AITER unified attention is second of the four kernels

The four lower rungs of the context ladder on AMD's unified attention kernel, chosen by hand. At 128,000 tokens of context it writes 12.7 times as fast as the kernel vLLM picks by itself; AMD's flash-attention kernel is faster still, most of all in reading.

Qwen3.8-27B · one AMD Instinct MI300X · vLLM 0.27.1 **Run** 20261125T211511Z-series-attn-rocm-aiter-unified-attn
Changed the attention kernel: ROCM_AITER_UNIFIED_ATTN chosen by hand, with AITER switched on
Load 4,000 to 128,000 tokens in, 64 out, one request at a time, after a warm-up

The context ladder showed the kernel vLLM picks by itself collapsing as the prompt grows. I replayed its four lower rungs on each of the three other attention kernels this install offers, one server per kernel. This is the one for AMD's unified attention kernel, `ROCM_AITER_UNIFIED_ATTN`. At 128,000 tokens of context it writes **12.7 times** as fast as the default, and it is still only second of the four: AMD's flash-attention kernel writes faster and reads the prompt in about half the time.

46.7
answer tokens a second at 128,000 tokens of context; 3.7 on the default kernel

12.7 times
as fast in writing as the default at 128,000 tokens, from the two servers' writing seconds

46 s
before the first token at 128,000 tokens; 76 s on the default kernel

2,799
prompt tokens read a second at 128,000, averaged over the prompt; 1,686 on the default kernel

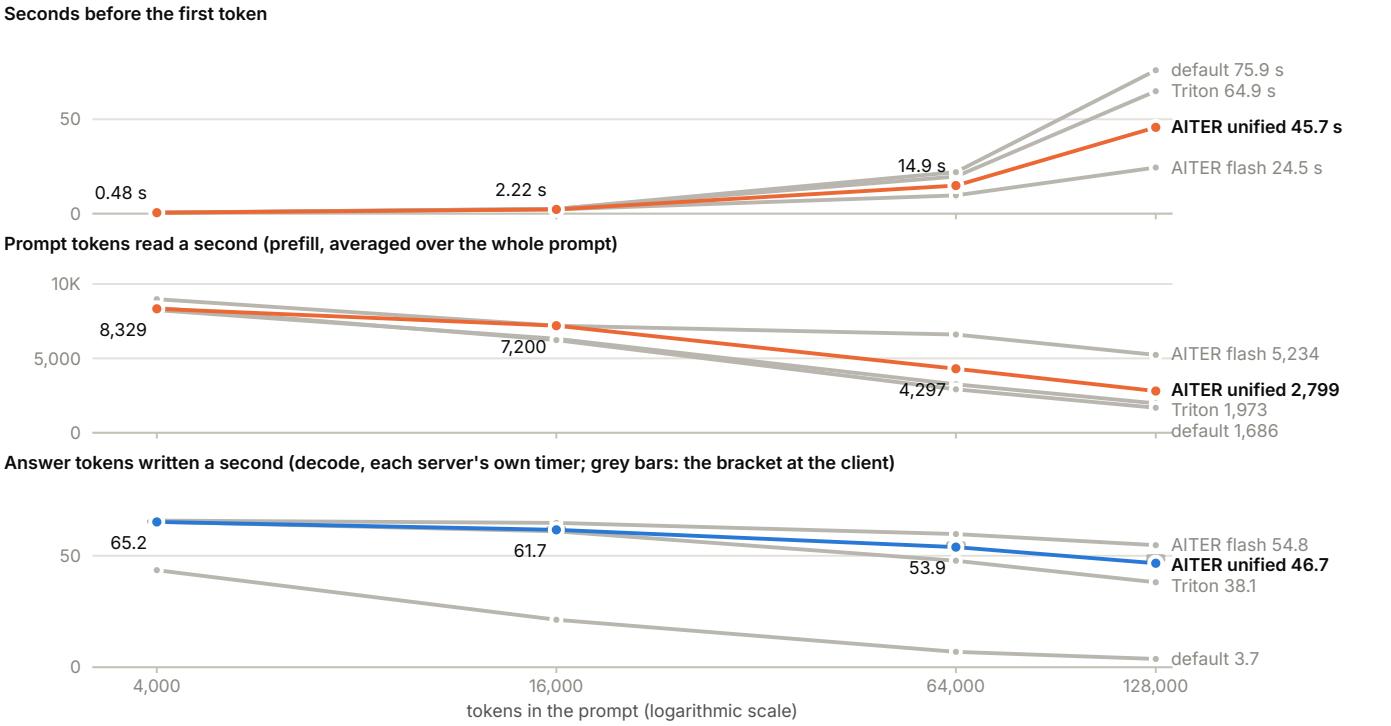


Figure 1 The four rungs on the four kernels. This document's kernel is in colour with its values; the other three are grey, each named at the end of its line with its value at 128,000 tokens. Top: the wait for the first token. Middle: how fast the prompt was read, averaged over the whole prompt. Bottom: how fast the answer was written; the grey bars are the two figures the client can compute, shown only as a bracket.
Source: cross/ladder.json: each run's summary.json single_requests[] (first_token_s, prefill_tok_s, decode_tok_s_server)

MEASURED
Second of the four in writing from 16,000 tokens up: 46.7 answer tokens a second at 128,000, against 54.8 on AITER flash and 38.1 on Triton. At 4,000 tokens the three chosen kernels are within a token a second of each other.

COMPUTED
From 4,000 to 128,000 tokens this kernel keeps 71.6% of its writing speed and 33.6% of its reading speed. The default kernel keeps 8.4% and 20.0%.

The machine, and the one part that was changed

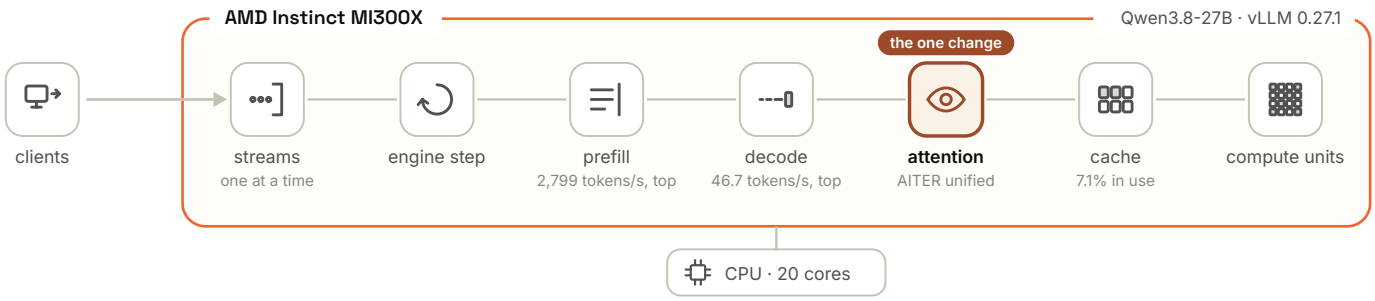


Figure 2 The serving settings are those of the context ladder; the attention kernel is the one change. "Top" is the 128,000-token rung.
Source: FACTS.json; summary.json server.attention_kernel and plans.plan-context-ladder-short

Four servers, one for each kernel

SERVER	ATTENTION KERNEL	CHOSEN	ENVIRONMENT SET	CACHE SLOTS	START-UP GRAPHS	READY, S	COMPILING, S
AITER flash	`ROCM_AITER_FA`	by hand	`VLLM_ROCM_USE_AITER=1`	1,723,325	12.90 GiB	206	37.3
AITER unified	`ROCM_AITER_UNIFIED_ATTN`	by hand	`VLLM_ROCM_USE_AITER=1`	1,812,820	7.42 GiB	194	36.8
Triton	`TRITON_ATTN`	by hand	`VLLM_ROCM_USE_AITER=1`	1,816,394	6.83 GiB	185	37.0
default	`ROCM_ATTN`	by vLLM	none	1,826,476	6.83 GiB	94	4.9

KEEP IN MIND

Like for like. The three servers with a kernel chosen by hand differ in `--attention-backend` only; all three have AMD's AITER library switched on. Against the default two things differ: the kernel, and AITER being on. The default's rungs come from its full ladder, which has the same seed.

READING, NOT MEASUREMENT

The slower starts are not a cost of a kernel. My reading: vLLM's compile cache did not hold these three configurations' graphs yet, so each compiled for about 37 s where the default's server needed 4.9 s.

1. A warm-up request of 1,000 tokens goes first. It is not a rung.
2. Then four requests, one at a time: 4,000, 16,000, 64,000 and 128,000 random token ids in, 64 tokens forced back. The next is sent only when the last token of the answer before has arrived.
3. Beside it a sampler reads the server's counters, the card and the CPU once a second.

Prompt plus answer of each request, against the model's limit

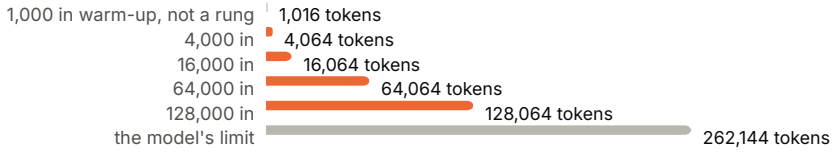


Figure 3 The plan against the model's limit: the four rungs reach half of it.
Source: plans/context-ladder-short.json; summary.json server.max_model_len

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	20261125T211511Z-series-attn-rocm-aiter-unified-attn, started 2026-11-25T21:15:24Z, server ready after 194 s

KEEP IN MIND

The load is made on the same machine as the server and shares its CPU cores. It talks to the server on the loopback address: no request leaves the machine.

Prompt length by kernel: the whole grid

Answer tokens a second, by kernel and prompt length

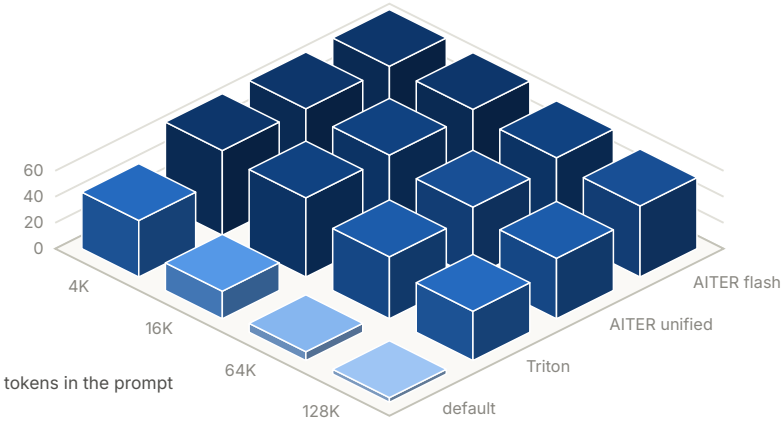


Figure 4 Writing speed on the four attention kernels at the four rungs. A view only: the exact values are in the first matrix beside it. The default kernel is the front row.
Source: cross/ladder.json decode_tok_s_server (each run's summary.json)

	tokens in the prompt 4K	16K	64K	128K
AITER flash	65.8	64.8	59.8	54.8
AITER unified	65.2	61.7	53.9	46.7
Triton	65.3	61.0	47.8	38.1
default	43.6	21.3	6.9	3.7

darker = more: answer tokens a second

Figure 5 Writing: the same values as the view, exactly.
Source: cross/ladder.json decode_tok_s_server

	tokens in the prompt 4K	16K	64K	128K
AITER flash	8,979	7,194	6,604	5,234
AITER unified	8,329	7,200	4,297	2,799
Triton	8,239	6,312	3,257	1,973
default	8,393	6,232	2,914	1,686

darker = more: prompt tokens read a second

Figure 6 Reading, averaged over each prompt.
Source: cross/ladder.json prefill_tok_s

	tokens in the prompt 4K	16K	64K	128K
AITER flash	0.45	2.22	9.69	24.5
AITER unified	0.48	2.22	14.9	45.7
Triton	0.49	2.54	19.7	64.9
default	0.48	2.57	22.0	75.9

darker = more: seconds before the first token (shaded by order of magnitude)

Figure 7 The wait for the first token, in seconds.
Source: cross/ladder.json first_token_s

	tokens in the prompt 4K	16K	64K	128K
AITER flash	1.5×	3.0×	8.6×	14.9×
AITER unified	1.4×	2.8×	7.7×	12.7×
Triton	1.4×	2.8×	6.8×	10.3×

darker = more: times as fast in writing as the default kernel

Figure 8 Writing against the default kernel: the default's writing seconds divided by each kernel's, cut to one decimal.
Source: computed from cross/ladder.json server_decode_s

The exact record of this run

PROMPT TOKENS	FIRST TOKEN, S	READING, S (SERVER)	PROMPT TOKENS/S	WRITING, S (SERVER)	ANSWER TOKENS/S (SERVER)	BRACKET AT THE CLIENT	MS PER ANSWER TOKEN	WRITING, AGAINST THE DEFAULT	CACHE IN USE, %
4,000	0.48	0.47	8,329	0.966	65.2	65.4 to 65.4	15.3	1.4×	-
16,000	2.22	2.20	7,200	1.021	61.7	62.4 to 62.4	16.2	2.8×	1.0
64,000	14.89	14.84	4,297	1.170	53.9	54.8 to 55.6	18.6	7.7×	3.6
128,000	45.72	45.61	2,799	1.348	46.7	48.2 to 49.8	21.4	12.7×	7.1
NOT A RUNG: THE WARM-UP REQUEST, 16 TOKENS ASKED BACK									
1,000	0.93	0.92	1,079	0.228	65.9	49.1 to 67.0	15.2	-	-

One row per request, from `summary.json` and `cross/ladder.json`. First token: at the client, from sending to the first text chunk. Reading and writing seconds: the server's own prefill and decode timers for that request. Answer tokens a second: the answer's tokens less one, divided by the writing seconds. The bracket: a token that carries no visible text sends no chunk, and these answers are random tokens, so counting chunks gives too low a speed and spreading all tokens over the time from the first chunk to the last too high a one; the server's own timer can sit just below it, because tokens without text at either end of an answer shorten the client's window.

Why it is second: a look back twice as dear as the fastest

Milliseconds for one answer token

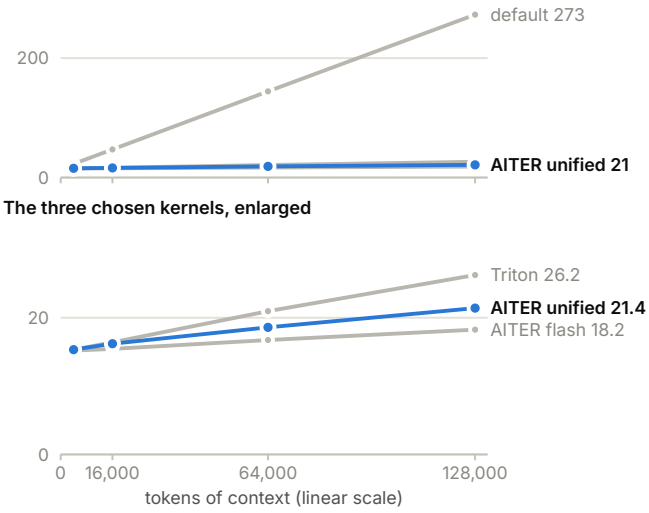


Figure 9 Writing. The time for one answer token against the context, on plain scales. Top: the four kernels on one scale. Bottom: the three chosen kernels, enlarged.
Source: computed from cross/ladder.json server_decode_s (each run's summary.json)

Seconds for one bite

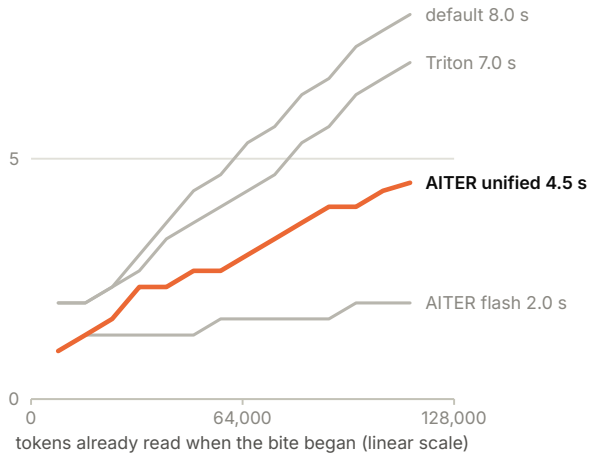


Figure 10 Reading. The seconds each full bite of the 128,000-token prompt took, against what had been read before it. The record has whole seconds, so each line is a centred mean over three bites.
Source: each run's timeline.json: the seconds between two moves of the server's work counter (est_tflops), which moves once per bite

MEASURED

The 128,000-token prompt went in as 16 bites of at most 8,192 tokens. On this kernel the first three took 1.3 s each on average and the last three full ones 4.3 s; on the default kernel the same bites took 2.0 s and 7.7 s.

COMPUTED

One answer token took 15.3 ms at 4,000 tokens of context and 21.4 ms at 128,000: 0.049 ms more for every 1,000 tokens of context. On the default kernel that figure is 2.01 ms, 41 times as much.

Added to one answer token by 1,000 tokens of context

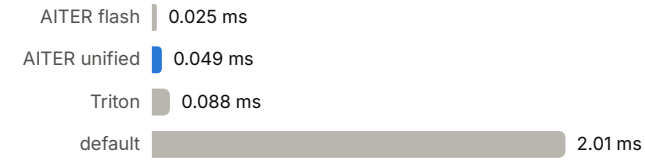


Figure 11 Writing: what 1,000 more tokens of context add to the time for one answer token, on each kernel.
Source: computed from cross/ladder.json server_decode_s: the rise from the 4,000-token rung to the 128,000-token rung

Reading speed for the tokens from 64,000 to 128,000

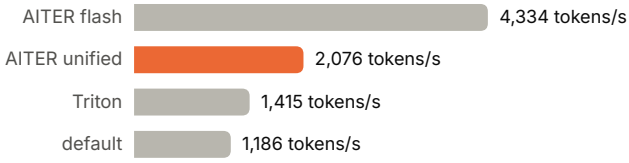


Figure 12 Reading: how fast the second half of the longest prompt went in, on each kernel.
Source: computed from cross/ladder.json first_token_s: tokens added over the rung before, divided by seconds added

One prompt, four kernels, second by second

Cache slots in use, thousands

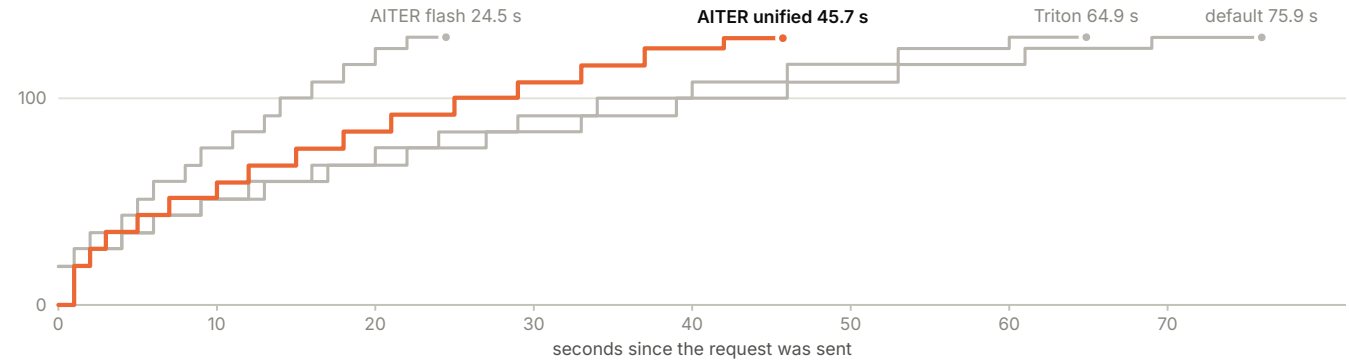


Figure 13 The 128,000-token prompt going into the cache on the four servers, from the second it was sent to its first token. The same prompt, token for token, on each. The steps are the bites; where a line ends, that server's first answer token left.
Source: each run's timeline.json (vllm:kv_cache_usage_perc every second, times its summary.json server.kv_cache_tokens) and its single_requests[stage=in-128000].first_token_s

SERVER	THE RUNG TOOK, S	TFLOPS, VLLM'S ESTIMATE	% OF AMD'S STATED PEAK	ESTIMATED WORK, QUADRILLION OPERATIONS	CPU BUSY, %	ENGINE CORES
AITER flash	25	784	60.3	19.6	10.7	2.03
AITER unified	46	429	32.9	19.7	10.6	2.00
Triton	66	290	22.2	19.1	10.7	2.01
default	92	207	15.9	19.1	10.6	2.00

The 128,000-token rung on each server. TFLOPS is vLLM's estimate from the model's shape over the whole request, not a hardware counter, and the peak it is divided by is AMD's stated figure. The estimated work is about the same on every kernel; only the time differs.

READING, NOT MEASUREMENT

vLLM's estimate counts too much attention for this model. What it adds for every bite already read is attention over the context in 64 layers; the model's configuration has full attention in 16 of its 64. At 128,000 tokens 67% of the estimated work is that attention. Counted for 16 layers, the same arithmetic gives 205 TFLOPS on this kernel, 15.7% of AMD's stated peak. My reading, not vLLM's figure.

READING, NOT MEASUREMENT

All four kernels pay for the look back over the context; they differ in how much. This one pays about twice what AITER flash pays for every token of context in writing, and its bites of the prompt take about twice as long at the end. That is enough to hold writing speed and not enough to hold reading speed.

MEASURED

Against the fastest kernel at 128,000 tokens: the first token comes after 46 s here and after 24 s on AITER flash; writing runs at 46.7 against 54.8 tokens a second. The two servers differ in the kernel only.

What I take from it

● With AITER unified attention chosen by hand, a request of 128,000 tokens waits 46 s for its first token and is then written at 46.7 tokens a second.

○ That is 12.7 times the default kernel's writing speed and 1.6 times as soon to the first token.

● It is second of the four in writing from 16,000 tokens up; AITER flash is ahead, by 17.3% at 128,000 tokens, and reads that prompt in about half the time.

◌ A large gain over the default, but AITER flash gives more on every count that grows with the context. On these four rungs I see none where this kernel is the one to choose.

Not results, and limits

NOT A RESULT

The 1,000-token request at the start is a warm-up, not a rung.

- Speed and behaviour only: random-token prompts and answers of 64 forced tokens say nothing about the quality of an answer.
- One request at a time on an otherwise idle server, each rung once. Two starts of one configuration differed by up to about 2% on these rungs.
- Four rungs, up to half the model's context.

● 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`.

AITER
AMD's library of GPU kernels for inference. Off by default in this install (`VLLM_ROCM_USE_AITER`).

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

Decode
the server writing the answer, one token per stream per engine step.

First token
(time to first token) from sending the request to receiving the first answer token. It contains waiting in line and prefill.

Context
all tokens a request holds: prompt plus answer so far. This model's native limit is 262,144.

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.

TFLOPS
trillions of arithmetic operations per second. Here always vLLM's estimate from the model's shape (`--enable-mfu-metrics`), not a hardware counter. It counts attention in all 64 layers where this model has it in 16: 1.7% high for short prompts, over half too high from 128,000 tokens of context.

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 · `cross/ladder.json` 3511faec1269 · `facts/box/20261125T192138Z/model-config.json` 191e0af23210 · `plans/context-ladder-short.json` b6a8613ab993 · `runs/20261125T201346Z-series/plan-context-ladder/timeline.json` 7ee787305964 · `runs/20261125T201346Z-series/summary.json` aab3940eb12e · `runs/20261125T211511Z-series-attn-rocm-aiter-unified-attn/plan-context-ladder-short/timeline.json` 49ed1e646107 · `runs/20261125T211511Z-series-attn-rocm-aiter-unified-attn/summary.json` 12b96900e2e1 · `runs/20261125T212044Z-series-attn-triton-attn/plan-context-ladder-short/timeline.json` f30fdb834d79 · `runs/20261125T212044Z-series-attn-triton-attn/summary.json` 6ecc28a27410 · `runs/20261125T212632Z-series-attn-rocm-aiter-fa/plan-context-ladder-short/timeline.json` 66856580a6bb · `runs/20261125T212632Z-series-attn-rocm-aiter-fa/summary.json` 0272c72e9f27

42 values bound by path, 13 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