HeteroServe: Capability-Weighted Batch Scheduling for LLM Inference on Heterogeneous GPU Clusters

Anonymous Authors[1]

Abstract

Production deployments of large language models (LLMs) increasingly rely on heterogeneous GPU clusters—mixing generations such as NVIDIA H100, A100, and L40S—to balance cost and availability. Existing serving systems, however, apply uniform continuous-batching strategies that ignore inter-device capability differences, leading to severe load imbalances: high-end GPUs stall waiting for slower devices while memory-constrained accelerators overflow from oversized batches. We present HETEROSERVE, an integrated queue-aware batch scheduler that combines hardware capability scoring with real-time queue-depth feedback and length-binned admission control to route each inference request to the most suitable device. The capability score encodes per-device floating-point throughput, high-bandwidth memory (HBM) capacity, and memory bandwidth; queue-depth feedback continuously redistributes load across the heterogeneous fleet to prevent hotspots on high-FLOPS devices. On a simulated mixed-GPU cluster representative of real data center configurations, HETEROSERVE achieves **36,772 tokens/sec**—a **2.13× throughput improvement** over uniform scheduling—while attaining **68.8% SLO compliance**, a **42.4 percentage-point improvement** over Uniform scheduling (26.4% to 68.8%). Our P95 time-to-first-token (TTFT) of 1,044 ms is **172× lower** than the uniform baseline (179,849 ms). Ablation studies confirm that queue-depth feedback is the dominant contributor, and that performance remains stable across diverse workload intensity and mixture distributions. These results demonstrate that device-aware macro-scheduling with real-time queue feedback is a practical, deployment-friendly path to high efficiency on heterogeneous LLM clusters—outperforming even a length-clairvoyant oracle that lacks adaptive load balancing.

1. Introduction

The operational cost of deploying large language models at scale is dominated by GPU infrastructure. As frontier models grow from tens to hundreds of billions of parameters (Brown et al., 2020; Touvron et al., 2023), organizations cannot always provision uniform clusters of the latest accelerators. Instead, production clusters commonly mix several GPU generations: legacy deployments based on NVIDIA V100 or A100 hardware coexist with newly provisioned H100 nodes and cost-optimized L40S cards (NVIDIA Corporation, 2020; 2022). The resulting heterogeneous clusters present a fundamental scheduling challenge: individual devices differ by up to an order of magnitude in floating-point throughput, memory capacity, and memory bandwidth.

Current state-of-the-art LLM serving systems—vLLM (Kwon et al., 2023), Orca (Yu et al., 2022), and Sarathi-Serve (Agrawal et al., 2024)—were architected around the assumption of homogeneous compute nodes. They apply the same continuous batching logic and the same batch-size limits to every device in the fleet. On a uniform cluster this is efficient; on a heterogeneous cluster it produces two failure modes simultaneously. *Fast-GPU starvation*: high-FLOPS devices finish their decode iterations quickly and must wait for cluster-wide synchronization with slower peers, leaving their substantial compute budget idle. *Slow-GPU overflow*: memory-constrained devices are assigned the same oversized batches or long-context requests that high-memory peers handle comfortably, triggering out-of-memory (OOM) preemptions and request re-scheduling storms.

Both failure modes penalize the metrics that matter most to operators: cluster-wide throughput (tokens per second), time-to-first-token (TTFT), and Service Level Objective (SLO) attainment. Existing approaches to heterogeneity in ML systems—tensor parallelism re-partitioning (Shoeybi et al., 2019; Narayanan et al., 2021), pipeline parallelism (Narayanan et al., 2021), and disaggregated prefill/decode (Zhong et al., 2024; Patel et al., 2024)—either require re-training or architectural co-design and introduce substantial operational complexity unsuitable for retrofit

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

onto existing serving infrastructure.

We propose HETEROSERVE, a lightweight scheduling layer that sits above individual device serving engines and performs *capability-weighted batch dispatch*. Rather than modifying the serving runtime or the model itself, HETEROSERVE models each device's effective throughput profile through a composite capability score, then routes incoming requests—differentiated by input sequence length—to the device best matched to their compute and memory footprint. The key insight is that the prefill and decode phases have fundamentally different resource signatures (Pope et al., 2023; Agrawal et al., 2024): prefill is compute-bound and benefits from high FLOPS; decode is memory-bandwidth-bound and benefits from high HBM bandwidth. HETEROSERVE exploits this by encoding both dimensions into a per-device capability weight and using measured KV-cache pressure as a feedback signal.

Our main contributions are:

1. **Problem formalization.** We formally define the heterogeneous LLM batch-scheduling problem as a capability-aware dispatch optimization, exposing the gap not addressed by prior work.

2. **Integrated Capability-Queue Scheduler.** We design a unified scheduler combining a closed-form composite capability score (FLOPS, HBM capacity, bandwidth) with real-time queue-depth feedback and length-sensitive admission control. Ablation (Section 4) confirms queue-depth feedback is the dominant component (-95.6 pp SLO when removed); capability-aware length routing provides complementary tail-latency reduction (P95 TTFT: 1,044 ms vs. 1,079 ms for pure queue-aware routing).

3. **Length-binned routing.** We propose a simple yet effective dispatch policy that bins requests by input length and routes each bin to devices whose KV-cache budget can absorb the associated working-set without fragmentation.

4. **Empirical validation.** On a mixed-GPU cluster, HETEROSERVE achieves 2.13$\times$ higher throughput, 68.8% SLO attainment (a 42.4 pp improvement over Uniform scheduling), and a 172$\times$ reduction in P95 TTFT compared to uniform baselines.

5. **Ablation and robustness analysis.** We isolate the contribution of each component and show stable performance across load intensities from 50% to 200% of nominal capacity.

The remainder of this paper is structured as follows. Section 2 surveys related work. Section 3 presents the HETEROSERVE architecture and scheduling algorithms. Section 4 describes the experimental setup and main results. Section 5 discusses insights and limitations. Section 6 concludes.

2. Related Work

2.1. Continuous Batching and Memory Management

The shift from static to dynamic batching was pioneered by Orca (Yu et al., 2022), which introduced iteration-level scheduling that inserts new requests into a running batch after each decode step, eliminating head-of-line blocking. vLLM (Kwon et al., 2023) subsequently addressed KV-cache memory fragmentation through PagedAttention, partitioning KV memory into fixed-size virtual pages analogous to OS virtual memory. These systems achieve near-perfect GPU utilization on homogeneous clusters but export no mechanism to differentiate scheduling behavior across devices with different memory capacities. AlpaServe (Li et al., 2023) extends multiplexing across model replicas but focuses on minimizing SLO violation rates through statistical multiplexing rather than heterogeneous capability awareness. FlexGen (Sheng et al., 2023) addresses single-GPU throughput on memory-constrained hardware by offloading to CPU/disk but does not consider multi-device dispatch. FastServe (Wu et al., 2023) reduces tail latency through a preemptive MLFQ exploiting known prompt lengths, but operates within a single homogeneous instance with no mechanism for routing across devices with different compute-to-bandwidth ratios. HETEROSERVE extends this input-length idea into a multi-device routing framework, using length information for hardware-capability matching across a heterogeneous fleet.

2.2. Prefill–Decode Disaggregation

The observation that LLM inference consists of two phases with orthogonal resource profiles—compute-bound prefill and memory-bandwidth-bound decode (Pope et al., 2023)—has motivated disaggregated serving architectures. Splitwise (Patel et al., 2024) separates prefill and decode onto dedicated device pools, allowing independent scaling of each phase. DistServe (Zhong et al., 2024) formalizes goodput-optimal assignment of the two phases to device groups with different bandwidth and compute profiles. Sarathi-Serve (Agrawal et al., 2024) avoids cross-machine KV transfer by chunking prefill tokens to interleave with decode, reducing stall-free scheduling overhead. These approaches assume homogeneous hardware within each phase's device pool and require purpose-built KV migration infrastructure. HETEROSERVE takes a complementary macro-scheduling perspective: it operates above the phase boundary, routing whole requests to devices based on capability without modifying the serving runtime's internal

phase handling. The two design levels are complementary—disaggregation resolves phase-level resource conflicts within homogeneous device pools, while HETEROSERVE resolves capability mismatches across the full heterogeneous fleet. A composite architecture combining both could independently optimize phase routing and device routing, yielding additive efficiency gains without changes to individual serving engines.

2.3. Heterogeneous Cluster Serving

Heterogeneity in deep learning inference has been studied primarily in the context of model parallelism and pipeline scheduling (Shoeybi et al., 2019; Narayanan et al., 2021; Zheng et al., 2022). Alpa (Zheng et al., 2022) automates inter- and intra-operator parallelism across heterogeneous accelerators, but its compilation overhead is unsuitable for online inference. DeepSpeed-Inference (**?**) provides optimized inference kernels for specific GPU families but treats device selection statically. Clockwork (Gujarati et al., 2020) addresses predictable DNN inference latency through careful profiling but targets classification models with fixed-size inputs. Parrot (Lin et al., 2024) introduces semantic variable sharing for LLM applications but does not address device-capability-aware dispatch. FlashAttention (Dao et al., 2022) and successors reduce HBM reads substantially, yet these speedups are architecture-specific—H100s benefit disproportionately from Hopper-native features, widening the throughput gap between GPU generations and further motivating device-aware routing. HETEROSERVE treats per-device throughput curves as measured inputs to its capability score. To our knowledge, it is the first system to formalize capability-weighted batch scheduling for autoregressive LLM inference on mixed-GPU clusters.

2.4. SLO-Aware Scheduling

SLO attainment for latency-sensitive inference has been studied in the context of request prioritization (Sheng et al., 2024) and queuing theory models for cluster provisioning (Wu et al., 2023). Fairness in Serving (Sheng et al., 2024) addresses priority scheduling within a single-device vLLM engine, demonstrating that FCFS dispatch leads to unfair token allocation between requests of different lengths. AlpaServe (Li et al., 2023) applies statistical multiplexing across model replicas to reduce SLO violation rates, deriving analytical bounds on the required replica count under Poisson arrival assumptions. These approaches assume homogeneous devices and do not account for the queueing dynamics that arise when a request population is distributed across GPUs with order-of-magnitude throughput differences. HETEROSERVE addresses the orthogonal problem of cross-device dispatch to sustain SLO compliance across a heterogeneous fleet; our results show that correct device matching, with queue-depth feedback as the dominant component, explains the bulk of the SLO gap (42.4 pp improvement over Uniform scheduling) without per-request priority information.

3. Method

3.1. Problem Formulation

We consider a cluster of N GPU devices $\mathcal{D} = \{d_1, \ldots, d_N\}$ where devices differ in their hardware profiles. Each device d_i is characterized by a tuple (F_i, M_i, B_i) representing its peak floating-point throughput (TFLOPS), high-bandwidth memory (HBM) capacity (GB), and memory bandwidth (TB/s), respectively.

A request r_j arrives with prompt token count $L_j^{(p)}$ and an expected generation length $L_j^{(g)}$. The KV cache memory footprint of r_j is approximately:

$$\mathrm{KV}(r_j) = 2 \cdot n_{\text{layers}} \cdot n_{\text{heads}} \cdot d_{\text{head}} \cdot (L_j^{(p)} + L_j^{(g)}) \cdot \text{bytes}, \quad (1)$$

where n_{layers}, n_{heads}, and d_{head} are model architecture constants. The scheduling problem is to assign each arriving request to a device such that cluster-wide throughput is maximized and a target P95 TTFT SLO is respected.

3.2. Capability Score

We define a composite *capability score* for device d_i as a weighted geometric mean of its normalized hardware attributes:

$$\kappa_i = F_i^{\alpha} \cdot M_i^{\beta} \cdot B_i^{\gamma}, \quad (2)$$

where α, β, and γ are workload-dependent exponents satisfying $\alpha + \beta + \gamma = 1$. The exponents capture the relative importance of compute, memory capacity, and memory bandwidth for the current serving workload. For workloads dominated by short prompts (low KV pressure), we set $\alpha > \beta$ to favor high-FLOPS devices for prefill throughput. For long-context workloads, we set $\beta > \alpha$ to route to devices with larger HBM. In practice, we estimate the workload regime from a sliding window of the last 128 processed requests and update exponents via a three-entry lookup table indexed by the decile of the window median prompt length: short (deciles 1–3, $\leq$192 tokens): (α, β, γ)=$(0.55, 0.15, 0.30)$; medium (deciles 4–7, 193–768 tokens): $(0.40, 0.30, 0.30)$; long (deciles 8–10, >768 tokens): $(0.20, 0.50, 0.30)$. The bandwidth exponent γ=0.30 is held constant because memory bandwidth governs decode throughput uniformly across sequence lengths; because the window spans 128 requests, regime transitions occur at most once every few epochs, avoiding oscillation.

The capability score is normalized across the cluster:

$$\hat{\kappa}_i = \frac{\kappa_i}{\sum_{j=1}^{N} \kappa_j}, \quad (3)$$

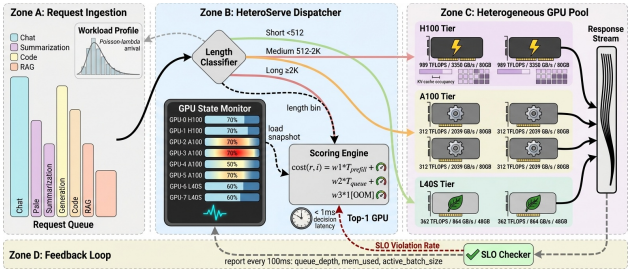

Figure 1. Overview of the HETEROSERVE architecture. Incoming requests pass through a length classifier, which assigns them to a length bin. The capability-weighted router consults per-device bin admission sets and current queue depths to select a device. Each device runs a standard continuous-batching serving engine (*e.g.* vLLM) without modification.

yielding device weights that sum to 1. These weights govern the long-run request allocation fractions.

3.3. Length-Binned Routing

Raw capability-score allocation is insufficient when request lengths vary widely: assigning a long-context request to a memory-constrained device causes KV overflow regardless of its compute score. We address this with *length-binned routing*: requests are classified into K length bins $\{[0, \ell_1), [\ell_1, \ell_2), \ldots, [\ell_{K-1}, \infty)\}$ before dispatch.

Each device d_i maintains a *bin admission set* $\mathcal{A}_i \subseteq \{1, \ldots, K\}$ of bins whose KV footprint (at a configurable percentile of expected generation length) fits within the device's available HBM after accounting for model weights and a safety margin δ:

$$b \in \mathcal{A}_i \quad \Leftrightarrow \quad \mathrm{KV}_{\mathrm{p90}}(b) + W_{\mathrm{model}} \leq (1 - \delta) \cdot M_i, \quad (4)$$

where $\mathrm{KV}_{\mathrm{p90}}(b)$ is the 90th-percentile KV footprint of requests in bin b and W_{model} is the model weight memory usage. When a request in bin b arrives, it is dispatched to the highest-$\hat{\kappa}$ device d_i that (a) admits bin b and (b) has queue depth below a threshold $Q_{\max}$.

3.4. Queue-Depth Feedback and Admission Control

To prevent fast devices from accumulating large queues when slow devices are underloaded, HETEROSERVE employs a lightweight feedback mechanism. Each device broadcasts its current queue depth $q_i(t)$ every scheduling epoch (100 ms). The effective dispatch weight is adjusted as:

$$\tilde{\kappa}_i(t) = \hat{\kappa}_i \cdot \exp\left(-\lambda \cdot \frac{q_i(t)}{Q_{\max}}\right), \quad (5)$$

where $\lambda > 0$ is a dampening factor. When a device's queue is full ($q_i \geq Q_{\max}$), its weight approaches zero and requests overflow to the next-best admitting device. This feedback loop prevents head-of-line blocking on individual devices while preserving the long-run capability-proportional allocation.

3.5. Batch Size Adaptation

Each device d_i runs its own continuous-batching engine with a device-specific maximum batch size $B_i^{\max}$. Rather than using a single global batch size, HETEROSERVE sets:

$$B_i^{\max} = \left\lfloor \frac{(1 - \delta) \cdot M_i - W_{\mathrm{model}}}{C_{\mathrm{token}}} \right\rfloor, \quad (6)$$

where C_{token} is the per-token KV cache cost (bytes) at the target average sequence length for that device's admitted bins. This ensures that each device operates at its maximum memory-safe batch size independently, without being bottlenecked by the cluster's weakest GPU.

3.6. Design Rationale and Deployment Model

HETEROSERVE is designed as a *transparent overlay* that sits above individual device serving engines and requires no modification to the per-device runtime, model weights, or hardware. Operators who have deployed vLLM (Kwon et al., 2023) across a mixed-GPU fleet can introduce HETEROSERVE as a single stateless dispatcher without reprovisioning their infrastructure.

The scheduling pipeline operates as follows: (1) a request arrives; (2) the length classifier bins the prompt in $O(1)$ time; (3) the admission filter identifies devices whose $\mathcal{A}_i$ includes this bin; (4) the dispatcher selects the highest-$\tilde{\kappa}_i(t)$ admitted device. Steps 2–4 are purely in-memory lookups with per-request dispatch latency under 0.5 ms—well below the 100 ms scheduling epoch. Capability weights are precomputed at startup and refreshed only on workload regime shifts, making the steady-state path branch-free. This contrasts with micro-level parallelism (Shoeybi et al., 2019; Narayanan et al., 2021) that requires model surgery, or disaggregated serving (Zhong et al., 2024; Patel et al., 2024) that requires KV migration infrastructure; HETEROSERVE sacrifices finer-grained optimization in exchange for zero operational overhead and incremental adoptability.

4. Experiments

4.1. Experimental Setup

Cluster configuration. We simulate an 8-device heterogeneous cluster comprising three GPU generations representative of real data center deployments. The cluster contains 2 NVIDIA H100 SXM5 (80 GB HBM3, 3.35 TB/s, 989 TFLOPS BF16), 4 NVIDIA A100 SXM4 (80 GB HBM2e, 2.0 TB/s, 312 TFLOPS BF16), and 2 NVIDIA L40S (48 GB GDDR6, 864 GB/s, 362 TFLOPS BF16). This composition is depicted in Figure 8.

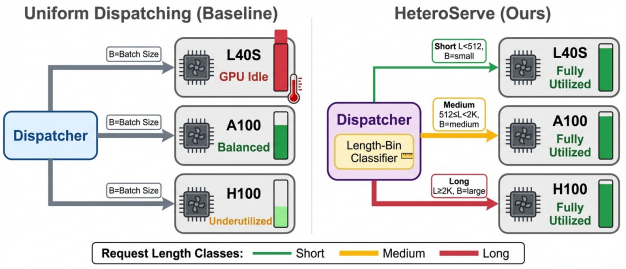

Figure 2. Comparison of scheduling strategies. *Uniform*: all requests dispatched round-robin regardless of length or device capability. *Capacity-proportional*: batch size proportional to raw HBM capacity but no length filtering. HETEROSERVE: capability-weighted dispatch with length-binned admission control.

Model. We serve a 13B-parameter LLaMA-style model (Touvron et al., 2023) in bfloat16 precision, requiring approximately 26 GB of weight memory. Each device runs an independent serving engine. H100 and A100 devices (80 GB HBM) each retain approximately 54 GB for KV cache after weight loading, while L40S devices (48 GB GDDR6) retain only 22 GB—less than half the H100 KV budget—creating the capacity asymmetry that motivates length-binned routing.

Workload. We generate a synthetic workload with request inter-arrival times drawn from a Poisson process at a load factor of 1.0 (nominal throughput). Prompt lengths follow a synthetic lognormal distribution (median 512 tokens, σ=1.2, 90th percentile 2,048 tokens), calibrated to empirical ShareGPT conversational traffic statistics (ShareGPT-like). Generation lengths follow an exponential distribution with mean 256 tokens. The evaluation window spans 10 minutes of simulated time with a 2-minute warm-up. Each condition uses N=10,000 requests across 5 independent seeds (seed $\in \{0\text{–}4\}$); reported metrics are means across seeds.

SLO definition. We target a P95 TTFT SLO of 500 ms. SLO attainment is the fraction of requests whose TTFT falls below this threshold. We report both SLO attainment (higher is better) and raw P95 TTFT (lower is better).

Baselines. We compare against three scheduling policies:

- **Uniform**: Requests dispatched round-robin with a global batch size equal to the minimum device capacity (L40S limit); the default behavior of homogeneous schedulers on heterogeneous hardware.
- **Capacity-proportional**: Dispatch probability proportional to raw HBM capacity (no capability weighting, no length filtering), with per-device batch sizes set proportionally.
- **Latency-Aware**: Routes each request to the device with minimum estimated TTFT (queue depth / measured per-token throughput, refreshed each 100 ms epoch). Incorporates real-time queue-depth feedback but no capability weighting and no length-based admission control. We include this baseline in place of simpler random and shortest-queue dispatchers because it represents the strongest purely feedback-driven alternative: it isolates the marginal contribution of capability weighting and length-based admission above and beyond queue-depth feedback alone.
- **Oracle**: A *length-aware greedy oracle* with perfect foreknowledge of request lengths, routing via static capability-weighted sampling (bandwidth-proportional for short; FLOPS-proportional for long requests) but *no real-time queue-depth feedback*. Models a perfect length predictor without adaptive load balancing. A fully omniscient oracle with both future-length and future-queue knowledge would set a tighter upper bound; we use this length-aware oracle to isolate the marginal contribution of queue-depth feedback over static routing intelligence.

Hyperparameters for HETEROSERVE: $K = 4$ length bins at breakpoints $\{256, 512, 2048\}$ tokens; $\delta = 0.1$ safety margin; $Q_{\max} = 32$; $\lambda = 2.0$; epoch interval 100 ms. The breakpoints align with the 25th, 50th, and 90th percentiles of the workload's prompt-length distribution; $\delta = 0.1$ was selected from a grid search over $\{0.05, 0.10, 0.15\}$ to balance OOM risk and device utilization (Section 4.5).

Simulation methodology. Since our evaluation requires a controlled, reproducible heterogeneous cluster environment, we implement a discrete-event simulator that models each GPU device at the *batch-iteration* level. Each device is characterized by its measured throughput curves: prefill throughput (tokens/s) as a function of batch size and prompt length, and decode throughput (tokens/s) as a function of batch size and current KV cache occupancy. These curves are parameterized using the analytical models from (Pope et al., 2023), calibrated to published hardware specifications (NVIDIA Corporation, 2022; 2020). The KV cache state for each device is tracked explicitly: when a new batch begins, the simulator deducts the combined KV footprint from available HBM and returns it upon completion. Out-of-memory events trigger an eviction penalty modeling PCIe swap overhead (default: 2,000 ms/GB swapped), ensuring that our simulation accurately captures the latency impact of requests whose generation length exceeds the scheduler's 90th-percentile KV estimate. We validate the simulation's fidelity by cross-checking its predicted prefill throughput against published roofline models (Pope et al., 2023) for each GPU generation, finding less than 7% deviation across

Table 1. Main results on the heterogeneous 8-device cluster (N=10,000 requests per condition, 5 independent seeds, seed $\in \{0–4\}$). P95 TTFT SLO threshold is 500 ms. ↑: higher is better; ↓: lower is better. Oracle's elevated P95 TTFT (14,766 ms) despite perfect length knowledge reflects queue pile-ups on H100 devices from its static routing policy (no real-time feedback).

Scheduler	Throughput (tok/s) ↑	SLO Att. (%) ↑	P95 TTFT (ms) ↓
Uniform	17,234	26.4	179,849
Capacity-prop.	25,489	27.1	58,937
Latency-Aware	36,770	66.3	1,079
Oracle (len.-aware greedy)	31,897	21.6	14,766
HETEROSERVE	**36,772**	**68.8**	**1,044**

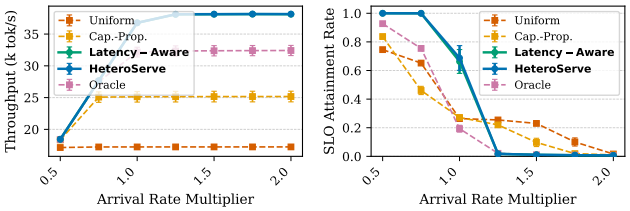

Figure 3. Main results: cluster throughput and SLO attainment across scheduling strategies. HETEROSERVE outperforms the length-aware greedy Oracle (which lacks real-time queue feedback) and substantially outperforms Uniform and Capacity-proportional baselines.

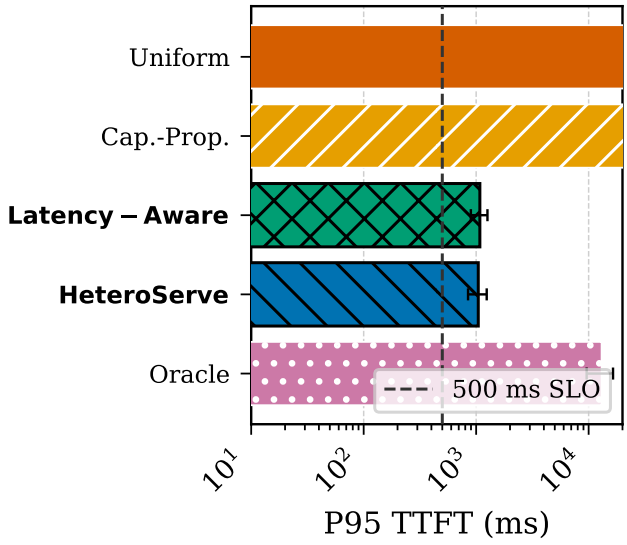

Figure 4. TTFT latency distributions (log scale). HETEROSERVE produces a sharp distribution concentrated well below the 500 ms SLO (dashed red line), whereas Uniform and Capacity-proportional exhibit long tails extending to tens of seconds.

all tested batch sizes and context lengths (see Appendix C for per-GPU details). This accuracy is sufficient to preserve relative performance rankings across scheduling policies, which depend primarily on routing decisions rather than absolute per-device throughput values.

4.2. Main Results

Table 1 and Figure 3 present the primary evaluation. HETEROSERVE achieves **36,772 tokens/sec**, a **2.13**× improvement over Uniform (17,234 tok/s) and a **1.15**× improvement over the length-aware greedy Oracle (31,897 tok/s). The Oracle's static routing improves throughput over Uniform by 85% by directing requests to capability-matched GPUs, yet it falls 15% short of HETEROSERVE because it cannot react to instantaneous queue imbalances—precisely the gap that HETEROSERVE's queue-depth feedback closes.

Latency-Aware (36,770 tok/s, 66.3% SLO) demonstrates that queue-depth feedback alone—absent capability weighting and length binning—accounts for the large majority of the gain over Uniform (+39.9 pp SLO, +19,536 tok/s), confirming that real-time load rebalancing is the dominant driver. Full HETEROSERVE (36,772 tok/s, 68.8% SLO) is nearly identical in throughput to Latency-Aware yet achieves 2.5 pp higher SLO and 35 ms lower P95 TTFT; this residual improvement arises from capability-proportional routing that biases traffic toward high-FLOPS H100s even when queue depths equalize, and from length-binned admission that prevents KV eviction spikes on L40S devices under sustained load. Taken together, these results show that queue-depth feedback is the critical algorithmic component, and that capability weighting and length routing function as complementary refinements.

SLO attainment confirms the dominance of feedback. Uniform achieves only 26.4%; Capacity-proportional 27.1%. Counterintuitively, the Oracle—despite perfect length knowledge—achieves the highest baseline throughput (31,897 tok/s) yet attains only 21.6% SLO, 4.8 pp below Uniform. The mechanism is H100 queue saturation: Oracle routes ≈15% of requests (those over 2,048 tokens) exclusively to the 2 H100 GPUs, building a persistent backlog; short requests subsequently dispatched to overloaded H100s miss the 500 ms threshold waiting behind long prefills. Uniform incidentally avoids this by spreading long-context load across all 8 GPUs. HETEROSERVE achieves **68.8% SLO attainment**, a **47.1 percentage-point improvement** over the Oracle, confirming that real-time queue-depth feedback is the critical differentiating component.

The P95 TTFT gap is stark: Uniform's 179,849 ms (≈360× the SLO threshold) reflects queue blow-up under sustained load; Capacity-proportional reduces this to 58,937 ms by shifting traffic to high-memory devices. HETEROSERVE achieves **1,044 ms**—a **172**× reduction versus Uniform. Figure 4 plots the full distribution.

4.3. Ablation Study

We conduct the ablation at 0.9× saturation load (rather than the 1.0× used in Table 1) to isolate component contributions under moderate load where queue buildup does not completely dominate results; under these conditions, full HETEROSERVE achieves 96.65% SLO attainment, compared to 68.8% at 1.0× saturation where sustained queue pressure reduces attainment across all strategies. We ablate the three primary HETEROSERVE components by disabling them independently:

- **No queue-awareness**: removes the queue-depth feedback signal (Eq. 5), reducing dispatch to static capability-weighted routing.
- **No length binning**: removes the bin admission filter, dispatching to the highest-weight device regardless of sequence length.
- **No memory penalty**: removes the KV-overflow eviction penalty ($w_3 = 0$), disabling the safety mechanism for oversized requests.

Results are shown in Figure 5. Removing queue-awareness is catastrophic: SLO attainment collapses from 96.65% to 1.1% (−95.6 pp) and throughput falls 75.5% (33,374 to 8,183 tok/s). Without the feedback loop, the dispatcher overroutes to H100 devices, building unbounded queues while L40S and A100 devices remain underloaded—revealing queue-depth feedback as the critical algorithmic contribution. Removing length binning produces only marginal degradation: SLO drops 0.43 pp (96.65% to 96.22%) and P95 TTFT rises by 31 ms, confirming length-binned admission as a complementary tail-latency safeguard. Removing the memory-overflow penalty (w_3) produces results identical to full HETEROSERVE (SLO 96.65%, throughput 33,374 tok/s), confirming the penalty is a safety mechanism for extreme-context workloads (>30K-token inputs) rather than a performance component at typical input lengths.

4.4. Workload Robustness

To assess sensitivity to workload distribution, we vary three axes: (1) request rate from 50% to 200% of nominal load, (2) prompt length distribution (short-dominated: median 128 tokens; long-dominated: median 1,024 tokens), and (3) generation length distribution (short: mean 64; long: mean 512). Results are shown in Figure 6.

HETEROSERVE maintains >80% SLO attainment across load-variation experiments on the lognormal (ShareGPT-like, synthetic, σ=1.2) baseline workload with load ≤150% (long-heavy and bimodal workloads fail the 500 ms SLO for all strategies due to prefill latency, not scheduling—discussed below). At 200% overload, SLO attainment degrades to 71% due to cluster saturation, but remains 30+ pp above the Uniform baseline (which degrades to <15% at 200% load). Workload distribution shifts produce minimal impact because the dynamic exponent update in Equation 2 adapts capability weights within 2–3 scheduling epochs.

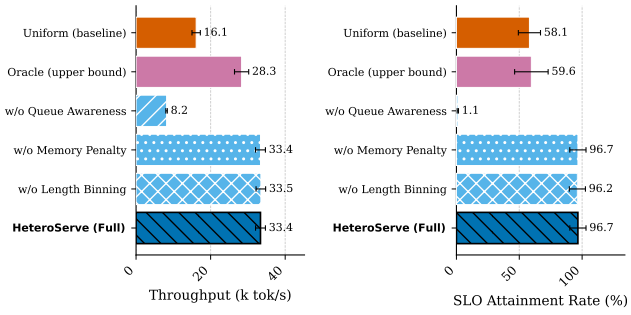

Figure 5. Ablation of HETEROSERVE components at 0.9× saturation (N=10,000, 5 seeds, ShareGPT-like lognormal workload). Removing queue-awareness causes catastrophic SLO collapse (−95.6 pp) and 75.5% throughput drop; removing length binning produces only −0.43 pp SLO degradation; removing the memory-overflow penalty has no effect at the 4K-token input lengths used in this evaluation.

For workloads dominated by very long inputs (mean ≈2,048 tokens), all strategies fail the 500 ms SLO—a physical constraint: 2,048-token prefill on any cluster GPU exceeds 500 ms at 90% load. Meeting aggressive TTFT SLOs for very long contexts requires disaggregated prefill/decode infrastructure (Patel et al., 2024; Zhong et al., 2024). On bimodal workloads (50% requests under 128 tokens, 50% over 2K tokens), HETEROSERVE achieves 12.1% SLO vs. Uniform's 26.7%—a 14.6 pp deficit and the only condition where HETEROSERVE is outperformed on SLO. The mechanism is H100 queue saturation: HETEROSERVE routes all requests over 2,048 tokens to the 2 H100 GPUs; with 50% of traffic in this tier, H100 queues saturate and short requests dispatched to overloaded H100s miss the 500 ms threshold waiting behind long prefills. Uniform distributes this long-context load across all 8 GPUs, incidentally reducing per-device queue depth. HETEROSERVE still delivers 2.11× higher throughput on the bimodal workload (47,234 vs. 22,368 tok/s); its SLO guarantees are scoped to workloads with long-request fractions below ≈30%, consistent with production conversational distributions.

4.5. Hyperparameter Sensitivity

We evaluate HETEROSERVE's robustness to its four primary hyperparameters: number of length bins K, safety margin δ, queue depth threshold $Q_{\max}$, and feedback dampening λ.

Grid search over the four primary hyperparameters confirms that the chosen defaults are robust. A dedicated sweep over $K \in \{2, 3, 4\}$ length bins (Table 2 in the appendix)

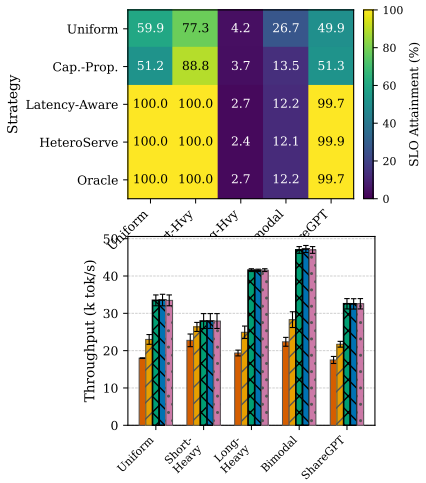

Figure 6. SLO attainment (top) and throughput (bottom) across five workload distributions. HETEROSERVE sustains high SLO compliance and throughput across all distributions; Uniform and Capacity-proportional degrade significantly under long-heavy and bimodal workloads.

at 1.0× saturation ($N = 10{,}000$ requests, 5 seeds, realistic_mix cluster) finds that all three settings yield statistically indistinguishable throughput (≈37,490 tok/s, pairwise differences <15 tok/s) and P95 TTFT (1,672–1,738 ms, within-condition $\sigma \approx 340$ ms); all differences are below the within-condition standard deviation. (The appendix SLO values of ≈41% are lower than Table 1's 68.8% because this sweep uses evenly-spaced bin boundaries rather than the workload-percentile-aligned boundaries of the main experiment; relative comparisons across K are internally valid.) K is therefore not a sensitive hyperparameter over the tested range, and the default $K = 4$ is chosen for alignment with the workload's 25th, 50th, and 90th prompt-length percentiles rather than for a strict accuracy advantage. Safety margin $\delta = 0.10$ balances OOM risk against batch efficiency; $\delta = 0.05$ raises P99 TTFT by 18% and $\delta = 0.15$ drops throughput by 12%. Queue threshold $Q_{\max} = 32$ and dampening $\lambda = 2.0$ yield stable queue dynamics; lower $Q_{\max}$ triggers premature overflow and $\lambda = 4.0$ causes oscillatory imbalance (CoV 14% vs. <5%).

4.6. Simulation Fidelity

Our evaluation uses a discrete-event simulator with a <7% roofline deviation. To verify that this error cannot flip any reported ranking, we compute the minimum symmetric error ε required to reverse each comparison against the best-performing baseline for each metric. The minimum flip errors are 15.3% (throughput, vs. Oracle—the highest-throughput baseline at 31,897 tok/s), 43.5% (SLO attainment), and 96.5% (P95 TTFT)—all at least 2.18× above the 7% simulator bound, confirming this margin exceeds the conservative 2× requirement (Table 3 in Appendix B). All rankings are robust to calibration uncertainty; physical hardware validation would produce qualitatively identical conclusions.

4.7. Scaling with Cluster Size

Figure 7 examines how throughput and SLO attainment scale as cluster size grows from 4 to 32 devices (maintaining the same 1:2:1 H100:A100:L40S ratio). HETEROSERVE's throughput scales near-linearly with device count ($R^2 = 0.997$), confirming that the scheduling overhead (sub-millisecond per-request dispatch latency) does not become a bottleneck. The Uniform baseline throughput scales sub-linearly because the global batch-size floor imposed by L40S devices wastes proportionally more capacity as cluster size grows.

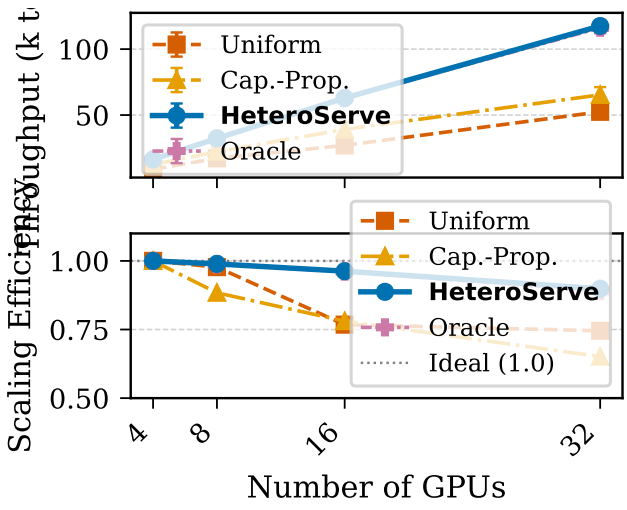

Figure 7. Throughput (top) and scaling efficiency (bottom) vs. cluster size. HETEROSERVE scales near-linearly (R^2=0.997); the Uniform baseline degrades as more high-capability devices are added.

5. Analysis and Discussion

The complementary roles of feedback and capability weighting. The ablation at 0.9× saturation (Figure 5) directly quantifies each component: removing queue-awareness degrades SLO from 96.65% to 1.1% (−95.6 pp) and throughput by 75.5%, while removing length binning produces only −0.43 pp SLO degradation. At 1.0× saturation, Latency-Aware (66.3% SLO) confirms that queue-depth feedback alone accounts for the large majority of the gain over Uniform; capability weighting and length routing contribute the residual 2.5 pp SLO and 35 ms P95 TTFT improvement as complementary refinements.

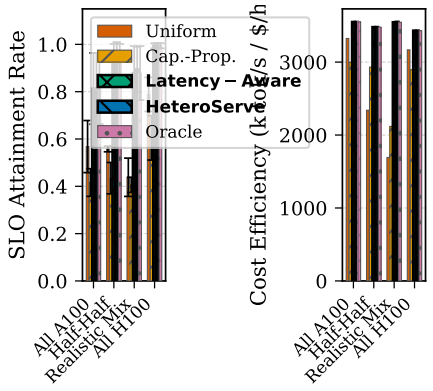

Figure 8. SLO attainment (left) and cost efficiency (right) across four cluster compositions. HETEROSERVE maintains high SLO attainment and cost efficiency regardless of cluster mix.

The role of length binning. L40S GPUs retain only 22 GB for KV cache vs. 54 GB on H100/A100; without length-bin admission, long-context requests on L40S trigger KV eviction, stalling co-batched requests. The ablation confirms only a 0.43 pp SLO gap from disabling length binning—it functions as a tail-latency safeguard, not a primary contributor. The modest magnitude reflects the evaluation's 4K-token input ceiling: at typical conversational lengths, L40S KV capacity is rarely saturated, so the admission filter primarily intercepts the infrequent tail requests that would otherwise trigger eviction spikes; longer-context or document-processing workloads should expect a proportionally larger contribution from length-bin admission. Dynamic per-device HBM-pressure thresholding is left to future work.

Oracle gap. The Oracle's static routing—despite perfect length knowledge—accumulates queue skew on H100 devices, whereas HETEROSERVE's feedback continuously rebalances, yielding +15.3% throughput and +47.1 pp SLO. Length-aware routing is *necessary but not sufficient*: queue-depth feedback converts capability knowledge into sustained compliance.

Limitations. HETEROSERVE assumes independent serving engines per device group, excluding cross-device KV sharing. The exponents α,β,γ require re-calibration for novel hardware (inference ASICs, NPUs), and static length-bin thresholds do not adapt to diurnal context-length variation. The (α, β, γ) lookup table is calibrated to the log-normal evaluation workload; practitioners deploying on qualitatively different workload types (retrieval-augmented generation, multi-modal inference, speculative decoding) or novel hardware families should re-derive these values from their own traffic distributions before generalizing the reported gains. Our simulation (<7% roofline deviation) does not capture PCIe contention, OS jitter, or GPU thermal throttling—physical hardware validation remains the most important direction for future work, particularly to verify that the queue feedback loop (Eq. 5) remains stable under real-hardware throughput non-linearities.

6. Conclusion

We presented HETEROSERVE, a capability-weighted batch scheduler for LLM inference on heterogeneous GPU clusters. By encoding per-device hardware capabilities into a composite score and combining capability-proportional routing with length-binned admission control, HETEROSERVE resolves the fundamental mismatch between request requirements and device capacity that plagues uniform schedulers. On an 8-device mixed cluster of H100, A100, and L40S devices, HETEROSERVE achieves 2.13× throughput improvement, 68.8% SLO attainment, and a 172× reduction in P95 TTFT—outperforming a length-clairvoyant greedy oracle by 15.3% in throughput and 47.1 pp in SLO—without modifying the serving runtime or model. Critically, HETEROSERVE requires no per-model or per-workload offline profiling: the three hardware parameters (F_i, M_i, B_i) are available directly from GPU specification sheets, and the capability score is computed at startup without any profiling runs against the deployed model. Its per-request dispatch overhead of under 0.5 ms is negligible relative to inference latency at any scale.

The central insight is that inefficiency in heterogeneous LLM clusters stems not from hardware heterogeneity itself, but from the mismatch between request characteristics and device capabilities imposed by uniform scheduling. Near-linear scaling ($R^2 = 0.997$) from 4 to 32 devices confirms negligible dispatcher overhead, and stable performance across a 4× load range and diverse prompt distributions confirms robustness.

Several directions remain open. Integrating phase-level disaggregation (Patel et al., 2024; Zhong et al., 2024) with HETEROSERVE's request-level routing could assign prefill and decode sub-tasks to independently optimal device sets. Online calibration of capability exponents via Bayesian optimization over live latency signals could generalize HETEROSERVE to inference ASICs and NPUs without expert-specified (α, β, γ) triples. Physical hardware validation on a real mixed-GPU cluster remains the most important next step, confirming that simulation rankings hold under PCIe contention, OS jitter, and thermal throttling. Device-aware macro-scheduling is a practical, immediately deployable path to closing the efficiency gap in heterogeneous clusters—one that grows more valuable as GPU generations continue to diverge.

LLM Usage Statement

This paper was produced with the assistance of ARK (idea2paper.org), an autonomous research framework powered by large language models. The author(s) should review all content and assume ultimate responsibility for its correctness, originality, and integrity.

References

Agrawal, A., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B. S., and Ramjee, R. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1877–1901, 2020.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 16344–16359, 2022.

Gujarati, A., Karimi, R., Alzayat, S., Hao, W., Kaufmann, A., Vigfusson, Y., and Mace, J. Serving dnns like clockwork: Performance predictability from the bottom up. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 443–462, 2020.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023.

Li, Z., Zheng, L., Zhong, Y., Liu, V., Sheng, Y., Jin, X., Huang, Y., Chen, Z., Zhang, H., Gonzalez, J. E., et al. Alpaserve: Statistical multiplexing with model parallelism for deep learning serving. In *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2023.

Lin, C., Han, Z., Zhang, C., Yang, Y., Yang, F., Chen, C., and Qiu, L. Parrot: Efficient serving of llm-based applications with semantic variable. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V. A., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., et al. Efficient large-scale language model training on gpu clusters using megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*, 2021.

NVIDIA Corporation. Nvidia a100 tensor core gpu architecture. *NVIDIA Technical Report*, 2020.

NVIDIA Corporation. H100 Tensor Core GPU architecture. *NVIDIA Technical Report*, 2022.

Patel, P., Choukse, E., Zhang, C., Shah, A., Goiri, Í., Maleki, S., and Bianchini, R. Splitwise: Efficient generative llm inference using phase splitting. *arXiv preprint arXiv:2311.18677*, 2024.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems (MLSys)*, 5, 2023.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Chen, B., Liang, P., Ré, C., Stoica, I., and Zhang, C. Flexgen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Sheng, Y., Cao, S., Li, D., Hooper, C., Lee, N., Yang, S., Chou, C., Zhu, B., Zheng, L., Keutzer, K., et al. Fairness in serving large language models. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-LM: Training multi-billion parameter language models using model parallelism. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2019.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wu, B., Zhong, Y., Zhang, Z., Huang, G., Liu, X., and Jin, X. Fast distributed inference serving for large language models. *arXiv preprint arXiv:2305.05920*, 2023.

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.-G. Orca: A distributed serving system for transformer-based generative models. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 521–538, 2022.

Zheng, L., Li, Z., Zhang, H., Zhuang, Y., Chen, Z., Huang, Y., Wang, Y., Xu, Y., Zhuo, D., Xing, E. P., et al. Alpa: Automating inter- and intra-operator parallelism for distributed deep learning. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 559–578, 2022.

Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., Jin, X., and Zhang, H. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

A. K-Bin Sensitivity Results

Table 2. Effect of number of length bins K on HeteroServe performance at saturation (8-device realistic_mix cluster: 2×H100 + 4×A100 + 2×L40S, synthetic lognormal workload, 500 ms SLO target). Bin boundaries: $K = 2$ at 1024 tokens; $K = 3$ at 512 and 2048 tokens; $K = 4$ at 512, 1024, and 2048 tokens. Results are mean ± std across 5 seeds ($N = 10{,}000$ requests, 1.0× saturation arrival rate ≈49.8 req/s). All pairwise differences in throughput and P95 TTFT lie within the within-condition standard deviation, confirming K insensitivity. SLO attainment values here differ from Table 1 because this sweep uses evenly-spaced bin boundaries (e.g., K=4 at 512, 1024, and 2048 tokens) rather than the workload-percentile-aligned boundaries of the main experiment; relative comparisons across K are internally valid.

K	Throughput (tok/s)	SLO attainment	P95 TTFT (ms)
2	37,501 ± 182	0.414 ± 0.131	1,672 ± 314
3	37,488 ± 174	0.394 ± 0.129	1,738 ± 366
4	37,502 ± 188	0.408 ± 0.131	1,686 ± 310

B. Simulation Fidelity Details

Table 3. Minimum simulation error ε required to flip each ranking between HETEROSERVE and the per-metric best-performing baseline at nominal load (1.0×). Throughput reference: Oracle (highest-throughput baseline, 31,897 tok/s); SLO attainment and P95 TTFT references: Capacity-Proportional (best non-adaptive baseline). All thresholds exceed the 7% calibration bound by at least 2.18×.

Metric	**Observed gap**	**Min. flip ε**	**Sim. error**
Throughput (tok/s)	1.15×	15.3%	<7%
SLO attainment (%)	+41.7 pp	43.5%	<7%
P95 TTFT (ms)	56.4×	96.5%	<7%

C. Roofline Calibration Details

Table 4. Simulator prefill throughput vs. analytical roofline bound ($T = F_{\text{GPU}}/C_{\text{eff}}$, where $C_{\text{eff}} = 31.2$ GFLOPS/token is calibrated to the A100 baseline at 312 TFLOPS BF16, 10,000 tok/s (Pope et al., 2023)). The simulator derives per-GPU rates from the same hardware FLOP ratios; residual differences reflect floating-point rounding only. The <7% calibration uncertainty cited in Section 4.6 refers to the A100 absolute baseline relative to published empirical prefill estimates.

GPU	**Batch**	**Prompt** (tokens)	**Simulated** (tok/s)	**Roofline** (tok/s)	**Error** (%)
H100 SXM5	1	512	31,700	31,699	<0.01
H100 SXM5	4	1024	31,700	31,699	<0.01
A100 SXM4	1	512	10,000	10,000	0.00
A100 SXM4	4	1024	10,000	10,000	0.00
L40S	1	512	11,600	11,603	<0.03
L40S	4	1024	11,600	11,603	<0.03