Small engine steps keep every stream answering while long prompts are read

The mixed load on the fast attention kernel, with room for 2,048 tokens in an engine step instead of 8,192. The engine makes three to four steps a second where it made under one, and no second passes without an answer token. The price: prompts are read about a fifth slower.

Qwen3.8-27B · one AMD Instinct MI300X · vLLM 0.27.1 **Run** 20261125T223438Z-series-fa-batch2k

Changed 2,048-token engine steps and a long-prompt cap of 2,048, against the fast-kernel run

Load 70% short, 25% medium, 5% long prompts; 8, 32, 128, 256 streams

On the fast attention kernel the mixed load still starved decoding: while a long prompt was read the engine made about one step a second, and every stream gets one token per step. I gave a step room for 2,048 tokens instead of 8,192, with a long prompt capped at the same 2,048. The engine then made **3.5 steps a second at 128 streams instead of 1.0**, and the server wrote 3.7 times the answer tokens.

0
seconds without an answer token in the four loaded minutes; 46 with large steps

4.40
engine steps a second at 32 streams, 3.46 at 128, 2.81 at 256

122.7
answer tokens a second at 32 streams; 11.1 with large steps, in a minute disturbed by kernel builds

23.3%
fewer prompt tokens read a second at 128 streams: the price



Figure 1 The four loaded minutes of both runs. Grey: the run with steps of 8,192 tokens. Colour: this run, with its single seconds thin and grey behind the 7-second centred mean. Small steps lift answers and steps, and lower the reading of prompts, in the two minutes with the most streams.
Source: plan-mixed-overlap/timeline.json of this run and of 20261125T215049Z-series-fa

MEASURED

At 128 and 256 streams the server wrote 282.0 and 256.2 answer tokens a second; with large steps 75.9 and 118.0. That is 3.7 times and 2.1 times.

KEEP IN MIND

At 32 streams the ratio is 11.0 times, and it flatters this run: in that minute the run with large steps stood still for 27 seconds building kernels it had never used. The page on that minute takes it apart.

The machine, and the part that was changed

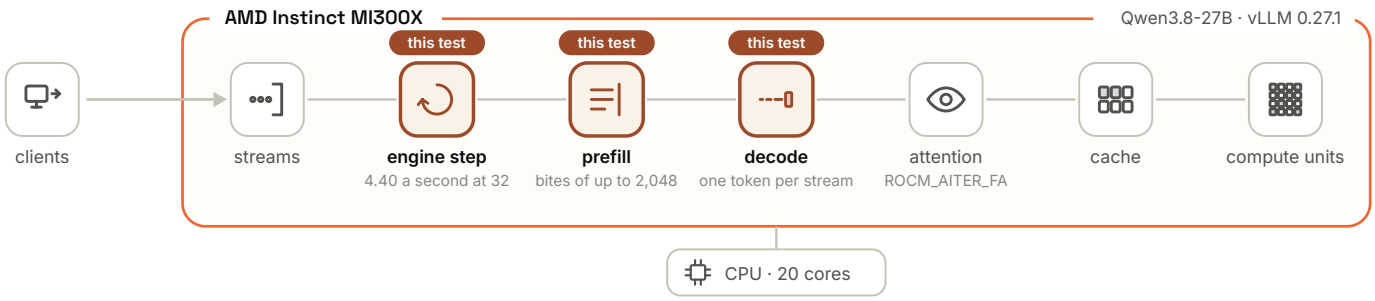


Figure 2 The card as every document in this series draws it. An engine step carries one new token for every stream that is answering and a bite of a prompt that is being read. This run makes the step smaller.
Source: FACTS.json; summary.json config and plans.plan-mixed-overlap.stages[stage=mix-32]

What differs between the two runs

Attention kernel	ROCM_AITER_FA in both
Room in one engine step	2,048 tokens here; 8,192 in the run before
Long-prompt cap	at most 2,048 tokens of one prompt in a step here; none before
Cache	1,728,754 token slots here; 1,734,958 before
The run before	20261125T215049Z-series-fa

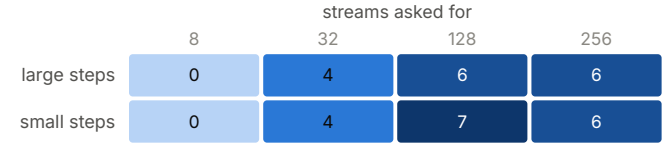
KEEP IN MIND
Two settings differ at once. This run cannot say which of the two does the work.

The plan

1. Four stages of 60 seconds: 8, 32, 128, then 256 requests kept open. A new one starts only when one ends.
2. Each request draws its prompt from a mix: 70% short (256 to 2,000 tokens), 25% medium (4,000 to 16,000), 5% long (32,000 to 64,000); answers of 64 to 1,024 tokens.
3. Then 120 seconds with no new requests; what is still open after that is closed by the load generator.

Settings

Model	Qwen3.8-27B: 27.8 billion parameters, BF16 weights, 262,144 tokens of context
Card	one AMD Instinct MI300X: 304 compute units, 191.7 GiB of memory
Server	vLLM 0.27.1+rocm723, ROCm 7.2.3, PyTorch 2.11.0+gitd0c8b1f
Machine	a cloud GPU virtual machine: 20 virtual cores, Intel Xeon Platinum 8568Y+
This run	20261125T223438Z-series-fa-batch2k, started 2026-11-25T22:34:51Z, server ready after 185 s



darker = more: long prompts sent in that minute

Figure 3 Long prompts sent in each minute, in both runs. The plan is a closed loop: a server that ends requests sooner is sent more of them. At 32 streams both runs were sent the same four long prompts as the minute began.
Source: plan-mixed-overlap/requests.jsonl of both runs (stage, asked_in)

KEEP IN MIND
The two runs were not sent the same requests: 342 here, 297 before. Request number N has the same lengths in both, and a faster server gets further down the list. The 32-stream minute is the like-for-like one in what was sent.



Figure 4 What one engine step holds in the two runs. The time is computed: one second divided by the engine steps a second at 128 streams.
Source: each run's summary.json (config, plans.plan-mixed-overlap.stages[stage=mix-128].engine_steps_per_s); FACTS.json

Minute by minute, before and after

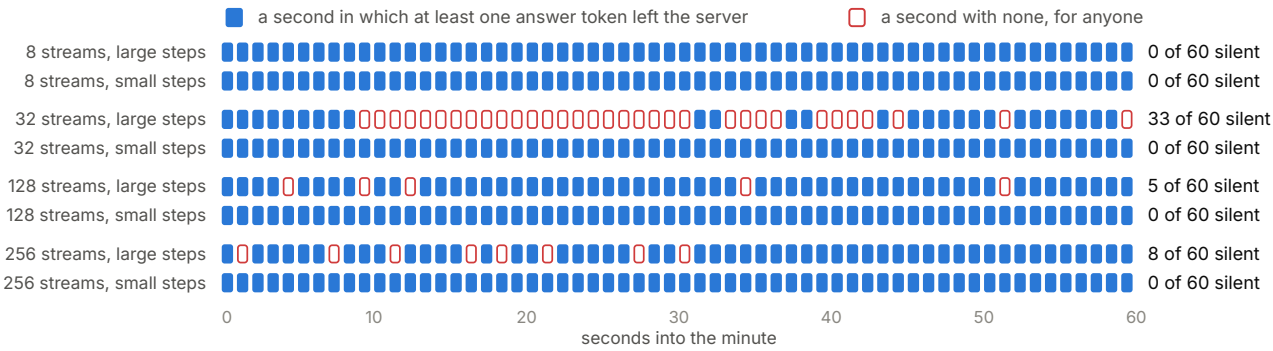


Figure 5 Every second of the four loaded minutes in both runs. With small steps no second is silent.
Source: plan-mixed-overlap/timeline.json of both runs (output_tok_s per one-second row)

Answer tokens a second, all streams together

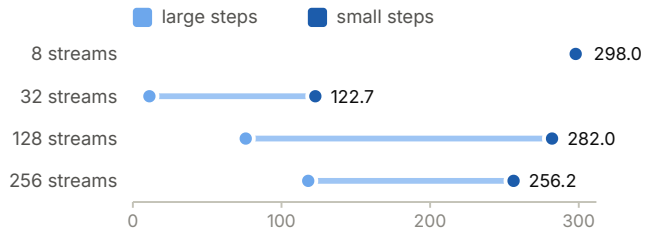


Figure 6 The gain. More answers in every loaded minute but the first, where no long prompt was in the mix.
Source: both runs' summary.json stages[].server_output_tok_s

Prompt tokens read a second

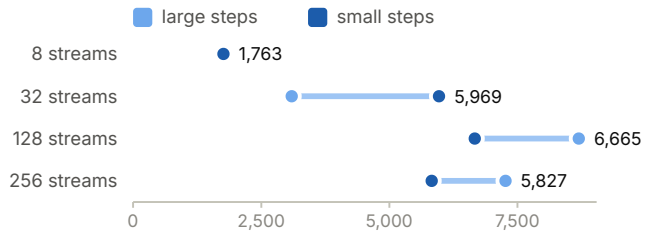


Figure 7 The price. Prompts are read faster at 32 streams and slower at 128 and 256.
Source: both runs' summary.json stages[].server_prompt_tok_s

The exact record of both runs

	ANSWER TOKENS/S	PROMPT TOKENS/S	ENGINE STEPS/S	SECONDS WITH NO ANSWER TOKEN	RUNNING AT MOST	WAITING AT MOST	CACHE AT MOST, %	FIRST TOKEN, MEDIAN S	REQUESTS SENT	CARD, W
8 STREAMS										
large steps	299.9	1,701	37.64	0 of 60	8	0	3.9	0.26	39	695
small steps	298.0	1,763	37.46	0 of 60	8	1	3.9	0.26	38	610
32 STREAMS										
large steps	11.1	3,095	0.47	33 of 60	30	22	19.0	6.77	25	316
small steps	122.7	5,969	4.40	0 of 60	32	22	26.3	7.17	31	744
128 STREAMS										
large steps	75.9	8,696	0.98	5 of 60	125	97	61.3	31.30	96	744
small steps	282.0	6,665	3.46	0 of 60	104	95	52.5	26.70	121	748
256 STREAMS										
large steps	118.0	7,266	0.88	8 of 60	174	131	91.8	65.19	137	746
small steps	256.2	5,827	2.81	0 of 60	100	170	59.8	≥ 95.66	152	724
DRAIN: NO NEW REQUESTS										
large steps	744.8	4,527	4.13	9 of 121	207	74	100.0	-	0	750
small steps	495.4	5,664	3.08	1 of 121	195	156	89.9	-	0	745

From each run's `summary.json`; seconds with no answer token are counted from its `timeline.json`. Tinted rows are this run. A stage row spans 59 s of its 60 s. The server counts a prompt's tokens when its reading ends. Running and waiting are the server's gauges, which move only when a step ends.

The minute at 32 streams, taken apart

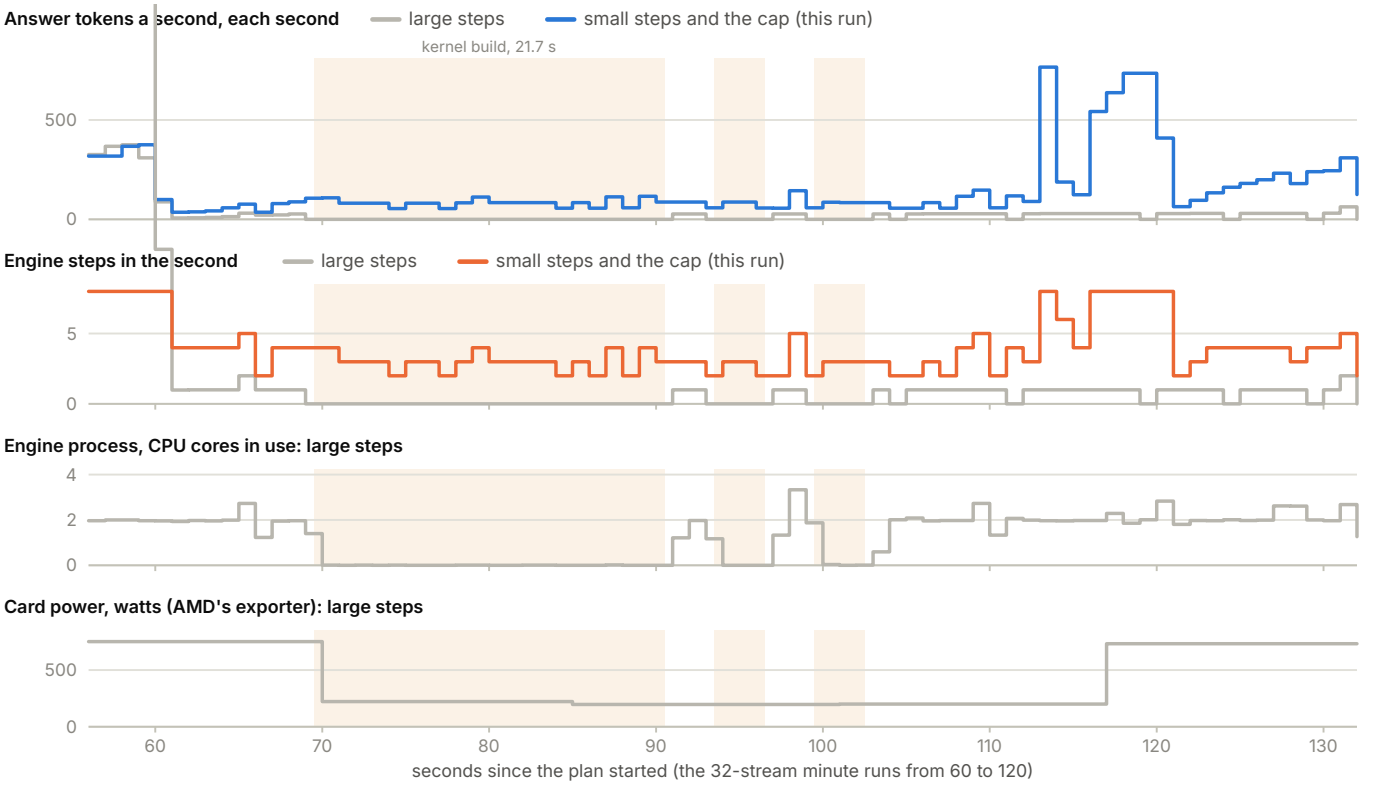


Figure 8 Both runs from second 56 to 132, without smoothing. Shaded: the seconds in which the engine process of the run with large steps used no CPU. Its card fell to idle power in the same seconds, and its log reports a kernel build of that length for each.
Source: plan-mixed-overlap/timeline.json of both runs; server.log of 20261125T215049Z-series-fa ("[aiter] finish build ... cost")

MEASURED

In its 32-stream minute the run with large steps did nothing for 27 seconds: no CPU for the engine process, the card at 196 W. Its log gives the reason: three attention kernels built on first use, 21.7, 3.8 and 3.7 s. This run started later and found them built.

MEASURED

The four long prompts sent as the minute began had their first token after 18.6 to 48.1 s here, and after 46.0 to 70.2 s with large steps, builds included. Once they are read, this run's last seconds of the minute reach 767 answer tokens a second.

COMPUTED

Counting only the seconds in which its engine worked, the run with large steps wrote 22.2 answer tokens a second in that minute and made 1.15 steps a second. Counted the same way this run wrote 132.6 and made 4.85: 5.9 times the tokens, not 11.0 times.

READING, NOT MEASUREMENT

My reading: every decoding stream gets one token per engine step, so shorter steps mean more turns for everyone. The price is that each step reads less of the waiting prompts.

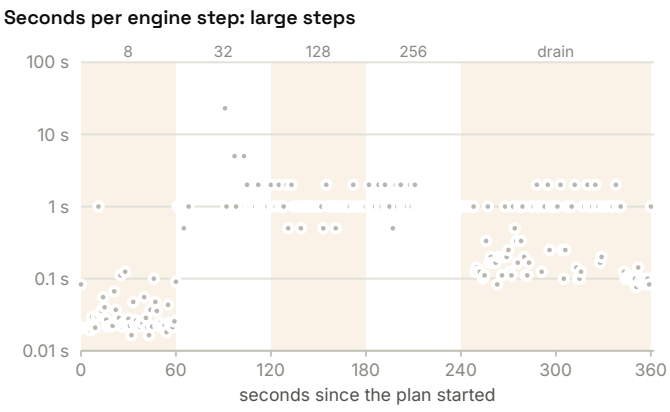


Figure 9 Large steps: once long prompts are in the mix a step takes about a second. The three gaps of 22, 4 and 4 seconds are the builds.
Source: timeline.json of 20261125T215049Z-series-fa (steps_s)

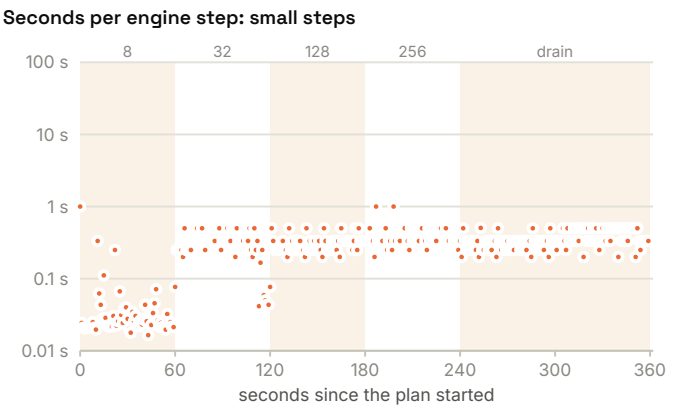


Figure 10 Small steps: a third of a second, for the whole plan. Computed from the step counter as in the other plot.
Source: plan-mixed-overlap/timeline.json (steps_s)

What the clients saw

Every request: large steps

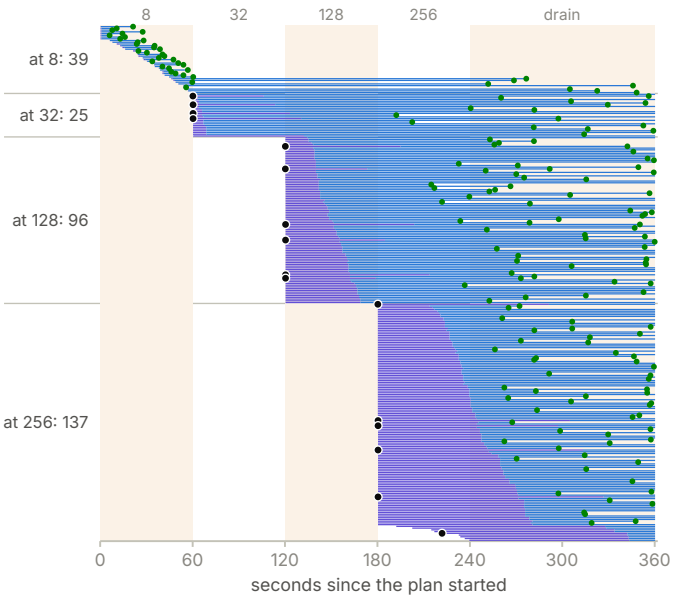


Figure 11 Every request of the run with large steps as one line, in the order sent. Violet: no answer token yet. Blue: answering. Green dot: ended well. Black dot: a long prompt.
Source: requests.jsonl of 20261125T215049Z-series-fa

Every request: small steps

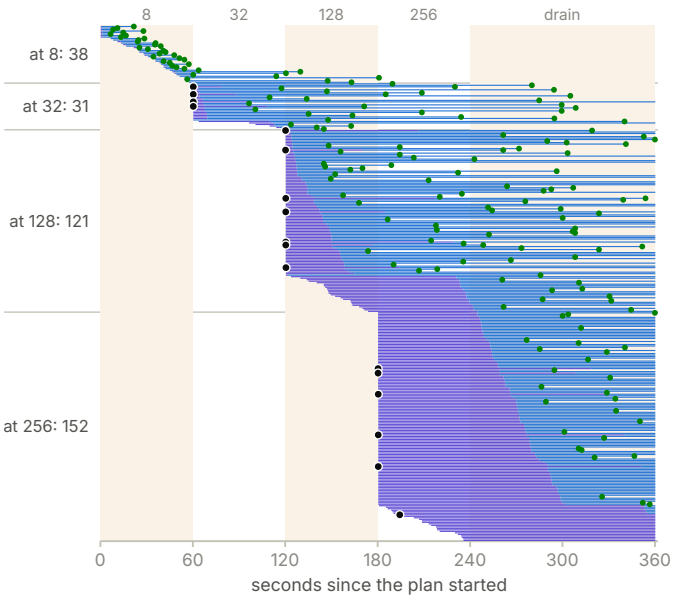


Figure 12 The same for this run. The violet wedge at the top right is the price: at 256 streams requests wait longer for their first token.
Source: plan-mixed-overlap/requests.jsonl (t_sent, t_first, t_end, end)

	REQUESTS SENT	LONG PROMPTS AMONG THEM	FIRST TOKEN, MEDIAN S	LONGEST, S	ENDED INSIDE THE PLAN	CUT WHILE ANSWERING	CUT BEFORE A FIRST TOKEN
SENT AT 8 STREAMS							
large steps	39	0	0.3	1.7	39	0	0
small steps	38	0	0.3	1.8	38	0	0
SENT AT 32 STREAMS							
large steps	25	4	6.8	71.9	15	10	0
small steps	31	4	7.2	48.1	29	2	0
SENT AT 128 STREAMS							
large steps	96	6	31.3	93.6	59	37	0
small steps	121	7	26.7	109.0	72	49	0
SENT AT 256 STREAMS							
large steps	137	6	65.2	146.4	54	83	0
small steps	152	6	91.7	172.2	27	107	18
ALL REQUESTS OF THE PLAN							
large steps	297	16	43.6	146.4	167	130	0
small steps	342	17	64.4	172.2	166	158	18

From each run's `requests.jsonl`. Tinted rows are this run. The median and the longest first-token time are over the requests that got a first token. "Ended inside the plan" compares within this pair only, and best at the 32-stream stair, where both runs were sent the same long prompts.

MEASURED

Of the requests sent at 32 streams, 29 of 31 ended inside the plan here and 15 of 25 with large steps. Over the whole plan the two runs ended almost the same number: 166 and 167.

NOT A RESULT

Requests "cut at the end of the plan" (176 here) are not refusals and not failures: the load generator closed them when the plan's time was over. All 342 requests got HTTP 200.

The same card, the same work, in a different order

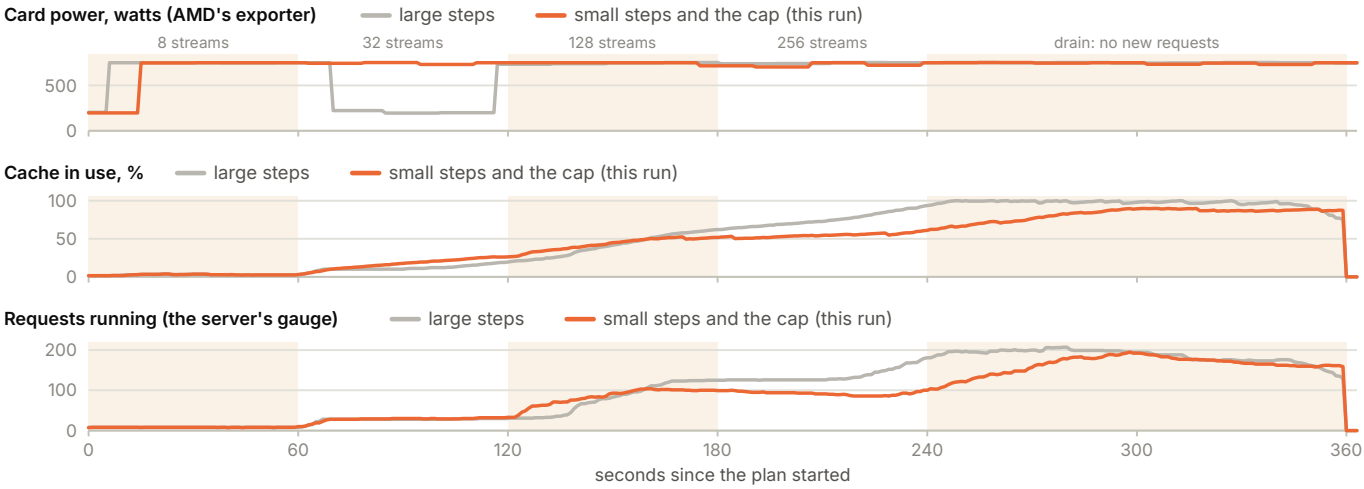


Figure 13 Both runs over the whole plan. Grey: large steps. Colour: this run. With large steps the server takes in more requests sooner and fills its cache; with small steps it holds fewer at once and stays below a full cache.
Source: plan-mixed-overlap/timeline.json of both runs (watts, cache_percent, running)

COMPUTED

○ Over the whole plan, drain included, this run wrote 117,480 answer tokens and the run with large steps 119,182 (the per-second counts of each timeline, added up): 1.4% apart.

MEASURED

The price shows at the top: at 256 streams this run kept at most 100 requests running against 174, the median first token came after at least 95.7 s against 65.2 s, and 18 requests were cut before a first token.

What I take from it

● With steps of 2,048 tokens and the cap, the engine makes three to four steps a second under this load and no second is silent.

○ The gain in answer tokens is 2.1 times to 3.7 times where nothing disturbed the comparison.

● Prompts are read 19.8% to 23.3% slower at 256 and 128 streams.

◌ Shorter steps give every stream more turns and each prompt less of every turn. Answers come earlier; capacity is not added.

Not results, and limits

- The ratio at 32 streams is inflated by the other run's kernel builds. Two settings changed together; which one works is open.
- A closed loop: the two runs met different requests after the 32-stream minute began. The bench has a second plan with the same requests at the same moments for every server.
- Prompts are random tokens. Nothing here is about the quality of an answer.

● measured: read from a run file · ○ computed: arithmetic on measured values · ◑ estimate: vLLM's, from the model's shape · ◌ my reading: interpretation, not measurement

Words used

Engine step
one pass of the model over a batch: one new token for every decoding stream, plus a bite of prefill.

Chunked prefill
a long prompt is read in bites across several engine steps, at most `max_num_batched_tokens` (8,192 here by default) per step for all requests together.

Every number in this document was read from these files at build time (sha256, first 12 characters):

`FACTS.json` 160f5ca0d495 · `GLOSSARY.md` 98323be796af · `plans/mixed-overlap.json` b86c1f4ef044 · `runs/20261125T215049Z-series-fa/plan-mixed-overlap/requests.jsonl` 694b08ff0653 · `runs/20261125T215049Z-series-fa/plan-mixed-overlap/t0` 2ee118d3a1b0 · `runs/20261125T215049Z-series-fa/plan-mixed-overlap/timeline.json` 2136a1e512e7 · `runs/20261125T215049Z-series-fa/server.log` a2aa2d010fcc · `runs/20261125T215049Z-series-fa/summary.json` ec0c01913c37 · `runs/20261125T223438Z-series-fa-batch2k/plan-mixed-overlap/requests.jsonl` c524013589a6 · `runs/20261125T223438Z-series-fa-batch2k/plan-mixed-overlap/t0` ddda6d548584 · `runs/20261125T223438Z-series-fa-batch2k/plan-mixed-overlap/timeline.json` e18647ef421b · `runs/20261125T223438Z-series-fa-batch2k/summary.json` 35fa706c2022

88 values bound by path, 13 figures drawn from data. built 2026-11-26 02:39Z, explainer-pdf-system v2.



CREATED AND DIRECTED BY

Peter Styk

HPC & AI Platform Engineer

GPU inference, data and performance

linkedin.com/in/peterstyk · github.com/styk-tv

Questions about a run? Connect on LinkedIn.

LinkedIn