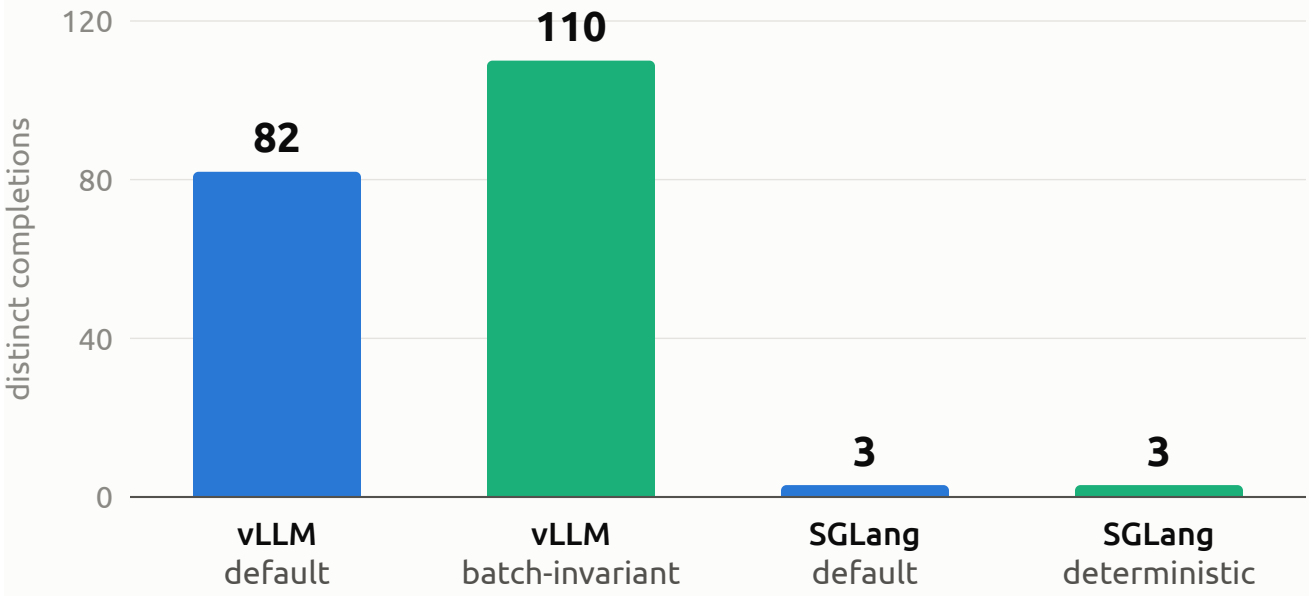


At the most users that pass the KV-fit rule (2,304 tokens per user). FP8 KV doubles the KV pool in every engine.

Configuration	KV pool (tokens)	Max users	Tok/s	$ / 1M out	J / token	Per-user tok/s	TTFT p50 @16
vLLM	113,326	32	2,020	$0.72	0.27	74	1.54 s
vLLM · FP8 KV	226,708	64	3,230	$0.45	0.18	57	1.46 s
SGLang	68,780	16	1,403	$1.04	0.38	91	0.49 s
SGLang · FP8 KV	137,560	32	2,152	$0.68	0.27	71	0.50 s
TensorRT-LLM	291,627	64	3,025	$0.48	0.20	52	1.14 s
TensorRT-LLM · FP8 KV	585,098	64	3,284	$0.44	0.19	58	1.18 s
vLLM · batch-invariant	71,966	16	1,019	$1.43	0.44	65	0.44 s
SGLang · deterministic	68,780	16	1,205	$1.21	0.40	79	0.57 s

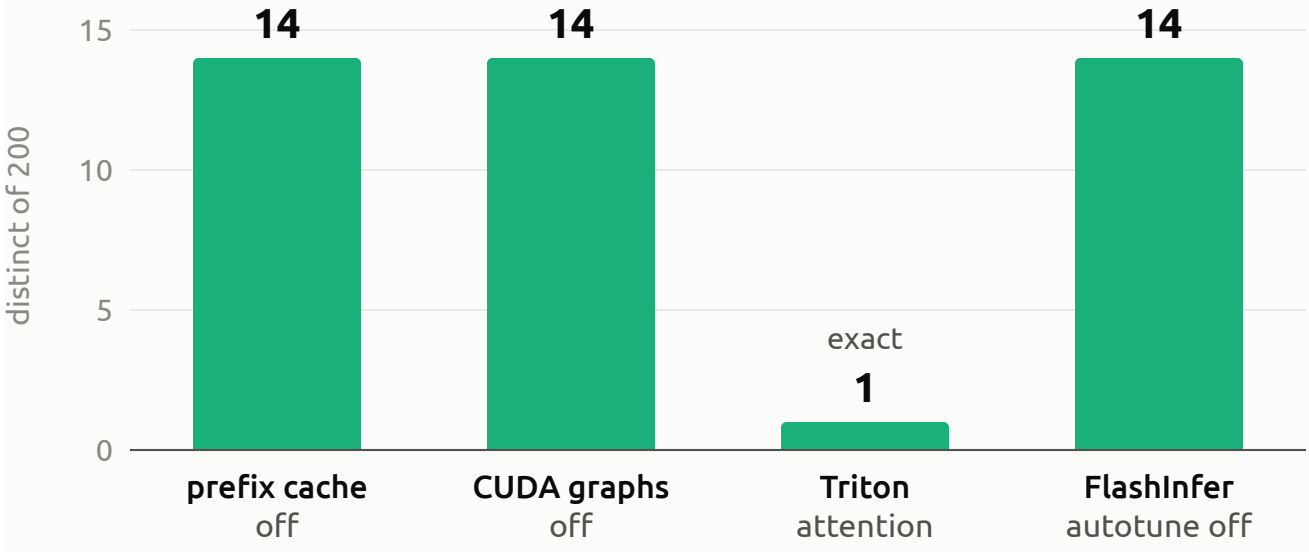
The FP8 saving comes from serving more users: at equal users, throughput changed −7% to +3% on vLLM, −2% to −1% on SGLang and 0% to +9% on TensorRT-LLM. At one user every engine costs $7–8 per million output tokens at ~1.6 J per token; the GPU pays off only with concurrency. SGLang's default KV pool is the smallest (its Hopper MXFP4 weight layout keeps more memory resident), yet it has the highest 16-user throughput and the lowest time to first token among the non-deterministic configurations. Run-to-run spread: SGLang 0.1–2.6% and both deterministic modes ≤ 0.7%; vLLM up to 17% (its first repeat at a new level is often the slowest — see section 5); TensorRT-LLM up to 10%.

4. Repeatability and its price



Blue = default, aqua = deterministic / batch-invariant mode. Temperature 0, identical prompt, 512 tokens, 24 concurrent clients (8 probe, 16 background).

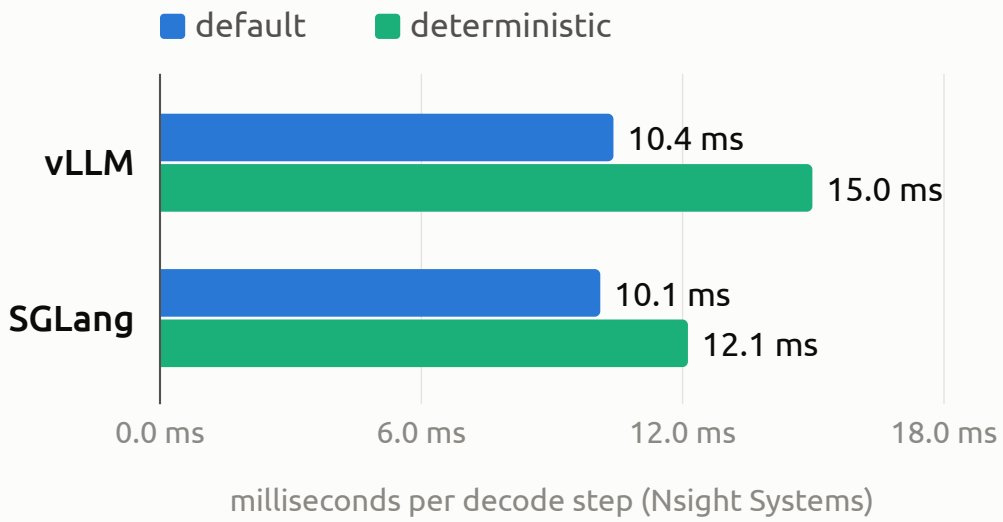
vLLM's batch-invariant mode (`VLLM_BATCH_INVARIANT=1`) did not make gpt-oss-120b repeatable on the H100, although its batch-invariant kernels were active. A local check on an RTX 3060 with a small dense model and the same vLLM image gave 1 distinct completion in 300 — the mode itself works — which pointed at the gpt-oss/Hopper path. The final session changed one thing at a time, 200 requests each:



All four runs with batch-invariant mode on.

Change (with batch-invariant mode)	Distinct /200	Largest identical group	Equal to solo run	Solo runs identical
`--no-enable-prefix-caching`	14	40	40	yes
`--enforce-eager`	14	48	0	no
`--attention-backend TRITON_ATTN`	1	200	200	yes
`--no-enable-flashinfer-autotune`	14	64	64	yes

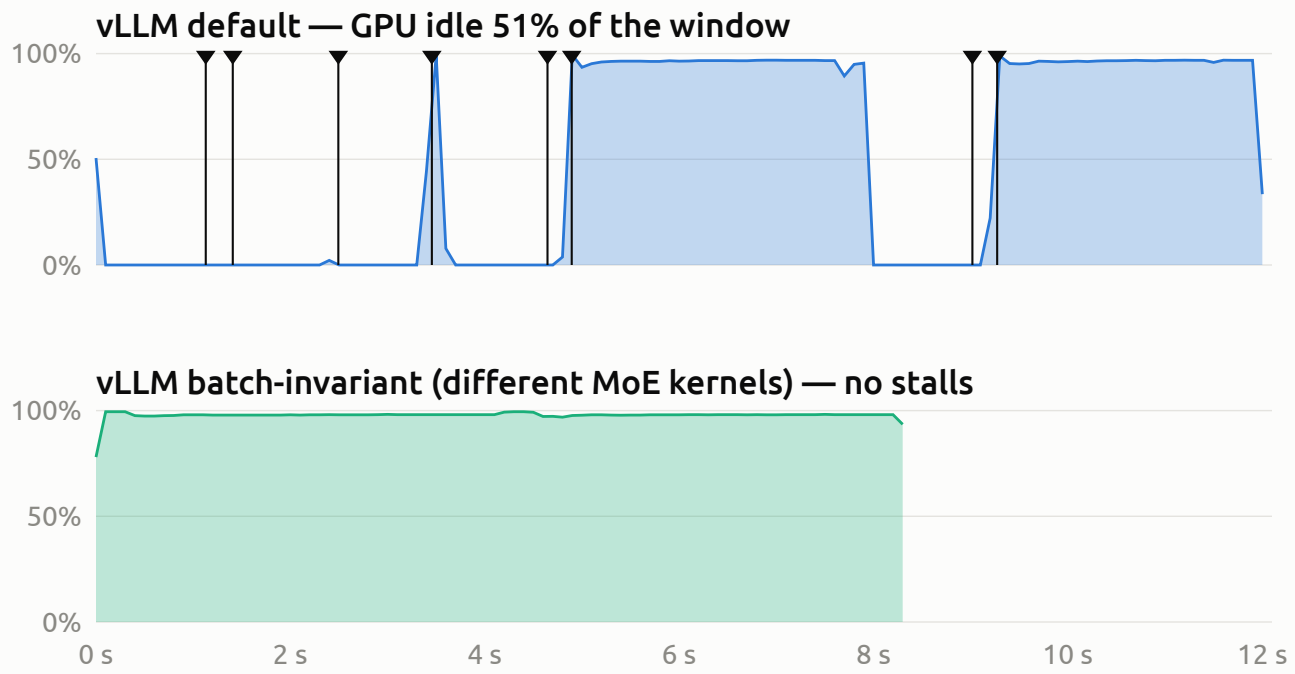
Of the four changes, only the attention backend mattered: with Triton attention instead of FlashAttention 3 the output was exact. A separate effect, seen in the local RTX 3060 check: with prefix caching on, the first (cache-miss) request differs from later (cache-hit) ones. The H100 solo runs were mixed (last column). The speed cost of Triton attention on the H100 was not measured.



Median time between decode steps (CUDA graph launches) in the Nsight captures.

The kernel tables show where the time goes. vLLM's batch-invariant mode replaces cuBLAS dense GEMMs with a persistent Triton matmul (36% of GPU time) and its Triton MoE kernels with batch-invariant-aware ones; SGLang's deterministic mode keeps its MoE kernels and swaps cuBLAS for DeepGEMM. Decode steps are 44% and 20% slower; cost per token 30% and 16% higher.

5. What the profiles showed

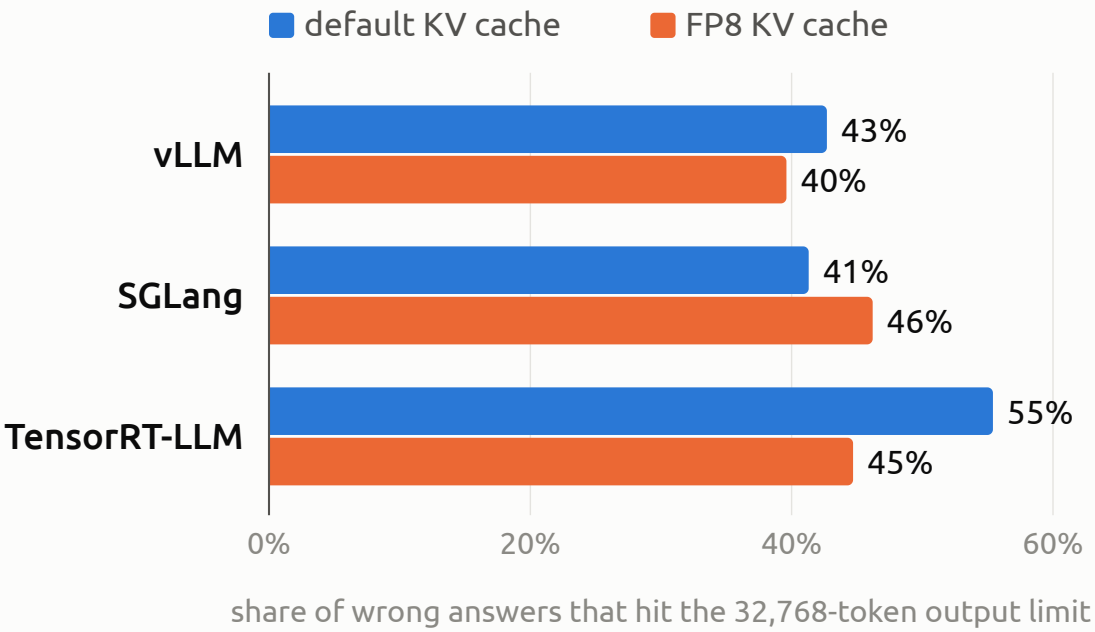


Share of each 100 ms with a kernel running. ▼ marks a GPU module load (cuModuleLoadData).

Capture	GPU busy	Median step	p90 step	Idle gaps > 50 ms	Gaps ending in a module load
vllm-default	48.8%	10.41 ms	11.71 ms	8 (5.95 s)	8
vllm-bi	97.9%	14.98 ms	16.39 ms	0 (0.0 s)	0
sglang-default	91.7%	10.11 ms	11.79 ms	2 (0.29 s)	2
sglang-det	92.7%	12.12 ms	13.8 ms	2 (0.11 s)	1

A vLLM server that had reported ready and served two warm-up requests still left the GPU idle for ~1 s at a time under 16-user load — 6 s of a 12 s window — and every gap ended with a new GPU module load. That is consistent with just-in-time compilation of the default MoE kernels for batch sizes not seen in warm-up. It is also consistent with vLLM's larger run-to-run spread. In production it means latency spikes for the first traffic after a restart unless the server is warmed up at every batch size.

6. Answers that never finish



Configuration	Hit 32,768 tokens	Share of wrong answers	Median tokens correct / wrong
vllm-default	128 (10.4%)	42.7%	4,312 / 27,755
trtllm-default	32 (14.0%)	55.4%	3,602 / 32,768
sglang-default	27 (11.8%)	41.3%	4,624 / 28,272
vllm-fp8kv	24 (10.5%)	39.6%	4,408 / 28,317
sglang-fp8kv	27 (11.8%)	46.2%	4,052 / 29,022
trtllm-fp8kv	26 (11.4%)	44.7%	4,671 / 28,098

44 of the 228 questions hit the output cap in at least one vLLM repeat; 78–85% of each other configuration's capped questions are among them, and the model solves those only about a third of the time. This is a property of the model on hard questions, not of an engine. At ~74 tokens/s per user a capped answer takes about seven minutes, so output limits and timeouts are part of the accuracy and latency budget.

7. A failure and its fix

Early in the project SGLang scored 12 of 18 on a calibration set where TensorRT-LLM scored 17 of 18. The cause was not the engine's math. The same configured “memory fraction” means three different things in the three engines — share of the whole GPU (vLLM), share of memory left after the weights (TensorRT-LLM), share of the whole GPU for weights plus KV cache (SGLang). With SGLang's MXFP4 weight layout keeping 70 GiB resident, its KV pool came out at 5,209 tokens — less than one long request — and answers were cut at about 5,000 tokens with an ordinary “length” finish. The server logged a warning at start-up and then served anyway.

Fixes: every measurement now stops if an engine's KV pool cannot hold one full-length request (speed levels use the stricter per-user KV-fit rule in section 1); SGLang runs at a memory fraction that gives a

68,780-token pool; and an upstream pull request to SGLang (**#52026, open**) adds a per-request warning when the KV pool silently lowers a request's output limit.

A second, operational failure: the Kubernetes step of one paid session silently did nothing — a shell function named `install` shadowed the system command and recursed until it crashed, and the step reported the clean-up's success. It was fixed, the runner now keeps the first failing exit status, and the deployment was re-tested on a local k3s cluster with the NVIDIA GPU Operator (ready in ~70 s; after deleting the pod on purpose, the service answered again after 73–206 s).

8. Tradeoffs: what I would choose

- **Default choice for gpt-oss-120b on one H100:** any of the three with FP8 KV cache; pick SGLang where time to first token under load matters, TensorRT-LLM or vLLM where the most concurrent users per GPU matters (64 users fit with FP8 KV).
- **Regulated or audit-heavy use that needs repeatable output:** SGLang's deterministic mode (16% more per token), or vLLM batch-invariant mode with Triton attention once its speed is measured.
- **Before taking traffic:** warm up at every batch size the scheduler will form; watch time to first token after restarts.
- **Output limits:** set them from the latency budget; a 32k limit on this model means multi-minute answers on hard prompts.

9. Limits

One GPU type (H100 SXM), one model, one cloud, one month's engine versions. Accuracy resolution is about ±3 points (228 questions). The speed workload is synthetic. Costs are GPU rental only. The batch-invariance ablation used 200 requests per change (baseline 1,000). The 37% compares the highest level each vLLM configuration passed (default 32 users, FP8 64); the KV-fit rule refused default at 64 (not shown to fail), and 48 was not measured. The repeat test used one prompt and one load (24 clients); FP8 KV, 1 or 64 users, and TensorRT-LLM were not repeat-tested, and probes ran largely in step (groups of 8), so they are not 1,000 independent samples. SGLang's less conservative scheduler (`--schedule-conservativeness 0.3`) matched its default at 1–16 users; the levels where it could matter did not fit its KV pool, so its effect is untested.

10. Cold start

From an empty VM to a verified model and pulled images took 11–19 minutes across five sessions, about 90% of it download. Engine ready times: vLLM 149 s median, SGLang 172 s, TensorRT-LLM 208 s.

Isaac Levinsky · MCMLV1 LLC — open to GPU inference and ML infrastructure roles, and contract work.

www.linkedin.com/in/isaaclevinsky/ · mcmlv1.com · github.com/IsaacLevinsky

All measurements November 2026. Engine names are trademarks of their owners; no affiliation.