Read alone in small bites, a full-context prompt takes 89 seconds instead of 73

The settings that kept every stream answering under mixed load, tried on one long prompt at a time. From 64,000 tokens up the first token comes later, by up to a fifth at the model's full context. Below that nothing changes, and decoding is as fast as before.

Qwen3.8-27B · one AMD Instinct MI300X · vLLM 0.27.1 **Run** 20261125T231329Z-series-fa-batch2k
Changed 2,048-token engine steps and a long-prompt cap of 2,048, against the fast-kernel run
Load one request at a time, 1,000 to 262,000 tokens in, 64 out

Small engine steps kept every stream answering when long prompts arrived in a mix. This is what the same settings cost a long prompt that is read alone. One request at a time, from 1,000 to 262,000 tokens, with room for 2,048 tokens in a step instead of 8,192: at the model's full context the first token comes after 89.3 s instead of 72.9.

22.5%
longer to the first token at 262,000 tokens

2,935
prompt tokens read a second at that length; 3,596 with large bites

2.04 s
to the first token at 16,000 tokens; 2.21 s with large bites: no cost down here

46.8
answer tokens a second once the full-context prompt is read; 45.7 with large bites

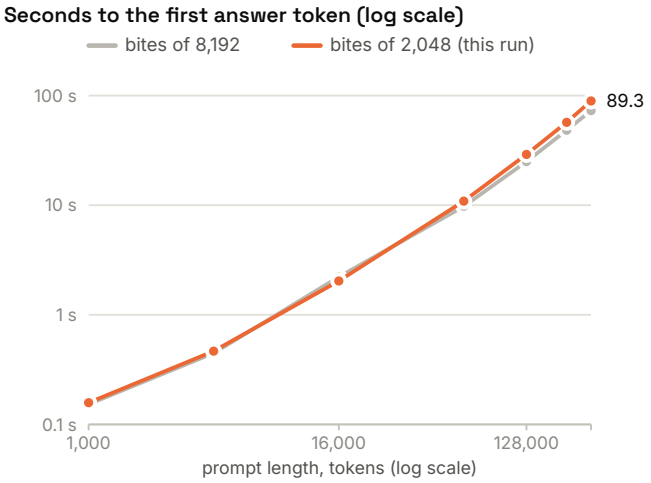


Figure 1 Seconds from sending a request to its first answer token, by prompt length, in the two runs. Both axes are logarithmic, so the gap at the right looks smaller than it is.
Source: summary.json plans.plan-context-ladder.single_requests[].first_token_s of this run and of 20261125T215049Z-series-fa

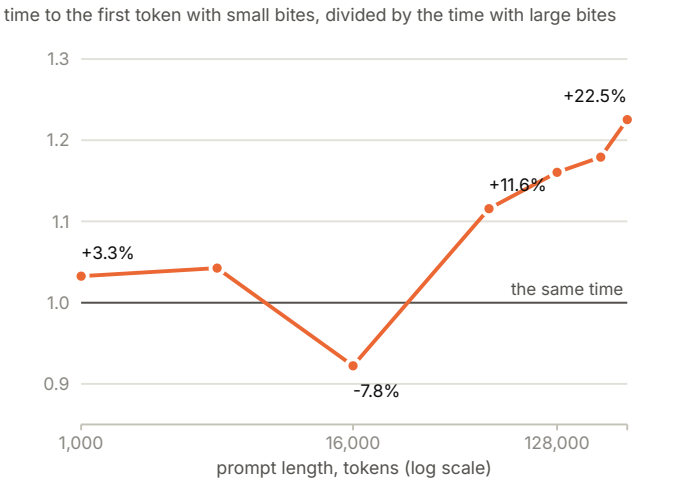


Figure 2 The same times as a ratio. Up to 16,000 tokens the two are a few hundredths of a second apart, or small bites are ahead; from 64,000 up small bites cost more the longer the prompt.
Source: computed from the two runs' first_token_s; every value is in the table on the next page

MEASURED
From 64,000 tokens up a prompt read in bites of 2,048 reaches its first token later: 10.9 s against 9.7 there, 89.3 s against 72.9 at the top.

MEASURED
Decoding does not change: 64.4 answer tokens a second after the shortest prompt in both runs, 46.8 and 45.7 after the longest (the server's own decode timer).

The machine, and the part that was changed

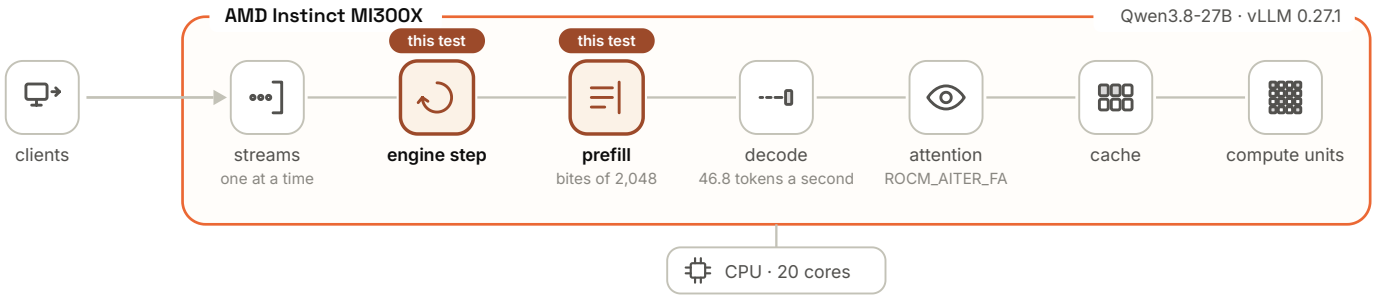


Figure 3 The card as every document in this series draws it. Prefill is the server reading the prompt; it reads a long one in bites, one bite per engine step. This run makes the bite a quarter of the size.
Source: FACTS.json; summary.json config, server.attention_kernel, plans.plan-context-ladder.single_requests

The plan

1. Seven requests, one at a time: 1,000, 4,000, 16,000, 64,000, 128,000, 200,000 and 262,000 tokens of prompt, each with an answer of 64 tokens.
2. The next request is sent when the one before has ended, so nothing shares a step with it.
3. Prompts are random token ids: nothing is reused between requests.
4. Two more requests that cannot fit the model's context follow; the server refused both at once, as on every ladder of this bench.

What differs between the two runs

Room in one engine step	2,048 tokens here; 8,192 in the run before
Long-prompt cap	2,048 tokens here; none before. For one prompt alone the two settings come to the same bite
Attention kernel	ROCM_AITER_FA in both
Cache	1,734,183 token slots here; 1,734,958 before
The run before	20261125T215049Z-series-fa

The exact record of both ladders

PROMPT, TOKENS	FIRST TOKEN, S: BITES OF 8,192	FIRST TOKEN, S: BITES OF 2,048	DIFFERENCE	PREFILL TOKENS/S: BITES OF 8,192	PREFILL TOKENS/S: BITES OF 2,048	DECODE TOKENS/S: BITES OF 8,192	DECODE TOKENS/S: BITES OF 2,048
1,000	0.15	0.16	+3.3%	6,555	6,321	64.4	64.4
4,000	0.45	0.47	+4.3%	8,951	8,578	65.5	65.6
16,000	2.21	2.04	-7.8%	7,234	7,844	64.7	64.7
64,000	9.74	10.87	+11.6%	6,570	5,888	59.5	59.9
128,000	25.00	29.01	+16.0%	5,121	4,412	54.6	54.8
200,000	48.38	57.05	+17.9%	4,134	3,506	50.2	50.2
262,000	72.85	89.27	+22.5%	3,596	2,935	45.7	46.8

From each run's `summary.json`. The first-token time is measured at the client; prefill tokens a second is the prompt divided by it. Decode is the server's own timer for the request. The tinted row is the model's full context. Like for like: the same plan and seed, the same kernel, one request at a time.

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	20261125T231329Z-series-fa-batch2k, started 2026-11-25T23:13:42Z, server ready after 98 s

More bites for the same prompt

Tokens of the longest prompt in the cache (the cache gauge times the cache size)

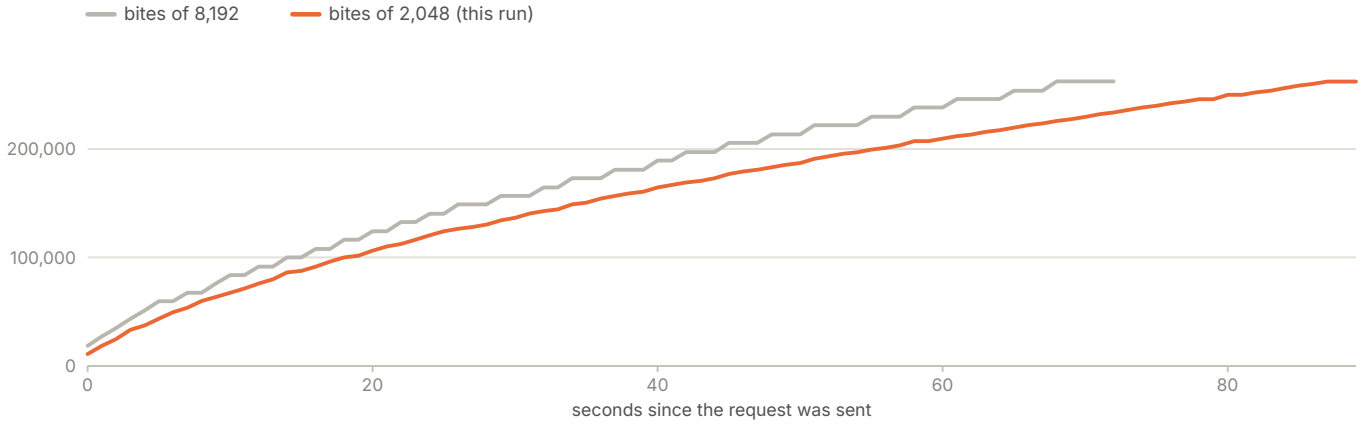


Figure 4 The longest prompt in both runs: how much of it is in the cache, second by second. Both curves bend: the more has been read, the slower the next part goes. With small bites the curve lies lower all the way.
Source: plan-context-ladder/timeline.json of both runs (cache_percent in the stage of the longest request); summary.json server.kv_cache_tokens

Prompt tokens read a second, over the whole prompt

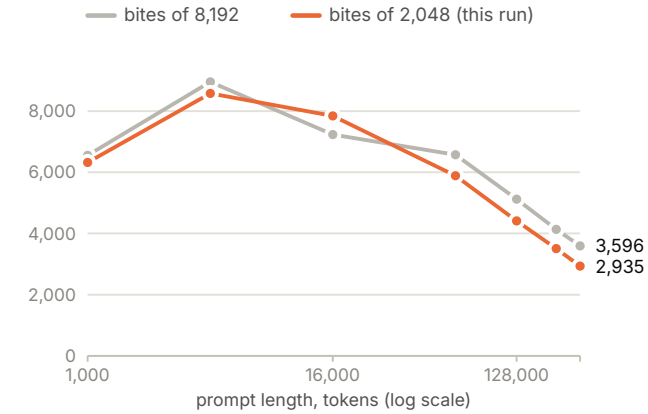


Figure 5 Reading speed over the whole prompt, by its length. On this kernel too a longer prompt is read more slowly; small bites lower the curve from 64,000 tokens up.
Source: summary.json plans.plan-context-ladder.single_requests[].prefill_tok_s of both runs

COMPUTED

A prompt of 262,000 tokens is 128 bites of 2,048 or 32 bites of 8,192: four times the engine steps for the same reading.

ESTIMATE

vLLM's estimate of the compute while the longest prompt is read: 728 TFLOPS here, 906 with large bites. Both are too high: the estimate counts attention in all 64 layers and this model has it in 16. Counted for 16 they come to 280 and 359 TFLOPS, 21.5% and 27.6% of AMD's stated peak (computed). The card draws 711 and 709 W.

READING, NOT MEASUREMENT

My reading: small steps are a trade. Every decoding stream gets more turns while prompts are being read, and each prompt is read in smaller bites, so a long prompt alone takes longer. Why four bites cost more than one of four times the size I did not measure; each step has work that does not depend on its size.

MEASURED

The other side of the trade is in the mixed load of the run 20261125T223438Z-series-fa-batch2k: with the same settings the engine made 3.46 steps a second at 128 streams where large steps made 0.98.

A trade, measured from its expensive side

What I take from it

● Read alone in bites of 2,048, a prompt of 262,000 tokens takes 89.3 s to its first token instead of 72.9.

● Up to 16,000 tokens there is no cost. Decoding is unchanged at every length.

◌ Small steps buy turns for the streams that are answering and pay with the reading of long prompts.

Not results, and limits

- One request for each length, played once: differences of a few hundredths of a second at the short rungs are within what one request can show.
- Two settings differ from the other run; for a prompt alone they come to the same bite.
- Prompts are random tokens. Nothing here is about the quality of an answer.
- TFLOPS is vLLM's estimate from the model's shape; the peak is AMD's stated figure.

● 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

Prefill
the server reading the prompt. Every prompt token passes through the model once before the first answer token.

First token
(time to first token) from sending the request to receiving the first answer token. It contains waiting in line and 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 · `cross/compute-estimate.json` b36121710947 · `plans/context-ladder.json` 2fe320742325 · `runs/20261125T215049Z-series-fa/plan-context-ladder/timeline.json` 6d9043e58e9a · `runs/20261125T215049Z-series-fa/summary.json` ec0c01913c37 · `runs/20261125T223438Z-series-fa-batch2k/summary.json` 35fa706c2022 · `runs/20261125T231329Z-series-fa-batch2k/plan-context-ladder/timeline.json` 6e2f0c3eb718 · `runs/20261125T231329Z-series-fa-batch2k/summary.json` af8932210ab6

66 values bound by path, 5 figures drawn from data. built 2026-11-26 02:38Z, 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