How Many Users Can One GPU Serve? A Capacity Law for LLM Inference from Consumer Cards to Datacenter Accelerators

Preprint, November 2026

Thanatchon Chongmankhong

Abstract

"How many users can one GPU serve?" is the first question in sizing a self-hosted large language model (LLM), yet benchmarks answer in tokens per second. We define capacity as the number of concurrently active users, each alternating between thinking and waiting, for whom at least 95% of requests meet a time-to-first-token (TTFT) and a time-per-output-token (TPOT) target. We build three tools that answer in this unit: a token roofline for one engine iteration, a birth–death model of a closed population with stochastic SLO terms, and an iteration-level simulator of continuous batching with paged KV memory. Calibrated on public profiles and benchmarks, the step-time model predicts held-out GPUs within 10% (vLLM) and held-out consumer cards' decode speed with Spearman $\rho_s = 0.99$; the simulator matches 90% of 48 MLPerf systems within 0.67–1.5×. For a chat workload replayed from Azure traces, one GPU serves about 0 (12–16 GB budget cards), 170–230 (RTX 4090/5090), 330–340 (RTX 6000 Ada, L40S, A100) and 690–1010 (H100–B200) concurrent users with an 8B model. Capacity is governed less by memory bandwidth than by prompt length and prefill compute (Sobol total-order indices 0.6 and 0.3 against 0.02 for bandwidth). A closed-form "capacity law" is within ±25% of the simulator in only 51% of cells, so our pre-registered rule-of-thumb hypothesis fails; we report why. Thai text needs 1.6–8.4× the tokens of English across five tokenizers and cuts capacity by more than the token ratio alone. Code, data manifests and all figures are released.

Keywords LLM serving · capacity planning · queueing · roofline · GPU · simulation · tokenization · Thailand

1 Introduction

Anyone who wants to run an open-weight LLM for a team, a classroom or an agency faces the same question before buying hardware: how many people can one GPU serve? Benchmarks rarely answer it. They report tokens per second at a fixed batch size [1], requests per second under an open-loop arrival process [2], or single-stream decoding speed. None of these is a number of users, and converting between them requires assumptions about how people use a chat system: how long they read and type between requests, how long their prompts are, and how fast a response must start and stream to feel usable.

This paper answers the question in the unit people ask it in. We define the capacity $N^\star$ of a GPU as the largest number of concurrently active users who each alternate between thinking and waiting, such that at least 95% of their requests meet a TTFT and a TPOT target (Section 3). We then build a model of $N^\star$ from first principles (Section 4), a simulator that serves as the reference (Section 5), and calibrate both on public data only (Section 6). With these we map capacity across 17 GPUs from a 12 GB RTX 3060 to a B200, five model sizes, three weight precisions, four workloads and three service levels (Section 7), test which shortcuts work (Section 8), what drives capacity (Section 9), what a user costs (Section 10), and what changes for Thai-language users (Section 11).

Contributions.

1. **A user-level unit with an explicit SLO**, and a ladder that converts between registered users, concurrent users, requests in flight and tokens per second (Figure 2).

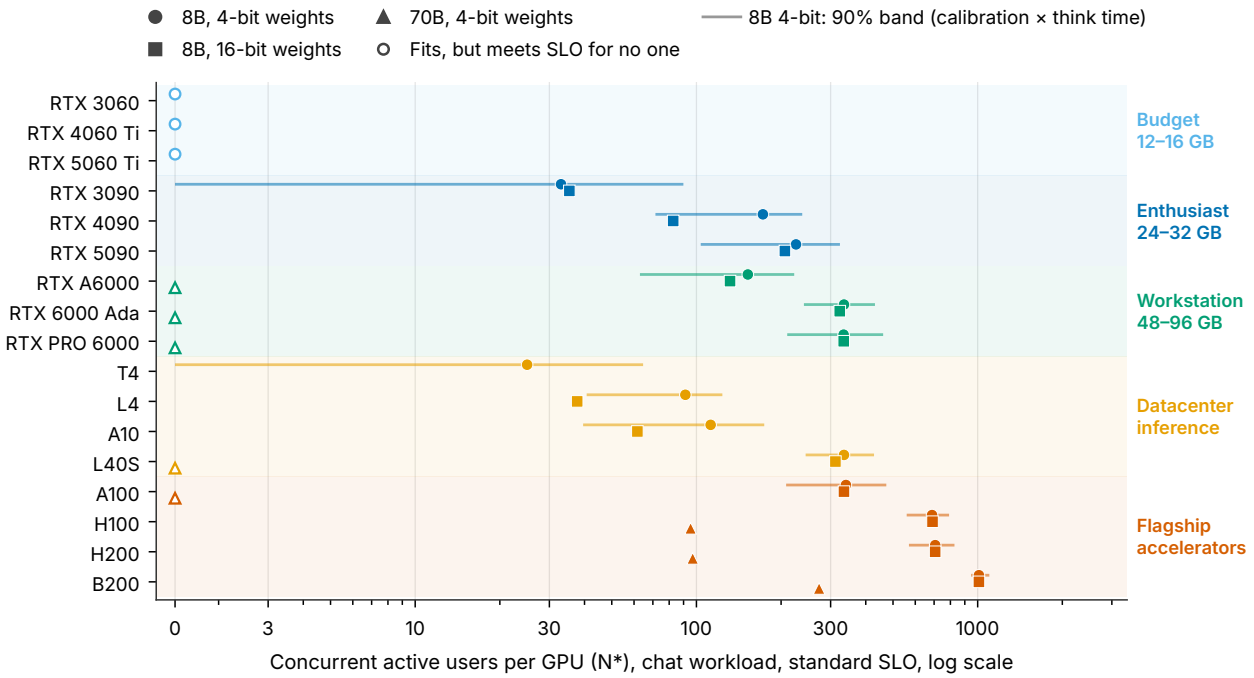


Figure 1: Concurrent active users one GPU can serve for a chat workload (Azure 2023 conversation trace, mean prompt 1,155 and output 211 tokens, mean think time 45 s) under the standard SLO (TTFT $\leq$ 2 s and TPOT $\leq$ 100 ms for 95% of requests). Points are simulator estimates (mean of three seeds); bars span the 90% band from calibration uncertainty combined with think times of 25–90 s. Open markers: the model fits in memory but no number of users meets the SLO. GPUs within a tier are ordered by release.

2. **A capacity model in three layers of fidelity**: a token roofline for one engine iteration; a closed-population birth–death model with a stochastic KV-memory limit, a prefill queue and Poisson prefill interruptions; and a closed-form "capacity law" based on square-root staffing. Seven propositions characterise the regimes (Section 4).
3. **An open iteration-level simulator** whose C kernel is differentially tested against a Python reference and validated against exact queueing results (Section 5).
4. **A five-layer validation** on public data, with the MLPerf hold-out compared only after calibration was frozen in version control (Section 6).
5. **A users-per-GPU atlas** with uncertainty, regime maps and costs per user for buying and renting (Sections 7 and 10).
6. **A Thai lens**: token inflation measured with five tokenizers rebuilt and verified against llama.cpp, and its effect on capacity (Section 11).

Main findings. (i) Capacity spans three orders of magnitude, from zero to about 1,000 concurrent chat users for an 8B model (Figure 1). (ii) Budget 12–16 GB cards fit an 8B model at 4-bit weights but cannot prefill the long-prompt tail of real chat traffic within 2 s; they serve 12–47 users only under a relaxed 5 s TTFT. (iii) Under the standard SLO the binding constraints are prefill compute and TTFT far more often than memory bandwidth; bandwidth efficiency explains about 2% of the variance of $N^\star$ in a global sensitivity analysis. (iv) The closed-form law is useful for orientation but not for sizing: it is within $\pm$25% of the simulator in 51% of cells, failing our pre-registered criterion of 80%. (v) Thai text costs 2.1$\times$ the tokens of English with the Llama 3 tokenizer and reduces capacity to a median 30% of the English value in latency-bound cells. (vi) Per user and month, buying a consumer RTX 5090 or RTX 4090 is cheapest (about \$0.65 at assumed prices); renting a flagship accelerator costs 3–5$\times$ more per user.

2 Background and related work

Serving systems. Modern engines schedule at the granularity of one iteration [3], store the key–value (KV) cache in fixed-size blocks [4], and split long prompts into chunks that ride along with decoding sequences [5]. Disaggregation of prefill and decode [6, 7] and goodput-oriented scheduling sharpen the trade-off between throughput and latency targets. Our simulator follows the iteration-level, paged, chunked-prefill design common to vLLM and its successors.

Performance models and simulators. The roofline model [8] bounds a kernel by memory bandwidth or compute; for transformer inference it explains why decoding at small batch is bandwidth-bound [9, 10]. Vidur [11] simulates LLM serving from operator profiles and is the source of our operator-level validation data. These tools predict throughput and latency for a given arrival process; they do not translate the result into a number of users with think times.

Benchmarks. MLPerf Inference [2] reports throughput under TTFT and TPOT limits for an open-loop Poisson arrival process; LLM-Inference-Bench [1] reports batch latency across accelerators and frameworks; community tables report single-stream llama.cpp speed across consumer cards [12]. We use all three, in different validation layers.

Queueing. Interactive users form a closed population: a user who is waiting does not submit a new request. The machine-repairman model and its product-form generalisations [13, 14] describe such populations; square-root staffing [15] sizes many-server systems. Our capacity law applies the same logic to concurrent decode slots, with load-dependent service from the roofline.

Tokenization and language. Tokenizers trained mostly on English need many more tokens for other languages, which raises cost and latency for their speakers [16, 17]. We quantify this for Thai on a parallel corpus [18] and carry it through to capacity.

3 What is a user?

Definition 1 (Concurrent active user and capacity)**.** *A concurrent active user alternates between a think period (reading, typing; mean Z) and a request. A request has a prompt of L tokens and an output of O tokens; its TTFT is the time from submission to the first output token, and its TPOT is the mean time between subsequent output tokens. For an SLO (θ, τ), the capacity $N^\star$ is the largest N such that, in steady state, at least 95% of requests have TTFT $\leq \theta$ and TPOT $\leq \tau$.*

Three SLO tiers bracket common practice (Table 1). The standard tier streams at 10 tokens per second, faster than most people read; the interactive tier mirrors MLPerf's interactive scenario; the relaxed tier corresponds to a patient internal tool.

Table 1: Service-level objectives. A request passes if both its TTFT and its TPOT are within target; capacity requires a 95% pass rate.

Tier	TTFT θ	TPOT τ	Streaming speed
Interactive	0.5 s	50 ms	20 tok/s
Standard (headline)	2 s	100 ms	10 tok/s
Relaxed	5 s	200 ms	5 tok/s

Figure 2 relates $N^\star$ to the quantities that procurement and benchmarks use. If R is the response time, a user is in flight a fraction $p = R/(R+Z)$ of the time, so $N^\star$ users generate about $pN^\star$ concurrent requests and $N^\star O/(R+Z)$ output tokens per second. Conversions further up the ladder (registered and daily active users) depend on organisation-specific shares and are left as parameters.

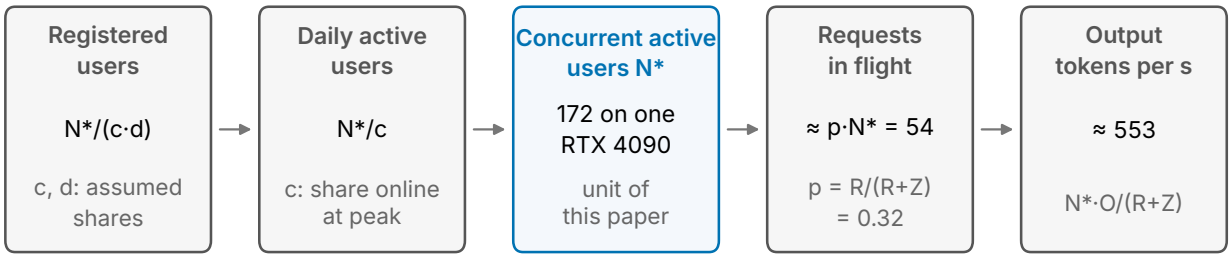


Figure 2: The user ladder with the RTX 4090 example (8B model, 4-bit weights, chat, standard SLO). Only the third box is the unit of this paper; the shares c and d are organisation-specific.

4 A first-principles capacity model

4.1 Token roofline of one iteration

An iteration processes n_d decode tokens (one per running sequence) and n_p prefill tokens, reads KV entries of S cached tokens, and performs causal attention work A for the prefill chunks. With weight bytes W, linear-layer FLOPs per token $F = 2P_r$ (P_r the parameters read per iteration), KV bytes per token κ, effective bandwidth $\beta' = \eta_\beta \beta$ and effective compute $C' = \eta_c C$, we model its duration as

$$t_{\text{iter}} = t_0 + a\, n_s + \text{smax}_k\left(\frac{W}{\beta'},\ \frac{F(n_d + n_p)}{C'}\right) + \frac{\kappa S}{\beta'} + \frac{\alpha A}{C'}, \tag{1}$$

where n_s is the number of scheduled sequences, t_0 and a capture per-iteration and per-sequence host overheads, $\alpha = 4\ell h d_h$ is the attention FLOP coefficient (ℓ layers, h query heads of dimension d_h), and $\text{smax}_k(x, y) = (x^k + y^k)^{1/k}$ with $k = 4$. The functional form (max, sum or smooth max) was chosen by out-of-sample error (Section 6).

4.2 KV memory as a stochastic knapsack

The KV budget is $K = uV - W_{\text{foot}} - V_0$ bytes ($u = 0.9$ of VRAM V, weight footprint W_{foot}, runtime overhead $V_0 = 1.5\,\text{GB}$), or K/κ tokens. A running sequence holds its prompt plus the part of its output generated so far, a random context c with mean m and standard deviation s. We allow n concurrent sequences when $nm + z\sqrt{n}\, s \le K/\kappa$ ($z = 1.645$), which gives

$$B_{\text{kv}} = \left\lfloor \left(\frac{-zs + \sqrt{z^2 s^2 + 4mK/\kappa}}{2m} \right)^2 \right\rfloor, \tag{2}$$

capped by the engine's limit on running sequences (256 by default).

4.3 A closed population of users

Let n be the number of active requests among N users. New requests arrive at rate $\lambda(n) = (N - n)/Z$. With n running, a decode iteration takes $t_d(n)$ from Equation (1) at the mean context, and each arriving prompt adds the marginal time $\Delta(n)$ of its prefill chunks. Prefills therefore occupy a share $u(n) = \lambda(n)\Delta(n)$ of GPU time, a request decodes one token per $t_d(n)/(1 - u(n))$, and its response time is $R(n) = \text{TTFT}_{\text{own}}(n) + (O - 1)\, t_d(n)/(1 - u(n))$. Completions occur at $\mu(n) = \min(n, B_{\text{kv}})/R(n)$. The number of active requests is a birth–death chain with stationary law

$$\pi_N(n) \propto \prod_{j=1}^{n} \frac{N - j + 1}{Z\, \mu(j)}, \tag{3}$$

and an arriving request sees n others with probability proportional to $\pi_N(n)(N - n)$. Given n, we add three stochastic terms: a wait for a sequence slot when $n \ge B_{\text{kv}}$; an $M/M/1$ wait in a prefill queue whose capacity is the token budget left by the decodes; and a Poisson number of prefill interruptions during the request's decode phase, each costing Δ. The pass probability is averaged over 20 quantiles of the prompt and output distributions, and $N^\star_{\text{PF}}$ is found by bisection. We call this the *product-form model*.

4.4 The capacity law

The closed form replaces the distribution of n by its first two moments:

1. **Streams:** $B^\star = \min(B_{\text{kv}}, B_\tau, B_\theta, 256)$, where B_τ is the largest n with mean TPOT $t_d(n)/(1 - u) \le \tau$ and B_θ the largest n with TTFT of the 95th-percentile prompt $\le \theta$; u is solved by fixed-point iteration with $\lambda = N/(R + Z)$.
2. **Duty cycle:** $p = R^\star/(R^\star + Z)$ with $R^\star = R(B^\star)$.
3. **Staffing:** $N_s = x^2/p$ where x solves $x^2 + z\sqrt{1 - p}\, x = B^\star$, the largest N with $Np + z\sqrt{Np(1 - p)} \le B^\star$.
4. **Prefill bound:** $N_{\text{pf}} = \rho^\star C_{\text{pf}}(R^\star + Z)/\bar{L}$, where $\rho^\star$ is the utilisation at which an $M/M/1$ prefill queue exceeds the remaining TTFT slack with probability 5%.

Then $N^\star_{\text{CF}} = \min(N_s, N_{\text{pf}})$, and the regime is the constraint that binds.

4.5 Propositions

Proofs are in Appendix A. Write β', C' for effective bandwidth and compute, b for bytes per weight and $\tau' = \tau - t_0 - W/\beta'$ for the TPOT slack left after streaming the weights.

Proposition 1 (Ridge batch)**.** *Without prefill, the linear layers switch from bandwidth- to compute-bound at* $B_{\text{ridge}} = bC'/(2\beta')$ *sequences, independent of model size.*

Proposition 2 (Fit is not serve)**.** *The elasticity of* B_{kv} *with respect to VRAM exceeds 1, and diverges as the KV budget approaches zero. Capacity inherits this elasticity whenever* B_{kv} *binds.*

Proposition 3 (When memory can bind)**.** *In the bandwidth regime without prefill interference, memory binds before latency if and only if* $K/\beta' < \tau'$ *and the mean context satisfies* $c \geq c^\star = a(K/\kappa)/(\tau' - K/\beta')$*. A GPU that cannot read its whole KV budget within the TPOT slack is never memory-bound.*

Proposition 4 (Product form and its Binomial limit)**.** *Equation* (3) *is the exact stationary law of the birth–death chain. If* $\mu(n) = n/R$ *for all* n*, it is* $\text{Binomial}(N, R/(R+Z))$*, and without prefill interference the law depends on the output length only through its mean.*

Proposition 5 (Staffing)**.** $N_s(B^\star)/B^\star$ *increases with* $B^\star$ *and tends to* $1/p$*; in the Binomial case the staffing rule is asymptotically exact as* $B^\star \to \infty$*.*

Proposition 6 (Token inflation, fluid limit)**.** *If all token counts scale by* $\rho > 1$ *and the binding constraint does not change, the fluid capacity* $B^\star/p$ *scales by a factor in* $[\rho^{-2}, \rho^{-1}]$ *in the memory regime and in* $[\rho^{-1}, 1]$ *in the latency regime.*

Proposition 7 (Pooling)**.** *With* k *GPUs behind one ideal queue, users per GPU* $N_s(kB^\star)/k$ *increase with* k*.*

5 Simulator

The simulator advances one engine iteration at a time. In each iteration it first serves running sequences in admission order (one decode token each, or the next prefill chunk), then admits waiting requests first-come-first-served while the 2,048-token budget, the 256-sequence cap and free KV blocks allow. Memory is allocated in 16-token blocks on demand; when a decode token needs a block and none is free, the most recently admitted sequence is preempted and later recomputed. Iteration time follows Equation (1). Users think for lognormal times (coefficient of variation 1) and draw prompt and output lengths jointly from the workload pool. Capacity is found by bisection on N with common random numbers, recording 2,500 requests after a 300-request warm-up.

The kernel is written in C (about 250 lines) and has a line-by-line Python reference; on identical inputs both return identical per-request metrics, including runs with preemption. Table 2 compares the simulator with exact queueing results; all five cases fall within their 95% confidence intervals.

Table 2: V1: simulator against exact queueing results (eight replications; ± is the 95% CI half-width).

Case	Simulator	Theory
Infinite-server closed loop: mean active (busy periods)	3.7108 ± 0.0222	3.7185
Load-dependent PS closed loop: mean active (busy periods)	3.4127 ± 0.0402	3.3971
Insensitivity: geometric output, same mean	3.4072 ± 0.0618	3.3971
M/G/1 FCFS: mean response time (s)	0.1655 ± 0.0014	0.1661
Little's law (closed loop): X (E[T] + Z) = N	50.0042 ± 0.0165	50.0000

6 Data, calibration and validation

Data. All data are public and were retrieved on 14 November 2026; each file is listed with its SHA-256 hash and source commit in the repository manifest. GPU specifications come from a Wikipedia-derived database [19], cross-checked and corrected where it disagrees with vendor datasheets (for example, it lists 3,360 GB/s for the H200 instead of 4,800 GB/s); every corrected value is flagged in Table 7. Model architectures for Llama models come from Vidur's configuration files [11]. Workloads come from the Azure LLM inference traces of 2023 [7] (Figure 13); RAG and reasoning workloads use stated assumptions (Table 9). Rental prices come from the SkyPilot catalogue (9 providers).

Pre-registration. Hypotheses, SLOs, workloads and every validation criterion were committed to version control before any capacity estimate was computed (tag `pap-locked`); the calibration was frozen (tag `calibration-frozen`) before MLPerf results were compared. Two deviations are logged in Appendix D.

Calibration (V2). We fit $(\eta_\beta, \eta_c, t_0, a)$ for vLLM and TensorRT-LLM jointly across GPUs and models on single-GPU batch latencies from LLM-Inference-Bench [1] (A100 40 GB, H100 SXM, GH200; four 7–8B models; batches 1–256;

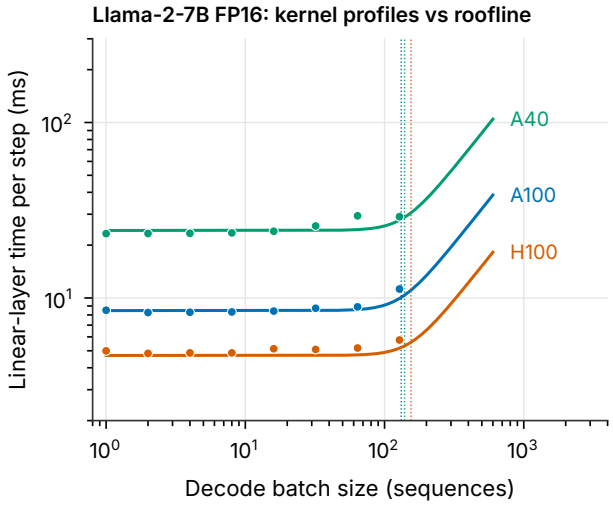


Figure 3: V2a: per-iteration linear-layer time from Vidur kernel profiles (dots) and the fitted roofline (lines) on three GPUs; dotted vertical lines mark the ridge batch of Proposition 1. MAPE is 3.1% (A100), 4.1% (A40) and 6.5% (H100).

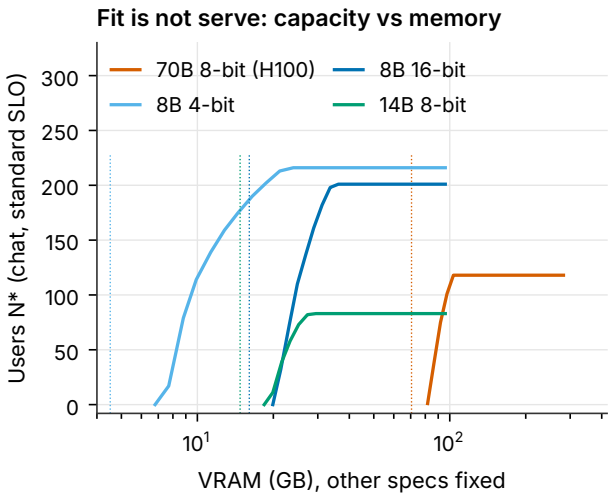


Figure 4: Fit is not serve (Proposition 2): capacity against VRAM with bandwidth and compute held fixed. Dotted lines mark each model's weight footprint; the 70B curve uses H100 bandwidth and compute, the others RTX 4090 speeds. Capacity rises steeply just above it and flattens once another constraint binds.

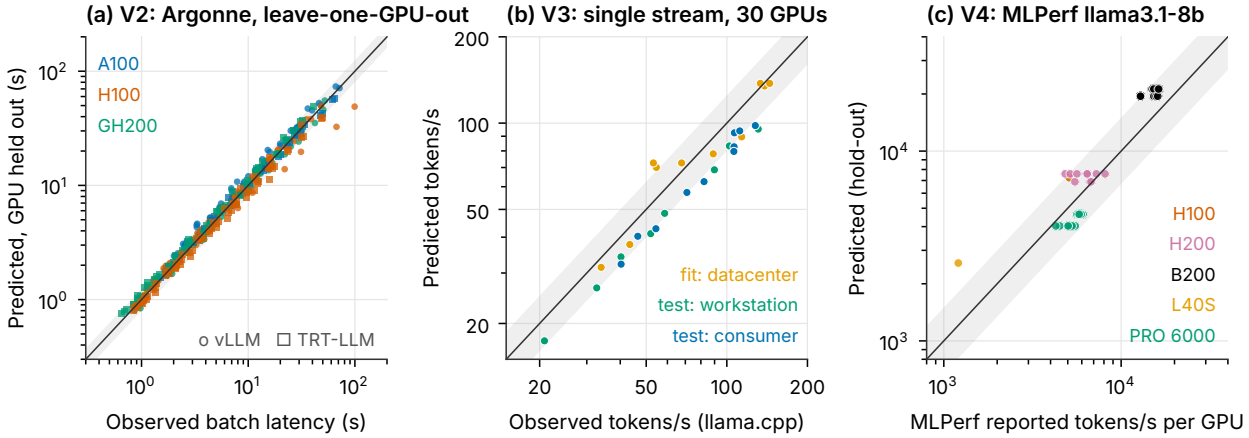


Figure 5: Validation. (a) V2: batch latency with each GPU held out of the fit. (b) V3: single-stream decoding speed with efficiency fitted on 10 datacenter-card rows only and predicted for 18 consumer and workstation rows. (c) V4: MLPerf Inference llama3.1-8b throughput per accelerator under the Server and Interactive limits, compared after calibration was frozen. Shaded: ±25% (a, b) and 0.67–1.5× (c).

128–2,048 tokens). Rows whose KV demand exceeds memory are held out for V2b, and four TensorRT-LLM rows that report latencies below the physical lower bound (all output tokens at 100% of peak bandwidth) are excluded as recording errors. For vLLM we obtain $\eta_\beta = 0.77$ (90% cluster-bootstrap interval 0.68–0.86), $\eta_c = 0.61$ (0.36–1.00), $t_0 = 2.2$ ms and $a = 81$ μs per sequence. Compute efficiency is weakly identified because these benchmarks are mostly bandwidth-bound; we propagate its uncertainty. The smooth max has the lowest leave-one-GPU-out error (9.6% against 10.5% for max and sum).

Cross-tier transfer (V3). Most consumer cards appear only in single-stream benchmarks. Fitting one bandwidth efficiency and one overhead on datacenter cards in the llama.cpp benchmark table [12] and predicting the consumer and workstation cards gives Spearman $\rho_s = 0.99$ and a median absolute error of 19.6% (Figure 5b). This supports transferring efficiencies across price tiers.

Table 3: Validation scorecard. Criteria were fixed in the pre-analysis plan except V2b.

Layer	Comparison	Criterion	Result	Verdict
V1	Simulator vs exact queueing results (5 cases)	inside 95% CI	5/5 inside	pass
V2a	Roofline vs Vidur kernel profiles (A100, H100, A40)	–	MAPE 3.1%, 4.1%, 6.5%	–
V2	Batch latency, vLLM, leave-one-GPU-out / model-out	MAPE $\leq 25\%$	10.4% / 7.6%	pass
V2	Batch latency, TensorRT-LLM, same	MAPE $\leq 25\%$	8.7% / 8.2%	pass
V2b	Memory-pressured rows via simulator (n = 24)	not pre-registered	MAPE 24.8% (naive 20.5%)	weak
V3	Single-stream speed, 18 held-out consumer/workstation cards	$\rho_s \geq 0.9$, median $\leq 25\%$	$\rho_s = 0.99$, median 19.6%	pass
V4	MLPerf llama3.1-8b per GPU, hold-out (n = 48)	0.67–1.5× for $\geq 70\%$	90%, median 1.18×	pass
V4†	Same, first run (output length mis-read)	same	65%, median 1.44×	fail
V5	Closed-form law vs simulator (n = 108)	$\pm 25\%$ for $\geq 80\%$	51%, median 1.06	fail
V5	Product-form model vs simulator	$\pm 25\%$ for $\geq 80\%$	61%, median 1.15	fail

Table 4: Users per GPU, chat workload, standard SLO (simulator; 90% band in brackets combining calibration bootstrap and think times of 25–90 s). "0": the model fits but meets the SLO for no one. Regime: binding constraint for 8B at 4-bit in the closed form.

GPU	8B, 4-bit	8B, 16-bit	70B, 4-bit	Regime
RTX 3060	0	no fit	no fit	ttft
RTX 4060 Ti	0	no fit	no fit	ttft
RTX 5060 Ti	0	no fit	no fit	ttft
RTX 3090	33 [0, 90]	35 [0, 55]	no fit	prefill
RTX 4090	172 [71, 238]	83 [69, 94]	no fit	latency
RTX 5090	226 [104, 324]	206 [120, 260]	no fit	latency
RTX A6000	152 [63, 223]	132 [69, 195]	0	latency
RTX 6000 Ada	335 [241, 431]	323 [233, 404]	0	latency
RTX PRO 6000	334 [210, 461]	334 [214, 464]	0	latency
T4	25 [0, 65]	no fit	no fit	prefill
L4	91 [41, 124]	38 [25, 41]	no fit	latency
A10	112 [40, 174]	62 [50, 69]	no fit	latency
L40S	335 [244, 428]	312 [227, 396]	0	latency
A100 80GB	340 [208, 473]	335 [209, 468]	0	latency
H100 SXM	688 [559, 791]	691 [561, 797]	95 [31, 175]	seqcap
H200	706 [570, 827]	706 [569, 828]	97 [32, 207]	seqcap
B200	1010 [948, 1100]	1010 [947, 1102]	273 [135, 406]	seqcap

MLPerf hold-out (V4). For 48 single-node NVIDIA systems in MLPerf Inference v5.1 and v6.0 (llama3.1-8b, Server and Interactive), we simulate Poisson arrivals with the TensorRT-LLM calibration, the submitted weight precision (FP8 or FP4) and the MLPerf limits (p99 TTFT 2 s and TPOT 100 ms; 0.5 s and 30 ms). 90% of systems fall within 0.67–1.5×, with median ratio 1.18 (Figure 5c). Our first run failed this test (65% inside, median 1.44) because we read MLPerf's `GEN_LEN` as tokens; the reference evaluation code counts characters and the reference system caps output at 128 tokens. We corrected the output length without changing the calibration and report both runs. The L40S system is over-predicted by 2.1× and the RTX PRO 6000 systems under-predicted by about 20%.

Memory pressure (V2b). For the 24 benchmark rows whose KV demand exceeds memory, the simulator's latency error (24.8%) is larger than that of a model that ignores memory (20.5%): our preemption policy is more pessimistic than the engines under heavy oversubscription. Capacity estimates operate below this regime, but we flag it as a limitation.

7 Users per GPU across tiers

Figure 1 and Table 4 report simulated capacity for the chat workload under the standard SLO. Four patterns stand out.

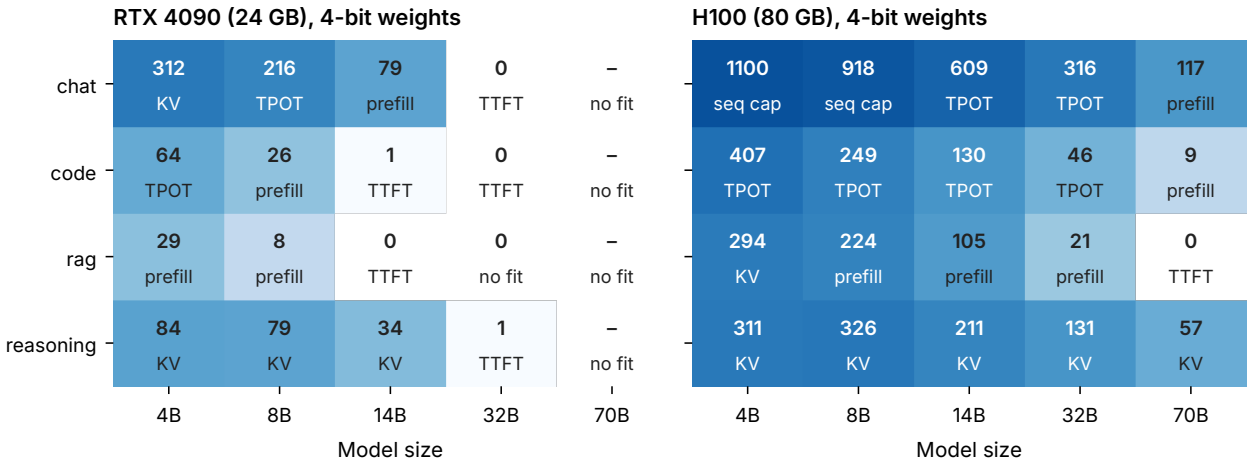


Figure 6: Capacity (product-form model) and binding constraint by workload and model size, 4-bit weights, standard SLO. TPOT: decode latency; KV: memory; prefill: prefill throughput; TTFT: first-token latency of long prompts; seq cap: engine limit of 256 sequences.

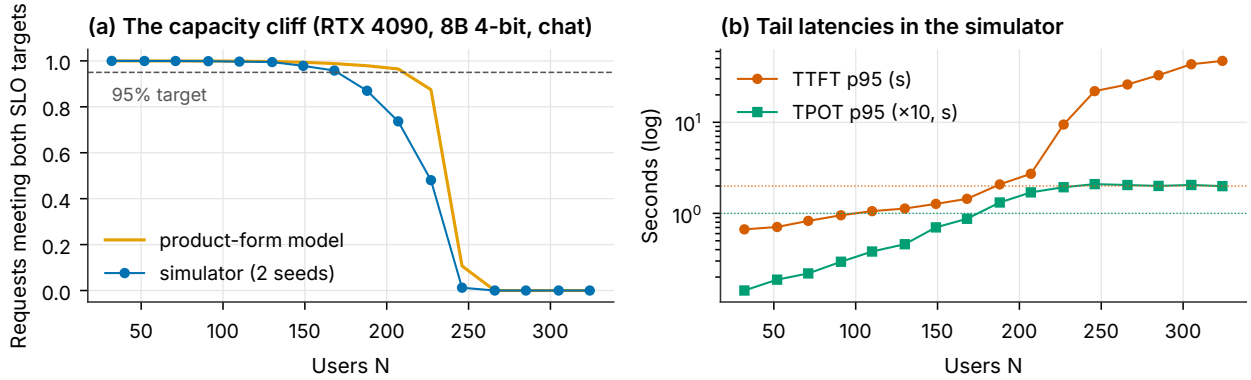


Figure 7: The capacity cliff. (a) Share of requests meeting both targets as users increase, RTX 4090, 8B 4-bit, chat. (b) Tail latencies in the simulator. The pass rate falls from 96% to 48% between 168 and 227 users.

Budget cards fit but do not serve. The RTX 3060, 4060 Ti and 5060 Ti hold an 8B model at 4-bit weights, yet none meets a 2 s TTFT for 95% of chat requests even with a single user: the trace's long prompts (95th percentile about 3,000 tokens) take longer than 2 s to prefill at 26–48 TFLOPS. Under the relaxed SLO they serve 12, 43 and 47 users.

Enthusiast cards are the value tier. An RTX 4090 serves 172 users (90% band 72–238) and an RTX 5090 226 (104–324) with 4-bit weights. At 16-bit weights the 4090 drops to 83 because the KV budget shrinks: Proposition 2 in action.

Workstation and datacenter-inference cards converge. The RTX 6000 Ada, RTX PRO 6000, L40S and A100 all serve 330–340 users with an 8B model, despite a 2.3× range in bandwidth, because prefill compute and TTFT bind before bandwidth does. None of them meets the standard SLO with a 70B model, whose prefill of long prompts exceeds 2 s.

Flagships are limited by the sequence cap. H100, H200 and B200 reach the engine's 256-sequence limit with an 8B model (688, 706 and 1,010 users). Raising the cap to 1,024 adds little in simulation (699, 716 and 1,080), because prefill throughput binds next. Only the flagships serve a 70B model under the standard SLO (95, 97 and 273 users).

Figure 6 shows how the binding constraint moves with workload and model size. Code completion, with long prompts and very short outputs, is TTFT- or prefill-bound almost everywhere; RAG with 6,000-token prompts is prefill-bound; reasoning with 2,500-token outputs is memory-bound on the RTX 4090 and latency-bound on the H100.

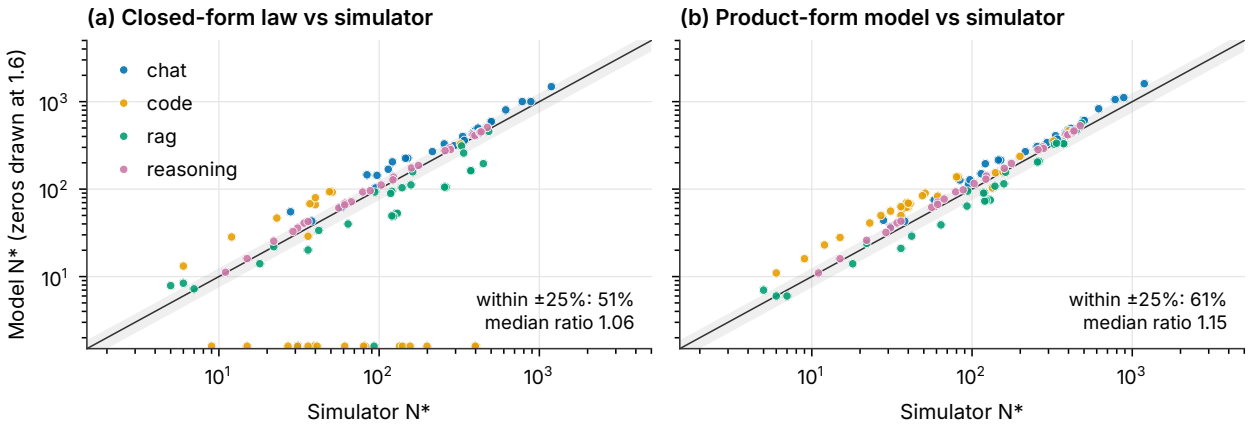


Figure 8: V5: model capacity against simulator capacity on 108 stratified cells. Shaded: ±25%.

Capacity is a cliff, not a slope (Figure 7): on the RTX 4090 the pass rate is 96% at 168 users and 48% at 227, as sequences accumulate, decode iterations lengthen and the prefill queue grows. The product-form model reproduces the cliff but places it about 20% too far to the right, which motivates the next section.

8 How good are the shortcuts?

We compared the closed-form law and the product-form model with the simulator on 108 cells sampled from the grid, stratified by workload and SLO (Figure 8). Monte Carlo variation across seeds is small (median coefficient of variation 4.1%).

The closed form is within ±25% of the simulator in 51% of cells (median ratio 1.06), so **H6 is not supported**. Errors have two sources. First, its TTFT check uses the 95th-percentile prompt as a hard gate; for the code workload this returns zero capacity in 17 of 27 cells where the simulator finds a positive value, because the simulator's pass rate is a smooth average over prompts. Second, both analytic models under-predict the congestion that the prefill queue and slot waits create near the cliff, so they over-predict chat capacity by about 20% (product-form median ratio 1.15, interquartile range 1.05–1.35). Scaling the product-form estimate by a factor fitted on half the cells does not help on the other half (50% within ±25%), because the error depends on the workload rather than being a common bias. We therefore report simulator values for every headline number and treat the closed form as an orientation tool: it identifies the binding constraint and the order of magnitude, not a purchase quantity.

9 What drives capacity?

Global sensitivity. We varied ten inputs over plausible ranges (Figure 9) and computed Sobol indices [20, 21] of the closed-form capacity with SALib [22]. Prompt length dominates (total-order index 0.64 for an RTX 4090 with an 8B model and 0.59 for an H100 with a 32B model), followed by compute efficiency (0.35 and 0.27), think time (0.24 and 0.26) and output length (0.15). Bandwidth efficiency contributes 0.02 and 0.01. The common intuition that LLM serving capacity is a matter of memory bandwidth holds for single-stream decoding, not for a population of users with realistic prompts.

H1: regime flip. We expected chat to be latency-bound and RAG memory-bound for an 8B FP16 model on GPUs with at least 24 GB. Chat is latency-bound on 5 of 13 such GPUs (others: memory 4, sequence cap 3, prefill 1), and RAG is never memory-bound: it is prefill-bound on 10 and TTFT-bound on 3. **H1 is not supported**. In a context sweep under the relaxed SLO, the context at which memory starts to bind in the simulator matches the closed form exactly on the RTX 4090 and L40S and differs by a factor of 2 on the A100 (14B) and H100 (8B), where prefill interference delays it. Proposition 3 explains why memory binds early on small cards: their whole KV budget can be read well within the TPOT slack.

H2: insensitivity. Holding means fixed, 79% of output-length and think-time shape variants change simulated capacity by at most ±10%, just below our 80% criterion (**partly supported**). Exponential think times change capacity by +1% (all within ±10%), deterministic outputs by 0%. By contrast, prompt-length variance matters (a lognormal

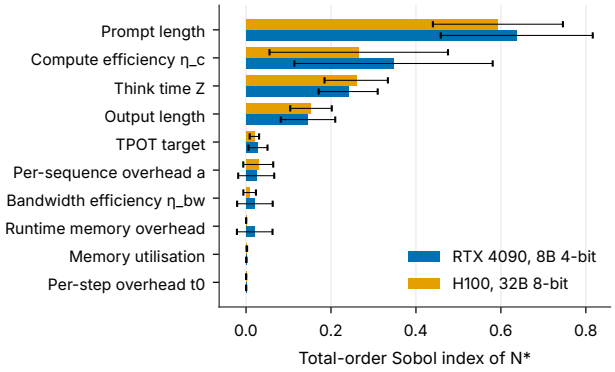


Figure 9: Total-order Sobol indices of capacity (closed form, 2,816 evaluations per configuration; bars: 95% bootstrap intervals).

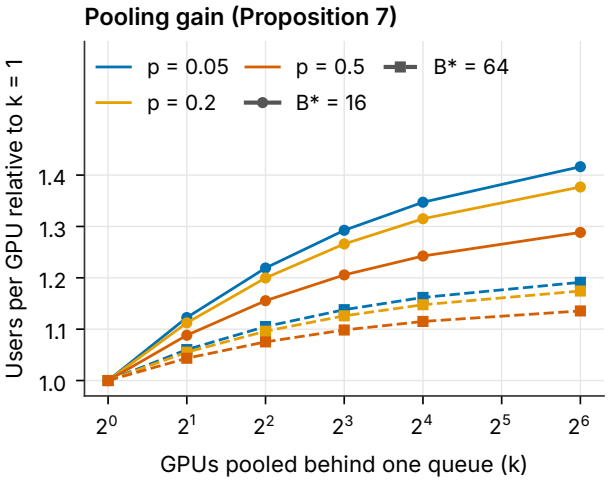


Figure 10: Pooling gain (Proposition 7): users per GPU relative to a single GPU when k GPUs share one queue, for duty cycles p and stream limits $B^\star$.

with coefficient of variation 1.5 lowers capacity by a median 20%), and deterministic think times lower it by 25%, because synchronised users arrive in waves.

H3: fit is not serve. Near the weight footprint, capacity rises faster than VRAM in 71% of sweep points (Figure 4): always for the 8B model on RTX 4090-class speed, but only half the time for a 70B model at H100 speed, where TTFT binds as soon as the model fits. **H3 is partly supported**: the elasticity of Proposition 2 reaches capacity only when memory binds.

Pooling. Sharing a queue across GPUs raises users per GPU (Figure 10): by 42% at 64 GPUs when $B^\star = 16$ and $p = 0.05$, and by about 19% when $B^\star = 64$. Single-GPU numbers are therefore conservative for clusters with ideal load balancing.

10 What does a user cost?

We compute the monthly cost of one GPU in two ways. *Buy*: purchase price plus host share annualised over four years at 5%, plus electricity at 0.7 of rated power, a host allowance and a datacenter PUE of 1.4 for server cards, at US$0.13 per kWh (about 4.4 THB). Purchase prices are assumptions (Table 7); consumer prices are near launch list prices and datacenter prices are market estimates with wide ranges. *Rent*: the median on-demand price per GPU across providers in the SkyPilot catalogue. Dividing by $N^\star$ gives the cost per concurrent user per month (Figure 11a, Table 5); dividing by tokens generated at capacity gives the API price at which self-hosting breaks even (Figure 11b).

The RTX 5090 and RTX 4090 are cheapest per user to own ($0.65 and $0.67 per month), followed by the RTX 6000 Ada ($0.78) and the L40S ($1.04). Flagships cost about twice as much per user for an 8B model but are the only option for 70B, where the B200 is cheapest ($4.67 per user and month). Renting costs 2–5× more per user than owning at full utilisation. Users per dollar is not monotone in price, and the best GPU differs between the 8B model (RTX 5090) and the 70B model (B200): **H4 is supported**. Break-even API prices are $0.08–0.15 per million output tokens to own and $0.15–0.6 to rent for the efficient cards. These assume 24-hour operation at capacity; at lower utilisation the self-hosting cost per token rises in proportion.

11 The Thai lens

We rebuilt five tokenizers from llama.cpp vocabulary files: byte-level BPE for Llama 3, Qwen 2.5/3 and GPT-2, Gemma 4's BPE with byte fallback, and Llama 2's SentencePiece. Each reproduces llama.cpp's expected token ids on all 45 of its test inputs. We tokenized 4,800 English–Thai sentence pairs from six domains of scb-mt-en-th-2020 [18].

Thai needs 1.62× the tokens of English with Gemma 4, 2.14× with Llama 3, 2.45× with Qwen, 4.09× with Llama 2 and 8.40× with GPT-2 (Table 6); dialogue is the least inflated domain and Wikipedia and government text the most.

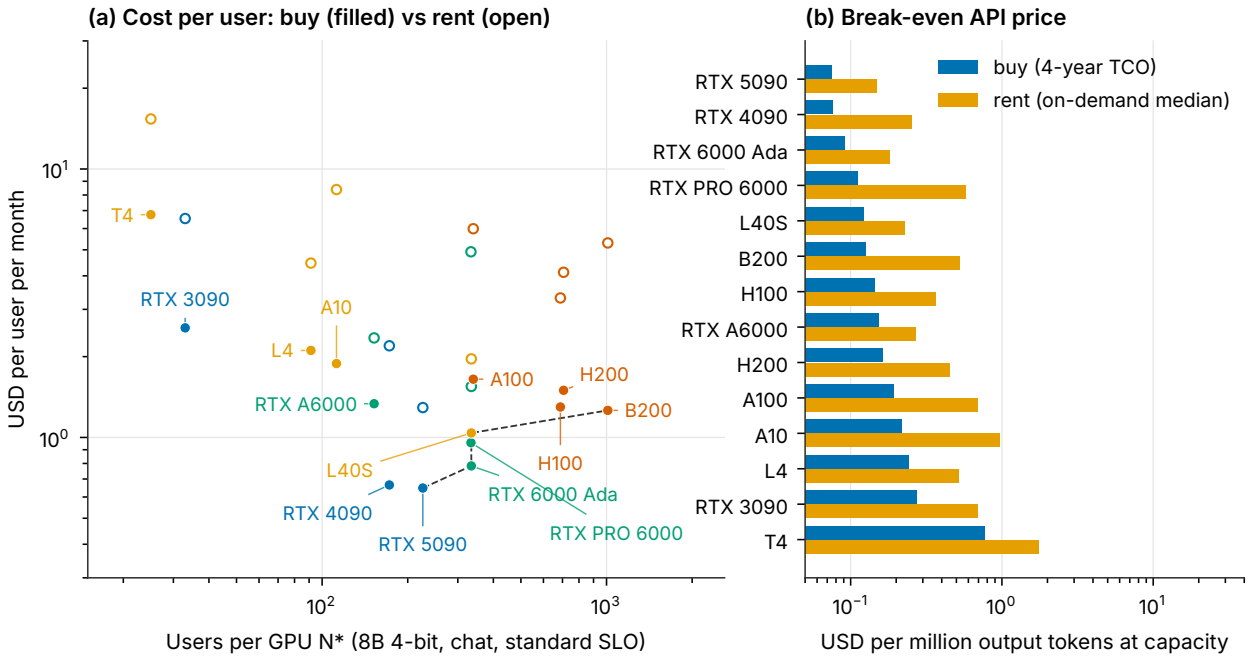


Figure 11: (a) Cost per concurrent user per month for an 8B model at 4-bit weights. Filled: buy (4-year TCO); open: rent; dashed: buy-side Pareto frontier. (b) Cost per million output tokens when running at capacity around the clock; below these prices an API is cheaper than self-hosting.

Table 5: Cost per user, 8B model at 4-bit weights, chat, standard SLO. Buy: 4-year TCO at assumed prices. Rent: median on-demand price. THB at 34 THB/USD.

GPU	$N^\star$	Buy ($/mo)	$/user (buy)	$/user (rent)	$/Mtok (buy)	$/Mtok (rent)	THB/user (buy)
RTX 5090	226	146	0.65	1.29	0.075	0.15	22
RTX 4090	172	115	0.67	2.19	0.077	0.25	23
RTX 6000 Ada	335	262	0.78	1.55	0.091	0.18	27
RTX PRO 6000	334	319	0.96	4.92	0.111	0.57	32
L40S	335	348	1.04	1.96	0.121	0.23	35
B200	1010	1,274	1.26	5.30	0.125	0.52	43
H100 SXM	688	894	1.30	3.31	0.144	0.37	44
RTX A6000	152	203	1.34	2.35	0.154	0.27	45
H200	706	1,058	1.50	4.12	0.163	0.45	51
A100 80GB	340	560	1.65	5.99	0.191	0.70	56
A10	112	212	1.88	8.38	0.217	0.96	64
L4	91	193	2.11	4.46	0.243	0.51	72
RTX 3090	33	84	2.56	6.54	0.272	0.70	87
T4	25	169	6.76	15.36	0.764	1.74	230

Scaling the chat workload by these ratios, capacity falls further than Proposition 6 allows: only 43% of 905 cells lie within its bounds, so **H5 is not supported**. The reason is a change of regime: longer Thai prompts push many cells into prefill- or TTFT-limited operation, where capacity can drop to zero (30% of Thai cells). In latency-bound cells the median Thai/English capacity ratio is 0.30, below $1/\rho = 0.47$ for Llama 3 (Figure 12b). Requiring Thai users to receive the same text speed as English users (TPOT divided by ρ) reduces capacity further on most cards (Table 10).

Illustrative Thai cases. Using simulated Thai capacity for the 8B model at 4-bit weights and buy-side costs at the stated assumptions: an office of 150 concurrent Thai users fits on one RTX 6000 Ada (159 users; about 8,900 THB per month, 59 THB per user), or two RTX 5090 cards (84 users each). A class of 500 concurrently active students needs four L40S (154 each; about 47,000 THB per month), four RTX 6000 Ada (about 35,700 THB) or two H100 (381 each; about 61,000 THB); pooling (Proposition 7) lowers these counts slightly. An agency that must keep data on premises and wants a 70B model needs a flagship accelerator: no 48 GB card meets the standard SLO with long Thai prompts.

Table 6: Thai token inflation ρ (Thai tokens / English tokens) on 4,800 parallel sentence pairs.

Tokenizer	llama.cpp tests	ρ (corpus)	Median [P10, P90]	Range over domains	Chars/token EN / TH
Gemma 4	45/45	1.62	1.38 [0.92, 2.13]	1.20–1.80	4.46 / 2.67
Llama 3 / 3.1	45/45	2.14	1.91 [1.38, 2.80]	1.72–2.31	4.56 / 2.07
Qwen 2.5 / 3	45/45	2.45	2.17 [1.50, 3.24]	1.89–2.69	4.40 / 1.75
Llama 2	45/45	4.09	3.69 [2.50, 5.48]	3.08–4.41	3.94 / 0.94
GPT-2	45/45	8.40	7.33 [4.95, 11.21]	6.15–9.29	4.64 / 0.54

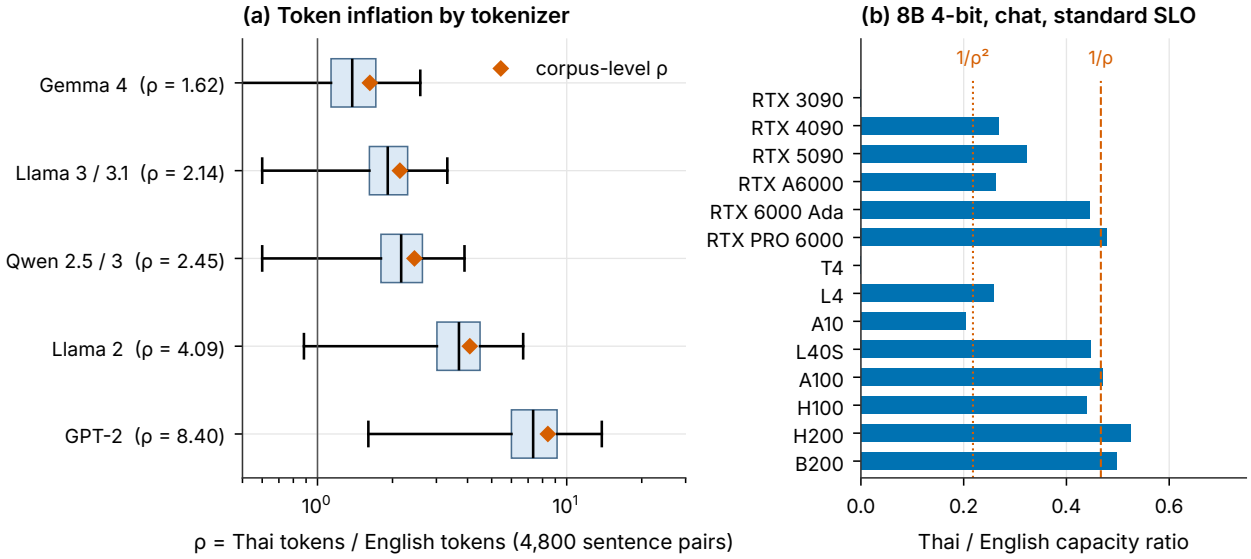


Figure 12: (a) Distribution of per-sentence token inflation by tokenizer; diamonds and axis labels give the corpus-level ratio. (b) Thai/English capacity ratio for Llama-3.1-8B at 4-bit weights; dashed and dotted lines are the fluid bounds of Proposition 6 for the Llama 3 tokenizer.

12 Discussion and limitations

No GPU was measured. Every capacity in this paper is a model or simulator prediction calibrated and validated against public measurements. The validation is broad (five layers, including a hold-out against 48 MLPerf systems), but none of it measured closed-loop user capacity directly on hardware. A short measurement campaign with a load generator that models think times would be the most valuable next step.

Engine generation. The vLLM calibration reflects the benchmark's 2024 software, including a per-sequence host overhead of 81 μs per iteration. Newer engines reduce this overhead, which would raise capacity at large batch. The TensorRT-LLM calibration, with a per-sequence overhead of 19 μs, gives an upper variant (Table 10).

Workload assumptions. Think times (45 s mean for chat) are assumptions; capacity is roughly proportional to $R + Z$, and our bands cover 25–90 s. Prefix caching, which would cut prefill for multi-turn chat, is switched off, so our numbers are conservative for workloads with shared prefixes. RAG and reasoning workloads are parametric.

Prices. Purchase prices are labelled assumptions; rental prices are a snapshot. Conclusions about which tier is cheapest per user are robust across the ranges in Table 7 for the 8B model, but the gap between consumer and workstation cards is within the uncertainty.

Models of the engine. Our simulator's preemption is more pessimistic than real engines under heavy memory oversubscription (V2b), and both analytic models over-predict capacity near the cliff. Compute efficiency is weakly identified, and it is the second most influential parameter; better prefill benchmarks across tiers would tighten every estimate in this paper.

13 Conclusion

One GPU serves anywhere from zero to about a thousand concurrent chat users for an 8B model, and the number depends more on prompts, prefill compute and how long people think than on memory bandwidth. Budget cards fit models they cannot serve under a 2-second first-token target; enthusiast cards deliver the most users per dollar; flagships are needed for 70B models and hit engine limits before hardware limits with small ones. A closed-form law identifies the binding constraint and the order of magnitude but is not accurate enough for sizing; a simulator calibrated on public data is. For Thai users, tokenizer choice matters as much as hardware: the same GPU serves roughly a quarter to a third as many Thai users as English users with Llama 3's tokenizer.

Code and data availability. Code, the data manifest (sources, commits, SHA-256 hashes), the pre-analysis plan with its deviation log, all results and figure scripts are in the accompanying repository. `make all` regenerates every number, table and figure from the raw data.

AI-assistance statement. The study design, code, analysis and draft were produced with the assistance of an AI system (Claude, Anthropic) under the author's direction; the author is responsible for the content. The bibliography was compiled without database access and should be verified against publishers' records before citation.

References

[1] Krishna Teja Chitty-Venkata, Siddhisanket Raskar, Bharat Kale, Farah Ferdaus, Aditya Tanikanti, Ken Raffenetti, Valerie Taylor, Murali Emani, and Venkatram Vishwanath. LLM-Inference-Bench: Inference benchmarking of large language models on AI accelerators. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024. arXiv:2411.00136.

[2] Vijay Janapa Reddi et al. MLPerf inference benchmark. In *Proceedings of the 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020.

[3] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022.

[4] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

[5] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[6] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[7] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

[8] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[9] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2023.

[10] Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, Yan Yan, Beidi Chen, Guangyu Sun, and Kurt Keutzer. LLM inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*, 2024.

[11] Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav S. Gulavani, Ramachandran Ramjee, and Alexey Tumanov. Vidur: A large-scale simulation framework for LLM inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

[12] Xiongjie Dai. GPU-Benchmarks-on-LLM-Inference: Multiple NVIDIA GPUs or Apple silicon for large language model inference? GitHub repository, https://github.com/XiongjieDai/GPU-Benchmarks-on-LLM-Inference, 2024. Commit aa72e0e, retrieved 14 November 2026.

[13] Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, 1975.

[14] Martin Reiser and Stephen S. Lavenberg. Mean-value analysis of closed multichain queuing networks. *Journal of the ACM*, 27(2):313–322, 1980.

[15] Shlomo Halfin and Ward Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29(3):567–588, 1981.

[16] Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[17] Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R. Mortensen, Noah A. Smith, and Yulia Tsvetkov. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[18] Lalita Lowphansirikul, Charin Polpanumas, Attapol T. Rutherford, and Sarana Nutanong. A large English–Thai parallel corpus from the web and machine-generated text. *Language Resources and Evaluation*, 56:477–499, 2022.

[19] voidful. gpu-info-api: GPU specifications as JSON. GitHub repository, `https://github.com/voidful/gpu-info-api`, 2025. Commit f9bf25e, retrieved 14 November 2026.

[20] Ilya M. Sobol'. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55(1–3):271–280, 2001.

[21] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto, and Stefano Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270, 2010.

[22] Jon Herman and Will Usher. SALib: An open-source python library for sensitivity analysis. *Journal of Open Source Software*, 2(9):97, 2017.

A Proofs

Proposition 1. Without prefill, the linear term of Equation (1) compares W/β' with FB/C'. With $W = bP_r$ and $F = 2P_r$ the two are equal at $B = bP_rC'/(2P_r\beta') = bC'/(2\beta')$, in which P_r cancels. With the smooth maximum, the step time at the ridge is $2^{1/k}$ times either term. □

Proposition 2. Let $K' = K/\kappa = (uV - W_{\text{foot}} - V_0)/\kappa$ and let x solve $mx^2 + zsx - K' = 0$, so that $B_{\text{kv}} = x^2$ (ignoring the floor). Differentiating, $dx/dK' = 1/(2mx + zs)$ and

$$\frac{d\ln B_{\text{kv}}}{d\ln K'} = \frac{2K'}{x(2mx+zs)} = \frac{2K'}{2mx^2+zsx} = \frac{2K'}{K'+mx^2} \in [1,2),$$

using $zsx = K' - mx^2$ and $mx^2 \le K'$. Since $d\ln K'/d\ln V = uV/(uV - W_{\text{foot}} - V_0) > 1$ whenever $W_{\text{foot}} + V_0 > 0$, the elasticity with respect to V exceeds 1 and diverges as $uV \downarrow W_{\text{foot}} + V_0$. If $B^\star = B_{\text{kv}}$, then $N^\star$ is increasing in $B^\star$ (Proposition 5) and inherits a positive elasticity, which exceeds 1 when the staffing ratio $N_s/B^\star$ is itself non-decreasing. □

Proposition 3. Ignore prefill interference and the KV safety margin, so $B_{\text{kv}} = K'/c$ for mean context c. In the bandwidth regime the decode step with n sequences takes $t_0 + W/\beta' + an + \kappa cn/\beta'$. Memory binds first iff this time at $n = B_{\text{kv}}$ is at most τ, i.e. $aK'/c + \kappa K'/\beta' \le \tau'$. Since $\kappa K' = K$, this requires $K/\beta' < \tau'$ and then $c \ge aK'/(\tau' - K/\beta') = c^\star$. If $K/\beta' \ge \tau'$ the inequality fails for every c, so latency binds first. □

Proposition 4. A birth–death chain on $\{0,\dots,N\}$ with birth rates $(N-n)/Z$ and death rates $\mu(n) > 0$ is reversible, and detailed balance $\pi(n)(N-n)/Z = \pi(n+1)\mu(n+1)$ gives Equation (3). If $\mu(n) = n/R$, then $\pi(n) \propto \frac{N!}{(N-n)!}\frac{R^n}{Z^n n!} = \binom{N}{n}(R/Z)^n$, which normalises to $\text{Binomial}(N, R/(R+Z))$. Without prefill interference, a running request receives one token per $t_d(n)$, so the active station is processor sharing with a rate that depends only on n, and the users' think station is an infinite server. For this network the stationary distribution depends on the service-requirement distribution only through its mean [13]. □

Proposition 5. With $a = z\sqrt{1-p}$, x solves $x^2 + ax - B = 0$, so $x^2 = B - ax$ and $N_s/B = (1 - ax/B)/p$. Since $x/B = x/(x^2 + ax) = 1/(x + a)$ decreases as x (hence B) increases, N_s/B increases and tends to $1/p$. In the Binomial case the requirement $P(n > B) \leq 0.05$ is, by the de Moivre–Laplace theorem, asymptotically $Np + z\sqrt{Np(1-p)} \leq B$, which is the staffing rule. □

Proposition 6. In the fluid limit $N^\star \approx B^\star/p$ with $p = R/(R + Z)$. Scaling all token counts by ρ scales R by a factor $r \in [1, \rho]$ when every component of R grows at most linearly, so $p(\rho)/p(1) = r(R + Z)/(rR + Z) \in [1, \rho]$. In the memory regime $B^\star = K'/c$ scales by exactly $1/\rho$, giving a ratio in $[\rho^{-2}, \rho^{-1}]$. In the latency regime $B^\star$ depends on contexts only through the KV-read term, which with a token-based TPOT target can only lower $B^\star$ by a factor in $[1/\rho, 1]$; combined with p this gives a ratio in $[\rho^{-2}, 1]$, and $[\rho^{-1}, 1]$ when the KV-read term is negligible, as in our latency-bound cells. The bounds fail when the binding constraint changes, which is what we observe. □

Proposition 7. Ideal pooling of k GPUs is a single system with $kB^\star$ slots, so users per GPU is $N_s(kB^\star)/k = B^\star \cdot N_s(kB^\star)/(kB^\star)$, which increases with k by Proposition 5. □

B Catalogue and provenance

Table 7: GPU catalogue. FP16: dense tensor TFLOPS with FP32 accumulation (GeForce cards run at half their FP16-accumulate rate). Price: assumed central purchase price (USD; range in the repository). Rent: median on-demand USD per GPU-hour (number of providers).

GPU	Tier	VRAM (GB)	BW (GB/s)	FP16	TDP (W)	Price	Rent
RTX 3060	Budget	12	360	26	170	300	0.10 (1)
RTX 4060 Ti	Budget	16	288	44	165	450	–
RTX 5060 Ti	Budget	16	448	48	180	450	–
RTX 3090	Enthusiast	24	936	71	350	800	0.30 (2)
RTX 4090	Enthusiast	24	1,008	165	450	1,800	0.52 (2)
RTX 5090	Enthusiast	32	1,792	210	575	2,500	0.40 (1)
RTX A6000	Workstation	48	768	155	300	4,500	0.49 (3)
RTX 6000 Ada	Workstation	48	960	364	300	7,000	0.71 (2)
RTX PRO 6000	Workstation	96	1,792	312	600	8,565	2.25 (2)
T4	DC inference	16	320	65	70	1,500	0.53 (3)
L4	DC inference	24	300	121	72	2,500	0.56 (3)
A10	DC inference	24	600	125	150	3,000	1.29 (3)
L40S	DC inference	48	864	362	350	8,000	0.90 (5)
A100 80GB	Flagship	80	2,039	312	400	12,000	2.79 (5)
H100 SXM	Flagship	80	3,352	989	700	25,000	3.12 (8)
H200	Flagship	141	4,800	989	700	32,000	3.99 (5)
B200	Flagship	180	8,000	2250	1000	40,000	7.34 (6)

Specifications marked as manual in the repository were corrected from vendor datasheets: A100 80 GB bandwidth, H200 bandwidth, B200 memory and bandwidth, RTX 6000 Ada FP16 throughput, and the L40S. The external benchmarks' hardware is assumed as follows: LLM-Inference-Bench A100 = A100 SXM 40 GB (Polaris), H100 = H100 SXM, GH200 = 96 GB at 4.0 TB/s; the single-stream speeds in V2 imply implausible efficiencies under the alternatives.

Table 8: Models. Parameters are computed from the architecture. KV: KiB per token at FP16. Footprint: weights at 16 and 4.5 bits per parameter (GB).

Model	Params (B)	Layers	d	Heads q/kv	KV	Footprint
Qwen3-4B	4.0	36	2,560	32/8	144	8.0 / 2.3
Llama-3.1-8B	8.0	32	4,096	32/8	128	16.1 / 4.5
Qwen2.5-14B	14.8	48	5,120	40/8	192	29.5 / 8.3
Qwen2.5-32B	32.8	64	5,120	40/8	256	65.5 / 18.4
Llama-3.1-70B	70.6	80	8,192	64/8	320	141.1 / 39.7

Table 9: Workload archetypes.

Archetype	Prompt tokens	Output tokens	Think time	Source
Chat	mean 1,155, median 1,020	mean 211, median 129	45 s	Azure 2023 conversation trace
Code	mean 2,048, median 1,469	mean 28, median 13	15 s	Azure 2023 code trace
RAG	lognormal, mean 6,000	lognormal, mean 300	60 s	assumption
Reasoning	lognormal, mean 1,500	lognormal, mean 2,500	120 s	assumption
Batch	as chat	as chat	0	upper bound

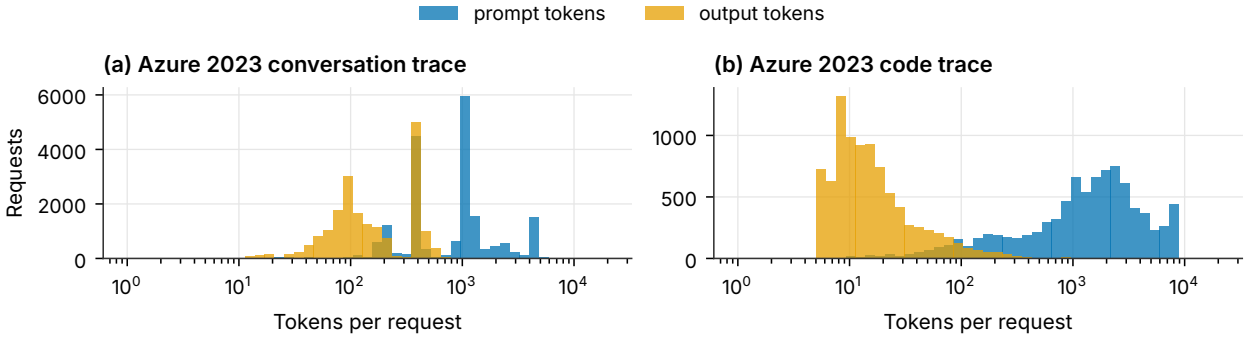


Figure 13: Prompt and output lengths in the Azure 2023 traces (19,366 conversation and 8,819 code requests).

C Sensitivity of headline capacities

Table 10: Capacity for Llama-3.1-8B at 4-bit weights under alternative settings (product-form model; simulator values in parentheses where run). Std: standard SLO; Int./Rel.: interactive/relaxed SLO; Code: code workload; TRT: TensorRT-LLM calibration; Seq 1024: sequence cap raised; Thai: Llama 3 token inflation; Thai char: TPOT target divided by ρ.

GPU	Std	Int.	Rel. (sim)	Code	TRT	Seq 1024	Thai (sim)	Thai char
RTX 3060	0	0	16 (12)	0	0	0	0 (0)	0
RTX 4060 Ti	0	0	55 (43)	0	0	0	0 (0)	0
RTX 5060 Ti	0	0	58 (47)	0	0	0	0 (0)	0
RTX 3090	47	0	108 (88)	0	39	47	0 (0)	0
RTX 4090	216	0	246 (219)	26	193	216	58 (47)	58
RTX 5090	289	76	341 (294)	38	268	289	93 (84)	92
RTX A6000	195	0	269 (218)	23	178	195	51 (40)	51
RTX 6000 Ada	388	271	490 (428)	83	406	388	173 (159)	116
RTX PRO 6000	403	213	531 (425)	68	413	403	193 (164)	145
T4	41	0	97 (75)	0	33	41	0 (0)	0
L4	116	0	173 (152)	11	112	116	30 (0)	20
A10	147	0	200 (161)	15	127	147	30 (0)	30
L40S	373	263	480 (425)	82	390	373	167 (154)	108
A100 80GB	411	213	538 (425)	68	420	411	193 (162)	151
H100 SXM	918	777	965 (736)	249	1050	939	403 (381)	341
H200	962	829	1012 (777)	251	1113	984	504 (432)	385
B200	1390	1338	1468 (1160)	517	1926	1416	692 (567)	630

D Pre-analysis plan, hypotheses and deviations

Table 11: Pre-registered hypotheses and verdicts.

	Hypothesis	Result	Verdict
H1	Regime flip between chat (latency) and RAG (memory)	chat latency-bound 38%; RAG memory-bound 0%	not supported
H2	Insensitivity to output and think-time shape	79% of shape variants within $\pm 10\%$	partly supported
H3	Fit is not serve (elasticity > 1 near the fit point)	71% of points elastic	partly supported
H4	Users per dollar non-monotone in price; argmax differs 8B vs 70B	argmax rtx5090 vs b200	supported
H5	Thai/English capacity inside Proposition 6 bounds	43% of 905 cells	not supported
H6	Closed form within $\pm 25\%$ of simulator in $\geq$ 80% of cells	51%	not supported

Deviations from the plan: (1) four TensorRT-LLM rows below the physical latency bound were excluded before calibration; (2) the first MLPerf comparison read `GEN_LEN` (a character count) as output tokens and lacked accelerator counts for v5.1 rows; it was rerun with 128-token outputs without changing the calibration, and both runs are reported. The analytic model was extended during development (prefill queue, KV safety margin, Poisson prefill interruptions) after comparisons with the simulator on three development cells that are not part of the V5 sample; a Morris screening of the simulator planned in the original design was not run. One TPOT/TTFT gate in the closed form (95th-percentile prompt) was fixed before V5 was computed.