BELIEF-SPACE REPLICA SCALING: THE PRICE OF MEAN-BLIND AUTOSCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

A cluster serving a reasoning model cannot see how much work it has already taken on. A request that has emitted 200 thinking tokens may stop at 210 or run to 20,000, and until it stops it holds KV-cache blocks and a share of decode bandwidth. Utilisation, queue depth and token rate all describe the present, while the quantity that fixes when capacity frees is the residual decode length of everything in flight. We show that this quantity is not measurable with respect to the filtration a serving stack generates, although its conditional law is, which makes capacity control for reasoning traffic a problem in belief space rather than one of regulation. Three consequences follow. Mean residual life obeys $m' = mh - 1$, so a heavy tail alone does not imply a growing residual. Policies measurable in the arrival-count filtration cannot separate two class mixes with different offered token load. And a controller that sees the belief only through its mean pays minimax regret $(1-\rho)^2/(4\rho)$ against mean-optimal capacity cost, which places reasoning serving in the uncertainty-dominated staffing regime where quantile prescriptions displace square-root rules. We then measure the premises on a four-replica vLLM cluster (Qwen3-1.7B, one NVIDIA L40S, 742 decoded prompts). The two classes a deployment mixes differ by $28.3\times$ in median output length, the served mixture reaches $\mathrm{CV} = 1.375$ (95% CI $[1.300, 1.444]$, a lower bound under censoring), and cold start is a median 31.90 s dominated by process bring-up rather than by the accelerator, so the actuation lag is structural. A sweep run *before* any length was measured put the belief-versus-point-estimate gap at exactly zero for $\mathrm{CV} \le 0.53$ and material only above $\mathrm{CV} \approx 0.87$, and the measured mixture clears that threshold with $P(\mathrm{CV} > 0.87) = 1.00$. We release the harness with every constant it needs already measured, and report no controller comparison on hardware: the GPU host became unreachable before that phase.

1 INTRODUCTION

A serving cluster for a reasoning model does not know how much work it is already doing. Utilisation, queue depth, request rate, a token-throughput indicator: every signal an autoscaler conventionally consumes is a statement about the present. The quantity that determines *when capacity will free* is the residual decode length of every in-flight request, and no instantaneous measurement observes it.

That reactive autoscalers therefore react late is not ours to claim. It is TokenScale's published contribution, established on a real GPU cluster against production traces (Lai et al., 2025), and we treat it as background. Nor is planning for autoscaling new: MPC over replica counts is deployed in production, sized against cold start, and already applied inside an LLM serving stack (Zou et al., 2023; Baghel, 2026; Basit et al., 2026). The gap we identify concerns what these controllers plan *over*. Their state is a forecast of the arrival process, and their unit of demand is load that has already materialised, whether a request count or, in recent production systems, a token rate (Lai et al., 2025; Li et al., 2025). Neither says how much work an admitted request has left to do.

Requests are far from interchangeable. Decoding 742 prompts on Qwen3-1.7B under vLLM 0.25.1 on one NVIDIA L40S, the two classes a deployment mixes (long prompts with thinking on, short prompts with thinking off) differ by $28.3\times$ in median output length, 5,016 versus 177 tokens, and the shortest long-class response we saw exceeds the short class's 99th percentile. On the $50/50$ served mixture $\mathrm{CV} = 1.375$ (95% CI $[1.300, 1.444]$), a lower bound under 8.3% right-censoring

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

(§2). Reacting late is structurally expensive here: cold start is a median $31.90\,\text{s}$ (n=9), of which $\approx 13\,\text{s}$ is interpreter start and imports *before* the engine exists, against $0.31\,\text{s}$ to load warm weights. A faster GPU does not remove that lag.

A capacity decision must therefore be taken $\approx 32\,\text{s}$ ahead of the load it serves, against in-flight work whose spread exceeds its mean. This is the setting in which replica scaling stops being a regulation problem and becomes control in *belief space* over residual decode work: the state is a posterior over the prompt-conditioned output-length distribution of everything in flight, propagated through dynamics conditioned on replica actions *and their lag*. Each ingredient has prior art. Output-length *distributions* are established for *scheduling* (Wang et al., 2026a; Zheng et al., 2026; Gan et al., 2026) and are by now predicted continuously as work proceeds (Wang et al., 2026b); length *moments* have even reached the replica count, through Janus's Gaussian-process model of SLO attainment (Zhou et al., 2026). Those moments are window aggregates over recent traffic, and no capacity controller conditions on per-request progress $g_{i,t}$, so none represents the residual work of the requests currently holding the cluster. We contribute:

1. **A belief-MDP formalization over residual decode work** (§3): unobservability (Prop. 1) and three consequences of it, the mean-residual-life identity (Prop. 2), rate-blindness (Prop. 3), and mean-blind regret (Thm. 1). Belief-MDP formulations as such are not novel; the claim is that Eq. (6) is a belief over the right latent variable, and that Thm. 1 prices the approximation deployed autoscalers actually make.
2. **The measured dispersion characterisation** the formalization needs (§2): n=96–329 completions per cell, mean-residual-life curves with bootstrap CIs, an exponential null rejected in 3 of 4 cells, and mean residual life reported explicitly as *not* established to be increasing.
3. **A prediction registered before measurement, then confirmed** (§6): a sweep placed the belief-versus-point SLO gap at exactly 0.000 for $\text{CV} \leq 0.53$ and material only above 0.87, and the measured mixture lands above it with $P(\text{CV} > 0.87) = 1.00$. At $\text{CV} \approx 0.5$ the mechanism would have been ruled out.
4. **Measured testbed constants and a released harness** (§4, §5): the structurally dominated cold start, $0.428\times$ co-residency interference at four replicas, a verified $0.8\,\text{req/s}$ capacity floor, and eight controllers whose belief/point ablation differs in exactly one method.

One experiment is missing, and we would rather name it here than bury it: we report no measured cost–SLO comparison between controllers on hardware (§6).

2 How dispersed is reasoning traffic?

The argument rests on a quantitative premise, so we lead with the measurement. It was taken on the testbed of §4, with output tokens counted by vLLM's `usage.completion_tokens`: thinking tokens included, since the KV cache and the decode budget pay for those too.

We decoded 742 prompts across four cells, prompt class (*long*: MATH Level 4–5 and AIME; *short*: low-step GSM8K, TriviaQA, CommonsenseQA) crossed with thinking mode, prompts disjoint across cells, one generation each (Table 2 in Appendix E, Fig. 2a). Two runs were needed, and the first is a lesson about censoring: at a 4,096-token cap 69.0% of the `long|think` cell truncated, so its upper quantiles all read exactly $4{,}096$. We re-measured the thinking cells at an $18{,}432$-token cap on disjoint prompts, never pooling samples across runs, since the two censor at different levels. Between classes the separation is nearly total, $28.3\times$ in median. Within class the spread survives: CV is 0.61–0.82, p_{99}/p_{50} reaches $6.6\times$ in the uncensored `short|think` cell, and the top decile of requests carries 20–28% of all tokens (Gini 0.31–0.38). Thinking mode multiplies median length by $\approx 6.1\times$ in both classes and is settable per request. Knowing the class is worth a great deal; it is not worth knowing the length.

A controller sees the arriving stream, not one class at a time. Pooling the two classes a deployment mixes, $50/50$ by request count, gives $\text{CV} = 1.375$ $[1.300, 1.444]$: the standard deviation of per-request work exceeds its mean, so the mean is not a summary of this workload. Under a linear replica price and a linear shortfall penalty the cost-minimising capacity is a *quantile* of the work distribution rather than its mean (Thm. 1); that gap grows with dispersion and vanishes when work is

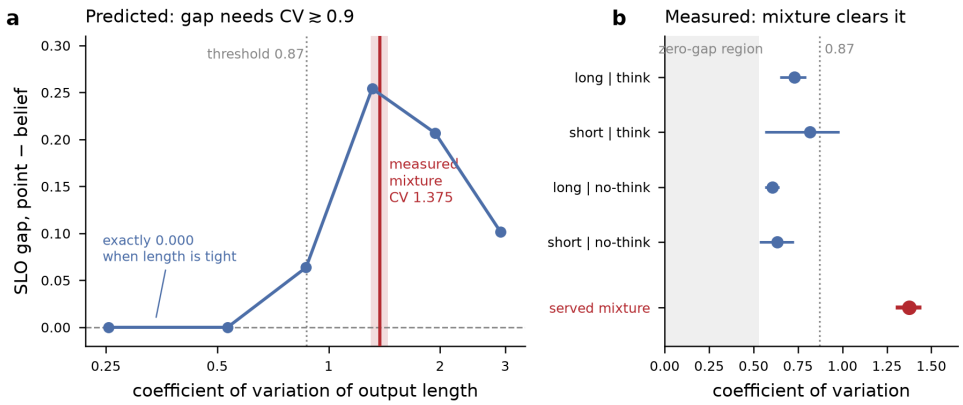

Figure 1: **The measurement lands where the mechanism matters.** (**a**) Belief-versus-point SLO gap against output-length dispersion, from a sweep run *before* any length was measured (§6; simulation, `calibrated:false`, σ-scaled lognormal, S1/S3/S5 $\times$ 2 seeds): exactly 0.000 at CV ≤ 0.53, the falsification check, and material only above ≈ 0.87. The panel locates the threshold in a model, not the magnitude of any gap on hardware. (**b**) Measured CV per cell and for the served mixture (n=96–329 per cell, 8,000 bootstrap resamples). Every cell clears the zero-gap region and the mixture clears 0.87 with $P(\text{CV} > 0.87) = 1.00$; censoring makes it a lower bound.

near-deterministic, so we treat the CV at which it appears as a falsifiable prediction (§6, Fig. 1) rather than a foregone conclusion.

Our sample is small next to a production characterisation, and the direction it shows is not in dispute: ServeGen reports reasoning outputs to be longer and more variable than non-reasoning ones across billions of requests (Xiang et al., 2026), and CASTILLO finds substantial spread even across repeated generations of one prompt (Perez-Ramirez et al., 2025). Neither supplies the object a capacity controller needs, dispersion of the *mixture* one deployment serves, resolved per class, with the thinking flag under our control. One external number bears on the objection that our belief is handed its conditioning for free: sampling 100 completions for each of 1,000 LMSYS-Chat prompts, Zheng et al. (2026) report a mean CV of 1.09 across the *prompt-conditioned* distributions, 78.6% of them above 1. Conditioning perfectly on the prompt, not merely on our two-way class label, still leaves dispersion above the threshold of §6; the residual uncertainty comes from sampling the decoder, not from a coarse taxonomy.

Mean residual life, at the strength the data supports. Fig. 2b shows $e(t) = \mathbb{E}[L - t \mid L > t]$ per cell with percentile-bootstrap bands, and Table 4 the tests (both in Appendix E); the unfavourable parts come first. $e(t)$ is **not** monotone increasing in any cell. In all four it first *falls*, by 28–59%, through the body of the distribution. In 3 of 4 it then turns back up, giving the U-shape that is the qualitative heavy-tail signature, while `long|no_think` decreases throughout and is 18.5% censored, so its upper tail is simply unobserved. That upturn is **not** significant at 95% anywhere: every rise CI includes zero, and at n=96–329 per cell the study is underpowered to establish an increasing $e(t)$. A non-exponential tail *is* established: we tested the memoryless null rather than assuming it and reject the flat-$e(t)$ null in 3 of 4 cells ($p = 0.0005, 0.0065, 0.028$; `long|think` at $p = 0.258$ cannot reject). We therefore adopt only the weaker premise, high dispersion with heavier-than-exponential tails, and §3 needs no more than that. It is also why §3 proves an *identity* rather than asserting monotonicity: the identity holds regardless, and it makes explicit that an increase requires $h_c m_c > 1$.

3 Belief-space replica control

Scaling a reasoning-model deployment differs from scaling a stateless service in one structural way: the work already admitted is not known. We make that precise here, because it determines what a

controller may condition on. Proofs are in Appendix A and each analytical claim is independently re-derived numerically by a released verification script.

Serving model. Time is discretised into intervals of length Δ seconds. The cluster holds $n_t \in \{n_{\min}, \dots, n_{\max}\}$ *active* replicas, each an inference server with continuous batching and a KV-cache budget of B tokens. Arrivals form a point process N with class marks $c \in \mathcal{C}$; a class-c request has output length $L \sim F_c$ in tokens, drawn independently of arrival times, with $\bar{F}_c = 1 - F_c$ and finite mean, and $g_{i,t}$ is the tokens it has emitted by t, observed exactly. With $\mathcal{I}_t$ the in-flight set and $\mathcal{Q}_t$ the admitted-but-not-running set, define

$$Q_t = |\mathcal{Q}_t| \qquad \text{(queue depth)}, \tag{1}$$

$$K_t = \tfrac{1}{n_t B} \textstyle\sum_{i \in \mathcal{I}_t} (\pi_i + g_{i,t}) \qquad \text{(KV-cache occupancy)}, \tag{2}$$

$$R_t = \textstyle\sum_{i \in \mathcal{I}_t} (L_i - g_{i,t}) \qquad \text{(residual decode work)}, \tag{3}$$

with π_i the prompt length of request i and μ the sustained decode throughput of one replica in tokens per interval, so $n_t \mu$ is cluster token capacity. Equations (1)–(2) are counters; (3) is not, and that asymmetry is the subject of Prop. 1. Capacity also does not appear on demand: a scale-up issued at t yields a serving replica only at $t + d$, $d = \lceil \tau_{\text{cold}} / \Delta \rceil \geq 1$, while scale-down takes effect next interval once resident requests drain, so with u_t requested at t,

$$n_{t+1} \;=\; \begin{cases} u_{t-d+1} & \text{if } u_{t-d+1} > n_t \quad \text{(scale-up, lagged)} \\ u_t & \text{if } u_t \leq n_t \quad \text{(scale-down, immediate).} \end{cases} \tag{4}$$

With $d > 0$ the capacity at t is fixed by information from $t - d$: the controller acts on an information set stale by construction and, by Prop. 1 below, incomplete even then.

What a capacity controller can observe. Let the *observable filtration* $\mathcal{F}_t^{\text{o}}$ be generated by everything the serving stack emits up to t: arrival epochs and class marks, prompt lengths, emitted-token counts $g_{i,t}$, completions with realised lengths, queue depths, occupancies, replica states. An implementable controller is a sequence of $\mathcal{F}_t^{\text{o}}$-measurable maps.

Proposition 1 (Residual work is not observable)**.** *Q_t and K_t are $\mathcal{F}_t^{\text{o}}$-measurable. R_t is not: if $\mathcal{I}_t \neq \emptyset$ and some F_c is non-degenerate on $(g_{i,t}, \infty)$, then R_t is not $\mathcal{F}_t^{\text{o}}$-measurable. Its conditional law is, however, determined by $\mathcal{F}_t^{\text{o}}$ and $\{F_c\}$:*

$$R_t \;\big|\; \mathcal{F}_t^{\text{o}} \;\sim\; \bigoplus\nolimits_{i \in \mathcal{I}_t} (L_i - g_{i,t}), \qquad \Pr\big[L_i - g_{i,t} > r \;\big|\; \mathcal{F}_t^{\text{o}}\big] = \tfrac{\bar{F}_{c_i}(g_{i,t} + r)}{\bar{F}_{c_i}(g_{i,t})}, \tag{5}$$

where $\bigoplus$ denotes the convolution of independent summands.

The consequence is not that autoscaling is impossible, only that its sufficient statistic is a distribution. The right side of (5) is the object length prediction established for scheduling (Wang et al., 2026a; Zheng et al., 2026; Gan et al., 2026; Wang et al., 2026b); we take it into the capacity decision.

The belief MDP. Define the belief state

$$b_t \;=\; \Big(\{(c_i, g_{i,t})\}_{i \in \mathcal{I}_t},\; \mathcal{Q}_t,\; K_t,\; \hat{\lambda}_t,\; n_t,\; (u_{t-d+1}, \dots, u_t) \Big), \tag{6}$$

where $\hat{\lambda}_t$ estimates the arrival rate and the trailing action window records warming replicas. By Prop. 1, $(c_i, g_{i,t})$ and $\{F_c\}$ determine the posterior over R_t through (5), so b_t is a sufficient statistic of $\mathcal{F}_t^{\text{o}}$ for the work in flight. Actions carry the lag of (4), and the per-interval cost prices GPU time and SLO violations,

$$c(b_t, u_t) \;=\; \underbrace{p\,(n_t + w_t)}_{\text{GPU-seconds}} \;+\; \underbrace{\beta_{\text{f}} \textstyle\sum_{i \in \mathcal{A}_t} (\text{TTFT}_i - s_{\text{f}})_+ \;+\; \beta_{\text{d}} \textstyle\sum_{i \in \mathcal{I}_t} (\text{ITL}_{i,t} - s_{\text{d}})_+}_{\text{SLO violation}}, \tag{7}$$

with w_t the warming (billed, not serving) replicas, p the price per replica-interval, $\mathcal{A}_t$ the arrivals in interval t, and $(s_{\text{f}}, s_{\text{d}})$ the TTFT and inter-token-latency targets; over horizon H the objective is $\min_\pi \mathbb{E}[\sum_{t=0}^{H} c(b_t, \pi(b_t))]$. Casting cluster control as a belief MDP is standard, and Dierks et al. (2021) already carry a posterior over an admitted workload's future resource usage into an admission decision. We claim no novelty for the formulation, only that (6) is a belief over the right latent variable: a request's remaining decode length.

3.1 CONSEQUENCES

Elapsed generation is not reassuring. Write $m_c(k) = \mathbb{E}[L - k \mid L > k]$ for the mean residual life of F_c and $h_c(k) = f_c(k)/\bar{F}_c(k)$ for its hazard.

Proposition 2 (Growth of expected residual work)**.** *Wherever F_c has a density, $m_c'(k) = m_c(k)h_c(k) - 1$; hence m_c is increasing at k if and only if $h_c(k)m_c(k) > 1$. In particular (i) if F_c is new-worse-than-used, i.e. $\bar{F}_c(k+r)/\bar{F}_c(k) \geq \bar{F}_c(r)$ for all $k, r \geq 0$, then $m_c(k) \geq m_c(0)$ for all k; and (ii) if $\bar{F}_c(x) = (x/x_0)^{-\alpha}$ for $x \geq x_0$ with $\alpha > 1$, then $m_c(k) = k/(\alpha - 1)$ exactly, which is strictly increasing and unbounded.*

For a request that has already generated a lot, expected remaining work can be *larger* than at admission, so waiting does not de-risk it. That is a property of the tail, not of heavy-tailedness loosely construed: for a lognormal F_c, m_c first *decreases*. The identity is the claim and the slogan is false; §2 finds exactly this shape without establishing the increase. Both the identity and the new-worse-than-used class are standard in reliability theory (Barlow & Proschan, 1975), and we restate them only because the loose version is what circulates here.

Rate-driven control is blind to the mix. Let $\mathcal{G}_t = \sigma(N_s : s \leq t)$ be the filtration of *unmarked* arrival counts — the input to rate thresholding and to rate forecasters.

Proposition 3 (Rate-blindness)**.** *Fix an arrival process N and two class-mix processes θ, θ' with $\mathbb{E}_\theta[L] \neq \mathbb{E}_{\theta'}[L]$. Then the law of $(N_s)_{s \leq t}$ is identical under θ and θ', so every $\mathcal{G}_t$-measurable policy induces the same action law under both, while the offered token load $\lambda\,\mathbb{E}[L]$ differs. No policy measurable with respect to $\mathcal{G}_t$ can distinguish the two regimes.*

The proposition is deliberately weak. Consequence metrics *are* informative about the mix, but no earlier than the moment long requests occupy capacity, and by (4) a remedy issued then arrives d intervals later. Reactive control can see the mix, and Lai et al. (2025) improves precisely that; it cannot see the mix *early*, and the mark that would let it is present at admission and discarded.

A point estimate of residual work is not enough. Reduce (7) to its one-shot core: choose n against unobserved work $W \geq 0$ at price p per replica and penalty β per token of shortfall, $C(n) = pn + \beta\mathbb{E}[(W - n\mu)_+]$, with $\rho = p/(\beta\mu) \in (0, 1)$ — a replica costs less than the violations it prevents, else never scale.

Theorem 1 (Mean-blind regret)**.** *The minimiser of C is $n^\star = q_{1-\rho}(W)/\mu$, the $(1-\rho)$-quantile of W in replica-equivalents. Consider any policy whose action depends on the belief only through $\bar{w} = \mathbb{E}[W]$. Then for the mean-matched family $\mathcal{B}_q$: $W = \bar{w}/q$ with probability q and 0 otherwise, $q \in (\rho, 1)$, together with the degenerate $W \equiv \bar{w}$, every mean-blind policy incurs worst-case regret at least*

$$\Gamma(q) \;=\; \frac{p\,\bar{w}}{\mu} \cdot \frac{(1-q)(q-\rho)}{q^2}, \qquad \textit{and} \qquad \sup_{q \in (\rho,1)} \frac{\Gamma(q)}{p\bar{w}/\mu} \;=\; \frac{(1-\rho)^2}{4\rho} \;\; \textit{at} \;\; q^\star = \frac{2\rho}{1+\rho}. \tag{8}$$

$n^\star$ is the newsvendor critical fractile (Arrow et al., 1951), and bounding the loss of a policy that knows only the mean is in the spirit of Scarf (1958); we invoke both to fix what optimal capacity *is*, and claim only the price Eq. (8) puts on the approximation an autoscaler actually makes. The bound is an identity rather than an asymptotic: it vanishes at $q = 1$ and at $q \leq \rho$, and grows without bound as SLO penalties come to dominate GPU price (80% of mean-optimal capacity cost at $\rho = 0.2$). It also bounds what we may claim, since it lower-bounds *mean-blind* policies only. The baseline it predicts against is therefore a *tuned-fractile* point estimate (Zhang et al., 2026a).

Which staffing regime this is. Sizing capacity against uncertain demand has a settled theory, and it is worth saying where reasoning serving falls in it. For a many-server queue with *known* offered load R, the classical prescription is square-root safety staffing, $n \approx R + \beta\sqrt{R}$, hedging stochastic fluctuation around a known mean (Halfin & Whitt, 1981; Borst et al., 2004). Bassamboo et al. (2010) show that this logic survives only while the coefficient of variation of demand stays below $1/\sqrt{R}$; above it a newsvendor quantile of the kind in Thm. 1 becomes the accurate prescription. Their uncertainty is in the arrival rate and ours in per-request work, so the correspondence is structural rather than exact, but the comparison is one-sided enough to survive that: $1/\sqrt{R}$ is 0.50 at the four

replicas of our testbed and still only 0.10 at a hundred, against a measured $\mathrm{CV} = 1.375$. Reasoning serving is no marginal case here, and the approximation Eq. (8) prices is one shipped systems make: llm-d's Workload Variant Autoscaler charges each in-flight request `AvgInputTokens + AvgOutputTokens` and divides arrival rate by a per-replica capacity derived from those averages (The llm-d Project, 2026).

The horizon is forced. Writing $\mathcal{R}_t(k)$ for the replica counts reachable at $t + k$, (4) gives $\mathcal{R}_t(k) = \{n \le n_t\}$ for $k < d$: if the constraint binds at $t + k$ the last admissible decision was at $t + k - d$. A controller with horizon below d cannot represent the action that prevents it, and any planner must predict R_{t+d}, which by Prop. 1 requires the belief rather than a measurement. Each consequence above predicts an experiment, paired in Appendix C with the scenario that tests it. The sharpest is Prop. 3: on constant-rate, shifting-mix traffic (S3), rate-driven and rate-forecasting controllers must act as if nothing happened while offered token load rises.

4 Testbed and workload

What a replica is here. The cluster is four co-resident vLLM 0.25.1 replicas on a single NVIDIA L40S (46 GB), each serving Qwen3-1.7B in bf16 at `--gpu-memory-utilization 0.21` behind a router that proxies the OpenAI-compatible streaming API and exposes cluster state to a controller (launch flags in the reproducibility statement). Each replica sizes an identical $55{,}200$-token KV cache (5.9 GiB); four occupy $40{,}409$ of $46{,}068$ MiB. This is a scaled-down analogue of a multi-GPU deployment, not a substitute: co-residency makes replicas contend for streaming multiprocessors, so we measure that contention rather than assume it away. Every figure and table for this section is in Appendix E, where Table 3 collects the constants.

The actuation lag is structural. Cold start to serving-ready is a median 31.90 s (IQR $[31.82, 32.40]$, n=9; a second session reproduced 31.66 s). The decomposition matters more than the value (Fig. 4a): roughly 13 s elapses in interpreter start and imports *before* the engine is constructed, while loading 3.22 GiB of warm weights takes under a second. Process bring-up, not the accelerator or I/O, dominates. Bring-up must also be *sequential*, since four engines initialising concurrently each profile free memory while the others allocate and all but one fail KV sizing, so four replicas take 126.8 s. Teardown is cheap (0.54 s), the asymmetry Eq. (4) encodes.

Co-residency interference. Driving k resident replicas at matched per-replica concurrency, aggregate throughput peaks near 801 tok/s at k=3 and *declines* 11% to 712 tok/s at k=4, which at concurrency 8 is $0.428\times$ of four times a single replica (Fig. 3). A control experiment separates the candidate causes: with four replicas resident but only one driven, throughput is 0.96–$1.12\times$ that of a lone replica, so residency is nearly free and the loss is simultaneous decode contention. The fourth replica has negative throughput value here, so a controller assuming linear scaling pays for capacity that reduces service.

Capacity is a floor, not a saturation point. We drove the cluster open-loop from a Poisson generator, since a closed-loop client throttles itself when the server slows and can never expose a queueing violation. Against an SLO of TTFT $p_{95} \le 2000$ ms, TBT $p_{95} \le 200$ ms, zero errors and completion ratio ≥ 0.9, all five rates from 0.1 to 0.8 req/s passed with large headroom. We report 0.8 req/s as a *verified floor*: the GPU-time budget stopped the sweep before a violation, so true capacity is higher.

Workload. The corpus is $4{,}397$ real prompts in the two classes of §2, with class membership taken from the source annotation rather than from a measured output length, emitted as five scenarios (S1–S5) in dev/test twins at fixed seeds. **S3 realises Prop. 3**: p_{long} steps $0.15 \to 0.85$ partway through the horizon while the arrival timestamps stay *byte-identical* to a fixed-mix twin at the same seed, so the two traces are one arrival realisation with different prompts attached and a rate-measurable controller provably sees nothing. S2 steps the rate $3\times$ with the mix fixed, the control in the other direction, on which a reactive controller *should* do well. Appendix F gives the hashes, the independence argument, the replay client and the scenario table.

5 THE HARNESS

The formalization predicts experiments (Appendix C), and we built the instrument that runs them. Eight controllers implement `decide`(state history, t) $\rightarrow$ target replicas over one interface: the six vLLM metric families verified present on the live server, plus a router-side view of per-request tokens forwarded and the thinking mode the router set. That view exists because the router proxies the token stream, and it carries nothing about any request's true total length. The baselines are calibrated rather than straw: **static-k**, whose rule-based kin have been reported to beat six deep-RL algorithms on cost across every workload tested (Zhang et al., 2026a); **HPA**, implemented from the Kubernetes autoscaler's primary documentation; **queue/concurrency**; **LeadingIndicator**, in the spirit of TokenScale's token velocity (Lai et al., 2025) but reproducing neither its disaggregated deployment nor its thresholds, so nothing here measures TokenScale; and **ForecastThreshold** (Baghel, 2026; Jaiswal et al., 2025), which forecasts to $t + d + H$ using the *measured* $d = 31.9\,\text{s}$ and so is not handicapped by ignoring actuation lag. Its limitation is its input: a forecaster extrapolates a scalar series, and on S3 that series is pathwise identical to a trace where nothing happens.

BeliefMPC plans over a receding horizon in a dynamics model conditioned on the replica action and its lag, with the belief state of Eq. (6); residual work enters as Monte-Carlo draws from each in-flight request's conditional tail $\bar{F}_{c_i}(g_{i,t} + r)/\bar{F}_{c_i}(g_{i,t})$. **PointMPC** is the ablation and differs in *exactly one method*: same dynamics, action set, cost, horizon and tuning protocol, with residual work replaced by the scalar class mean $- g_{i,t}$. The learned dynamics correction is offered to both arms, so the ablation cannot confound belief with better dynamics. **Oracle** reads true lengths and is excluded from fronts. Tuning is dev-only and enforced in code, on the same budget and objective for every controller. Appendix B gives the HPA semantics we did and did not reproduce, the tuning protocol, and the metric-name guard that stops a baseline silently reading KV pressure as zero.

6 A REGISTERED PREDICTION, AND THE EXPERIMENT WE DID NOT RUN

The prediction, and its confirmation. The load-bearing empirical result here is not a win over a baseline but a prediction that preceded the measurement. Before any output length had been measured, we ran an adversarial sweep on our own mechanism: holding class medians fixed and scaling the length distribution's σ, we re-ran both arms at their tuned parameters and recorded the SLO gap against CV. The sweep was designed to *falsify*, since had the gap persisted at low CV the belief arm would have been winning for some other reason. It did not persist. The gap is exactly 0.000 at $\text{CV} = 0.25$ and 0.53, first appears at 0.87 ($+0.064$) and peaks at 1.31 ($+0.254$), which is the right shape: with near-deterministic lengths a point estimate *is* the distribution. That made the contribution conditional on the world, since the mechanism matters only if real reasoning traffic has $\text{CV} \gtrsim 0.9$.

It does (Fig. 1). The served mixture has $\text{CV} = 1.375$ $[1.300, 1.444]$, so $P(\text{CV} > 0.87) = 1.00$, and censoring makes 1.375 a lower bound. Every individual class clears the zero-gap threshold ($P(\text{CV} > 0.53) \geq 0.978$), though only `short|think` has appreciable mass above 0.87, so a single-class workload would sit lower on the curve. Two limits bound what this buys: the curve is simulated, locating the *threshold* in a model and not the *magnitude* of any real gap, and the sweep scaled a lognormal σ whereas the measured mixture's dispersion comes from mixing two separated classes, the same statistic by a different mechanism.

Simulation: a labelled pipeline check, not a result. We built a discrete-event multi-replica simulator exposing the real router's state interface and ran 307 dev-split tuning runs plus a 165-run sanity matrix. **None of its numbers is a result of this paper**, for a structural reason rather than one of calibration quality: the simulator draws lengths from a length model, and `BeliefMPC`'s belief is a belief about *that same object*, sampled from it. The belief arm therefore holds a perfectly specified prior, which no deployment offers, so a gap there is guaranteed by construction; and the thinking-mode flag in the emitted traces is a noiseless proxy for the length class, so the "prompt-conditioned" belief gets its conditioning free. The simulator establishes only that the pipeline runs, the arms are behaviourally distinguishable, and the mechanism moves in the predicted direction. It also produced one result a reviewer should hold against us: on the aggregate dev objective, static-k occupies four of eight front points and no adaptive controller dominates it, reproducing the calibrated-baseline warning of Zhang et al. (2026a) in our own harness (Appendix E.2).

We report no controller comparison on real hardware: no measured cost–SLO Pareto front, burst response, or belief-versus-point ablation on the cluster, and the real-router backend is written and contract-tested but never executed. That bounds our empirical claims to four: the dispersion characterisation, the testbed constants, the formalization, and the released artifact.

7 Related Work

Serving engines and request-level schedulers: a different knob. Modern engines raise throughput by managing the KV cache and the batch rather than the replica count (Kwon et al., 2023; Yu et al., 2022; Agrawal et al., 2024b; Zhong et al., 2024; Patel et al., 2023; Qin et al., 2024; Zheng et al., 2023a; Wu et al., 2023; Sun et al., 2024), with theory for non-clairvoyant KV-cache scheduling (Feng et al., 2026b;a). All act *within* a fixed replica set, deciding which requests run next rather than how much capacity exists.

Output-length prediction: predicts, but does not plan. Because decode length is unknown at admission, a substantial literature predicts it (Jin et al., 2023; Zheng et al., 2023b; Qiu et al., 2024), including chain-of-thought length from hidden states for reasoning models (Yang et al., 2026). Closest to our premise, recent work argues that a *point estimate* misrepresents a stochastic decoding process, prompt-conditioned length being a heavy-tailed *distribution* (Wang et al., 2026a), and builds schedulers on tail-inflated or distributional cost models (Zheng et al., 2026; Gan et al., 2026; Li et al., 2026; Sundaresan, 2026). STAR predicts each request's *remaining* length continuously from hidden states, exactly the latent quantity of Prop. 1, and migrates work between decode instances with it (Wang et al., 2026b). This line establishes our premise rather than competing with it, and a good residual-length estimator would sharpen our belief rather than replace it. Almost none of it carries the uncertainty into a *capacity* decision; the prediction feeds an ordering, a batching, a migration, or a memory reservation inside a fixed replica set, as in MAPS, which calibrates predicted lengths into coverage-guaranteed bounds spent across a *fixed* decode pool (Zhang et al., 2026b).

Autoscaling: plans, but is length-blind. General-purpose autoscalers regulate a utilization signal with a stabilization window and tolerance band (Augustyn et al., 2024; Rzadca et al., 2020) and have moved toward forecast-then-optimize designs, several explicitly uncertainty-aware (Pan et al., 2023; Hang et al., 2024; Zou et al., 2023; Xue et al., 2022; Meng et al., 2024; Nguyen et al., 2025). ADAPT is closest on mechanism, optimizing replica counts over a rolling horizon with MPC and sizing the lookahead from an online provisioning-delay estimate (Baghel, 2026); its dynamics are a forecaster plus a delay estimate rather than a learned action-conditioned model, and its state carries neither KV-cache pressure nor residual decode work. Learned resource managers follow the same pattern (Qiu et al., 2020; Mao et al., 2018). The nearest formulation is the cluster-admission POMDP of Dierks et al. (2021), which does hold a posterior over an admitted workload's future resource usage, though over a VM's scaling parameters and decided once at admission. Otherwise the uncertainty all of these model is over the *arrival process*, not how much work each admitted request contains. The operations-research line they descend from is older and sharper on exactly this point: square-root safety staffing hedges fluctuation around a *known* offered load (Halfin & Whitt, 1981; Borst et al., 2004), and Bassamboo et al. (2010) identify the regime where that hedge yields to a newsvendor quantile (Arrow et al., 1951; Scarf, 1958), of which Thm. 1 is a serving-side instance.

LLM-specific autoscalers, and the closest systems. Chiron scales on hierarchical backpressure (Patke et al., 2025), Aladdin sizes workers from latency models (Nie et al., 2024), ENOVA targets serverless multi-GPU deployments (Huang et al., 2024), operator-granularity scaling reallocates at finer units (Cui et al., 2025), and a second group attacks the actuation delay itself (Zhang et al., 2024; Fu et al., 2024; Yousefijamarani et al., 2025). Five systems deserve explicit comparison. TokenScale diagnoses the failure mode motivating this work, that utilization and request counts are lagging indicators giving slow reaction, SLO violation and over-provisioning, and answers with *Token Velocity* (Lai et al., 2025); HeteroScale reaches the same conclusion across tens of thousands of production GPUs, picking decode tokens per second (Li et al., 2025). WarmServe and SageServe pre-position capacity by prewarming weights (Lou et al., 2025) or by an ILP over routing and VM scaling (Jaiswal et al., 2025), and DualScale actuates per-iteration prefill frequency by MPC (Basit et al., 2026). llm-d's Workload Variant Autoscaler sizes replicas from an M/M/1/k model whose per-request charge averages recent input and output tokens (The llm-d Project, 2026). Closest of all

is Janus, whose Elasticity GP takes the mean *and variance* of recent output lengths, predicts SLO attainment from workload and instance count, and searches the count (Zhou et al., 2026). Dispersion already reaches the capacity decision, so we claim novelty for neither the diagnosis, nor MPC in a serving stack, nor dispersion-awareness. What survives is the conditioning. A token rate counts work already done rather than work still owed, forecast-then-act assumes homogeneous per-request work, and Janus's moments are *window* aggregates describing the population a request is drawn from, not the state of one in flight. None uses $g_{i,t}$, so none can form the conditional tail (5) that Prop. 1 identifies as the sufficient statistic for R_t; Janus is variance-aware rather than mean-blind, so it sits outside Thm. 1, which is why our ablation is a tuned fractile rather than a mean. Simulators, analytical planners and production workload characterisations serve here for workload construction, not as baselines (Agrawal et al., 2024a; Chen et al., 2026; Wang et al., 2024; Xiang et al., 2026).

Reasoning-model cost and learned world models. Output length is not incidental for reasoning models: RL-trained reasoners emit long chains of thought (DeepSeek-AI et al., 2025), test-time compute is a scaling axis (Snell et al., 2024; Muennighoff et al., 2025), and much work budgets thinking tokens (Chen et al., 2024; Han et al., 2024; Sui et al., 2025). That line *reduces* length from the model side; here it is a random variable to *plan around* from the serving side. Learning a dynamics model and planning inside it is well established in control (Hafner et al., 2018; 2019; 2023; Schrittwieser et al., 2019; Chua et al., 2018; Hansen et al., 2023; de Boer et al., 2005; Williams et al., 2015), but not for replica scaling.

8 Limitations and next steps

No controller comparison on hardware (§6): no cost–SLO Pareto front, burst response, or belief-versus-point ablation was measured. **Scale:** one L40S with four co-resident replicas of a 1.7B model is a small analogue of production; the $0.428\times$ interference is specific to co-residency, and a larger model changes the KV/compute balance and the cold-start decomposition. **Sample size and censoring:** 96–329 completions per cell establishes dispersion and rejects an exponential tail in three cells, but cannot resolve $e(t)$ in the far tail; two cells are censored, so their upper quantiles and the mixture CV are lower bounds, and 0.8 req/s is the largest *tested* rate. **Prompt conditioning:** our belief conditions on a class the traces supply noiselessly, whereas a deployment needs a length predictor with real error. The external dispersion number of §2 bounds what perfect conditioning could buy but does not substitute for measuring our own predictor; that error-injection sweep is implemented and unrun, and is the most immediate next step, the only one that does not need the GPU back. **The formalization's reach:** Thm. 1 bounds *mean-blind* policies only, and a fixed-upper-fractile controller can match the optimum in any single regime, which is why the correct baseline is `PointMPC`: if it matches the belief planner where the bound is large, the mechanism is falsified.

9 Conclusion

For reasoning traffic the admitted work is not observable while its conditional law is, which makes replica scaling control in belief space over residual decode work rather than regulation of a measured signal. Three things follow. Expected residual work grows only where $h_c m_c > 1$, so a heavy tail is by itself no reason to expect a long request to get longer. No policy measurable in the arrival-count filtration can separate two class mixes, however good its forecaster. And a controller that compresses the belief to its mean pays $(1-\rho)^2/(4\rho)$ of mean-optimal capacity cost, a charge that is not hypothetical: llm-d's reference autoscaler sizes replicas from averaged input and output tokens.

Both premises hold on a real vLLM cluster. The two classes a deployment mixes differ by $28.3\times$ in median output length and the served mixture reaches $\mathrm{CV} = 1.375$, above a threshold fixed before any length was measured and far above the $1/\sqrt{R}$ boundary at which quantile capacity displaces square-root staffing. Cold start is a median 31.90 s spent mostly before the engine object exists, so the lag is structural and no faster accelerator removes it. The controller comparison remains, on a harness released with every constant it needs already measured: the lag, the co-residency interference, the capacity floor, and the length distributions the belief is a belief about.

REPRODUCIBILITY STATEMENT

The harness, the measurement artifacts and the verification scripts are available for review, anonymised, at `https://anonymous.4open.science/r/REPLACE-BEFORE-SUBMISSION`. Every number in this paper traces to one of those artifacts, and they carry per-field provenance flags rather than a single global one.

Testbed. vLLM 0.25.1 with torch 2.11.0+cu130, model `Qwen/Qwen3-1.7B` in bf16, on a single NVIDIA L40S (46,068 MiB) with a 16-vCPU / 124 GiB host. Launch flags per replica: `--dtype bfloat16 --max-model-len 8192 --gpu-memory-utilization 0.21 --max-num-seqs 64 --enforce-eager --reasoning-parser qwen3`; the tail-cap re-measurement used `--max-model-len 20480`. Required environment: `VLLM_USE_FLASHINFER_SAMPLER=0` and `VLLM_ATTENTION_BACKEND=FLASH_ATTN` (no CUDA toolkit on the host, so JIT compilation fails). Replicas must be brought up sequentially or all but one fail KV sizing.

Three interface facts that silently invalidate measurements, each verified live on this build. (i) vLLM 0.25.1 exposes `vllm:kv_cache_usage_perc` and has *no* `vllm:gpu_cache_usage_perc` (absent from all 66 `vllm:*` families); a controller coded against the latter name reads KV pressure as permanently zero, raises no error, and becomes a silently broken baseline. Our state accessor raises on that name instead of returning zero. (ii) This build streams thinking tokens in `delta.reasoning`, not `delta.reasoning_content`; a client reading only the latter scores every reasoning request as a zero-token success. (iii) Killing an API server by port does not free GPU memory, because the EngineCore subprocess outlives the parent, and a failed server becomes a zombie that `os.kill(pid, 0)` reports as alive.

What reproduces which number. Cold start, KV budget, GPU occupancy, teardown, and the serving curve come from the testbed harness and are tabulated in `testbed_report_v2.md` with n and spread for every quantity; unmeasured quantities carry a sentinel -1 rather than an estimate. The serving curve of Fig. 3 is `serving_curve.csv`; the capacity floor is `capacity.json` (SLO, per-rate detail, and the null `lowest_failing_rps` that makes it a floor). Length distributions, mean-residual-life curves and tests come from `measured_lengths.json` (full per-cell sample vectors, 742 requests, validating as `calibrated:true`), `length_analysis.json` and `cv_verdict.json`; bootstrap settings are 8,000 resamples for CV CIs, 10,000 for the MRL rise test, 4,000 for the flat-tail test. The CV-to-gap curve of Fig. 1b is `tail_sensitivity.csv`, whose header carries `calibrated=False` and names each uncalibrated required field.

Workload. Corpus `corpus.jsonl`, 4,397 prompts, SHA-256 `bf74118878847d47407f5126f79da6249323db82010946e8560c0bc14ae9e824`, rebuilt deterministically by `build_corpus.py` at sampling seed 20270918. Ten scenario traces (S1–S5 $\times$ dev/test) at seeds 1001–1005 and 2001–2005, each with its SHA-256 in `manifest.json`, which also records the corpus hash so a trace can always be matched to the corpus it was generated against; `verify_scenarios.py` re-runs six property checks against the emitted files, including the byte-identical-arrivals check for S3.

Simulation and controllers. `sim_cluster.py selftest` runs seven physical and interface checks, `run_eval.py contracttest` the backend contract and metric-name guards; `tune.py` reproduces the dev-only sweep (dev-only status is asserted in code). All simulation outputs carry `calibrated:false` in their header, and the top-level flag is the AND over fields marked `required_for_claims`, so one placeholder poisons a whole run by design. No simulation number appears as a result in this paper.

ETHICS STATEMENT

This is infrastructure work on serving efficiency; it involves no human subjects, no personal data, and no model training. Two considerations apply.

Compute and energy. All measurements ran on a single NVIDIA L40S. The GPU-time budget was the binding constraint on this work and shaped what we could measure: several planned experiments (cold start under load, drain time, router overhead under load, and the controller comparison itself) were cancelled on that budget rather than blocked by a defect, and we report them as not measured rather than estimating them. The motivation for the work is a reduction in wasted accelerator time:

over-provisioned reasoning-model serving burns energy on idle replicas, and mis-provisioned serving burns it on preempted and re-run generations.

Dataset licences. Prompts are drawn from GSM8K, CommonsenseQA, MATH (Level 4–5) and AIME 1983–2024, with per-dataset licence status recorded in the corpus documentation. GPQA was considered and dropped because access is gated, and was not substituted with a different dataset, so no cell of the design is silently backfilled. The redistribution status of the TriviaQA subset is flagged as unresolved in that documentation, and the corpus builder reproduces the file from upstream sources rather than requiring redistribution. We use these datasets only as prompt text to elicit generations of varying length; no answer correctness is measured or claimed, so the work makes no benchmark claim about any of them.

Disclosure of AI Use

Generative AI was used extensively throughout this work, under human direction and review, and this section is reconstructed from per-track activity logs kept while the work happened rather than from recollection afterwards. Task-category names below are those of the ICLR 2027 AI Policy for Authors.

Tasks with required disclosure, all of which were performed with generative-AI assistance. *Help develop theoretical models or conceptual frameworks*: the whole of §3 — the discrete-interval serving model, Eqs. (1)–(3), the lagged dynamics (4), the observable filtration, the belief state (6) and the cost (7). *Formulate mathematical claims* and *provide critical ingredients for proving* them: all four numbered statements and their proof strategies — conditioning the in-flight indicator on $\{L_i > g_{i,t}\}$ for the normalised tail; differentiating $m = S/\bar{F}$ for the mean-residual-life identity; factorising the marked point process for rate-blindness; convexity plus $C'(n) = p - \beta\mu \Pr[W > n\mu]$ for the critical fractile, with the substitution $v = 1/q$ giving the closed-form maximiser. *Assist in the writing of proofs*: every proof body in Appendix A as it appears. *Propose or refine hypotheses*: narrowing the thesis from the lag of reactive control to the unobservability of in-flight work, and scoping Prop. 3 to rate-measurable policies after concluding the broader claim was both false and already published. *Design or provide feedback on research methodology or experiments*: the S1–S5 scenario design and the rate/mix independence mechanism; the falsifiable predictions in §3.1; the requirement that the point-estimate ablation be a *tuned-fractile* variant rather than a mean, so as not to be a straw man; the two-class corpus membership rules; the sampling plan for the length study, including diagnosing that a $4{,}096$-token cap censored 69% of the long thinking class and designing the higher-cap re-measurement on disjoint prompts. *Implement methods*: the rate/mix independence mechanism in the trace generator, all eight controllers, the simulator, the tuner and the evaluation harness. *Clean and reformat dataset*: the corpus build from HuggingFace Parquet shards — extraction, class filters, SHA-1 de-duplication and stratified capping; no prompt text was generated or paraphrased, and no synthetic prompt was substituted for any dataset, including the dropped one. *Interpret results*: the statistical readings reported here, including the conservative reading of mean residual life (§2) and the conclusion that the lognormal $e(t)$ is non-monotone, which falsified a looser phrasing the section brief had assumed. *Generate synthetic data sets*: only in two labelled places — synthetic constant-rate timestamps as input to a client benchmark, and the placeholder lognormal length parameters, labelled at the module, report and call site. *Assist with translation* and *support qualitative and thematic data analysis* are not applicable to this work.

Tasks with recommended disclosure. *Sourcing and searching for information*, *identify relevant literature*, *summarize or analyse existing literature*, and *discover research topics or identify gaps*: the survey behind §7 — 776 unique arXiv records from 25 abstract-scoped queries and 583 OpenAlex works, screened by reading abstracts rather than titles, with regex co-occurrence screens used to locate the threat-to-novelty cluster mechanically. *Format references*: `refs.bib` was generated programmatically from fetched API records, with fields taken from the retrieved record rather than composed. *Create or edit software code*: every module of the released artifact, plus the verification scripts, self-tests and the numerical script that independently re-derives each analytical claim. *Create or modify scientific figures*: all figures, from the measured artifacts. *Draft parts of a research paper*: all sections of this manuscript, including this disclosure. *Suggest experimental parameters*, *brainstorming*, *propose a title*, *suggest a structure*, and *edit for readability* were also AI-assisted.

Verification, and what we take responsibility for. We have reviewed all AI-assisted work. Each analytical claim in §3 is independently re-derived numerically by a verification script (finite differ-

ences, quadrature, root finding, brute-force grid minimax) and compared against its closed form. Measurement code was exercised by a smoke harness on every code path before any measurement run, including an assertion that a thinking request records non-zero output tokens — the failure mode that silently invalidated an earlier sweep. Reported aggregates were recomputed from raw per-request records rather than trusted from an experiment's own summary, and every number in the internal technical reports was programmatically interpolated from measured JSON rather than typed. Two arithmetic errors introduced by the assistant in generated report prose were caught in review and corrected against the data. The Kubernetes HPA semantics were fetched from the primary documentation rather than recalled, and a text extract with its SHA-256 is retained. Every bibliography entry was checked against a real retrieved record, and claims we could not verify are flagged as such in the text rather than asserted. We take responsibility for the final content of this work, including all text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav Gulavani, Ramachandran Ramjee, and Alexey Tumanov. Vidur: A Large-Scale simulation framework for LLM inference. *arXiv preprint arXiv:2405.05465*, 2024a.

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. *arXiv preprint arXiv:2403.02310*, 2024b.

Kenneth J. Arrow, Theodore Harris, and Jacob Marschak. Optimal inventory policy. *Econometrica*, 19(3):250–272, 1951.

Dariusz Augustyn, Łukasz Wyciślik, and Mateusz Sojka. Tuning a kubernetes horizontal pod autoscaler for meeting performance and load demands in cloud deployments. *Applied Sciences*, 2024. doi: 10.3390/app14020646.

Himanshu Singh Baghel. ADAPT: a Self-Calibrating proactive autoscaler for container orchestration. *arXiv preprint arXiv:2605.15788*, 2026.

Richard E. Barlow and Frank Proschan. *Statistical Theory of Reliability and Life Testing: Probability Models*. Holt, Rinehart and Winston, 1975.

Omar Basit, Yunzhao Liu, Z. Jonny Kong, and Y. Charlie Hu. DualScale: Energy-Efficient disaggregated LLM serving via Phase-Aware placement and DVFS. *arXiv preprint arXiv:2602.18755*, 2026.

Achal Bassamboo, Ramandeep S. Randhawa, and Assaf Zeevi. Capacity sizing under parameter uncertainty: Safety staffing principles revisited. *Management Science*, 56(10):1668–1686, 2010. doi: 10.1287/mnsc.1100.1203.

Sem Borst, Avishai Mandelbaum, and Martin I. Reiman. Dimensioning large call centers. *Operations Research*, 52(1):17–34, 2004. doi: 10.1287/opre.1030.0081.

Huamin Chen, Xunzhuo Liu, Yuhan Liu, Junchen Jiang, Bowei He, and Xue Liu. inference-fleet-sim: A Queueing-Theory-Grounded fleet capacity planner for LLM inference. *arXiv preprint arXiv:2603.16054*, 2026.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of o1-Like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

Xingqi Cui, Chieh-Jan Mike Liang, Jiarong Xing, and Haoran Qiu. From models to operators: Rethinking autoscaling granularity for large generative models. *arXiv preprint arXiv:2511.02248*, 2025.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the Cross-Entropy method. *Annals of Operations Research*, 2005. doi: 10.1007/s10479-005-5724-z.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. doi: 10.1038/s41586-025-09422-z. Nature volume 645, pages 633-638 (2025).

Ludwig Dierks, Ian A. Kash, and Sven Seuken. On the cluster admission problem for cloud computing. *Journal of Artificial Intelligence Research*, 71:1–40, 2021. doi: 10.1613/jair.1.12346.

Yiding Feng, Siyu Liu, Zonghan Yang, and Yuhao Zhang. General Non-Clairvoyant KV-Cache scheduling via Regime-Aware routing. *arXiv preprint arXiv:2607.09248*, 2026a.

Yiding Feng, Zonghan Yang, and Yuhao Zhang. Competitive Non-Clairvoyant KV-Cache scheduling for LLM inference. *arXiv preprint arXiv:2601.22996*, 2026b.

Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. ServerlessLLM: Low-Latency serverless inference for large language models. *arXiv preprint arXiv:2401.14351*, 2024.

Zhenghao Gan, Yichen Bao, Yifei Liu, Chen Chen, Quan Chen, and Minyi Guo. SageSched: efficient LLM scheduling confronting demand uncertainty and hybridity. *arXiv preprint arXiv:2603.07917*, 2026.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Shlomo Halfin and Ward Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29(3):567–588, 1981. doi: 10.1287/opre.29.3.567.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-Budget-Aware LLM reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

Haitian Hang, Xiu Tang, Jianling Sun, Lingfeng Bao, David Lo, and Haoye Wang. Robust Auto-Scaling with probabilistic workload forecasting for cloud databases. In *IEEE 40th International Conference on Data Engineering (ICDE)*, 2024. doi: 10.1109/icde60146.2024.00308.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

Tao Huang, Pengfei Chen, Kyoka Gong, Jocky Hawk, Zachary Bright, Wenxin Xie, Kecheng Huang, and Zhi Ji. ENOVA: autoscaling towards cost-effective and stable serverless LLM serving. *arXiv preprint arXiv:2407.09486*, 2024.

Shashwat Jaiswal, Kunal Jain, Yogesh Simmhan, Anjaly Parayil, Ankur Mallick, Rujia Wang, Renee St. Amant, Chetan Bansal, Victor Rühle, Anoop Kulkarni, Steve Kofsky, and Saravan Rajmohan. SageServe: optimizing LLM serving on cloud data centers with forecast aware Auto-Scaling. *arXiv preprint arXiv:2502.14617*, 2025. doi: 10.1145/3771576. Proceedings of the ACM on Measurement and Analysis of Computing Systems, Vol. 9, No. 3, Article 61. December 2025.

Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. S^3: increasing GPU utilization during generative inference for higher throughput. *arXiv preprint arXiv:2306.06000*, 2023.

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. *arXiv preprint arXiv:2309.06180*, 2023.

Ruiqi Lai, Hongrui Liu, Chengzhi Lu, Zonghao Liu, Siyu Cao, Siyang Shao, Yixin Zhang, Luo Mai, and Dmitrii Ustiugov. TokenScale: timely and accurate autoscaling for disaggregated LLM serving with token velocity. *arXiv preprint arXiv:2512.03416*, 2025.

Rongzhi Li, Ruogu Du, Zefang Chu, Sida Zhao, Chunlei Han, Zuocheng Shi, Yiwen Shao, Huanle Han, Long Huang, Zherui Liu, and Shufan Liu. Taming the chaos: Coordinated autoscaling for heterogeneous and disaggregated LLM inference. *arXiv preprint arXiv:2508.19559*, 2025.

Yueying Li, Yuanfan Chen, Jiayang Chen, Esha Choukse, Haoran Qiu, G. Edward Suh, Rodrigo Fonseca, Ziv Scully, and Udit Gupta. Beyond prediction: Tail-Aware scheduling for LLM inference. *arXiv preprint arXiv:2606.18431*, 2026. Forty-Third International Conference on Machine Learning (2026).

Chiheng Lou, Sheng Qi, Rui Kang, Yong Zhang, Chen Sun, Pengcheng Wang, Xuanzhe Liu, and Xin Jin. WarmServe: enabling One-for-Many GPU prewarming for Multi-LLM serving. *arXiv preprint arXiv:2512.09472*, 2025.

Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters. *arXiv preprint arXiv:1810.01963*, 2018.

Chunyang Meng, Haogang Tong, Tianyang Wu, Maolin Pan, Yang Yu, and Yi Jiang. BASE: Burst-Adaptive autoscaling via stacked ensembles for SLO assurance and cost efficiency. *arXiv preprint arXiv:2402.12962*, 2024. doi: 10.1109/TSC.2026.3668105.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Chanh Nguyen, Monowar Bhuyan, and Erik Elmroth. Taming cold starts: Proactive serverless scheduling with model predictive control. *arXiv preprint arXiv:2508.07640*, 2025. doi: 10.1109/MASCOTS67699.2025.11283271. 33rd International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication System (MASCOTS 2025).

Chengyi Nie, Rodrigo Fonseca, and Zhenhua Liu. Aladdin: Joint placement and scaling for SLO-Aware LLM serving. *arXiv preprint arXiv:2405.06856*, 2024.

Zhicheng Pan, Yihang Wang, Yingying Zhang, Bin Yang, Yunyao Cheng, Peng Chen, Chenjuan Guo, Qingsong Wen, Xiduo Tian, Yunliang Dou, Zhiqiang Zhou, Chengcheng Yang, Aoying Zhou, and Bin Yang. MagicScaler: Uncertainty-Aware, predictive autoscaling. *Proceedings of the VLDB Endowment*, 2023. doi: 10.14778/3611540.3611566.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. *arXiv preprint arXiv:2311.18677*, 2023.

Archit Patke, Dhemath Reddy, Saurabh Jha, Chandra Narayanaswami, Zbigniew Kalbarczyk, and Ravishankar Iyer. Hierarchical autoscaling for large language model serving with chiron. *arXiv preprint arXiv:2501.08090*, 2025.

Daniel F. Perez-Ramirez, Dejan Kostic, and Magnus Boman. CASTILLO: characterizing response length distributions of large language models. *arXiv preprint arXiv:2505.16881*, 2025.

Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A KVCache-centric disaggregated architecture for LLM serving. *arXiv preprint arXiv:2407.00079*, 2024.

Haoran Qiu, Subho S. Banerjee, Saurabh Jha, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. FIRM: an intelligent Fine-Grained resource management framework for SLO-Oriented microservices. *arXiv preprint arXiv:2008.08509*, 2020.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T. Kalbarczyk, Tamer Başar, and Ravishankar K. Iyer. Efficient interactive LLM serving with proxy model-based sequence length prediction. *arXiv preprint arXiv:2404.08509*, 2024.

Krzysztof Rzadca, Paweł Findeisen, Jacek Świderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Paweł Krzysztof Nowak, Beata Strack, Piotr Witusowski, Steven Hand, and John Wilkes. Autopilot: Workload autoscaling at google scale. In *Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys)*, 2020.

Herbert Scarf. A min-max solution of an inventory problem. In *Studies in the Mathematical Theory of Inventory and Production*, pp. 201–209. Stanford University Press, 1958.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019. doi: 10.1038/s41586-020-03051-4.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving. *arXiv preprint arXiv:2406.03243*, 2024.

Aravind Sundaresan. Clairvoyant: Predictive Shortest-Job-First admission for serial LLM inference. *arXiv preprint arXiv:2606.07248*, 2026.

The llm-d Project. llm-d workload variant autoscaler. `https://github.com/llm-d/llm-d-workload-variant-autoscaler`, 2026. Open-source variant-optimization autoscaler for distributed LLM inference.

Jing Wang, Yu-Yang Qian, Ke Xue, Chao Qian, Peng Zhao, and Zhi-Hua Zhou. Robust length prediction: A perspective from Heavy-Tailed Prompt-Conditioned distributions. *arXiv preprint arXiv:2604.07931*, 2026a.

Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Yuchu Fang, Yeju Zhou, Yang Zheng, Zhenheng Tang, Xin He, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, and Xiaowen Chu. BurstGPT: a real-world workload dataset to optimize LLM serving systems. *arXiv preprint arXiv:2401.17644*, 2024.

Zhibin Wang, Zetao Hong, Xue Li, Zibo Wang, Shipeng Li, Qingkai Meng, Qing Wang, Chengying Huan, Rong Gu, Sheng Zhong, and Chen Tian. STAR: Decode-Phase rescheduling for LLM inference. In *Proceedings of the 35th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 2026b. doi: 10.1145/3806645.3807813.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Bingyang Wu, Yinmin Zhong, Zili Zhang, Shengyu Liu, Fangyue Liu, Yuanhang Sun, Gang Huang, Xuanzhe Liu, and Xin Jin. Fast distributed inference serving for large language models. *arXiv preprint arXiv:2305.05920*, 2023.

Yuxing Xiang, Xue Li, Kun Qian, Yan Zhang, Wenyuan Yu, Ennan Zhai, Xin Jin, and Jingren Zhou. ServeGen: workload characterization and generation of large language model serving in production. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2026.

Siqiao Xue, Chao Qu, Xiaoming Shi, Cong Liao, Shiyi Zhu, Xiaoyu Tan, Lintao Ma, Shiyu Wang, Shijun Wang, Yun Hu, Lei Lei, Yangfei Zheng, Jianguo Li, and James Zhang. A meta reinforcement learning approach for predictive autoscaling in the cloud. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022. doi: 10.1145/3534678.3539063.

Yuedong Yang, Xiwen Wei, Mustafa Munir, and Radu Marculescu. Fuel gauge: Estimating Chain-of-Thought length ahead of time in large multimodal models. *arXiv preprint arXiv:2603.10335*, 2026.

Zahra Yousefijamarani, Xinglu Wang, Qian Wang, Morgan Lindsay Heisler, Taha Shabani, Niloofar Gholipour, Parham Yassini, Hong Chang, Kan Chen, Qiantao Zhang, Xiaolong Bai, Jiannan Wang, Ying Xiong, Yong Zhang, and Zhenan Fan. HFX: joint design of algorithms and systems for Multi-SLO serving and fast scaling. *arXiv preprint arXiv:2508.15919*, 2025.

Gyeong-In Yu, Joo Seong Jeong, et al. Orca: a distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022. Author list truncated by the metadata API; see unverified.md.

Dingyan Zhang, Haotian Wang, Yang Liu, Xingda Wei, Yizhou Shan, Rong Chen, and Haibo Chen. BLITZSCALE: fast and live large model autoscaling with O(1) host caching. *arXiv preprint arXiv:2412.17246*, 2024.

Guilin Zhang, Chuanyi Sun, Kai Zhao, Xu Chu, Shahryar Sarkani, and John Fossaceca. When does deep RL beat calibrated baselines? a benchmark study on adaptive resource control. *arXiv preprint arXiv:2605.26418*, 2026a.

Tiancheng Zhang, Yulin Chen, Yunfeng Zhao, Shaoyuan Huang, Cheng Zhang, and Xiaofei Wang. MAPS: Memory-aware predictive scheduling framework for large language model serving. *arXiv preprint arXiv:2609.15359*, 2026b.

Haoyu Zheng, Yongqiang Zhang, Fangcheng Fu, Xiaokai Zhou, Hao Luo, Hongchao Zhu, Yuanyuan Zhu, Hao Wang, Xiao Yan, and Jiawei Jiang. Scheduling LLM inference with Uncertainty-Aware output length predictions. *arXiv preprint arXiv:2604.00499*, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*, 2023a.

Zangwei Zheng, Xiaozhe Ren, Fuzhao Xue, Yang Luo, Xin Jiang, and Yang You. Response length perception and sequence scheduling: An LLM-Empowered LLM inference pipeline. *arXiv preprint arXiv:2305.13144*, 2023b.

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. *arXiv preprint arXiv:2401.09670*, 2024.

Tianbao Zhou, Yi Wang, Yu Zhou, and Zirui Liu. Janus: Multi-llm serving at production scale. In *Proceedings of the ACM SIGOPS 32nd Symposium on Operating Systems Principles (SOSP)*, 2026.

Ding Zou, Wei Lu, Zhibo Zhu, Xingyu Lu, Jun Zhou, Xiaojin Wang, Kangyu Liu, Haiqing Wang, Kefan Wang, and Renen Sun. OptScaler: a collaborative framework for robust autoscaling in the cloud. *arXiv preprint arXiv:2311.12864*, 2023. doi: 10.14778/3685800.3685829.

A PROOFS

Proofs of the statements in §3, in order of appearance.

Proposition 1. Q_t and K_t are finite functions of counters in $\mathcal{F}_t^{\mathrm{o}}$ by (1)–(2). For R_t, the event $\{i \text{ in flight at } t\}$ is exactly $\{L_i > g_{i,t}\}$, so conditioning on $\mathcal{F}_t^{\mathrm{o}}$ conditions L_i on the half-line $(g_{i,t}, \infty)$ and reveals nothing further: the decoder's stop event is not recorded until it occurs. Hence $L_i - g_{i,t}$ has the normalised tail law in (5), non-degenerate by hypothesis, so $\mathrm{Var}[R_t \mid \mathcal{F}_t^{\mathrm{o}}] > 0$ and R_t cannot a.s. equal an $\mathcal{F}_t^{\mathrm{o}}$-measurable random variable. Independence across i given classes and elapsed counts gives the convolution.

Proposition 2. Let $S(k) = \int_k^\infty \bar{F}_c$, so $m_c = S/\bar{F}_c$ and $S' = -\bar{F}_c$; then $m_c' = (S'\bar{F}_c - S\bar{F}_c')/\bar{F}_c^2 = -1 + m_c h_c$, giving the identity and the sign condition. For (i), NWU reads $\bar{F}_c(k+r) \geq \bar{F}_c(k)\bar{F}_c(r)$; integrating over $r \in (0,\infty)$ and dividing by $\bar{F}_c(k)$ gives $m_c(k) \geq m_c(0)$. For (ii), $S(k) = k^{1-\alpha}x_0^\alpha/(\alpha-1)$ and $\bar{F}_c(k) = k^{-\alpha}x_0^\alpha$, so $m_c(k) = k/(\alpha-1)$.

Proposition 3. Marks are drawn independently of epochs (§3), so the marked process factorises and the epoch marginal does not depend on the mark law; a $\mathcal{G}_t$-measurable policy is a function of that marginal alone, while offered token load $\lambda\mathbb{E}_\theta[L]$ differs by hypothesis.

Theorem 1. C is convex with $C'(n) = p - \beta\mu\Pr[W > n\mu]$, vanishing at $\Pr[W > n\mu] = \rho$, giving the critical fractile. Fix $q \in (\rho, 1)$. Under $W \equiv \bar{w}$ the optimum is $n_1^\star = \bar{w}/\mu$ with $C_1^\star = p\bar{w}/\mu$; under $\mathcal{B}_q$, since $q > \rho$, it is $n_2^\star = \bar{w}/(q\mu)$ with $C_2^\star = p\bar{w}/(q\mu)$. Both have mean $\bar{w}$, so a mean-blind policy plays one n for both. On $[n_1^\star, n_2^\star]$ the regret $C_1(n) - C_1^\star = p(n - \bar{w}/\mu)$ increases while $C_2(n) - C_2^\star$ has slope $p - \beta\mu q < 0$, so the minimax n equalises them; solving gives $\Gamma(q) = p\bar{w}(1-q)(q-\rho)/(\mu q^2)$. With $v = 1/q$ the relative regret is $-\rho v^2 + (1+\rho)v - 1$, maximised at $v = (1+\rho)/(2\rho)$ with value $(1-\rho)^2/(4\rho)$.

B CONTROLLER IMPLEMENTATION DETAIL

HPA, from the primary source. Read from the Kubernetes documentation (*Horizontal Pod Autoscaling*), page last modified 2026-03-15, retrieved 2026-08-12; a text extract with its SHA-256 is kept with the artifact. Implemented faithfully: the ratio rule $\lceil n \cdot m/m^* \rceil$ on a per-pod average; the tolerance band (default 0.1, with the per-direction `tolerance` field documented as beta from v1.35 and the same default applying cluster-wide before that); the scale-down stabilisation window as a rolling *maximum* over previously computed desired states (default 300 s); scale-up stabilisation 0; the default policies (scale-up $\max(100\%, 4 \text{ pods})$ per 15 s with `selectPolicy: Max`, scale-down 100% per 15 s); and a 15 s cadence matching the documented sync period. Deliberately not reproduced: the missing-metrics logic and the CPU readiness-delay special cases. In our setting a booting vLLM replica exposes no metrics endpoint at all, so it is simply absent from the average rather than synthesised — the honest analogue, and one that makes our HPA slightly *less* jumpy than production HPA on scale-up, i.e. it does not disadvantage the baseline.

The leading-indicator arm. Drives on signal $=$ tok_rate $+ \kappa\,\mathrm{d}(\text{tok_rate})/\mathrm{d}t$ with a backlog term, because a saturated cluster cannot *report* a token rate above what it can serve — without that term, throughput-driven control is blind exactly when overloaded. Not reproduced from TokenScale: the

disaggregated prefill/decode deployment and independent scaling of the two pools (our testbed is four co-resident monolithic replicas on one GPU, so there is no split to scale), the SLO-attainment predictor, the admission/routing changes, and the absolute thresholds, which are model- and hardware-specific and here tuned by the same protocol as every other baseline.

Forecasting. Three forecasters — EWMA, Holt's linear trend, and a ridge AR model fitted on dev — crossed with two signals (request rate, token rate). The token variant is the fairer, stronger version: it can see token load, but only after that load has been generated.

Tuning protocol. Dev-only, asserted in code, with the same sweep budget and objective for every controller: 307 sweep runs, then a 165-run sanity matrix of 11 controller variants $\times$ 5 dev scenarios $\times$ 3 seeds. Per-metric target ranges were introduced after a first sweep selected an unreachable KV *fraction* target that silently turned HPA into static-1 — the kind of defect that manufactures a win for the proposed method. Five further defects found by our own checks are recorded with the artifact, including a decode loop that could emit tokens faster than the step-time model permits, idle replica-seconds billed after all work finished (a $12\times$ cost error on one run), and an oracle that ignored the KV bound and so produced an "upper bound" with a 0.48 TTFT violation rate.

One interface guard. vLLM 0.25.1 exposes `vllm:kv_cache_usage_perc` and has *no* `vllm:gpu_cache_usage_perc` (verified absent from all 66 `vllm:*` families). A controller coded against the older name reads KV pressure as permanently zero, raises no error, and becomes a silently broken baseline — and a fake win for us. Our state accessor and Prometheus parser therefore *raise* on that name rather than returning zero.

C Predictions and the experiments that test them

Prop. 1 is a structural claim and is not itself an experiment; the other three each predict one. Prop. 3 predicts that under constant-rate, shifting-mix traffic (S3) rate-driven and rate-forecasting controllers act as if nothing happened while token load rises, and our generator makes the hypothesis exact by holding arrival epochs pathwise identical across mix schedules at fixed seed. Thm. 1 predicts a cost–SLO gap against tuned point estimates widening as SLO penalty rises relative to GPU price — a signed prediction along the front, not a single win. Prop. 2 makes m_c versus elapsed tokens a diagnostic worth reporting (§2). All three are falsifiable.

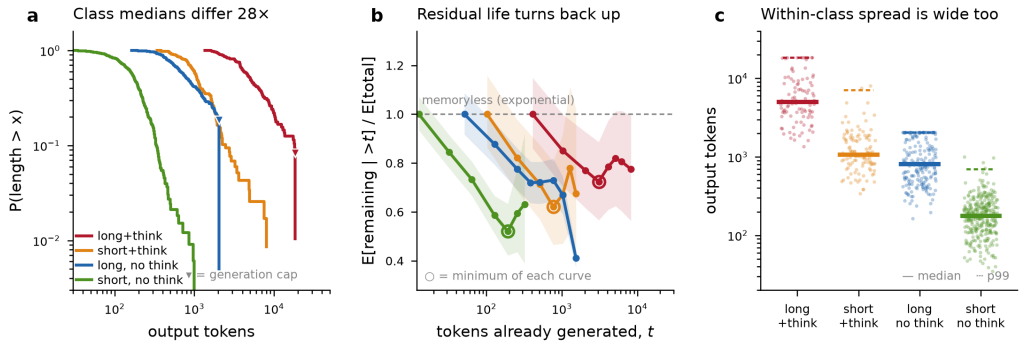

Figure 2: **Measured output-length distributions and their residual life.** (**a**) Survival functions per cell on log axes; triangles mark the generation cap, so curves reaching it are right-censored. (**b**) Mean residual life $e(t) = \mathbb{E}[L - t \mid L > t]$ normalised by $e(0)$, with percentile-bootstrap bands; open circles mark each curve's minimum. The dashed line at 1 is the memoryless (exponential) reference. **Read this panel carefully:** $e(t)$ *falls* by 28–59% before any upturn, and while it turns back up in three of four cells, no cell's upturn is significant at 95% (Table 4). What is established is that the tail is not exponential, in three of four cells. (**c**) Per-request lengths with the median and p_{99} marked, showing the within-class spread that survives conditioning on class.

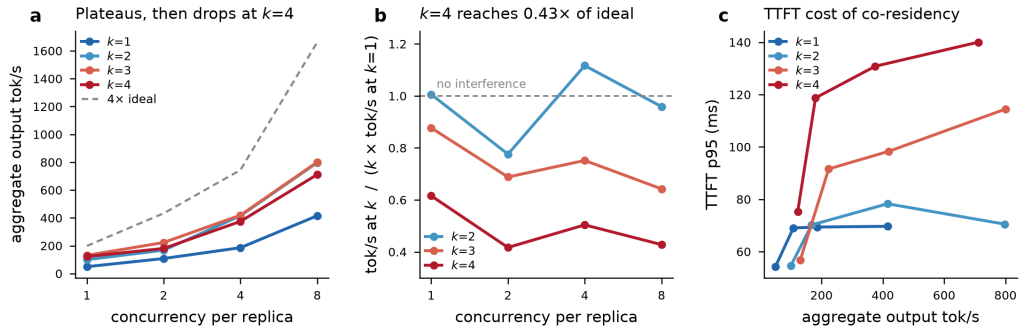

Figure 3: **Co-resident replicas do not scale linearly.** Closed-loop measurement, 50/50 class mix, 50 s per phase after a 10 s warm-up, k replicas resident and all k driven. (**a**) Aggregate output throughput against per-replica concurrency; the dashed line is four times a single replica. (**b**) Ratio to linear scaling: at matched concurrency 8, k=4 reaches 0.428×. (**c**) TTFT p_{95} against achieved throughput — the latency price of co-residency. Aggregate throughput peaks at k=3 (801 tok/s) and falls 11% at k=4 (712 tok/s), so the fourth replica subtracts throughput on this hardware.

D Notation

Table 1 lists every symbol used in §3, with the observability column separating what the serving stack reports from what the belief has to supply.

E Additional Measurements

E.1 Hill tail-index estimates

Reported for completeness and leaned on nowhere. The Hill estimator assumes a Pareto tail; a lognormal also yields a finite, sample-size-dependent estimate, so these are one tail statistic among several and are **not** evidence of a power law. `short|think` gives $\hat{\alpha} \approx 1.80$ (below 2, i.e. infinite variance if the tail were genuinely Pareto), but with $k = 11$ tail points and s.e. 0.54 that estimate is very imprecise. `long|think` is omitted-in-effect by its own uncertainty ($\hat{\alpha} = 17.80$, s.e. 5.63,

$k = 10$, threshold 17,355 tokens — i.e. estimated almost entirely from samples at or near the generation cap).

E.2 Dev-split pipeline check (simulation; `calibrated:false`)

Table 6 is the 165-run dev sanity matrix (11 controller variants $\times$ 5 dev scenarios $\times$ 3 seeds) from the simulator described in §6. **Every row is `calibrated:false` and none is a result of this paper**: output lengths are drawn from the same object the belief conditions on, so the belief arm holds a perfectly specified prior. The table is included because it shows the pipeline runs, the arms are distinguishable, and — read pessimistically, which is how it should be read — that calibrated static-k is a strong baseline.

Two readings of this table are worth separating. The aggregate scalar objective is not a fair arbiter of the belief/point ablation, because the tuner selected the point arm at a cheap, loose operating point (1150 GPU-s at SLO penalty 0.315) and the belief arm at an expensive, strict one (1510 at 0.022); neither dominates. The matched-SLO form on dev is more informative about the mechanism and equally compromised by the circularity: across all 27 of its configurations the point arm never reaches an SLO penalty below 0.294, while the belief arm reaches 0.010; where the point arm can compete at all (loose SLO ≤ 0.30) it is cheaper (1071 versus 1379 GPU-s). That truncated front is consistent with the mechanism — a point estimate systematically under-predicts KV exhaustion, so no threshold retuning buys reliability — and *equally* consistent with one arm holding a perfectly specified prior. Both readings survive this data; only the real cluster separates them.

E.3 Adversarial CV sweep, full table

Table 7 is `tail_sensitivity.csv` in full: class medians held fixed, the length distribution's σ scaled, both MPC arms re-run at their tuned parameters (S1/S3/S5 $\times$ 2 seeds). The measured mixture $\mathrm{CV} = 1.375$ $[1.300, 1.444]$ falls between the $\mathrm{CV} = 1.31$ and $\mathrm{CV} = 1.94$ rows. Simulation output, `calibrated:false`: it locates the threshold in a model and says nothing about the magnitude of any real gap.

F Workload construction and verification

The corpus is 4,397 real prompts in two reasoning-length classes (GSM8K with ≤ 2 calculator annotations, TriviaQA and CommonsenseQA as *short*; MATH Level 4–5 and AIME as *long*), SHA-256 `bf74118878847d47...`; class membership is a property of the source annotation, not of a measured output length. GPQA was dropped as access-gated and not substituted, so no cell is backfilled from another dataset. Five scenarios (S1–S5) are emitted as dev/test twins at fixed seeds with SHA-256 in a manifest, each stochastic component drawing from its own hash-derived RNG substream. That gives the property the experiment needs *by construction*: the arrival-time generator takes no mix argument, so a mix schedule cannot perturb a timestamp as a matter of data flow. **S3 realises Prop. 3**: p_{long} steps $0.15 \to 0.85$ at $0.4\times$ the horizon while the arrival timestamps stay *byte-identical* to a fixed-mix twin at the same seed (max $|\Delta t| = 5 \times 10^{-7}$ s, JSONL rounding). The two traces are the same arrival realisation with different prompts attached: verified on the emitted files, the windowed rate moves 0.008–0.114 req/s against a Poisson noise band of ± 0.193 while the long fraction moves $\approx 0.15 \to 0.84$. A rate-measurable controller provably sees nothing — the theory's prediction made into an experiment. S2 is the control in the other direction ($3\times$ rate step, mix fixed), on which a reactive controller *should* do well, so an S3 result cannot be dismissed as a broken baseline; S4 puts peak rate in phase with longest outputs; S5 alternates the mix with period $1.5\times$ cold start. The replay client is open-loop by construction — the sender awaits only `asyncio.sleep` and dispatches each request as a detached task — and sent $211/211$ requests on schedule at 21.2 req/s against a slow stand-in endpoint (p_{95} lateness 2.35 ms, 117 in flight) where a send-then-await client could have issued 2.

F.1 Scenario traces

All ten traces replayed with every request sent and zero errors at $60\times$ time compression (p_{95} send lateness 2.19–2.32 ms), with the `enable_thinking` flag dispatched correctly for every request.

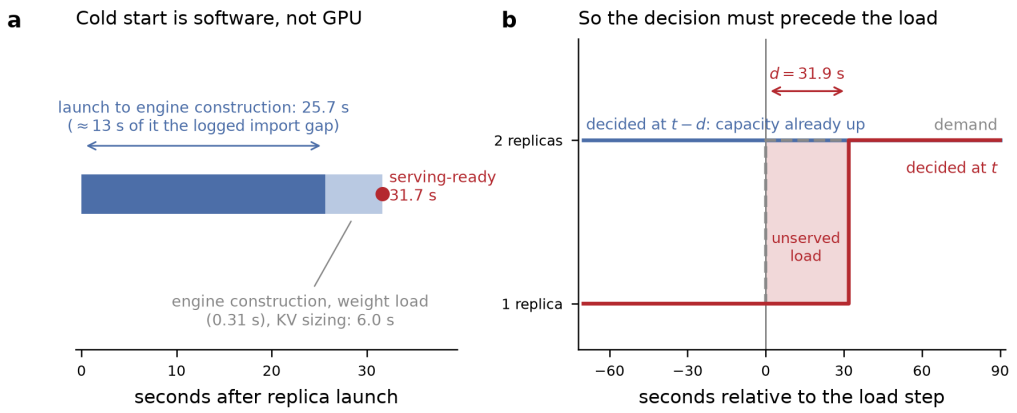

Figure 4: **The actuation lag is structural, and it forces the decision ahead of the load.** (**a**) Anatomy of *one* replica's cold start, from its own launch-log milestones; this replica reached serving-ready at 31.67 s, consistent with the median 31.90 s over n=9 launches quoted in the text. Most of the interval elapses before the engine object exists: the logged gap between the first line and the start of engine construction is ≈ 13 s of interpreter start and imports, while loading 3.22 GiB of warm weights takes 0.31 s. Log timestamps have 1 s resolution, so the internal milestone positions are accurate only to that. (**b**) Consequence of $d = 31.9$ s at a step in demand: a decision taken when the step is observed leaves the shaded interval unserved, whereas the same capacity decided d earlier is already serving. Teardown is 0.54 s (n=3), so the asymmetry in Eq. (4) is measured, not assumed.

Horizon is 300 s per trace at `time_scale=1`, so the full set costs 3,000 s of replay plus drain, and the five test twins alone 1,500 s.

Table 1: Symbols used in §3. "Obs." marks quantities measurable with respect to the observable filtration $\mathcal{F}_t^{\circ}$, i.e. readable from the serving stack at time t; the two entries marked *latent* are the reason the state is a belief.

Symbol	Meaning	Obs.
Time, capacity, actions		
Δ	control interval length (seconds)	—
t	control-interval index, $t \in \{0, \dots, T\}$	—
H	planning horizon, in control intervals	—
n_t	active (serving) replica count at t	✓
w_t	warming replicas at t (billed, not serving)	✓
$n_{\min}, n_{\max}$	replica-count bounds	—
u_t	replica count requested at t (the action)	✓
τ_{cold}	replica cold-start time (seconds)	—
d	actuation lag in intervals, $d = \lceil \tau_{\text{cold}}/\Delta \rceil \geq 1$	—
$\mathcal{R}_t(k)$	replica counts reachable at $t + k$ from a decision at t	—
B	per-replica KV-cache budget (tokens)	—
μ	sustained decode throughput of one replica (tokens per interval)	—
Requests, classes, lengths		
$\mathcal{C}$, c	request class set; class of a request	✓
c_i	class of request i	✓
L, L_i	output (thinking) length in tokens; that of request i	latent
F_c, $\bar{F}_c$, f_c	class-c output-length CDF, survival function, density	—
π_i	prompt length of request i (tokens)	✓
$g_{i,t}$	tokens already emitted by request i at t	✓
$h_c(k)$	hazard of F_c, $f_c(k)/\bar{F}_c(k)$	—
$m_c(k)$	mean residual life, $\mathbb{E}[L - k \mid L > k]$	—
$k^{\star}$	elapsed count solving $h_c(k^{\star})m_c(k^{\star}) = 1$	—
α, x_0	Pareto tail index and scale in Prop. 2(ii)	—
State		
N, N_s	arrival point process with class marks; unmarked count	✓
λ, $\hat{\lambda}_t$	arrival rate; its estimate at t	✓
θ, θ'	class-mix processes (mark laws)	latent
$\mathcal{I}_t$	in-flight (running) request set at t	✓
$\mathcal{Q}_t$, Q_t	admitted-not-running set; queue depth $\lvert\mathcal{Q}_t\rvert$	✓
$\mathcal{A}_t$	requests arriving in interval t	✓
K_t	KV-cache occupancy, Eq. (2)	✓
R_t	residual decode work, $\sum_{i \in \mathcal{I}_t}(L_i - g_{i,t})$	latent
b_t	belief state, Eq. (6)	✓
$\mathcal{F}_t^{\circ}$	observable filtration (§3)	—
$\mathcal{G}_t$	filtration of unmarked arrival counts, $\mathcal{G}_t \subset \mathcal{F}_t^{\circ}$	—
$\bigoplus$	convolution of independent summands	—
Cost and analysis		
p	price per replica-interval (GPU-seconds priced)	—
β_{f}, β_{d}	TTFT and inter-token-latency violation prices	—
s_{f}, s_{d}	TTFT and inter-token-latency SLO targets	—
TTFT_i, $\text{ITL}_{i,t}$	time-to-first-token; inter-token latency	✓
$c(b_t, u_t)$	per-interval cost, Eq. (7)	—
π	control policy, a map from belief to action	—
W, $\bar{w}$	one-shot unobserved work; its mean $\mathbb{E}[W]$	latent
β	shortfall price per token in the one-shot reduction	—
$C(n)$	one-shot cost of n replicas	—
ρ	price ratio $p/(\beta\mu)$, assumed in $(0, 1)$	—
$q_{1-\rho}(W)$	$(1-\rho)$-quantile of W (critical fractile)	—
$n^{\star}$	cost-minimising replica count, $q_{1-\rho}(W)/\mu$	—
$\mathcal{B}_q$, q	mean-matched two-point family and its mass parameter	—
$\Gamma(q)$	minimax regret of a mean-blind policy, Eq. (8)	—
$q^{\star}$	regret-maximising q, $2\rho/(1+\rho)$	—
$(x)_+$	positive part, $\max(x, 0)$	—

Table 2: Measured output-length distributions, n=742 completions in four disjoint cells, in output tokens from vLLM's `usage.completion_tokens` (thinking tokens included). *trunc.* is the fraction stopping at the generation cap rather than emitting end-of-sequence, so those cells' upper quantiles are lower bounds; bootstrap CIs are percentile, 8,000 resamples. The final row pools the two classes a deployment serves, 50/50 by request count with n=96 from each — the dispersion a capacity controller actually faces, and a lower bound given censoring in the long cell.

cell	n	cap	median	IQR	p_{99}	max	**CV**	CV 95% CI	trunc.
`long|think`	96	18,432	5,016	[3,431, 8,958]	18,432	18,432	0.730	[0.649, 0.796]	8.3%
`short|think`	117	18,432	1,075	[826, 1,734]	7,100	8,037	0.816	[0.564, 0.984]	0.0%
`long|no_think`	200	2,048	817	[512, 1,680]	2,048	2,048	0.606	[0.564, 0.645]	18.5%
`short|no_think`	329	1,024	177	[117, 243]	695	996	0.633	[0.533, 0.727]	0.0%
served mixture	96+96	—	—	—	—	—	**1.375**	**[1.300, 1.444]**	—

Table 3: Measured testbed constants. Quantities marked *floor* or *bound* were limited by the GPU-time budget rather than by an observed breakdown, and are reported as such. Cold start is to serving-ready on an idle GPU; the interpreter/import share precedes engine construction, so it is not reduced by a faster accelerator.

quantity	value	n	note
co-resident replicas	4	2 bring-ups	all reached serving-ready
KV cache per replica	55,200 tokens (5.9 GiB)	9 launches	identical every launch
GPU memory at k=4	40,409 of 46,068 MiB	2	87.7% occupied
cold start to serving-ready	31.90 s, IQR [31.82, 32.40]	9	the actuation lag τ_{cold}
interpreter + imports	≈13 s	9	before engine construction
warm weight load	0.31–0.68 s	9	3.22 GiB of weights
sequential 4-replica bring-up	126.8 s	2	parallel bring-up fails KV sizing
teardown to 0 MiB	0.543, 0.543, 0.545 s	3	incidental measurement
co-residency interference	0.428× linear at k=4	132 req	matched concurrency 8
peak aggregate throughput	801 tok/s at k=3	—	−11% at k=4 (712 tok/s)
single-replica capacity	$\geq$ 0.8 req/s (*floor*)	5 rates	all rates passed; sweep truncated

Table 4: Mean residual life $e(t) = \mathbb{E}[L - t \mid L > t]$ per cell. *rise* is measured from each curve's own minimum to the largest t with ≥ 20 samples still at risk; CIs are percentile bootstrap, 10,000 resamples. *flat-tail* p tests the observed range of $e(t)$ against an exponential null with the same mean at the same sample size (4,000 resamples). The tail is established to be non-exponential in three cells; the upturn is *not* significant in any cell.

cell	$e(0)$	$\min e(t)$	at t	$e(t_{\max})$	shape	rise	rise 95% CI	flat-tail p
`long|think`	6,887	4,986	3,072	5,339	U-shaped	+656	[−195, +1,681]	0.258
`short|think`	1,433	890	768	968	U-shaped	+226	[−7, +528]	0.028
`long|no_think`	1,046	430	1,536	430	decreasing	−54	[−139, +39]	0.0005
`short|no_think`	200	104	192	126	U-shaped	+22	[−16, +69]	0.0065

Table 5: Hill estimates. See the caveat above; no claim in this paper depends on these values.

cell	$\hat{\alpha}$	k (tail points)	threshold (tokens)	s.e.
`long|think`	17.80	10	17,355	5.63
`short|think`	1.80	11	2,239	0.54
`short|no_think`	3.37	32	333	0.60

Table 6: Dev sanity matrix, simulation only, `calibrated: false`. GPU-s is mean billed replica-seconds; SLO pen. is the aggregate penalty term; *changes* is mean replica changes per run. Oracle reads true lengths and is excluded from fronts. Four of the eight front points are static-k.

controller	GPU-s	±sd	SLO pen.	TTFT viol.	TBT viol.	changes	on front
`static_k1`	811	213	0.651	0.393	0.258	0.0	yes
`hpa`	1072	443	0.394	0.244	0.149	0.7	yes
`point_mpc`	1150	468	0.315	0.191	0.124	1.3	yes
`static_k2`	1295	261	0.048	0.032	0.016	1.0	yes
`queue`	1361	433	0.102	0.066	0.037	4.7	—
`belief_mpc`	1510	446	0.022	0.012	0.010	2.9	yes
`forecast`	1592	495	0.010	0.006	0.005	5.9	yes
`oracle`*	1773	609	0.000	0.000	0.000	4.7	(privileged)
`static_k3`	1902	375	0.002	0.001	0.001	1.0	yes
`leading`	1927	467	0.006	0.004	0.002	4.7	—
`static_k4`	2522	500	0.000	0.000	0.000	1.0	yes

Table 7: Belief-versus-point SLO gap against dispersion (simulation, `calibrated: false`). The gap is exactly zero at low CV — the falsification check the sweep was designed to run — and material only above CV ≈ 0.87.

σ scale	CV	belief GPU-s	belief SLO	point GPU-s	point SLO	SLO gap
0.25	0.254	337	0.000	337	0.000	**+0.000**
0.50	0.533	354	0.004	351	0.004	**+0.000**
0.75	0.869	588	0.028	448	0.091	+0.064
1.00	1.311	1191	0.002	882	0.256	+0.254
1.25	1.942	1995	0.017	1354	0.223	+0.207
1.50	2.913	3011	0.023	2526	0.124	+0.102

Table 8: The ten emitted traces. Dev/test twins match on every parameter except the seed. V2 in the verification suite checks that S3's arrival timestamps are byte-identical to a fixed-mix twin at the same seed; V3 checks that its windowed arrival rate shows no step beyond Poisson counting noise while V4 checks the mix step is present.

scenario	split	seed	requests	frac. long	trace SHA-256 (prefix)
S1	dev	1001	225	0.5022	`2d8c19b0d6ccd8be`
S1	test	2001	200	0.4650	`7753943035bc8554`
S2	dev	1002	390	0.5179	`be2423ed9326c924`
S2	test	2002	361	0.5097	`508a71fb31a3702d`
S3	dev	1003	221	0.5656	`4c366a3e2d2e01f5`
S3	test	2003	208	0.5865	`e597bc4581a5bb52`
S4	dev	1004	197	0.6802	`c58cd94564fb1691`
S4	test	2004	225	0.6311	`31095e934d00ee94`
S5	dev	1005	217	0.4332	`d99f8ce3ee27b10c`
S5	test	2005	222	0.4730	`9d0180954eb5b039`