Speculative Decoding on Older GPUs

A stock EAGLE-3 head roughly halves TPOT on an older inference GPU and lifts its standard-class knee 1.0 to 2.25×, but prefill still binds

Pranav Varshney
Nodus Compute

Viswa Kotra
Nodus Compute

Abstract

Older GPUs stay in service long after newer ones arrive, and a fleet gets more AI out of them only if they can serve latency-bound work. Our report 028 found that an older inference GPU serves none of Qwen3-8B's interactive chat, because its TPOT with one request at a time is already above 30 ms. Nodus, the infrastructure execution layer for AI workloads, measured whether stock speculative decoding changes that. On one such GPU serving Qwen3-8B in BF16 on stock vLLM 0.30.0, with real chat prompts and the arrivals of a public production trace, the stock EAGLE-3 head Tengyunw/qwen3_8b_eagle3 with 3 draft tokens per step roughly halved TPOT, from 33.4 ms to 16.6 ms with one request at a time and to a median 0.53 of the plain engine's under load. On identical windows, the share of output tokens whose own request met the interactive class, a p95 TTFT of 0.5 s and a p95 TPOT of 30 ms, rose from 0% to between 88% and 98%. The highest load meeting the standard class, 1 s and 50 ms, rose by 1.0 to 2.25× over 5 paired seeds, a mean of about 1.7×. Speculation does not shorten prefill, so windows with a burst of long prompts still fail on TTFT: pooled over each load, the interactive class was met on 4 of 5 seeds, and on one fresh seed the standard knee did not move. Outputs at temperature 0 were not token-identical: 37 of 64 answers differed from the plain engine's, at near-tied tokens. The pre-registered gate failed, an independent verifier rated the result PARTIAL, and the method is EAGLE-3, not Nodus's own. What is new is the measurement on an older GPU.

1 Introduction

A fleet keeps its GPUs for years, so getting more AI out of the GPUs that already exist is worth as much as buying newer ones. Older generations stay in service, but at a latency target their value can be zero. Our report 028 [1] measured an older inference GPU with the same stock engine and found that it serves none of Qwen3-8B's interactive chat. With one request at a time its TPOT was already 33.6 ms at p50, above the 30 ms target, so no load meets that class. That limit is decode speed, and decode speed is what speculative decoding targets: a cheap draft proposes several tokens and the target model checks them in one forward pass, keeping the target's output distribution [2] [3].

Speculative decoding and its serving behavior are established. EAGLE-3 [4] trains a light draft head on the target model's own features and reports large speedups with one request and smaller ones at larger batches. SpecInfer [3] verifies trees of drafted tokens. Serving systems tune speculation to load: SmartSpec [5], revised as TurboSpec [6], picks the speculation length per batch in vLLM to maximize goodput, MagicDec [7] shows that speculation can pay at large batch for long contexts, and AdaServe [8] adapts speculation per request to

TPOT SLOs. Our reports 009 [9] and 015 [10] measured FP8 weights plus an off-the-shelf EAGLE-3 head on a current flagship GPU at 2.40× the SLO-met requests per GPU-second of BF16 serving for Qwen3-8B, and 1.866× and 2.051× for Qwen3-14B and Qwen3-32B. The older GPU has no FP8 tensor cores, so this study asks a narrower question in BF16: does stock speculation let an older GPU meet chat latency classes it cannot meet today, and how much of them.

The metrics are the knee, the highest load at which the pooled p95 TTFT and p95 TPOT meet an SLO class, and the share of output tokens whose own request meets the class, both measured on identical windows for the plain and the speculative engine.

2 Experimental setup

System. One older inference GPU at its default power limit and clocks runs stock vLLM 0.30.0 [11] with report 028's flags: a maximum length of 9216, a memory fraction of 0.90, at most 256 sequences and a step budget of 2048 tokens, with prefix caching and chunked prefill at their defaults. The model is Qwen3-8B [12] in BF16. The baseline B is that engine. The candidate C adds the stock EAGLE-3 head Tengyunw/qwen3_8b_eagle3, 763 MB, which drafts 3 tokens per step, with nothing tuned. The head costs memory: B's engines held 27,152 tokens of KV cache, the warm C engines 17,808 and the first C engine, built with a cold compile cache, 10,208.

Workload. Prompts are real chat text: 25,242 prompts from 8,000 UltraChat-200k [13] conversations, every prefix that ends at a user turn rendered with the Qwen3 chat template, 14 to 3,294 tokens long with a mean of 592 and a p95 of 1,482, drawn uniformly at their natural lengths. Arrivals and answer caps come from the public 2023 LLM inference conversation trace released with Splitwise [14], scaled in time to each load with its bursts kept. Answers are capped at the trace's answer length and 512 tokens and end at the model's own end-of-sequence token. Every headline run decodes at temperature 0. The same seed, load and repeat give identical arrivals, prompts and caps in both arms, and the prefix cache is reset at the start of every run.

SLO classes. The interactive class needs a p95 TTFT of at most 0.5 s and a p95 TPOT of at most 30 ms, and the standard class 1.0 s and 50 ms. Both need at least 99% of requests completed. A load passes a class only if the p95 TTFT and p95 TPOT over all its requests meet it.

Load search. The ladder is report 028's. Load grows 1.5× per step from 0.3 requests per second until the standard class fails, then bisects on a grid of 10.7% steps and rechecks the best load with a fresh window. A window has arrivals for the longer of 15 s and the time of 20 requests, plus up to 30 s of drain. The knee is the highest passing load below the lowest failing one. The researcher ran seeds 1, 2 and 3 for both arms, repeated B's seed 1 on a fresh engine as an A/A and repeated C's seeds 1 and 3 on fresh engines. The head was chosen by a pre-registered rule from a screen of speculators: the lowest unloaded p95 TPOT among those whose saturated throughput was at least 0.8× B's. A 2.0 GB EAGLE-3 head and a Qwen3-0.6B draft model did not fit in memory next to the target.

3 Results

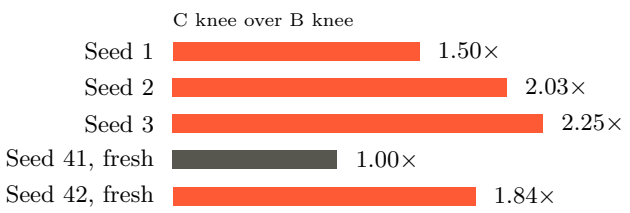


Figure 1: Knee of the standard class with the EAGLE-3 head over the plain engine, per paired seed. On seed 41 a burst of long prompts capped both arms at the same load.

Table 1: Knees in requests per second for the standard class (1 s, 50 ms) and the interactive class (0.5 s, 30 ms), and the share of C's output tokens whose own request met the interactive class on windows identical for both arms. B met the interactive class at no load on any seed, and 0.0% of its tokens met it. Seeds 41 and 42 were run by the verifier.

Seed	B standard knee	C standard knee	C over B	C interactive knee	C tokens meeting interactive
1	1.0000	1.5000	1.50×	0.9036	93.1%
2	1.1067	2.2500	2.03×	0.4444	94.9%
3	0.9036	2.0331	2.25×	0, TTFT	98.1%
41, fresh	0.9036	0.9036	1.00×	0.8165	87.9%
42, fresh	1.2247	2.2500	1.84×	0.3629	94.5%
Mean of 5			1.72×		

TPOT roughly halves. With one request at a time, B's TPOT was 33.4 ms at p50 and 34.2 ms at p95, and C's 16.6 ms and 18.4 ms, with a mean accepted length of 2.578 tokens per verification step. Under load, on the 20 loads at which both arms met the standard class over the 5 paired seeds, C's pooled p95 TPOT was 0.47 to 0.64 of B's, with a median of 0.53. On the fresh seeds 41 and 42 it was 21.0 to 30.1 ms against 38.8 to 47.4 ms, and B never went below 30 ms at any load. The mean accepted length per window stayed between 2.36 and 2.64 in the ladders. In a saturated closed loop of 256 clients, C still produced 1.09× and 1.21× B's output tokens per second on seeds 1 and 2 with a smaller KV cache, so no measured load made speculation a loss on this GPU.

The decode half of the interactive class is fixed, the prefill half is not. On identical windows, 0.0% of B's output tokens came from requests that met the interactive class on every pair, against 87.9% to 98.3% of C's: 92.3% to 98.3% on the researcher's pairs and 87.9% and 94.5% on the fresh seeds. Pooled over each load, C met the interactive class on seeds 1, 2, 41 and 42, up to 0.9036, 0.4444, 0.8165 and 0.3629 requests per second. It failed on seed 3 and on that seed's repeat on a fresh engine. At 0.30 requests per second the p95 TTFT was 0.519 s while the p95 TPOT was 24.2 ms, and because a class stops at its lowest failing load, seed 3 scores 0 even though 1.0 requests per second passed. With 14 to 30 requests per window the p95 is close to the window's maximum, so one or two prompts over about 2,000 tokens decide it, as the older GPU prefills Qwen3-8B at about 4k tokens per second. Speculation does not change that.

The standard knee rises 1.0 to 2.25×. Over the 5 paired seeds the knee rose by 1.50, 2.03, 2.25, 1.00 and 1.84×, a mean of 1.72×. Below B's knee both arms delivered the same standard-class tokens on identical windows, so the gain is the knee moving up, not more work at low load. On seed 41 there was no gain. In the 1.0 requests per second window, prompts of 1,758, 2,830 and 1,125 tokens arrived within 0.5 s, and the p95 TTFT reached 1.00 s for B and 1.09 s for C while C's p95 TPOT was 30.3 ms, so both arms stopped at 0.9036. C's TTFT was no better overall: over the 20 common passing loads its p95 TTFT was a median 1.01 of B's, between 0.83 and 1.28, and in the seed 41 burst it was slightly worse because its KV cache is smaller and the head adds its own prefill.

The registered token metric is not an effect size. The pre-registered metric, SLO-met output tokens per GPU-hour at the standard knee, rose from 0.61 to 1.56 million on seeds 1 to 3, 2.59×, but on the fresh seeds it read 1.00× and 12.2×. It counts only the knee's two short windows, whose actual request count varies about 5× at a fixed nominal load. On seed 42, B's knee windows served 48.7 tokens per GPU-second while B served 362.4 at 1.0 requests per second. This report therefore quotes the knee ratio.

Outputs at temperature 0 are not token-identical. On 64 fixed prompts with 128-token answers and 8 requests in flight, 27 of 64 C answers (42%) matched B's exactly and 37 (58%) differed, against 64 of 64 for B against itself and 40 of 64 for C against itself on two engines. Another EAGLE-3 head at two draft lengths and n-gram lookup, which has no draft network, matched B on 26 to 27 of 64, which points to the verification step rather than the head. At the first divergence C's token was B's second-ranked token in 33 of the 37 differing answers, the median gap between B's top two log-probabilities was 0.00 nats, and 62% of divergences had a gap of at most 0.05 nats, against 2.0% of all 7,905 positions.

Speculative decoding preserves the target's output distribution in exact arithmetic [2], but the target scores the 3 drafted tokens and one more position in one batched forward pass whose BF16 rounding differs from scoring one position per step, and near-tied tokens can flip. Answer quality was not measured. A customer that needs reproducible output at temperature 0 should leave speculation off.

Single-seed cells. At temperature 0.7, C met the interactive class up to 0.9036 requests per second where B met none and raised the standard knee 1.22×, but only 64.6% of C's tokens met the interactive class request by request. On the workload as first registered, with prompts matched to the trace's lengths, of which 9.5% exceed 2,000 tokens, neither arm met the interactive class at any load, and the standard knee rose 2.25× with 1.59× the tokens. The head matters more than speculation itself: a second EAGLE-3 head of the same size accepted 2.012 tokens per step against 2.578, with a p95 TPOT of 24.6 ms against 18.4 ms, and n-gram lookup made TPOT worse, 36.9 ms at p95, with only 2.9% of its tokens meeting the interactive class. vLLM's stock load-adaptive speculation length crashed the engine at start with this head.

4 Independent verification

A separate verifier re-judged every ladder from the per-request logs with its own code, matched every number, and confirmed identical arrivals, prompts and caps in both arms. On the same GPU it ran fresh seeds 41, 42 and 43, never used before, with the researcher's code unchanged except for the container name and output directory. Its time slot ended before C's seed 43, so seed 43 has B only. The fresh seeds reproduced the TPOT reduction, the per-request share of 87.9% and 94.5%, and C meeting the pooled interactive class on both seeds where B met none. Its A/A on B's seed 41 repeated the knee exactly, with 288.9 against 289.2 tokens per GPU-second, 0.1% apart. The fresh seeds also gave the 1.00× knee on seed 41 and the 12.2× token reading on seed 42.

The verifier rated the result PARTIAL. The pre-registered gate failed: the interactive class was not met on seed 3, the standard gain failed its every-seed condition on seed 41, and the exact-identity guard failed. The 2.59× headline is not a stable effect, and the defensible size is the knee ratio of 1.0 to 2.25×, a mean of about 1.7×. The headline workload replaced the registered trace-matched one after the plain engine's screen showed a p95 TTFT of 0.70 s with one request at a time on it and before any ladder ran, which was disclosed. The baseline uses the same stock flags as the candidate with no scheduler tuning, and its standard knee is TPOT-marginal at 46 to 53 ms against 50 ms, so the result holds for the stock configuration. The method is established in EAGLE-3 [4], SpecInfer [3], SmartSpec [5], MagicDec [7] and AdaServe [8], and is not Nodus's own. What is new is the operating point: no published measurement was found of whether stock speculation makes these SLO classes reachable for Qwen3-8B in BF16 on the older GPU.

5 Conclusion

Stock speculative decoding gives an older GPU back part of chat. On the older GPU serving Qwen3-8B in BF16, a stock EAGLE-3 head roughly halved TPOT, lifted the share of output tokens meeting a 0.5 s and 30 ms class request by request from 0% to between 88% and 98%, and raised the knee of the 1 s and 50 ms class by 1.0 to 2.25× over 5 paired seeds, a mean of about 1.7×. It fixes the decode half of the interactive class and leaves the prefill half: pooled classes still fail on windows with bursts of long prompts, and with prompts matched to the production trace's lengths no arm met the interactive class on the one seed measured. Its outputs at temperature 0 are not token-identical. For a fleet, speculation on older GPUs belongs on TPOT-bound endpoints, chosen per model by the measured acceptance length and with TTFT checked separately. The study covers one GPU, one model, one draft head, one engine version with stock flags, English chat prompts and 5 paired seeds.

References

[1] Nodus Research. The SLO Frontier per GPU Generation. Report 028, 2026.

[2] Y. Leviathan, M. Kalman, Y. Matias. Fast Inference from Transformers via Speculative Decoding. ICML 2023.

[3] X. Miao et al. SpecInfer: Accelerating Generative Large Language Model Serving with Tree-based Speculative Inference and Verification. ASPLOS 2024.

[4] Y. Li, F. Wei, C. Zhang, H. Zhang. EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test. NeurIPS 2025.

[5] X. Liu et al. Optimizing Speculative Decoding for Serving Large Language Models Using Goodput. arXiv preprint arXiv:2406.14066v1, 2024.

[6] X. Liu et al. TurboSpec: Closed-loop Speculation Control System for Optimizing LLM Serving Goodput. arXiv preprint arXiv:2406.14066v3, 2025.

[7] R. Sadhukhan et al. MagicDec: Breaking the Latency-Throughput Tradeoff for Long Context Generation with Speculative Decoding. ICLR 2025.

[8] Z. Li et al. AdaServe: Accelerating Multi-SLO LLM Serving with SLO-Customized Speculative Decoding. arXiv preprint arXiv:2501.12162, 2025.

[9] Nodus Research. FP8 and Per-Tenant Speculative Decoding from 8B to 32B. Report 009, 2026.

[10] Nodus Research. FP8 and Speculative Decoding at 14B and 32B. Report 015, 2026.

[11] W. Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP 2023.

[12] A. Yang et al. Qwen3 Technical Report. arXiv preprint, 2025.

[13] N. Ding et al. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. EMNLP 2023.

[14] P. Patel et al. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. ISCA 2024.