QWEN3.8-27B · DRAFT HEAD LEVEL SWEEPS, ONE SETTING CHANGED

The draft head: 71.4% faster for one stream, 30.7% slower among 32

The model guesses three tokens ahead and keeps the guesses that match. One stream gets 94.8 answer tokens a second instead of 55.3. 32 streams on the same random prompts get 516.2 instead of 745.9, and real conversations show the same shape, smaller.

Qwen3.8-27B · one AMD Instinct MI300X · vLLM 0.27.1 **Run** 20261125T200505Z-mtp

Changed the model's draft head, guessing 3 tokens ahead **Load** random prompts at 1 and 32 streams; real conversations at 1, 8 and 32

This model carries a small draft head that guesses the next 3 tokens; the full model checks the guesses in its next step and keeps the ones that match. I switched it on and played two loads, each of which another run had played without it. For one stream it is a clear gain: 94.8 answer tokens a second against 55.3 on random prompts, 71.4% more. For 32 streams on the same prompts it is a loss: 516.2 against 745.9, 30.7% less. Real conversations show the same shape, smaller.

71.4% more
for one stream on random prompts
94.8 against 55.3 answer tokens a second

30.7% less
for 32 streams on the same prompts
516.2 against 745.9

53.7% more
for one stream on real conversations
32.3% more for 8, 7.7% less for 32

46.7%
of the guesses kept in real conversations, one stream
61.4% on random prompts

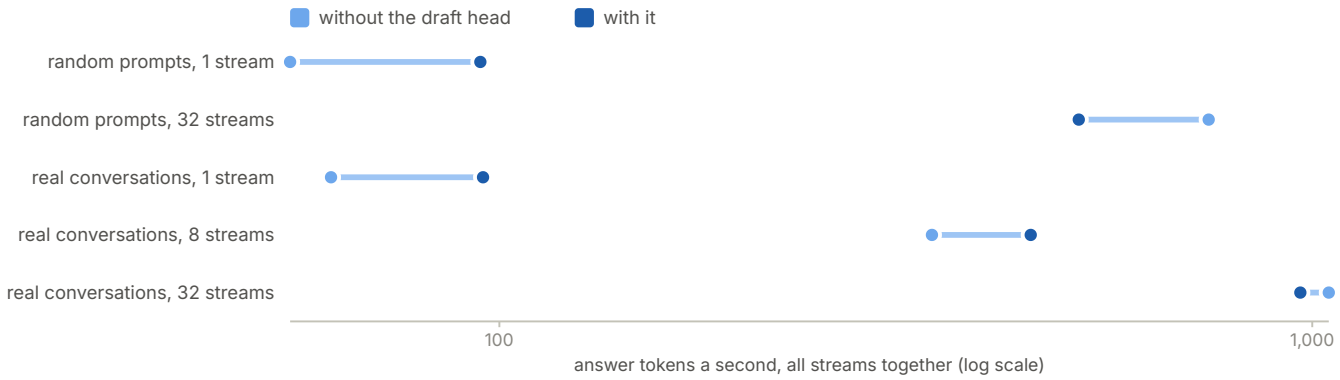


Figure 1 Answer tokens a second without the draft head (light) and with it (dark), on a logarithmic axis so that the same distance is the same share. The line between the two dots is the change: to the right a gain, to the left a loss. The table under the figure has the values.
Source: summary.json of this run (with), of 20261125T192827Z-probe (random prompts, without) and of 20261125T201346Z-series (real conversations, without): sweeps.*.levels[].output_tok_s

LOAD	WITHOUT	WITH THE DRAFT HEAD	CHANGE	GUESSES KEPT	REQUESTS
random prompts, 1 stream	55.3	94.8	+71.4%	61.4%	8
random prompts, 32 streams	745.9	516.2	−30.7%	47.5%	128
real conversations, 1 stream	62.1	95.5	+53.7%	46.7%	8
real conversations, 8 streams	340.5	450.5	+32.3%	44.5%	32
real conversations, 32 streams	1,047.9	967.0	−7.7%	43.6%	128

The plain side of each comparison is another run that played the same load without the draft head: random prompts in the probe run (20261125T192827Z-probe), real conversations in the series run (20261125T201346Z-series). Serving settings are the baseline's in all three; the counters switched on differ slightly. "Guesses kept" is the server's count of guessed tokens kept, of all it guessed.

MEASURED
Every request ended well in all five levels, with the draft head and without. The gain is largest where the card has the least to do, and turns into a loss at 32 streams on both loads.

READING, NOT MEASUREMENT
My reading, after taking the levels apart: at 32 streams a typical engine step still gives a stream more tokens with the draft head than without. What the level loses, it loses in the steps that take in new prompts, and with the draft head there are many more of those.

The machine, and the part this test changes

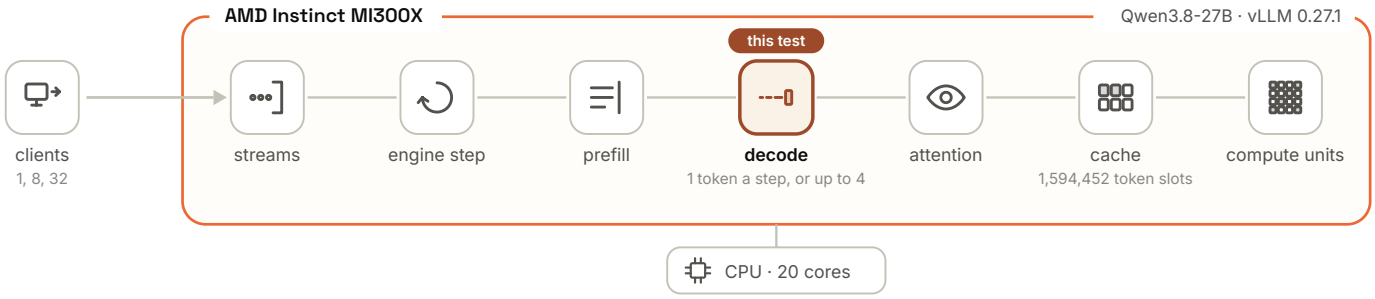


Figure 2 The draft head changes how decode works: an engine step can give a stream up to 4 tokens instead of one.
Source: FACTS.json; summary.json server.kv_cache_tokens

Guess ahead, then check

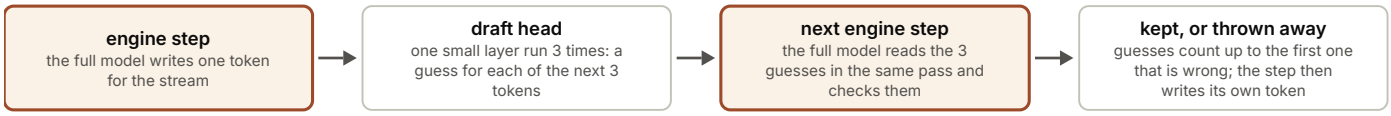


Figure 3 Two engine steps of one stream. The guesses cost arithmetic in the draft head and more in the full model, which has to read them; a kept guess is a token that needed no step of its own.
Source: summary.json config (the server's speculative setting); server.log (the draft head is one layer of the model, sharing its embedding and output weights)

1. The server starts with the baseline's settings plus the draft head, guessing 3 tokens ahead, and vLLM's estimates on.
2. Random prompts: 1,024 random tokens in, exactly 256 out, at 1 and 32 streams, as in the baseline.
3. Real conversations: the file of real chats vLLM's own benchmarks use, on the chat endpoint, answers cut at 256 tokens, at 1, 8 and 32 streams.
4. A level is 4 requests per stream, at least 8, after 2 warm-ups.

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	20261125T200505Z-mtp, started 2026-11-25T20:05:18Z, server ready after 207 s

COMPUTED

The draft head costs room. This server started with 1,594,452 cache token slots, 12.5% fewer than the probe run's 1,824,149, and loaded 51.89 GiB of weights against 51.1.

KEEP IN MIND

Each level was played once in each run, on the same machine as the server. Speed only: nothing here says whether the answers with the draft head are the same answers.

The exact record

LOAD, STREAMS	REQUESTS	FAILED	LEVEL TOOK S: WITHOUT	WITH	FIRST TOKEN S, MEAN: WITHOUT	WITH	MS PER ANSWER TOKEN: WITHOUT	WITH	ENGINE STEPS: WITHOUT	WITH	TOKENS A STEP GIVES A STREAM
random, 1	8	0	37.0	21.6	0.14	0.15	17.6	10.0	2,560	914	2.84
random, 32	128	0	43.9	63.5	2.54	1.24	33.0	54.8	1,292	616	2.43
chats, 1	8	0	33.0	21.5	0.06	0.08	15.9	10.2	2,560	1,077	2.40
chats, 8	32	0	23.5	17.7	0.25	0.77	22.1	13.5	1,287	607	2.34
chats, 32	128	0	30.7	33.3	0.90	0.49	26.5	29.4	1,289	619	2.31

"Random" is the random prompts, "chats" the real conversations; the number is the streams. Times are the load tool's, measured at the client. Engine steps are the server's own count across the level, warm-up requests included. "Tokens a step gives a stream" is its count of guesses kept, divided by its count of drafts, plus the step's own token.

How many guesses are kept

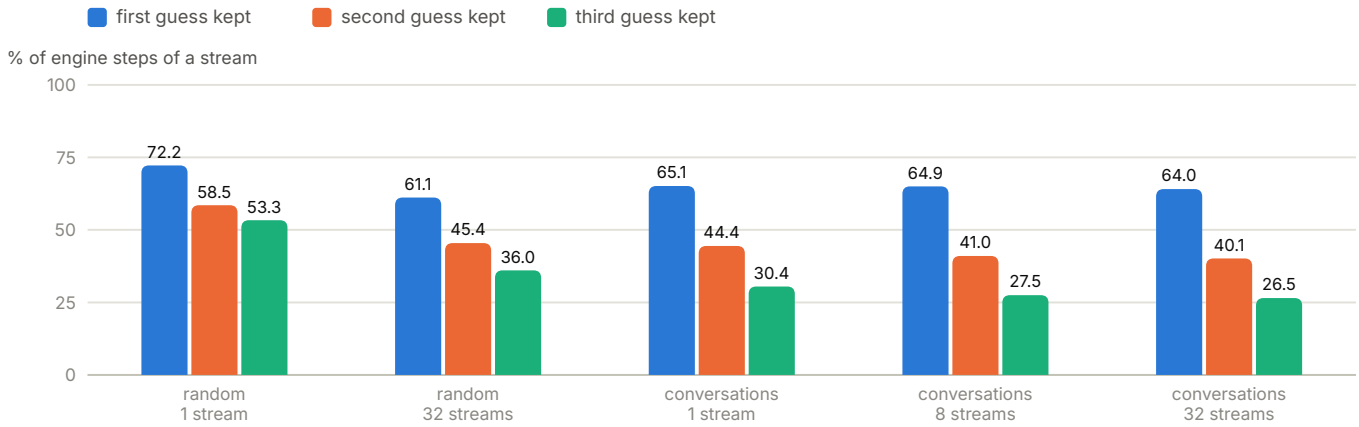


Figure 4 In how many of a stream's engine steps the first, the second and the third guess were kept. A guess can only be kept if the one before it was. On random prompts the later guesses hold up; in real conversations the third is kept in about three steps in ten.
Source: the server's counters before and after each level (c<N>.before.prom, c<N>.after.prom: vllm:spec_decode_num_drafts, vllm:spec_decode_num_accepted_tokens_per_pos)

MEASURED

Of all guesses made, 61.4% were kept for one stream on random prompts and 47.5% for 32. In real conversations it was 46.7%, 44.5% and 43.6% at 1, 8 and 32 streams: the number of streams hardly changes it.

READING, NOT MEASUREMENT

The random prompts flatter the draft head. Their forced answers are mostly one character repeated, as the load tool's record of the text shows, and that is easy to guess. Real conversations are the fairer measure of how many guesses are kept. The server itself warns at start that guessing several tokens with one draft layer may lower the share kept.

A step takes longer, and gives more

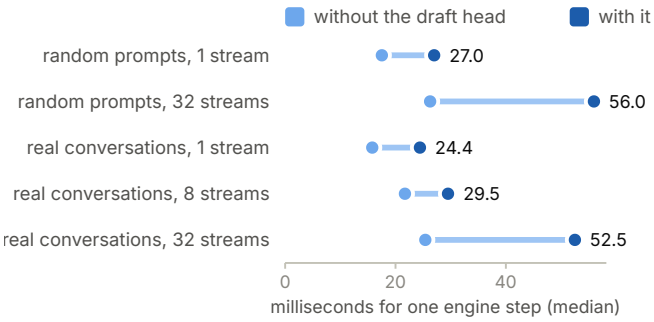


Figure 5 The cost: how long one engine step takes, as the median gap between two chunks of a stream. With the draft head a step is about one and a half times as long for one stream and twice as long among 32.
Source: c<N>.json of each run (median_itl_ms: with the draft head a chunk is what one step sent)

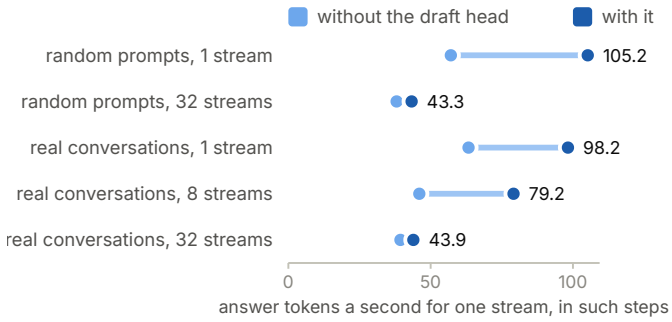


Figure 6 What that buys: answer tokens a second for one stream in such steps. Computed from the step's length and the tokens it gives. A gain at every level, 32 streams included.
Source: computed from c<N>.json (median_itl_ms) and the server's counters (guesses kept per draft)

LOAD, STREAMS	ONE STEP, MS: WITHOUT	WITH	AS LONG AS BEFORE	TOKENS IT GIVES A STREAM: WITHOUT	WITH	TOKENS A SECOND FOR A STREAM IN SUCH STEPS: WITHOUT	WITH	GAIN
random, 1	17.5	27.0	1.5 times	1	2.84	57.1	105.2	+84.2%
random, 32	26.3	56.0	2.1 times	1	2.43	38.1	43.3	+13.8%
chats, 1	15.8	24.4	1.5 times	1	2.40	63.3	98.2	+55.1%
chats, 8	21.7	29.5	1.3 times	1	2.34	46.0	79.2	+71.9%
chats, 32	25.4	52.5	2.0 times	1	2.31	39.4	43.9	+11.5%

Where the 32-stream levels lose it: prompts arrive all the time

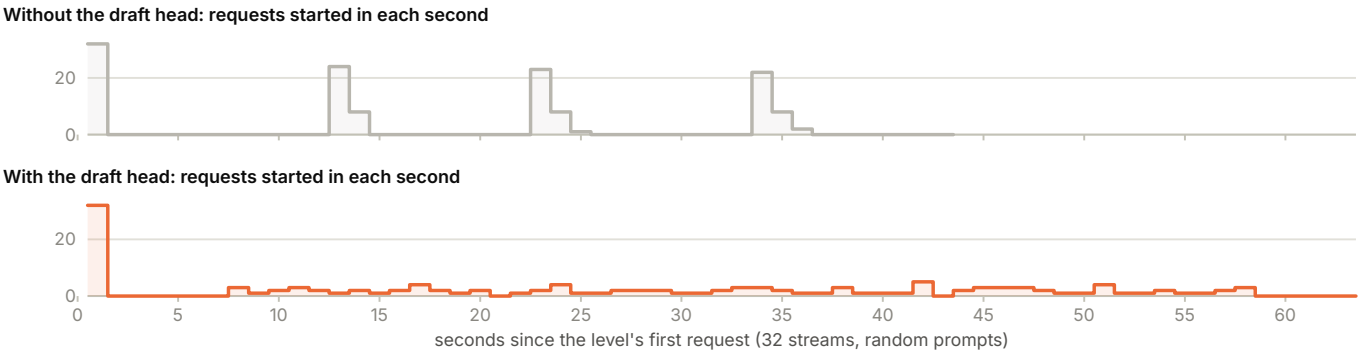


Figure 7 When new requests start, in the 32-stream level on random prompts. Without the draft head every answer takes the same number of steps, so all 32 end together and 32 new prompts arrive together. With it, answers end when their guesses let them, and new prompts arrive one or two at a time in most seconds.
Source: random-1k-256/c32.json of the probe run and of this run (start_times)

32 STREAMS	ENGINE STEPS COUNTED	OF THEM TAKING IN PROMPTS, AT LEAST	SECONDS IN WHICH A REQUEST STARTED	ENGINE STEPS ONE FULL ANSWER TOOK	GAPS OVER A TENTH OF A SECOND	THE LEVEL TOOK, S
random prompts, without	1,292	25	8 of 43	256 to 256	0.8%	43.9
random prompts, with the draft head	616	90	49 of 63	66 to 232	21.1%	63.5
real conversations, without	1,289	18	8 of 30	253 to 257	0.2%	30.7
real conversations, with the draft head	619	74	25 of 33	68 to 148	10.6%	33.3

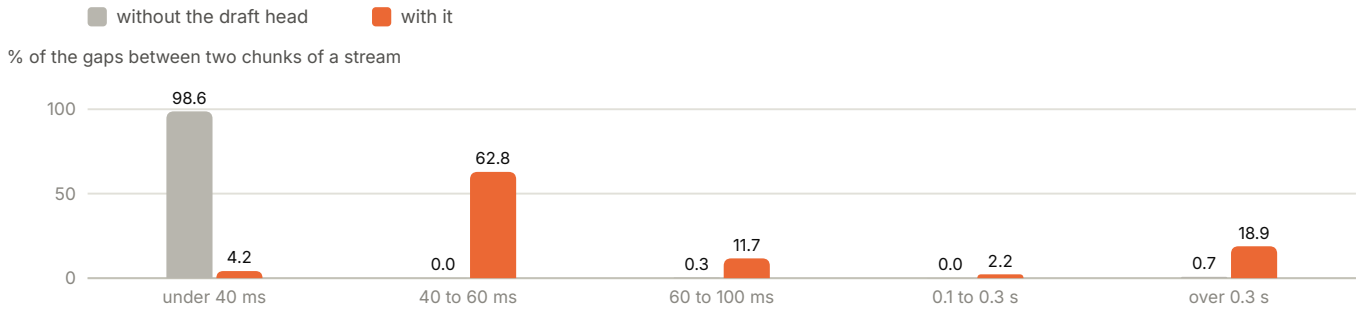


Figure 8 The gaps between two chunks of one stream, by length, in the same level. Without the draft head almost every gap is one short step. With it the typical step is longer, and one gap in five is a step that also took in a prompt.
Source: random-1k-256/c32.json of the probe run and of this run (itls: every gap of every request)

MEASURED

With the draft head the same prompts were taken in over several times as many engine steps, and one gap in five was longer than a tenth of a second. A step that takes in a prompt is a long step for every stream that is answering.

READING, NOT MEASUREMENT

So I do not put the loss at 32 streams down to the guessing itself. This load keeps the plain server's streams in step, which lets it take in prompts eight at a time between rounds of pure answering; the draft head breaks the rounds. A real service has no such rounds to begin with. I did not measure the draft head against a plain server on arrivals that are spread out, and that is the comparison that would settle it.

Underneath, for one stream: fewer bytes per token, more arithmetic

ONE STREAM, VLLM'S ESTIMATES	WITHOUT	WITH THE DRAFT HEAD	CHANGE
RANDOM PROMPTS			
GB read per answer token	46.9	17.0	−63.7%
arithmetic, TFLOPS	12.7	23.4	+84.5%
REAL CONVERSATIONS			
GB read per answer token	50.0	21.2	−57.4%
arithmetic, TFLOPS	5.4	11.2	+107.5%

READING, NOT MEASUREMENT

◌ A step reads the model's weights once whatever it writes, and with the draft head it writes more than two tokens; every guess is checked, kept or not. That fits the probe run's reading that one stream is bound by memory traffic: the draft head trades arithmetic, which one stream has to spare, for memory traffic, which it has not.

What I take from it

- ○ One stream: 71.4% more answer tokens a second on random prompts, 53.7% more in real conversations.
- ○ 8 streams, real conversations: 32.3% more.
- ○ 32 streams: 30.7% less on random prompts, 7.7% less in real conversations.
- ● In real conversations 43.6% to 46.7% of the guesses are kept.
- ● The server starts with 1,594,452 cache token slots instead of 1,824,149; no level here came near a full cache.

READING, NOT MEASUREMENT

◌ The reading agreed for this run: with spare capacity, checking several guessed tokens in one step is nearly free, so kept guesses are pure gain; on a saturated card the rejected guesses are wasted work. The first half holds in every level here. For the second I found the wasted work smaller than the gain even at 32 streams, and the loss in how prompts arrive.

- Each level played once in each run; the plain sides come from two other runs, with slightly different counters on.
- AMD's exporter is left out: these levels are too short for its readings, which trail by up to its refresh.

● 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

Draft head
(MTP, multi-token prediction) a small part of the model that guesses the next 3 tokens; the full model checks them in one step and keeps those that match.

Draft tokens accepted
the share of guessed tokens the full model kept.

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

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

FACTS.json 160f5ca0d495 · GLOSSARY.md 98323be796af · runs/20261125T192827Z-probe/random-1k-256/c1.after.prom 04337507a2c2 · runs/20261125T192827Z-probe/random-1k-256/c1.before.prom 9a8e580b65c3 · runs/20261125T192827Z-probe/random-1k-256/c1.json 37f8347fd437 · runs/20261125T192827Z-probe/random-1k-256/c32.after.prom 845e4233ffa0 · runs/20261125T192827Z-probe/random-1k-256/c32.before.prom 04337507a2c2 · runs/20261125T192827Z-probe/random-1k-256/c32.json e5ea8f77b051 · runs/20261125T192827Z-probe/summary.json 064f25b0020b · runs/20261125T200505Z-mtp/random-1k-256/c1.after.prom 1c7f3f0750e1 · runs/20261125T200505Z-mtp/random-1k-256/c1.before.prom db90616b59d7 · runs/20261125T200505Z-mtp/random-1k-256/c1.json af6f01f56d68 · runs/20261125T200505Z-mtp/random-1k-256/c32.after.prom 5661e97ac3d0 · runs/20261125T200505Z-mtp/random-1k-256/c32.before.prom 1c7f3f0750e1 · runs/20261125T200505Z-mtp/random-1k-256/c32.json 00d9f2240865 · runs/20261125T200505Z-mtp/sharegpt-chat-256/c1.after.prom 0056189f2b48 · runs/20261125T200505Z-mtp/sharegpt-chat-256/c1.before.prom 835d3d23764a · runs/20261125T200505Z-mtp/sharegpt-chat-256/c1.json d59361ff2e5f · runs/20261125T200505Z-mtp/sharegpt-chat-256/c32.after.prom 7b827e8297a0 · runs/20261125T200505Z-mtp/sharegpt-chat-256/c32.before.prom 7b2b514dd694 · runs/20261125T200505Z-mtp/sharegpt-chat-256/c32.json a679576da3ed · runs/20261125T200505Z-mtp/sharegpt-chat-256/c8.after.prom eaef64b51f47 · runs/20261125T200505Z-mtp/sharegpt-chat-256/c8.before.prom 0056189f2b48 · runs/20261125T200505Z-mtp/sharegpt-chat-256/c8.json 1659bf4f5666 · runs/20261125T200505Z-mtp/summary.json d9d06d350561 · runs/20261125T201346Z-series/sharegpt-chat-256/c1.after.prom 3a77f2bb7e8c · runs/20261125T201346Z-series/sharegpt-chat-256/c1.before.prom 11e41eadb74c · runs/20261125T201346Z-series/sharegpt-chat-256/c1.json 9008f455c5c8 · runs/20261125T201346Z-series/sharegpt-chat-256/c32.after.prom c98e077ae78b · runs/20261125T201346Z-series/sharegpt-chat-256/c32.before.prom 9c0789d331c2 · runs/20261125T201346Z-series/sharegpt-chat-256/c32.json 3d1900196099 · runs/20261125T201346Z-series/sharegpt-chat-256/c8.after.prom 60d2b6fa7b9d · runs/20261125T201346Z-series/sharegpt-chat-256/c8.before.prom 57fbea3cb239 · runs/20261125T201346Z-series/sharegpt-chat-256/c8.json 04bf2c507b59 · runs/20261125T201346Z-series/summary.json aab3940eb12e

52 values bound by path, 8 figures drawn from data. built 2026-11-26 02:37Z, 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