On the fast kernel a long prompt's bite takes one second, and the mix gets through

The mixed load with one setting changed: the attention kernel. While long prompts are read the engine makes a step a second, where the default kernel needed twenty seconds and more. All 16 long prompts are read. Decoding is still starved meanwhile.

Qwen3.8-27B · one AMD Instinct MI300X · vLLM 0.27.1 **Run** 20261125T215049Z-series-fa
Changed the attention kernel: ROCM_AITER_FA instead of ROCM_ATTN, against the series run
Load 70% short, 25% medium, 5% long prompts; 8, 32, 128, 256 streams

The same mix of short, medium and long prompts, with one setting changed: the attention kernel. On the default kernel a step that carried a bite of a long prompt took 20 to 33 seconds at 128 and 256 streams. On AMD's flash-attention kernel the engine makes **0.98 steps a second at 128 streams**: a second a step. The server read every long prompt it was sent, 16 of them, and no request was left without a first token.

0.98
engine steps a second at 128 streams, 0.88 at 256; the default kernel made 0.05 and 0.03

8,696
prompt tokens read a second at 128 streams

167 of 297
requests ended inside the plan; none was cut before its first token

118.0
answer tokens a second at 256 streams, with 174 requests running at most

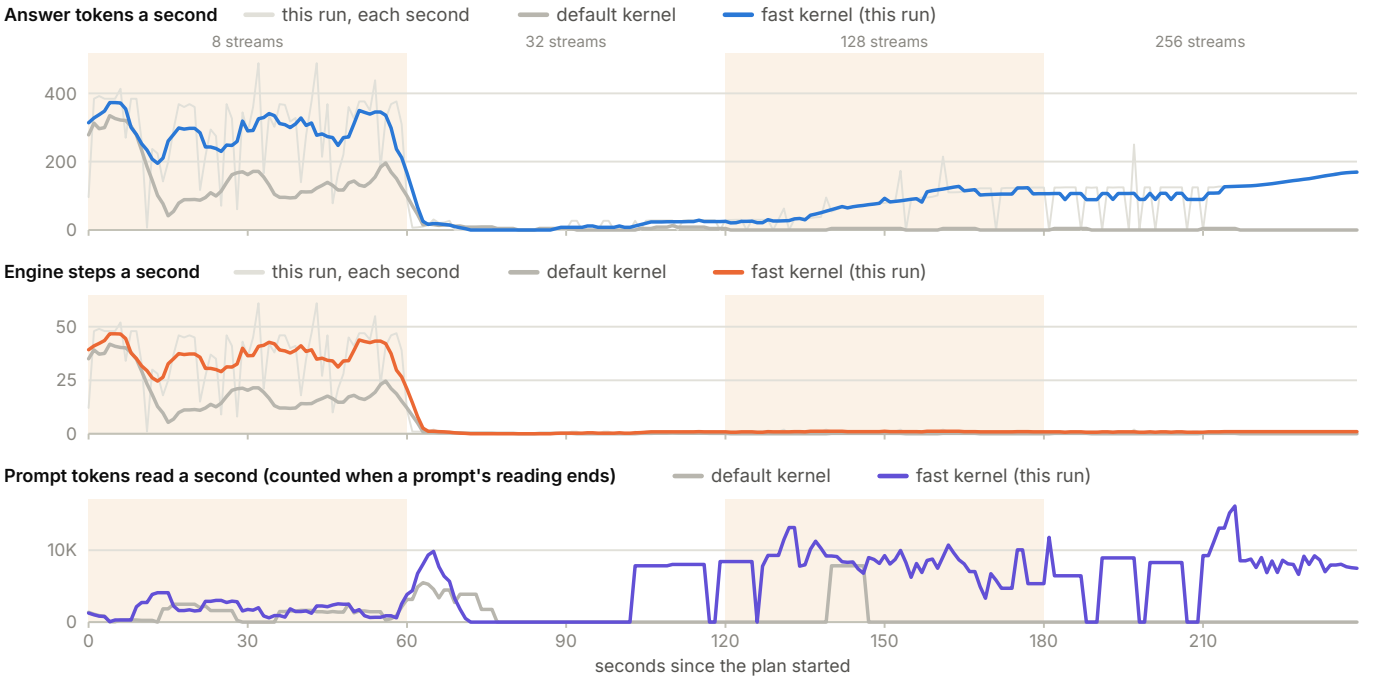


Figure 1 The four loaded minutes. Colour: this run, with its single seconds thin and grey behind the 7-second centred mean. Grey line: the run on the default kernel. In the first minute, with no long prompt in the mix, the fast kernel already makes almost twice the steps. In the minute at 32 streams this run's engine stood still for kernel builds.
Source: plan-mixed-overlap/timeline.json of this run and of 20261125T201346Z-series

MEASURED
At 128 and 256 streams the server wrote 75.9 and 118.0 answer tokens a second and read 8,696 and 7,266 prompt tokens a second. On the default kernel those two minutes gave 1.6 and 1.1 answer tokens a second.

KEEP IN MIND
One setting differs from the grey run, but the two were not sent the same requests: a closed loop sends a faster server more. At 8 streams neither was sent a long prompt, at 128 and 256 both were sent six. At 32 streams this run was sent four and the other two: not like for like, and I write no ratio for it.

The machine, and the part that was changed

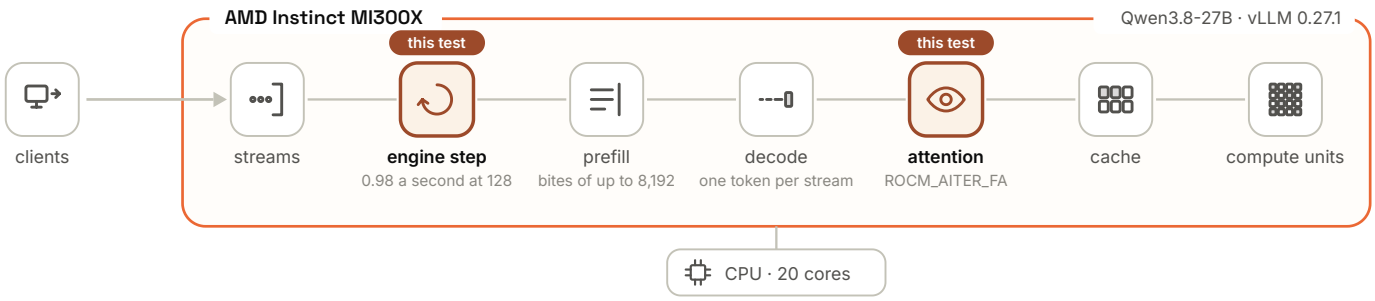


Figure 2 The card as every document in this series draws it. Attention is the code that lets each token look back over the context. This run sets it by hand to AMD's flash-attention kernel; everything else is the series run's.
Source: FACTS.json; summary.json server.attention_kernel and plans.plan-mixed-overlap.stages[stage=mix-128]

What differs between the two runs

Attention kernel	ROCM_AITER_FA here, by the --attention-backend setting; ROCM_ATTN in the run before, by vLLM's own preference order
Room in one engine step	8,192 tokens in both
Cache	1,734,958 token slots here; 1,826,476 before
The run before	20261125T201346Z-series

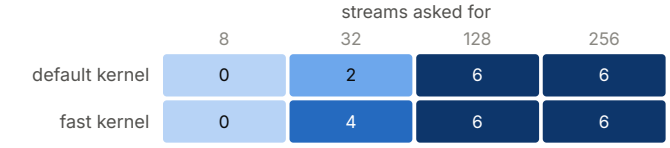
KEEP IN MIND
The fast kernel's server starts with 5.0% fewer cache slots. In this plan the cache fills only in the drain.

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	20261125T215049Z-series-fa, started 2026-11-25T21:51:02Z, server ready after 97 s



darker = more: long prompts sent in that minute

Figure 3 Long prompts sent in each minute, in both runs. Where the rows agree the two runs carried a like load in that minute; at 32 streams they do not.
Source: plan-mixed-overlap/requests.jsonl of both runs (stage, asked_in)

KEEP IN MIND
The two runs were not sent the same requests: 297 here, 270 before. Request number N has the same lengths in both, and a faster server gets further down the list. How many requests ended inside the plan is a fact about each run, not a ratio between them.

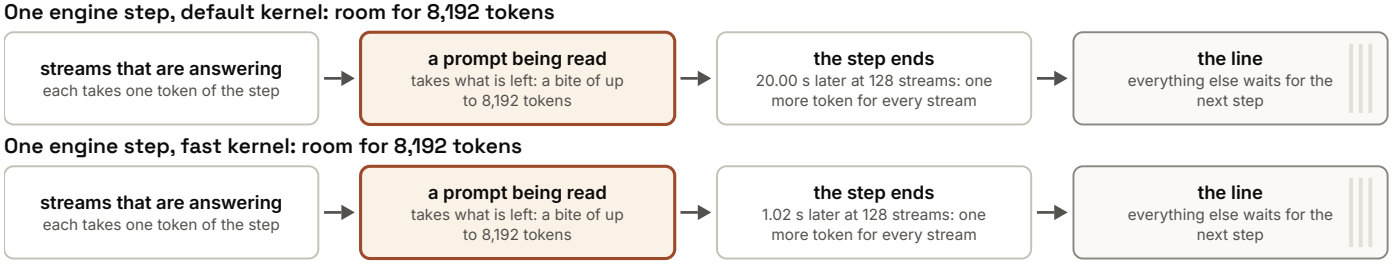


Figure 4 What one engine step holds on the two kernels: the same room, a very different time. The time is computed: one second divided by the engine steps a second at 128 streams.
Source: each run's summary.json (config, plans.plan-mixed-overlap.stages[stage=mix-128].engine_steps_per_s); FACTS.json

Minute by minute, on the two kernels

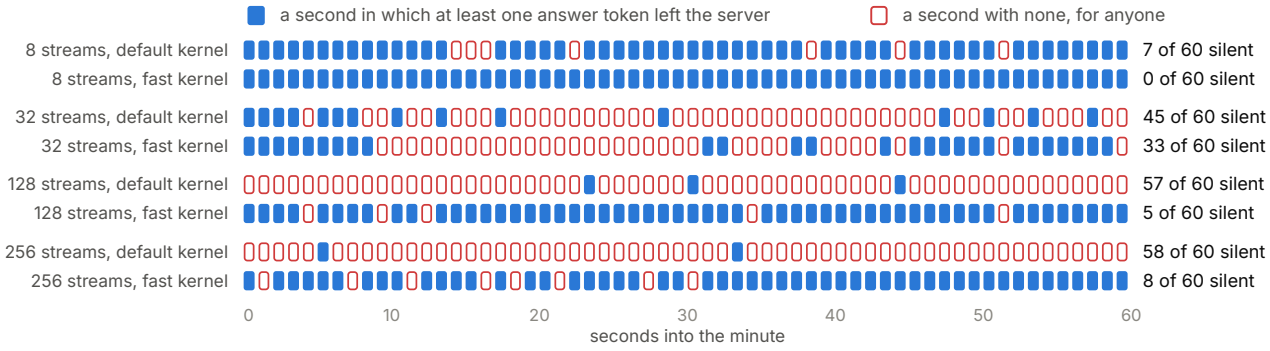


Figure 5 Every second of the four loaded minutes in both runs. The two rows at 32 streams are not like for like: this run was sent four long prompts and the other two, and this run's silent seconds there are the kernel builds and little else. At 128 and 256 streams hardly a second is silent on the fast kernel.
Source: plan-mixed-overlap/timeline.json of both runs (output_tok_s per one-second row)

Answer tokens a second, all streams together

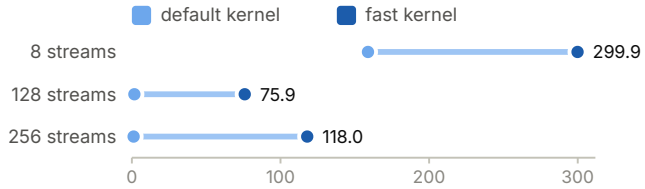


Figure 6 Answer tokens a second in the three minutes in which both runs were sent as many long prompts.
Source: both runs' summary.json stages[].server_output_tok_s

Engine steps a second (log scale)

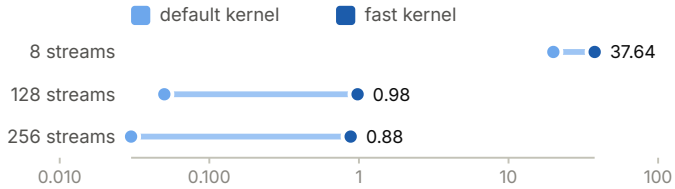


Figure 7 Engine steps a second in the same minutes. Every running stream gets one token per step.
Source: both runs' summary.json stages[].engine_steps_per_s

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										
default kernel	158.8	1,066	19.91	7 of 60	8	0	3.7	0.69	21	413
fast kernel	299.9	1,701	37.64	0 of 60	8	0	3.9	0.26	39	695
32 STREAMS										
default kernel	6.3	1,104	0.24	45 of 60	32	18	13.5	5.57	25	505
fast kernel	11.1	3,095	0.47	33 of 60	30	22	19.0	6.77	25	316
128 STREAMS										
default kernel	1.6	943	0.05	57 of 60	32	96	14.9	≥ 211.93	96	366
fast kernel	75.9	8,696	0.98	5 of 60	125	97	61.3	31.30	96	744
256 STREAMS										
default kernel	1.1	0	0.03	58 of 60	32	224	15.8	≥ 179.91	128	303
fast kernel	118.0	7,266	0.88	8 of 60	174	131	91.8	65.19	137	746
DRAIN: NO NEW REQUESTS										
default kernel	14.2	2,204	0.24	93 of 121	96	224	37.8	-	0	359
fast kernel	744.8	4,527	4.13	9 of 121	207	74	100.0	-	0	750

From each run's `summary.json`; seconds with no answer token are counted from its `timeline.json`. Tinted rows are this run. The two rows at 32 streams are not like for like (four long prompts against two, and this run's kernel builds). 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 same bite in a second instead of twenty, and still one token per step

Seconds per engine step: default kernel

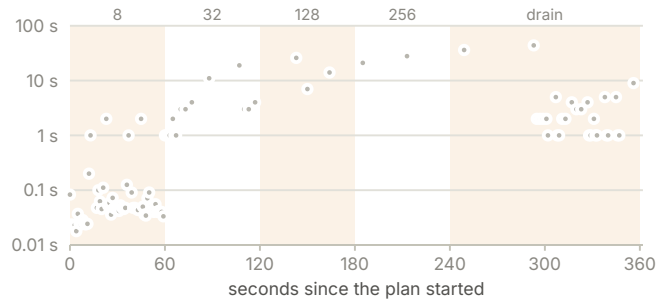


Figure 8 Default kernel: from the moment a long prompt is read, a step takes seconds, then tens of seconds.
Source: timeline.json of 20261125T201346Z-series (steps_s)

Seconds per engine step: fast kernel

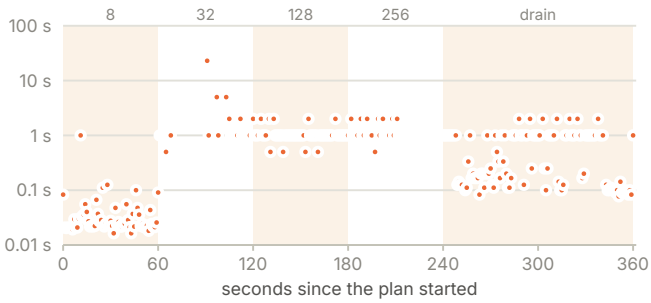


Figure 9 Fast kernel: a step takes about a second while prompts are read, less in the drain. Computed from the step counter: one second divided by the steps in it, or the seconds since the last step.
Source: plan-mixed-overlap/timeline.json (steps_s)

MEASURED

In the two minutes at 128 and 256 streams the engine made 59 and 53 steps. The default kernel made 3 and 2.

ESTIMATE

vLLM's estimate of the compute at 256 streams is 550 TFLOPS, 42.3% of AMD's stated peak, with the card at 746 W. On the default kernel the same minute read 26.6 TFLOPS at 303 W.

READING, NOT MEASUREMENT

My reading: decoding is still starved above 8 streams. Each engine step carries a bite of up to 8,192 prompt tokens that takes about a second even on this kernel, and every decoding stream gets one token per step. With 125 requests running, 75.9 answer tokens a second is less than one token a second each.

KEEP IN MIND

The heading's "twenty" is one second divided by the default kernel's steps a second at 128 streams: 20 s. Over the two minutes at 128 and 256 streams the fast kernel made 22.3 times the steps. The two servers were sent as many long prompts in those minutes, but did not carry the same backlog into them.

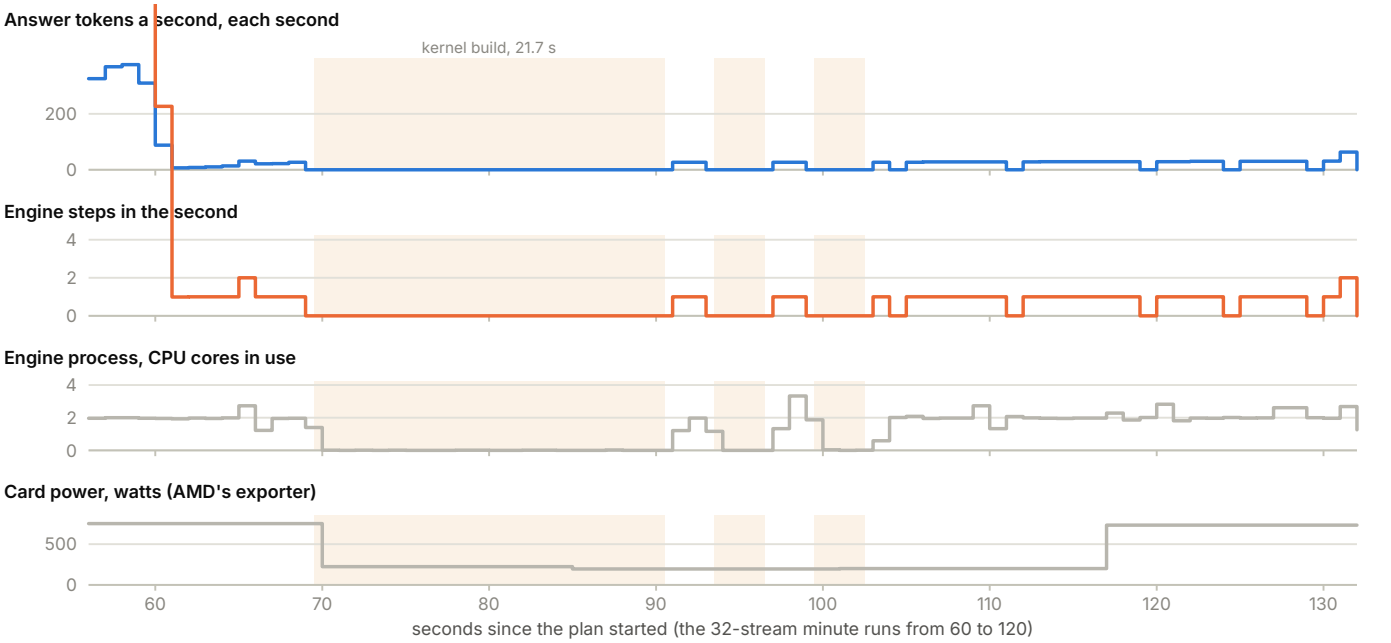


Figure 10 This run from second 56 to 132, without smoothing. Shaded: the seconds in which the engine process used no CPU. The card fell to idle power in the same seconds, and the server's log reports a kernel build of that length for each: attention kernels built on first use.
Source: plan-mixed-overlap/timeline.json; server.log ("[aiter] finish build ... cost")

NOT A RESULT

The 11.1 answer tokens a second of the 32-stream minute are not what this setting gives: in 27 of its seconds the engine was building kernels (21.7, 3.8 and 3.7 s by its log), a cost paid once. In the seconds it worked it wrote 22.2 a second (computed).

What the clients saw

Every request: default kernel

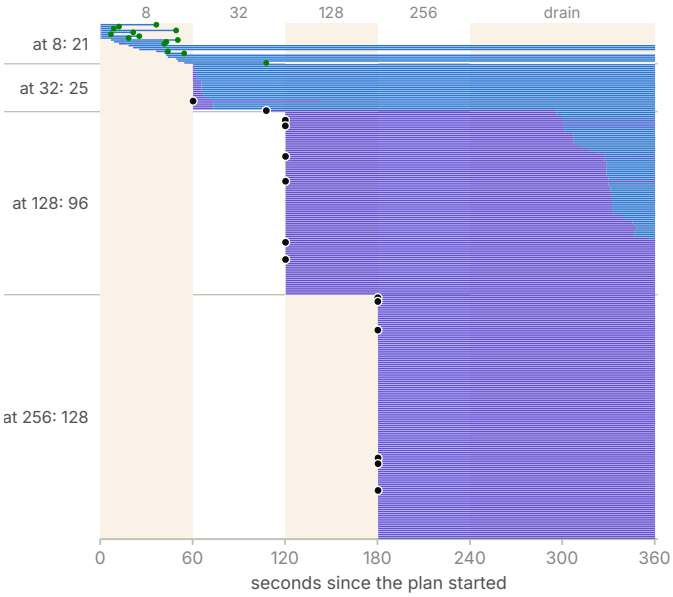


Figure 11 Every request of the run on the default kernel as one line, in the order sent. Violet: no answer token yet. Blue: answering. Green dot: ended well. Black dot: a long prompt.
Source: requests.jsonl of 20261125T201346Z-series

Every request: fast kernel

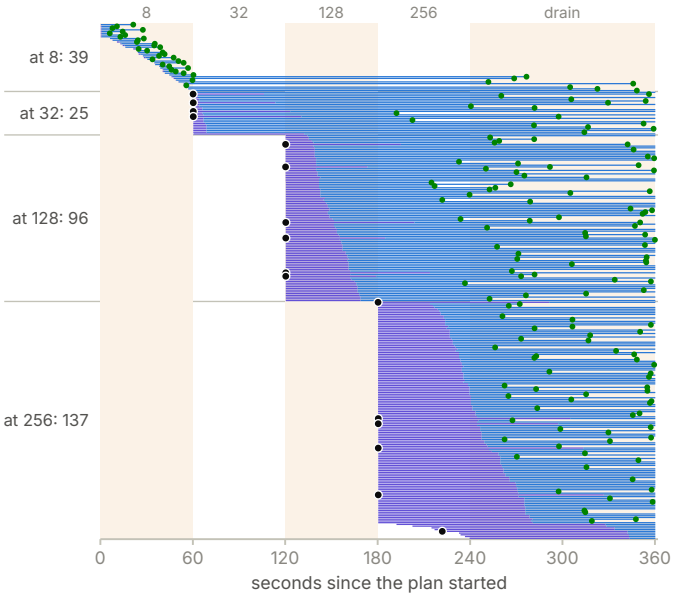


Figure 12 The same for this run. Every request gets its first token, and lines end in green all the way down. Two pictures of two runs, not a ratio: the runs were sent different requests.
Source: plan-mixed-overlap/requests.jsonl (t_sent, t_first, t_end, end)

	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							
default kernel	21	0	0.7	5.0	14	7	0
fast kernel	39	0	0.3	1.7	39	0	0
SENT AT 32 STREAMS							
default kernel	25	2	5.6	188.3	0	25	0
fast kernel	25	4	6.8	71.9	15	10	0
SENT AT 128 STREAMS							
default kernel	96	6	209.8	236.0	0	63	33
fast kernel	96	6	31.3	93.6	59	37	0
SENT AT 256 STREAMS							
default kernel	128	6	-	-	0	0	128
fast kernel	137	6	65.2	146.4	54	83	0
ALL REQUESTS OF THE PLAN							
default kernel	270	14	187.2	236.0	14	95	161
fast kernel	297	16	43.6	146.4	167	130	0

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" is a fact about each run: the two were sent different requests.

MEASURED

Every one of the 297 requests got a first token. The 16 long prompts waited 46 to 146 s for theirs, and 7 of them ended inside the plan. A short prompt sent at 128 streams waited 15 to 49 s.

NOT A RESULT

Requests "cut at the end of the plan" (130 here) are not refusals and not failures: the load generator closed them when the plan's time was over, all of them in the middle of an answer. All 297 requests got HTTP 200.

A busy card, and in the drain a full cache

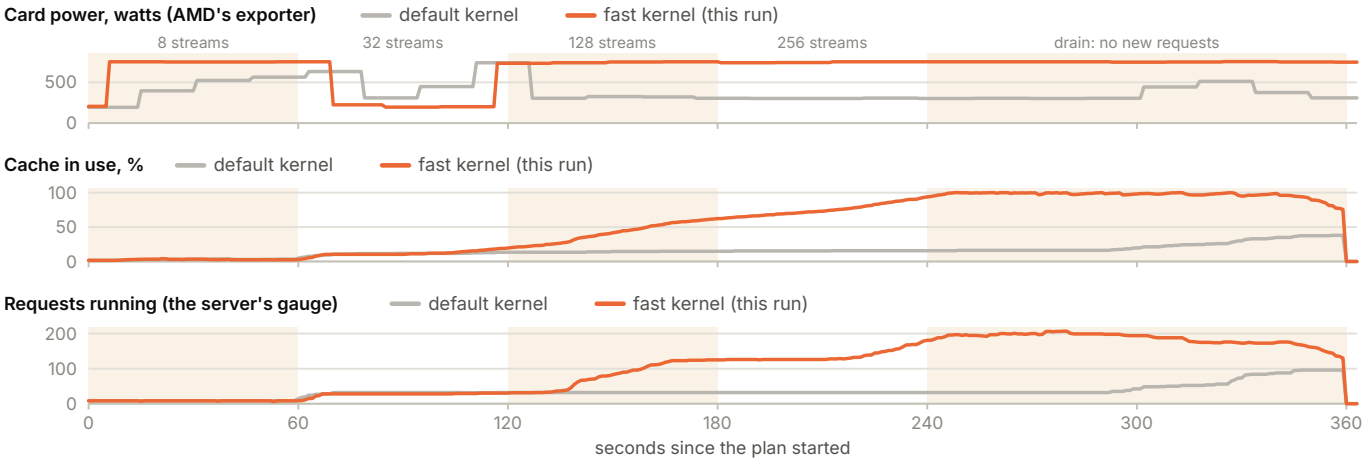


Figure 13 Both runs over the whole plan. Grey: default kernel. Colour: this run. The card draws full power from the first minute on, except during the kernel builds. With prompts read this fast the cache fills, and in the drain it is full.
Source: plan-mixed-overlap/timeline.json of both runs (watts, cache_percent, running)

MEASURED

In the drain the cache reached 100.0% with 207 requests running at most, and the server preempted 4 of them. With no new prompts to read, it wrote 744.8 answer tokens a second over those two minutes.

MEASURED

At 128 and 256 streams the card drew 744 and 746 W. On the default kernel it drew 366 and 303 W in those minutes, little more than idle.

What I take from it

● On this kernel a step that carries a long prompt's bite takes about a second. Every request gets a first token, and 167 of 297 end inside the plan.

◌ Decoding is still starved while prompts are read: one token per step, a step a second.

◌ The default kernel's slow steps were not heavy work for the card; this kernel puts the card to work.

Not results, and limits

- The 32-stream minute: kernel builds, and four long prompts where the other run was sent two.
- A closed loop: no ratio of requests ended against another run; the bench's fixed-arrival plan gives that.
- TFLOPS is vLLM's estimate, too high where long prompts are read: attention is counted in all 64 layers and the model has it in 16. The peak is AMD's stated figure.
- 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

Attention kernel
(`--attention-backend`) the GPU code that lets each token look back over the context. Four exist here: `ROCM_ATTN` (the default pick), `ROCM_AITER_FA` (AITER flash attention), `ROCM_AITER_UNIFIED_ATTN`, `TRITON_ATTN`.

Preempted
a running stream thrown out of a full cache and put back in line; its work is redone later.

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/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/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

94 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