# One long prompt among 32 streams, and everyone's answer stops

Short, medium and a few very long prompts sent together. With 8 streams the server writes 158 answer tokens a second. With 32, one prompt of 54,698 tokens is being read, the engine makes a step every four seconds, and all streams together get 6.

Qwen3.8-27B · one AMD Instinct MI300X · vLLM 0.27.1 **Run** 20261125T201346Z-series
**Changed** nothing: baseline settings, default attention kernel **Load** 70% short, 25% medium, 5% long prompts; 8, 32, 128, 256 streams

I sent one server a mix of prompts, 70% short, 25% medium and 5% very long, and raised the streams kept open from 8 to 32, 128 and 256. With 8 streams it wrote 158.8 answer tokens a second. In the second minute one prompt of 54,698 tokens was being read among 32 streams: the engine made a step every 4 seconds, and all 32 streams together got 6.3 answer tokens a second. Above that it was about one.

**6.3**
answer tokens a second for 32 streams together; 158.8 with 8

**0.24**
engine steps a second at 32 streams; 19.91 at 8

**45 of 60**
seconds at 32 streams with no answer token for anyone

**14 of 270**
requests ended inside the plan; none was refused, none failed

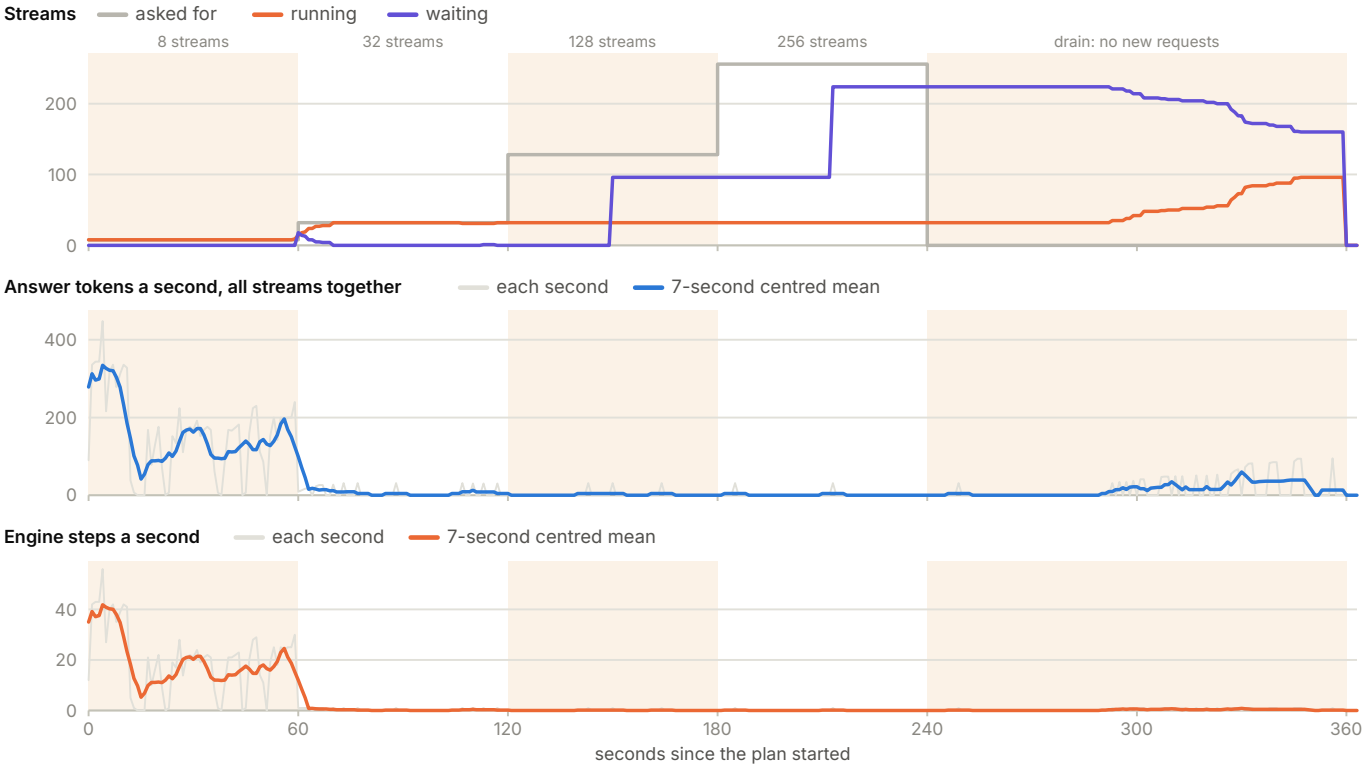



Figure 1 The whole plan, second by second. Top: streams asked for (grey steps), running and waiting, as the server's gauges give them; they move only when an engine step ends. Middle and bottom: answer tokens and engine steps a second, each second in grey and a 7-second centred mean in colour. Shaded bands are the stages.
Source: plan-mixed-overlap/timeline.json (vLLM's counters and gauges sampled every second)

MEASURED
In the third and fourth minutes the server ran 32 requests and no more, though it allows 1,024 and its cache was 15.8% full. The rest waited, 224 of them by the end of the fourth minute, and the server's reason for every one was "capacity".

MEASURED
Nothing was refused and nothing failed: all 270 requests got HTTP 200. 14 ended inside the plan; 256 were still open when its time ran out and were closed by the load generator.

## The machine, and the three parts this test leans on

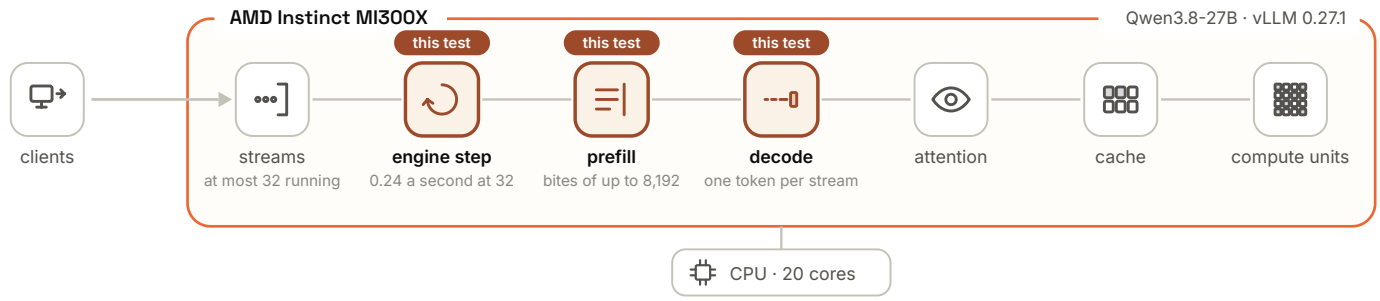



Figure 2 The card as every document in this series draws it. This test is about one engine step being shared: a bite of a prompt that is being read (prefill) and one new token for every stream that is answering (decode) travel in the same step.
Source: FACTS.json; summary.json plans.plan-mixed-overlap.stages[stage=mix-32]

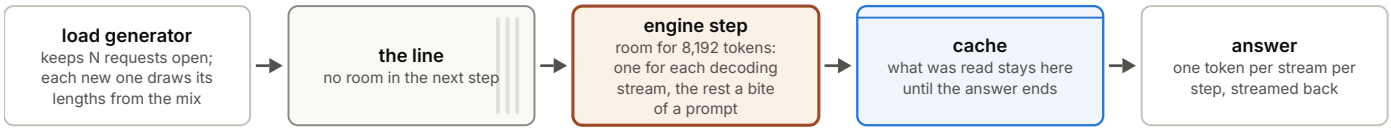



Figure 3 A closed loop: the load generator keeps a fixed number of requests open. Whatever does not fit into the next engine step waits in the line.
Source: loadgen.py; plans/mixed-overlap.json; summary.json server.chunked_prefill

## The plan

1. Four stages of 60 seconds, back to back: 8, 32, 128, then 256 requests kept open. A new one starts only when one ends.
2. Every request draws its prompt length from a mix: 70% short (256 to 2,000 tokens), 25% medium (4,000 to 16,000), 5% long (32,000 to 64,000). Its answer is forced to a length between 64 and 1,024 tokens.
3. Prompts are random token ids, so no two share a beginning and nothing is reused between requests.
4. After the last stage the generator waits 120 seconds for the open requests, then closes those still open.
5. Beside it a sampler reads the server's counters, the card and the CPU once a second.

| | prompt length drawn | | |
|---|---|---|---|
| | short | medium | long |
| sent at 8 streams | 15 | 6 | 0 |
| sent at 32 streams | 18 | 5 | 2 |
| sent at 128 streams | 69 | 21 | 6 |
| sent at 256 streams | 93 | 29 | 6 |

darker = more: requests

Figure 4 What was sent: requests by the stage they were sent in and by prompt length. No long prompt was drawn in the first minute.
Source: plan-mixed-overlap/requests.jsonl (stage, asked_in)

## 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 | 20261125T201346Z-series, started 2026-11-25T20:13:59Z, server ready after 94 s |
| Attention kernel | ROCM_ATTN, by vLLM's own preference order |
| Room in one engine step | 8,192 tokens, the default on this card |
| Stream slots | 1,024 |
| Cache | 1,826,476 token slots |

KEEP IN MIND

The load is made on the same machine as the server and shares its CPU cores; no request leaves the machine. The lengths are drawn in order from one seeded sequence, so request number N has the same lengths in every run of this plan. A faster server ends requests sooner and gets further down the sequence.

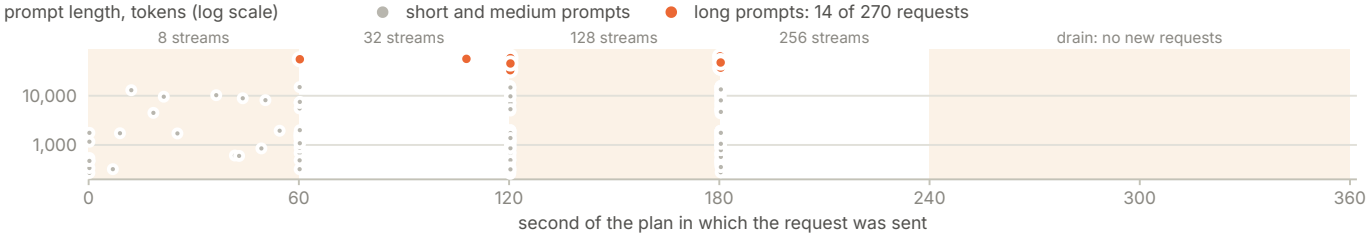



Figure 5 Every request as a dot: when it was sent and how long its prompt was. A closed loop sends a burst when a stage begins, then one request for each that ends. After the first minute almost none end.
Source: plan-mixed-overlap/requests.jsonl (t_sent, asked_in)

## Eight streams run. Thirty-two already crawl

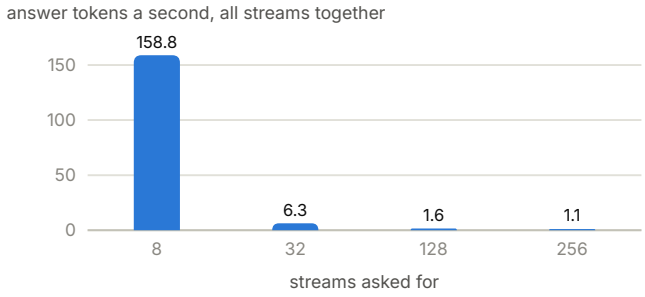



**Figure 6** Answer tokens a second for all streams together, by stage.
Source: summary.json plans.plan-mixed-overlap.stages[].server_output_tok_s

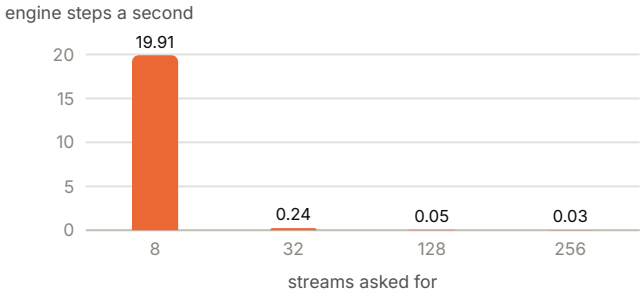



**Figure 7** Engine steps a second, by stage. Every running stream gets one token per step.
Source: summary.json plans.plan-mixed-overlap.stages[].engine_steps_per_s

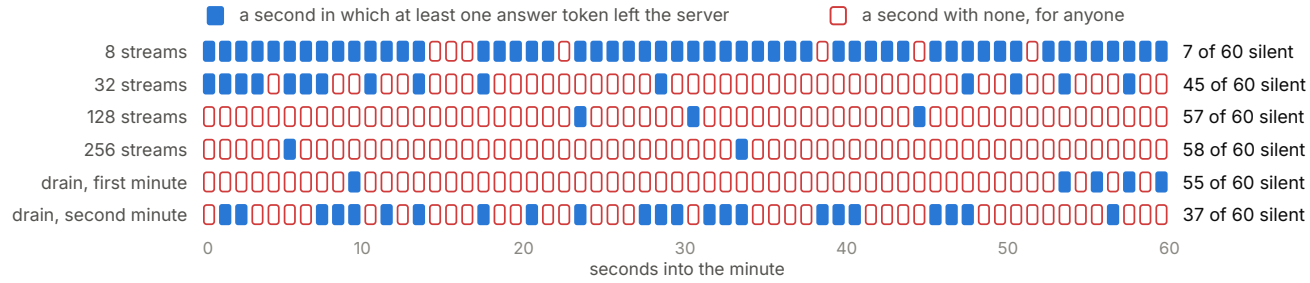



**Figure 8** The same plan as seconds, a row for each minute. With 8 streams the server falls silent now and then. From 32 on, silence is the rule, and it stays the rule after the last request was sent.
Source: plan-mixed-overlap/timeline.json (output_tok_s per one-second row)

### The exact record, the server's side

| STREAMS ASKED FOR | RUNNING AT MOST | WAITING AT MOST | ANSWER TOKENS/S | PROMPT TOKENS/S | ENGINE STEPS/S | ENGINE STEPS, COUNTED | SECONDS WITH NO ANSWER TOKEN | CACHE AT MOST, % | TFLOPS, EST. | CARD, W | MATRIX UNITS, % |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 8 | 0 | 158.8 | 1,066 | 19.91 | 1,197 | 7 of 60 | 3.7 | 67.2 | 413 | 5.6 |
| 32 | 32 | 18 | 6.3 | 1,104 | 0.24 | 15 | 45 of 60 | 13.5 | 135.1 | 505 | 9.4 |
| 128 | 32 | 96 | 1.6 | 943 | 0.05 | 3 | 57 of 60 | 14.9 | 32.5 | 366 | 0.2 |
| 256 | 32 | 224 | 1.1 | 0 | 0.03 | 2 | 58 of 60 | 15.8 | 26.6 | 303 | 0.1 |
| none new (drain) | 96 | 224 | 14.2 | 2,204 | 0.24 | 28 | 93 of 121 | 37.8 | 117.4 | 359 | 13.6 |

One row per stage from `summary.json`; a row spans 59 s of its 60 s stage. "Engine steps, counted" and "seconds with no answer token" are from the one-second rows of `timeline.json`. The server counts a prompt's tokens in the second its reading ends, so the zero in the prompt-tokens column at 256 streams means no prompt was finished in that minute, not that none was read. TFLOPS is vLLM's estimate, too high for long prompts (attention counted in 64 layers, the model has it in 16); power and matrix units are AMD's exporter.

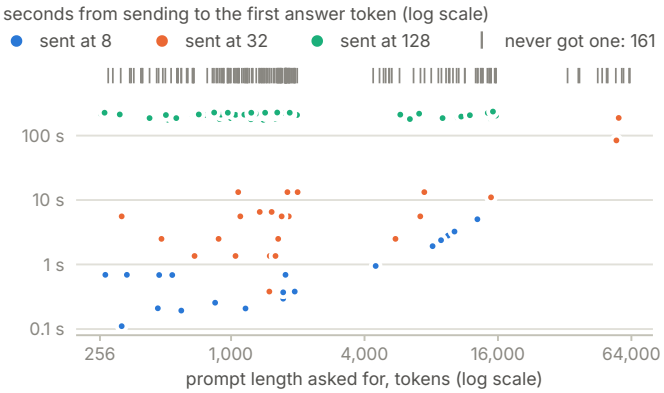



**Figure 9** Each dot is one request: how long its prompt was and how long it waited for its first answer token. With 8 streams the wait grows with the prompt. Sent at 128 streams, every length waits about as long.
Source: plan-mixed-overlap/requests.jsonl (asked_in, t_first - t_sent)

**MEASURED**

With 8 streams the mix held no long prompt: 15 short and 6 medium ones. Even so, 7 of that minute's seconds passed with no answer token for anyone, each of them between the sending of a medium prompt and its first token.

**MEASURED**

With 32 streams the engine made 15 steps in the whole minute, then 3 and 2 in the two minutes after. Asking for more streams added nothing: the server kept 32 running and the rest in line.

## A step that carries a bite of a long prompt takes seconds, and every stream waits for it

seconds per engine step (log scale): one dot for every second in which a step ended. Bars above: a long prompt, from sent to first token

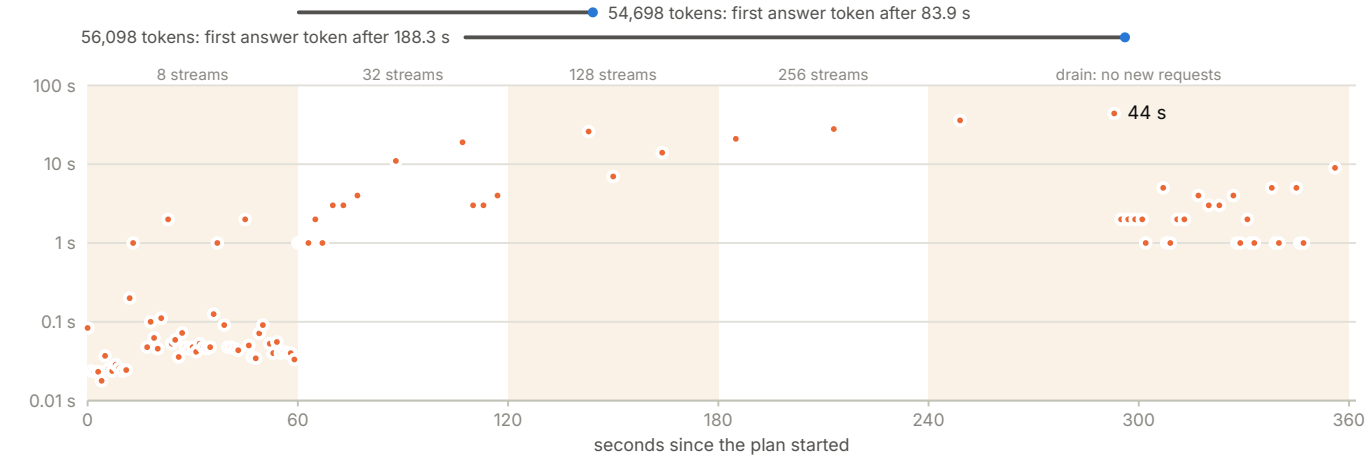



Figure 10 How long one engine step took, over the plan. In the first minute a step takes a few hundredths of a second. From the moment a long prompt is being read (the two bars above the plot) a step takes seconds, then tens of seconds. Computed from the step counter: in a second with several steps, one second divided by their number; otherwise the seconds since the last second in which a step ended.
Source: plan-mixed-overlap/timeline.json (steps_s); plan-mixed-overlap/requests.jsonl (the two long prompts the server read)

| THE PROMPT BEING READ | SENT AT SECOND | THE PROMPT BEFORE IT WAS READ BY SECOND | COUNTED AS READ AT SECOND | ENGINE STEPS IN BETWEEN | SECONDS EACH TOOK | FIRST ANSWER TOKEN AFTER |
|---|---|---|---|---|---|---|
| 54,698 tokens | 60 | 73 | 143 | 7 | 4 · 11 · 19 · 3 · 3 · 4 · 26 | 83.9 s |
| 56,098 tokens | 108 | 143 | 295 | 7 | 7 · 14 · 21 · 28 · 36 · 44 · 2 | 188.3 s |

### Every step while those two prompts were read

| STEP ENDED AT SECOND | IT TOOK, S | ANSWER TOKENS GIVEN | PROMPT TOKENS COUNTED AS READ | CACHE GAUGE, % | GAUGE ROSE BY, TOKEN SLOTS | CARD, W |
|---|---|---|---|---|---|---|
| 73 | 3 | 31 | 12,313 | 10.96 | 8,600 | 631 |
| 77 | 4 | 31 | - | 11.43 | 8,600 | 631 |
| 88 | 11 | 31 | - | 11.85 | 7,700 | 309 |
| 107 | 19 | 31 | - | 12.06 | 3,800 | 447 |
| 110 | 3 | 30 | - | 12.49 | 7,900 | 447 |
| 113 | 3 | 30 | - | 12.96 | 8,600 | 740 |
| 117 | 4 | 30 | - | 13.51 | 10,000 | 740 |
| 143 | 26 | 31 | 54,698 | 13.98 | 8,600 | 326 |
| 150 | 7 | 31 | - | 14.40 | 7,700 | 326 |
| 164 | 14 | 31 | - | 14.87 | 8,600 | 321 |
| 185 | 21 | 31 | - | 15.29 | 7,700 | 305 |
| 213 | 28 | 31 | - | 15.76 | 8,600 | 301 |
| 249 | 36 | 31 | - | 16.23 | 8,600 | 301 |
| 293 | 44 | 31 | - | 17.08 | 15,500 | 303 |
| 295 | 2 | 34 | 58,016 | 17.54 | 8,400 | 303 |

From `timeline.json`. Tinted: a step in which the server counted prompts as read. The gauge's rise is computed with the cache size (1,826,476 slots), rounded to hundreds.

READING, NOT MEASUREMENT

My reading: every decoding stream gets one token per engine step, so a step that carries a long prompt's prefill bite makes all of them wait. A step has room for 8,192 tokens. The streams that are answering take one each, the prompt being read takes all the rest, and nothing is left to start another request. That would be why the server says "capacity" with its cache 15.8% full.

COMPUTED

While the second long prompt was read, the cache gauge rose from 13.98% to 16.23% over five steps: about 8,200 token slots a step, the size of one bite. Each of those steps gave 31 answer tokens: 0.3% of what the step carried.

KEEP IN MIND

The server's "tokens per step" counter does not show the bite: it counts a whole prompt in the step that finishes reading it. It reads 31 for most of these steps and 54,729 for the step that finished the first long prompt, so its stage means (4,601 at 32 streams, 18,264 at 128) are not the size of a step.

KEEP IN MIND

Not explained: the steps are not equally slow. Three of the first prompt's seven took three or four seconds, with the card drawing 447 W or more. The others took longer the more of the prompt had been read, with the card near 303 W. I cannot say what decides between the two.

## What a client saw: a first token, then almost nothing

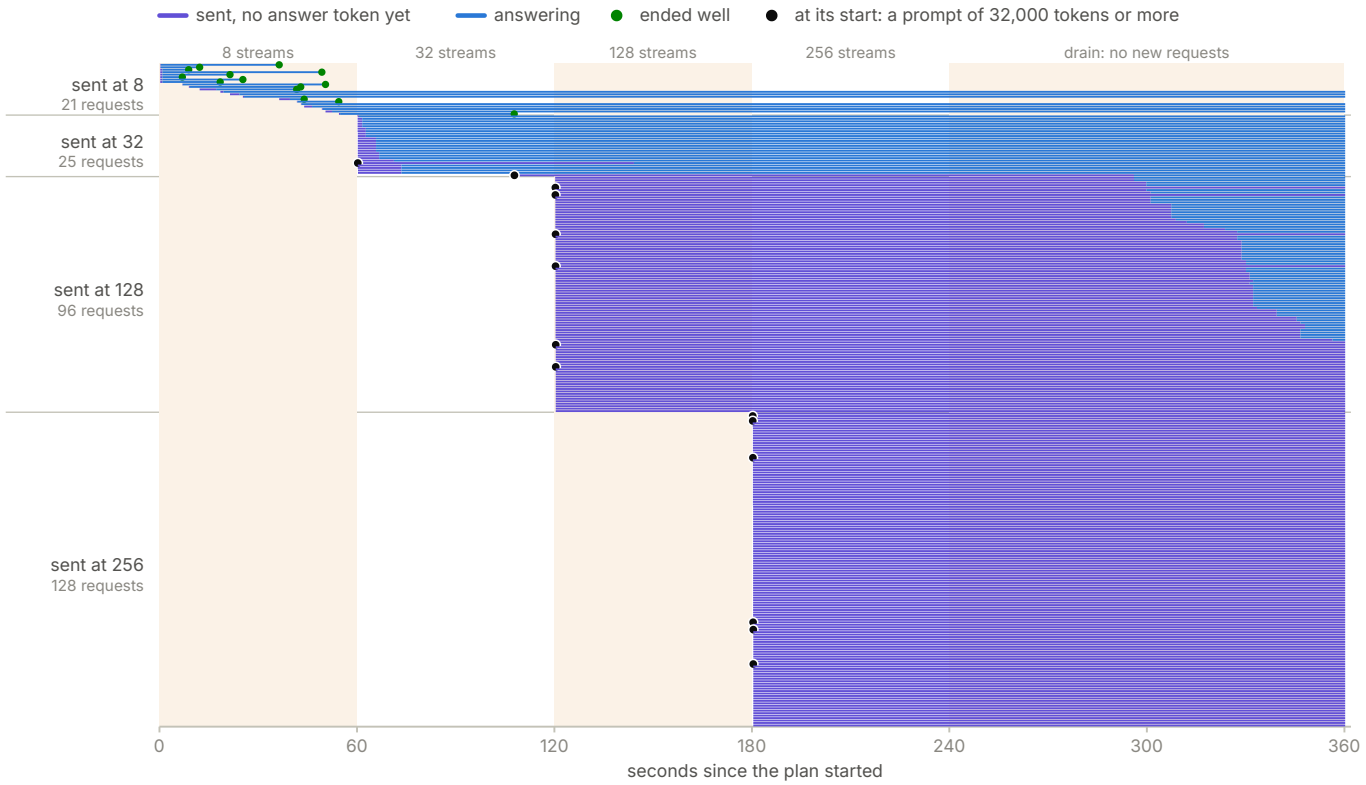



Figure 11 Every request of the plan as one line, in the order sent, top to bottom. Violet: sent and still without an answer token. Blue: answering. A green dot: it ended well. Everything that reaches the right edge was still open when the plan ended.
Source: plan-mixed-overlap/requests.jsonl (t_sent, t_first, t_end, end)

### The exact record, the clients' side

| REQUESTS | SENT | SHORT | MEDIUM | LONG | FIRST TOKEN, MEDIAN S | LONGEST, S | ENDED INSIDE THE PLAN | CUT WHILE ANSWERING | CUT BEFORE A FIRST TOKEN |
|---|---|---|---|---|---|---|---|---|---|
| sent at 8 streams | 21 | 15 | 6 | 0 | 0.7 | 5.0 | 14 | 7 | 0 |
| sent at 32 streams | 25 | 18 | 5 | 2 | 5.6 | 188.3 | 0 | 25 | 0 |
| sent at 128 streams | 96 | 69 | 21 | 6 | 209.8 | 236.0 | 0 | 63 | 33 |
| sent at 256 streams | 128 | 93 | 29 | 6 | - | - | 0 | 0 | 128 |
| all | 270 | 195 | 61 | 14 | 187.2 | 236.0 | 14 | 95 | 161 |

From `requests.jsonl`. The median and the longest first-token time are over the requests that got a first token. The client takes the first piece of visible text as the first token; a token without visible text sends nothing, so when steps are seconds apart a first-token time can be a step or two late.

MEASURED

All 25 requests sent at 32 streams got a first token. Then each got a token only when a step ended: in the five minutes that followed, none of them received more than 48 pieces of text, of answers that were to be 69 to 843 tokens long.

MEASURED

Of the 96 sent at 128 streams, 63 got a first token, after 176 to 236 s. None of the 128 sent at 256 streams got one.

MEASURED

14 long prompts were sent and the server read two: 54,698 tokens in 83.9 s and 56,098 in 188.3 s. Alone on this same server, a prompt of 64,000 tokens had its first token after 22.0 s.

NOT A RESULT

The 256 requests "cut at the end of the plan" are not refusals and not failures: the load generator closed them when the 120 s after the last stage were over. 95 were in the middle of an answer, 161 had no first token yet. How many ended inside the plan is itself the result.

## The same plan with a switch changed: what can be compared, and what cannot

KEEP IN MIND

The plan was played on four other servers, and they did not meet the same prompts. It is a closed loop and draws each prompt's length as the request is sent, so a faster server is sent more requests and draws further into the sequence. In the 32-stream minute this run was sent two long prompts, the runs with the cap and with small steps one each, the two fast-kernel runs four each. At that stair only two pairs are like for like: the cap against small steps, and the fast kernel against all three.

| | streams asked for 8 | 32 | 128 | 256 |
|---|---|---|---|---|
| no switch | 0 | 2 | 6 | 6 |
| the cap | 0 | 1 | 7 | 6 |
| small steps | 0 | 1 | 7 | 6 |
| fast kernel | 0 | 4 | 6 | 6 |
| all three | 0 | 4 | 7 | 6 |

darker = more: long prompts sent in that minute

**Figure 12** Long prompts sent in each minute, by run. Read this first: rows that differ here did not carry the same load.
Source: each run's plan-mixed-overlap/requests.jsonl (stage, asked_in)

| | streams asked for 8 | 32 | 128 | 256 |
|---|---|---|---|---|
| no switch | 158.8 | 6.3 | 1.6 | 1.1 |
| the cap | 144.8 | 8.9 | 9.0 | 54.4 |
| small steps | 144.6 | 18.8 | 3.7 | 2.7 |
| fast kernel | 299.9 | 11.1 | 75.9 | 118.0 |
| all three | 298.0 | 122.7 | 282.0 | 256.2 |

darker = more: answer tokens a second (shaded by order of magnitude)

**Figure 13** Answer tokens a second in the same runs and minutes.
Source: cross/mixed.json (each run's summary.json)

| | streams asked for 8 | 32 | 128 | 256 |
|---|---|---|---|---|
| no switch | 19.91 | 0.24 | 0.05 | 0.03 |
| the cap | 18.21 | 0.33 | 0.22 | 0.57 |
| small steps | 18.19 | 0.76 | 0.12 | 0.09 |
| fast kernel | 37.64 | 0.47 | 0.98 | 0.88 |
| all three | 37.46 | 4.40 | 3.46 | 2.81 |

darker = more: engine steps a second (shaded by order of magnitude)

**Figure 14** Engine steps a second. Every running stream gets one token per step.
Source: cross/mixed.json

| THE ROW, AND WHAT DIFFERS FROM THIS RUN | ATTENTION KERNEL | REQUESTS THE PLAN SENT | LONG PROMPTS AMONG THEM | ENDED INSIDE THE PLAN |
|---|---|---|---|---|
| **no switch**: nothing: this run | ROCM_ATTN | 270 | 14 | 14 |
| **the cap**: long-prompt cap at 2,048 tokens | ROCM_ATTN | 269 | 14 | 14 |
| **small steps**: 2,048-token engine steps | ROCM_ATTN | 269 | 14 | 27 |
| **fast kernel**: fast attention kernel | ROCM_AITER_FA | 297 | 16 | 167 |
| **all three**: fast attention kernel, 2,048-token engine steps and long-prompt cap | ROCM_AITER_FA | 342 | 17 | 166 |

"What differs" is the cross file's own wording, against the settings of this run. "Ended inside the plan" is a fact about each run, not a comparison between rows: the rows were sent different requests.

MEASURED

Like for like at 32 streams, one long prompt each, default kernel: 8.9 answer tokens a second with the cap, 18.8 with small steps. Four long prompts each, fast kernel: 11.1 with steps of 8,192 tokens, 122.7 with small steps and the cap.

MEASURED

At 128 and 256 streams every run was sent six or seven and six long prompts in the minute; what each still carried from the minute before differs. There the three runs on the default kernel stay under one engine step a second, with or without a control. The two on the fast kernel make 0.88 to 3.46 steps a second.

KEEP IN MIND

Even the fast-kernel pair is disturbed at 32 streams. In 27 of that minute's seconds the fast-kernel run's engine process used no CPU: its log shows it building three attention kernels it had not used before (21.7, 3.8 and 3.7 s). The run with all three came later and found them built, so its lead in that minute is larger than the settings alone would give.

KEEP IN MIND

"All three" differs from "fast kernel" in two settings, step size and cap, so it cannot say which of the two does the work. The bench has a second plan for this comparison, with the same requests sent at the same moments to every server; its runs are the ones to trust for a ranking.

## While everyone waited, the card was nearly idle

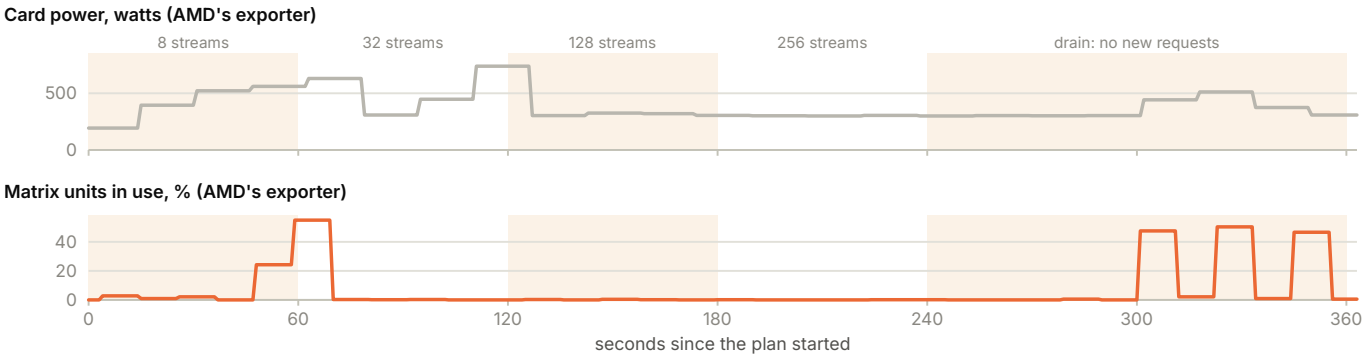



**Figure 15** The card underneath the same plan, as AMD's device exporter reports it (new readings every 11 to 16 s). The card works in the first minute, in short bursts while the first long prompt is read, and again when the plan drains. In between it draws little more than it does idle.
Source: plan-mixed-overlap/timeline.json (watts, matrix_units_percent)

MEASURED

At 128 and 256 streams the card drew 366 and 303 W, with its matrix units 0.2% and 0.1% in use. At the top of the staircase on this same server it drew 751 W with 50.0% of them in use.

MEASURED

The CPU is not what it waits for: the machine was 10.6% busy and the engine process used 2.0 cores, the same as with one short stream in the staircase (2.0).

### What I take from it

- ● With 5% long prompts in the mix this server falls from 158.8 answer tokens a second at 8 streams to 6.3 at 32 and 1.6 at 128.
- ● One long prompt at a time is enough. It costs time, not requests: nothing was refused, nothing failed.
- ◌ Every decoding stream gets one token per engine step, so a step that carries a long prompt's bite makes all of them wait.
- ◌ The slow steps are not heavy work for the card. I read them as the default attention kernel going through a long context narrowly. Not confirmed in the source.

### Not results, and limits

- Requests "cut at the end of the plan" were closed by the load generator: not refused, not failed.
- Prompts are random tokens: nothing repeats, and this says nothing about the quality of an answer.
- One plan, one seed, played once. Another seed would place the first long prompt elsewhere.

● 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.

**Capacity**
(waiting reason) the server's label for a request it cannot schedule now, because the cache or the step's token budget is used up. Reported in `vllm:num_requests_waiting_by_reason`.

Every number in this document was read from these files at build time (sha256, first 12 characters):
`FACTS.json` 160f5ca0d495 · `GLOSSARY.md` 98323be796af · `cross/mixed.json` 4db3ea4c9fb2 · `plans/mixed-overlap.json` b86c1f4ef044 · `runs/20261125T201346Z-series/plan-mixed-overlap/requests.jsonl` ec7778099e60 · `runs/20261125T201346Z-series/plan-mixed-overlap/t0` f79227400789 · `runs/20261125T201346Z-series/plan-mixed-overlap/timeline.json` 328dd803edea · `runs/20261125T201346Z-series/summary.json` aab3940eb12e · `runs/20261125T213149Z-series-longprefill2k/plan-mixed-overlap/requests.jsonl` 774120396827 · `runs/20261125T213149Z-series-longprefill2k/plan-mixed-overlap/t0` c945be467c45 · `runs/20261125T213149Z-series-longprefill2k/summary.json` 110822c67acb · `runs/20261125T214036Z-series-batch2k/plan-mixed-overlap/requests.jsonl` d80dc5271519 · `runs/20261125T214036Z-series-batch2k/plan-mixed-overlap/t0` fffb5e8db961 · `runs/20261125T214036Z-series-batch2k/summary.json` 3d4fa8438e60 · `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/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/summary.json` 35fa706c2022

118 values bound by path, 15 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