PURETENSOR
INTELLIGENCE DISTILLED

Tuning a Day-Zero Model Like an Experiment

Paired confirmations, tokens per step against milliseconds per step, and a judge the agent cannot move

Paper: PT-R-2026-066 v1.0

Author: PureTensor Ltd

Date: 5 November 2026

Status: Published

Abstract

A new dense open-weight model, Ripple-65B-Preview (64.8 billion parameters, Apache-2.0), was published on 4 November 2026 together with a 4.04-billion-parameter speculative drafter (a small model that guesses several tokens ahead so the large model can check them in one pass) and a 4-bit build. There was no community recipe for our GPU class. We tuned it with a harness that keeps a configuration only when the lower bound of a paired, order-balanced 95% confidence interval clears a minimum useful gain, and that its proposer, human or agent, cannot change without the run stopping.

On one 96 GB workstation GPU the 4-bit build with its drafter decoded at 135.2 tokens per second on a code prompt and 97.9 on a reasoning prompt, against 24.0 for both without speculation. On two such GPUs it reached 212.4 and 152.0. The two-card gain over one card was a paired, confirmed x1.589 (95% CI 1.512 to 1.641). Splitting every run into tokens per step and milliseconds per step showed that the whole gain was step time (x0.632 and x0.633) and that, in this setting (one boot configuration, one GPU, one model), the run-to-run noise sat in acceptance: code step time was 51.1 to 51.2 ms on three fresh boots while the five timed runs inside one boot typically spread by 16% (median over boots and cases).

On a smaller model (Qwen3-8B) we also let a headless agent propose configurations under a hashed judge. The plain sweep alone had already confirmed x1.219. The agent made six proposals, one of which, lowering the minimum match length of prompt-lookup speculation to 1, confirmed at x1.357 over defaults. That is a count of one move, and it measures nothing about a rate. The guarantee about the judge is detection plus isolation, not a read-only filesystem, and its live check never fired.

1. A Model Hours Old

The usual output of a launch-day speed test is one number: the best of several runs. With a model hours old, three things are unknown at once: what a good configuration is, how much two identical runs differ, and which levers move which part of the speed.

Ripple's weights, drafter, vendor serving plugin and launch recipe were all public on 4 November. We downloaded the three weight sets that evening (BF16 at 129.6 GB, the drafter at 8.08 GB, the 4-bit build at 57.2 GB, with MLP weights in NVFP4, a 4-bit floating-point format)

and ran the vendor plugin unmodified. The drafter proposes blocks of 16 tokens.

We wanted two properties. Every change we keep carries a paired confidence interval and a stated minimum useful gain, so that a lucky boot cannot pass as an improvement. And whoever proposes configurations cannot alter the evaluator unnoticed, the failure we described in our October decode-record reproduction. This paper describes the procedure, what it found on one and two GPUs, and what an agent proposer added on a small model. It claims no decode record.

2. The Harness

The harness launches a server per configuration, runs a frozen benchmark, applies gates and writes one ledger row per trial. The objective is the geometric mean over two cases (one code prompt, one reasoning prompt) of the median batch-1 streamed decode rate at temperature 0 (completion tokens minus 1, over last token time minus first token time). Low fidelity is one warm-up plus two timed runs per case; full fidelity is two warm-ups plus five.

A gated trial never counts, however fast. The 16-item quality set must be within one item of the baseline; outputs must not end in a repetition loop; compression ratio and repeats must stay within bounds of the baseline; the model must not hit the token cap where the baseline stopped naturally; and no output may be under half the baseline's length. For every request the judge also records tokens per step (accepted tokens plus one per verify step, from server-reported counts), milliseconds per step, a survival curve (the fraction of verify steps in which at least j drafted tokens were accepted) and an output hash.

Confirmation decides whether a change is kept. Fresh server boots are run in pairs, A the reference and B the candidate, in a seeded order balanced between AB and BA. A bootstrap resamples pairs and then runs within each boot. The change is kept only if the lower bound of the 95% interval on the objective ratio exceeds 1 plus a minimum useful gain (3% here) and each case is non-inferior (its lower bound above 0.98). The noise floor is reported separately, not added to the bar. Pairing shares slow drift between arms; balancing the order stops either always booting first.

Finally the judge program, the benchmark prompts and the quality set are hashed together. The hash is stored in every ledger row and the harness refuses to run if it changes; rows are compared only within the same judge hash and space-core hash.

3. One Card: Screening and the Final Numbers

Hardware was a workstation host with two RTX PRO 6000 Blackwell cards (96 GB each, PCIe Gen5, no NVLink). The one-card runs used one card; the other holds about 38 GB of small idle services. The engine was SGLang 0.5.19 with a 32K context, one running request and CUDA graphs at batch size 1.

An arm sweep (one warm-up and three timed runs per case, median reported) took the vendor's flags as arm A.

Arm	Code tok/s	Reasoning tok/s	ms per step	Outcome
A, vendor flags, 16-token blocks	135.6	99.7	51.6	winner
B, 32-token blocks	130.0	101.9	53	no gain
C, Marlin FP4 GEMM	133.1	102.0	51.6	equal, worse prefill
E, cuDNN FP4 GEMM	124.2	93.6	55	slower
G, Triton attention	112.8	91.1	55	slower

Arm	Code tok/s	Reasoning tok/s	ms per step	Outcome
J, online FP8 attention	116.7	95.6	53	slower
M, decode-mode verify attention	133.0	100.0	51.2	equal
H, no speculation	24.0	24.0	41.7 per token	reference

Table 1. One-card arm sweep (screening: one GPU model, one engine version, SGLang 0.5.19, batch 1, three timed runs). A slower result here says nothing about these components elsewhere. The vendor's flags won; no arm clearly beat them.

Three further arms did not run (an FP4 backend unsupported on this GPU generation in the engine version we ran, a torch-compile failure at graph capture, and an FP8 drafter the vendor loader rejected in our version; we later made it load, Section 5). The 32-token arm gained nothing, consistent with the drafter having been trained on 16-token blocks (we did not verify this). On one card the vendor recipe was the best point we found.

A 30-step profile (profiler summary only) showed both matrix-multiplication groups streaming at about 1.46 TB/s, about 82% of an assumed peak of about 1.8 TB/s, so a batch-1 verify step is bandwidth bound.

The final one-card runs (median of five after two warm-ups, batch 1, temperature 0, CPU boost on) were the figures we submitted to a public local-inference speed leaderboard, which, as we understand its rules, verifies runs from server-reported draft counts.

Case	Decode tok/s	Tokens per step	Prompt tokens
code, speculation on	135.2	6.9	216
reasoning, speculation on	97.9	5.1	279
15.7K-token prompt, on	98.5	5.4	15,700
code, no speculation	24.0	n/a	218
reasoning, no speculation	24.0	n/a	278

Table 2. Final one-card figures. Speculation is worth 5.6 times on code (135.2 / 24.0) and 4.1 times on reasoning (97.9 / 24.0).

A first pass (127.3 on code) lacked draft counts and was not submitted; the re-run with counts gave 135.2, a 6% spread inside the within-boot spread of Section 4.

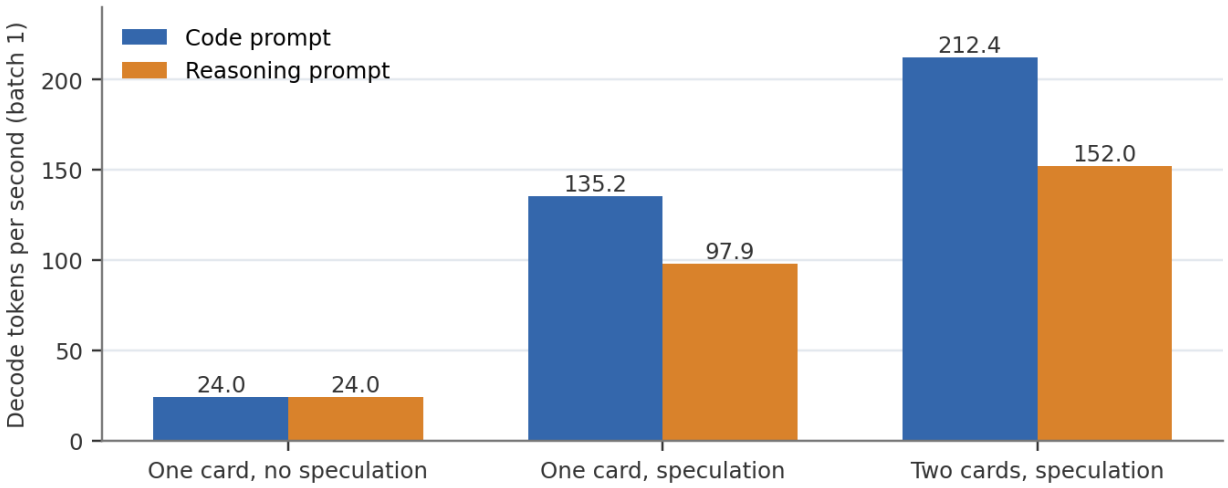


Figure 1. Batch-1 decode speed on code and reasoning prompts: one card without speculation, one card with speculation, two cards with speculation. Values are in Table 6.

4. Where the Noise Lives

We first measured how much the unchanged configuration disagrees with itself, in three fresh boots at full fidelity.

Boot	Objective tok/s	Code tok/step	Reasoning tok/step	Code ms/step	Reasoning ms/step
1	115.0	6.88	5.08	51.1	51.7
2	113.9	6.80	5.05	51.2	51.6
3	114.4	6.49	5.33	51.2	51.7

Table 3. Three fresh boots of the unchanged one-card configuration. Step time is flat; tokens per step moves.

Boot-to-boot, the standard deviation of the log objective was 0.005 (0.5%), and the three boot objectives span 0.98% of their median. Inside a single boot the five timed runs of a case typically spread by 16% (median over boots and cases of range over median). Code step time held at 51.1 to 51.2 ms on every boot (reasoning 51.6 to 51.7) while code tokens per step ranged from 6.49 to 6.88. Without speculation the same boots gave 24.0 tok/s with a spread under 0.1.

So in this setting the noise comes from how many drafted tokens happen to be accepted; the step is a fixed cost. Qwen3-8B without speculation gave 93.55 to 93.59 tok/s across six boots.

As in our October decode-record reproduction, we report tokens per step and milliseconds per step separately.

One gate needed a change. The first baseline tripped the repetition gate; we raised its allowance to 16, with the reason recorded, before measuring any candidate, and the next baseline passed.

5. Two Cards: A Paired Confirmation

Screening at low fidelity gave 187.6 tok/s for two-card tensor parallelism with SGLang's custom all-reduce, against 116.3 for the one-card baseline at full fidelity (a ratio of 1.613), and 175.9 with NCCL only. The first two launches failed SGLang's memory-balance check, because the second card carries other residents; the space now turns that check off for this host. Step time fell from 51.2 to 32.4 ms.

We confirmed with three pairs at full fidelity, orders BA, AB, AB, minimum useful gain 3%.

Pair	Order	One card (A) tok/s	Two cards (B) tok/s	Pair ratio
1	BA	113.1	178.4	1.577
2	AB	118.6	184.8	1.558
3	AB	113.8	180.7	1.588

Table 4. Paired confirmation of two cards over one.

The verdict was KEEP, with an objective ratio of 1.589 and a 95% interval of 1.512 to 1.641 (lower bound 1.512 against a bar of 1.03). Per case, code was 1.541 [1.467, 1.621] and reasoning 1.638 [1.522, 1.701].

Case	Tokens per step ratio	Milliseconds per step ratio	Speed ratio
code	0.973	0.632	1.541

Case	Tokens per step ratio	Milliseconds per step ratio	Speed ratio
reasoning	1.037	0.633	1.638

Table 5. Two cards over one, decomposed. Median step times were 51.2 and 51.7 ms on one card, 32.3 to 32.4 and 32.7 ms on two.

The whole gain is step time. Tokens per step sits either side of 1 (0.973 and 1.037), inside the noise of Section 4: two cards make each verify step cheaper and leave the drafter alone.

The ratio of arm medians is 180.7 / 113.8 = 1.588, the geometric mean of the pair ratios is 1.574, and a plain t-interval on the pair log-ratios gives 1.537 to 1.612; 1.589 is the bootstrap point estimate. The screen read 187.6 where the confirmation arms read 178.4, 184.8 and 180.7, about 3.8% above their median, which is why a screen is never reported as a result.

The two-card leaderboard runs are in Table 6 (212.37 and 151.95 before rounding).

Configuration	Code tok/s	Reasoning tok/s
One card, no speculation	24.0	24.0
One card, speculation	135.2	97.9
Two cards, speculation	212.4	152.0

Table 6. Verified batch-1 decode speed by configuration (two-card ratios over one card: 1.571 code, 1.552 reasoning).

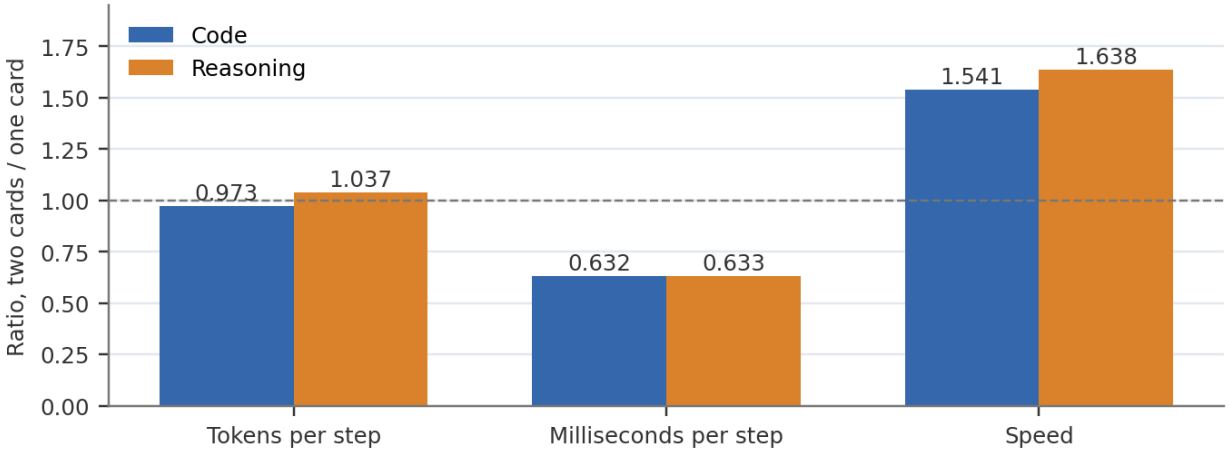


Figure 2. Two cards over one, split into tokens per step and milliseconds per step ratios, with the resulting speed ratio. Values are in Table 5.

One lever was screened and not confirmed. A drafter quantised online to FP8 gave +3.9% on one card (118.8 at low fidelity against the 114.4 full-fidelity median of the noise boots) and +1.4% on two (190.2 against 187.6). The one-card confirmation was stopped before its first pair when a background service reclaimed the card (launches at 06:45 and 06:47 failed the free-memory check, 57,402 MiB free against 84,000 MiB needed). Resolving the two-card 1.4% against a 1% bar needs about 49 pairs at the harness's sizing rule. Neither is a finding.

6. Pricing a Configuration Without Booting It

If a verify step checks only w drafted positions instead of 16, expected tokens per step is 1 + S_1 + ... + S_(w-1), where S_j is the survival curve, so the one-card baseline prices a narrower verify without a boot.

A 12-token verify keeps 0.892 of the 16-token tokens per step on code and 0.939 on reasoning; an 8-token verify keeps 0.729 and 0.827. Breaking even would need a step 6% to 11% cheaper

at width 12 and 17% to 27% cheaper at width 8, which a bandwidth-bound step does not offer.

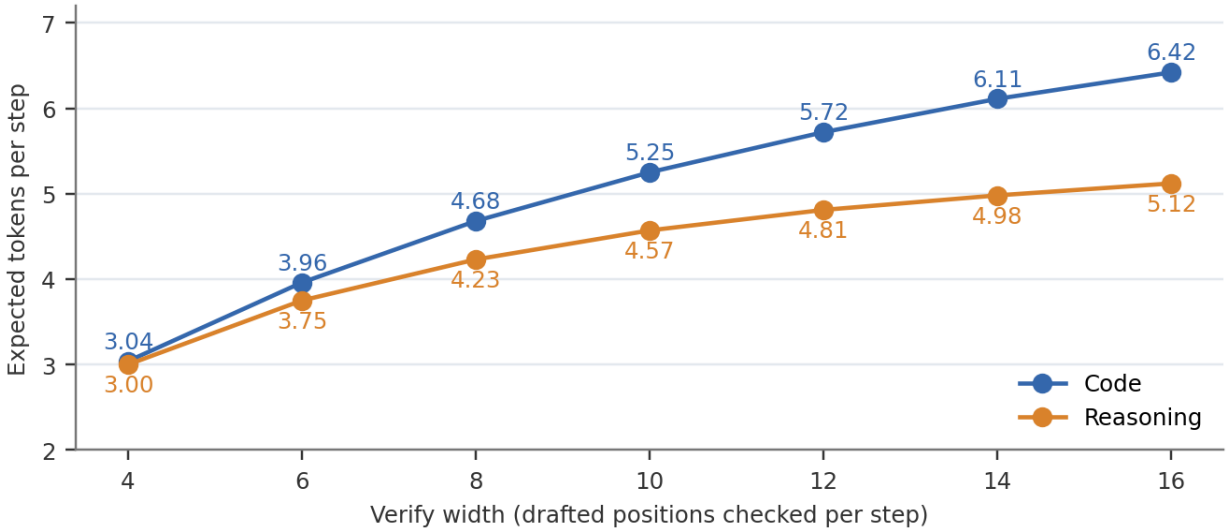


Figure 3. Expected tokens per step against verify width, from the one-card survival curves. Values are in Table 7.

Width	4	6	8	10	12	14	16
code	3.04	3.96	4.68	5.25	5.72	6.11	6.42
reasoning	3.00	3.75	4.23	4.57	4.81	4.98	5.12

Table 7. Expected tokens per step by verify width, from the one-card baseline survival curves.

We never booted a 12- or 8-token verify, and the pruning rule, written after the Ripple runs and tested by replay against the real ledgers, has never skipped a configuration in a live sweep.

7. What an Agent Added on a Small Model

Our only evidence on what an agent adds to a plain sweep is a study on 1 November: Qwen3-8B in BF16 on one shared GPU, 29 ledger rows, 1.36 GPU-hours of trial time.

A headless frontier-model agent ran on a different host from the GPU host. Its tools were reading, writing and editing files, and two read-only harness commands. It read a mirror of the ledger, wrote proposal files and a notebook, and exited; the harness host then ran the proposals. After each agent turn the loop hashed the judge, the harness program, the agent policy, the loop script and the space core, and any change stopped the run before a proposal executed. The agent could add values to a knob but not touch gates, defaults, model, context length or sampling.

Of the 29 ledger rows, six were agent proposals (one a launch failure); the rest are baseline, sweep and confirmation rows.

The sweep's twelve screens moved only speculative decoding: n-gram speculation with a minimum match of 2 gave 112.9 with 3 draft tokens and 114.9 with 6 at low fidelity, while the other knobs were within 3% or, for asynchronous scheduling off, 9% worse. Its confirmation (two boots each side, bar 1%) gave a ratio of 1.219, CI [1.205, 1.231], from 93.57 to 114.18 tok/s (code 1.141, reasoning 1.302).

The agent ran three turns. It computed that the baseline was already at about 85% of the memory-bandwidth ceiling and concluded that only draft acceptance had headroom. Its first proposals were flat or failed: a GPU-side n-gram variant (113.8), suffix decoding (the image lacked a required package; the trial crashed after 10 seconds), 10 draft tokens (114.1, tokens per step identical to 6) and a longer maximum lookup of 8 (114.3, flat). From that it reasoned

that the limit was finding a match, not match length, and proposed lowering the minimum match length to 1.

That trial read 127.0, +11% over the confirmed 114.2, with tokens per step rising from 1.13 to 1.26 on code and from 1.31 to 1.46 on reasoning. The agent had predicted +2% and asked for a confirmation, which (three pairs of fresh boots, bar 3%) went from 93.58 to 126.91 tok/s: ratio 1.357, CI [1.343, 1.366], code 1.266 [1.264, 1.270], reasoning 1.455 [1.425, 1.469]. A last proposal (3 draft tokens, minimum match 1) gave 124.1, no gain.

There are two honest readings. First, the plain sweep alone found x1.219; the agent's increment is the minimum-match change, about x1.11 over the sweep winner (126.9 against 114.2). That comparison is not paired and spans two time windows, because x1.357 was confirmed against the defaults. Second, one agent-found move out of six proposals is a count of one: an agent can propose a worthwhile change under this harness, and we know nothing about how often.

Here every gain came with higher tokens per step; we did not decompose step time on this model.

8. What These Numbers Will Not Carry

Coverage is narrow: one GPU class, one model for the day-zero case, and one small-model agent study with three iterations. Everything is batch 1; a single-stream tune says nothing about throughput under concurrency, where our September post on speculative decoding on production seats shows the two can disagree.

The headline confirmation rests on three pairs, which give few distinct bootstrap resamples. Our own t-interval (1.537 to 1.612) differs from the reported 1.512 to 1.641; both clear the bar widely, but the exact width is not something to cite. Without a null experiment (the same configuration in both arms) we have not measured whether the intervals are calibrated. The leaderboard ratios (1.571, 1.552) come from different single runs and differ from the paired reasoning ratio (1.638); the tokens-per-step figures in Table 2 come from different runs than the ledger rows and differ in the second decimal. The 135.2 is the better of two passes (127.3 first).

The one-card baselines ran on a card shared with small idle residents; that matters for absolute figures and not for the paired ratio.

The judge guarantee needs the most careful reading. The loop hashes the judge and harness after each agent turn and refuses to run proposals if anything changed. But the agent held write and edit tools on its own mirror, so this is detection plus isolation, not a read-only filesystem. The check never fired and we know of no attempted edit, so it is covered only by unit tests. The judge also changed once after the study, in a human-reviewed revision about two hours after the agent run finished, so the study's rows are not comparable with later rows.

The small-model gates were weak: the quality gate scored 12 of 16 on every arm including the baseline, and the reasoning case hit the 768-token cap on every arm, so its tok/s measures a truncated reply. On Ripple the quality gate passed 16 of 16 on every trial, but it is a small set.

What survives: the one-card figures on this hardware; the two-card gain as a step-time effect confirmed at x1.589; acceptance carrying the noise here; and the statement that, once, an agent under a hashed judge found a move a plain sweep did not. What does not survive: any rate for agent contributions, any claim that pruning has saved a boot, and any claim for the unconfirmed FP8-drafter gains.

9. What We Changed in Practice

Every verdict now states whether tokens per step or milliseconds per step moved. A sweep winner must repeat on a second fresh boot before it moves the sweep. Confirmation size is

derived from measured boot noise and the screened gain (3 to 8 pairs), with a floor of three pairs for a KEEP. A verify width is skipped without a boot when the best case keeps under 95% of tokens per step.

The next experiment is the null: the same configuration in both arms, enough pairs to see how often the interval excludes the bar by accident. After that, the same agent budget against the same sweep on a second model.

10. Methods and Artefacts

Servers were SGLang 0.5.19 (torch 2.13.0, CUDA 13.2) for Ripple and vLLM (a nightly image pinned by digest) for Qwen3-8B. The decode rule and prompts are those of a public local-inference leaderboard. Profiling used the torch profiler and an Nsight Systems trace of 39 warmed steps. The objective is the geometric mean of per-case medians; confirmations are paired, order-balanced, fresh-boot comparisons with a hierarchical bootstrap (pairs, then runs within each boot) for the 95% interval. Noise is the standard deviation of the log objective across three fresh boots plus the within-boot relative spread. We retain the full ledger for every trial, server logs, confirmation and noise records, the agent's notebook and proposal files, the profile summaries and the leaderboard submission records.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-066. The small-model study was run on 1 November 2026 and the Ripple measurements between the evening of 4 November and the early morning of 5 November 2026; the raw artefacts are retained.