RAVEL: REVERSIBLE SLO-AWARE VIRTUAL PLACEMENT FOR GEO-DISTRIBUTED LLM SERVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Regional capacity constraints and rising LLM inference demand motivate pooling GPU resources across geographically distributed clusters, but heterogeneous WAN paths, serving conditions, and request-level service-level objectives (SLOs) make placement feasibility request-specific. A region that is substitutable for one request may be indispensable to another, creating a request-dependent opportunity cost for regional capacity. Placement finality therefore creates a time–information tradeoff. Finalizing too early can consume capacity that later demand uniquely needs, while delaying execution consumes finite SLO slack. Using fixed-delay late binding as a diagnostic, we find that short deferral can improve SLO attainment but that the gain reverses as the delay grows. In this setting, most recoverable loss arises because already-arrived demand is hidden from subsequent capacity planning. RAVEL addresses this tradeoff by decoupling demand visibility from placement finality. Each arriving request becomes visible to capacity planning immediately, while its geographic placement remains revisable until successful engine handoff. RAVEL combines transferable prospective accounting with bounded pre-handoff replanning so that earlier placements can release capacity as new demand and serving state become observable. We implement RAVEL as an asynchronous routing layer for vLLM and evaluate it on five serving replicas across three clusters connected by physical WAN links, using Qwen3-1.7B and Qwen3-8B. Across five paired Qwen3-1.7B replays at $16\times$ load, RAVEL improves request-level SLO attainment over the best-baseline comparator by 4.1, 7.5, and 17.4 percentage points on LMSYS, Burst, and DeepResearch, respectively. In the Qwen3-1.7B single-run load sweep at $16\times$, RAVEL also reduces P95 TTFT by 49.1–80.5% and P95 end-to-end latency by 15.5–56.8%.

1 INTRODUCTION

Growing LLM inference demand and uneven regional GPU availability motivate pooling serving capacity across geographically distributed clusters, but regional capacity is not equally useful to every request. Heterogeneous WAN paths (Jain et al., 2013; Hong et al., 2013), accelerator and serving conditions (Mei et al., 2025; Xia et al., 2026; Toniolo et al., 2026; Bernhard & Katla, 2026), and request-specific latency objectives (Zhong et al., 2024; Zhang et al., 2026; Liu et al., 2026; 2024; Wang et al., 2025) make placement feasibility request-specific. A region that is substitutable for one request may be indispensable to another. Regional capacity therefore has a request-dependent opportunity cost: assigning one request to a region can consume capacity that it could obtain elsewhere but a later request cannot. An SLO miss can consequently be allocation-induced even when sufficient serving capacity exists in aggregate. Related tail-latency effects also arise in large shared services (Dean & Barroso, 2013).

Consider two regions, A and B. Request r_1 can satisfy its SLO in either region, whereas a later request r_2 can succeed only in A. Assigning r_1 to A is reasonable when r_1 arrives, but once r_2 arrives, moving r_1 to B can free A for r_2. This opportunity exists only while r_1 remains revisable and B remains feasible for it.

Placement finality creates a time–information tradeoff. Keeping a placement revisable allows newly observed demand and serving state to inform the decision, while delaying execution consumes finite SLO slack. Existing work primarily improves the spatial placement decision using load, cache

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.

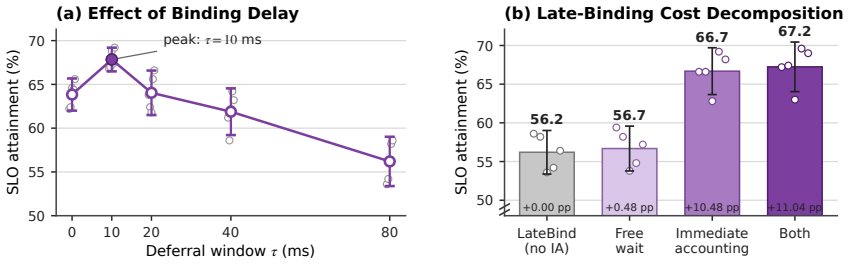


Figure 1: **Diagnosing fixed-delay late binding.** DeepResearch at $12\times$ load with Qwen3-1.7B (five runs). **(a)** Request-level SLO attainment versus binding delay. **(b)** Waiting-cost and demand-visibility interventions at $\tau = 80$ ms.

locality, WAN conditions, request-level SLOs, or adaptive feedback (Srivatsa et al., 2025; Yuan et al., 2026; Xia et al., 2026; Toniolo et al., 2026; Liu et al., 2026; Lim et al., 2026); some systems also revise or rebalance queued assignments as conditions evolve (Yuan et al., 2026). The central question is how to preserve useful placement flexibility within finite SLO slack as demand and serving state continue to evolve.

A fixed-delay diagnostic isolates what makes such deferral useful. Figure 1(a) shows that short deferral improves request-level SLO attainment, but the gain reverses as the delay grows. At $\tau = 80$ ms, Figure 1(b) separates the cost of waiting from the visibility of waiting demand. Excluding the waiting interval from SLO accounting improves attainment by only 0.5 percentage points, whereas making already-arrived demand prospectively visible improves it by 10.5 points; both interventions retain the same physical delay. In this diagnostic, hidden already-arrived demand accounts for most of the recoverable loss. Deferral can therefore provide additional information while simultaneously degrading the planning state needed to use it.

RAVEL addresses this tension by decoupling demand visibility from placement finality. Each request is tentatively placed and counted immediately, while its geographic destination remains revisable until successful engine handoff. Request-specific feasibility constrains valid destinations, prospective accounting transfers with each reassignment, and slack-bounded observation and bounded pre-handoff replanning allow earlier placements to release capacity as new demand and serving state become observable. This immediate-visibility, deferred-finality lifecycle preserves placement flexibility without hiding already-arrived work from intervening capacity decisions. Section 2 formalizes this lifecycle, and Section 3 presents its realization.

We implement RAVEL as an asynchronous routing layer for vLLM (Kwon et al., 2023) and evaluate it on five serving replicas across three physical-WAN clusters using Qwen3-1.7B and Qwen3-8B. Across five paired Qwen3-1.7B replays at $16\times$ load, RAVEL improves request-level SLO attainment over the fixed workload-specific comparator by 4.1, 7.5, and 17.4 percentage points on LMSYS, Burst, and DeepResearch, respectively. The Qwen3-8B extension preserves RAVEL's SLO-attainment advantage across all 18 workload–load settings.

This paper makes three contributions:

- **A time–information tradeoff in geo-distributed placement.** We identify the request-dependent opportunity cost of regional capacity and characterize the tradeoff between preserving placement flexibility and consuming finite SLO slack. A fixed-delay diagnostic shows that hidden already-arrived demand accounts for most recoverable loss in that setting, motivating the separation of demand visibility from placement finality.
- **An immediate-visibility, deferred-finality lifecycle.** We introduce a placement lifecycle in which arrived demand immediately influences capacity planning while Router-owned geographic assignments remain revisable until successful engine handoff. RAVEL realizes this lifecycle through request-specific feasibility, transferable prospective accounting, bounded active observation and replanning, and ownership-defined finality.
- **A physical-WAN implementation and evaluation.** We implement RAVEL as an asynchronous routing layer for vLLM and evaluate it across three workloads and two Qwen3 model sizes, showing improved request-level SLO attainment with low Router-side planning overhead.

2 When Should Placement Become Final?

We formalize geo-distributed placement through request-specific regional feasibility and the timing of demand visibility and geographic finality. For request i, let a_i denote its arrival time, t_i^{vis} the time when its prospective demand first enters the planning state used by subsequent placement decisions, and t_i^{final} the time after which the Router can no longer revise its geographic destination. Visibility refers to representation in the planning state, rather than merely knowledge that the request has arrived. Crucially, t_i^{vis} and t_i^{final} need not coincide.

2.1 Request-Specific Regional Feasibility

Regional feasibility is request- and time-specific. For request i and replica r, let $\Phi_{i,r}(t) = 1$ when r is predicted, using only information available at time t, to satisfy every applicable SLO constraint. This prediction depends on the request objective and remaining slack, the WAN path, current serving conditions, and prospective contention (Zhong et al., 2024; Zhang et al., 2026; Liu et al., 2026; 2024). Section 3.2 describes RAVEL's estimator.

Let c denote a region and $\mathcal{R}_c$ its replicas. The feasible-region set is

$$\mathcal{F}_i(t) = \{c \mid \exists r \in \mathcal{R}_c : \Phi_{i,r}(t) = 1\}. \tag{1}$$

Arrivals, completions, accounting updates, and elapsed time can change $\mathcal{F}_i(t)$ by altering prospective contention or consuming SLO slack. Capacity that is substitutable for one request may therefore be indispensable to another, inducing the request-dependent opportunity cost introduced in Section 1.

2.2 Separating Demand Visibility from Placement Finality

The timing of t_i^{vis} and t_i^{final} distinguishes three placement lifecycles.

One-shot placement. A one-shot lifecycle couples visibility and finality at arrival: $t_i^{\text{vis}} = t_i^{\text{final}} = a_i$. The request immediately contributes to capacity planning, but its destination cannot be revised as later demand or serving-state information becomes available.

Fixed-delay diagnostic. Our LateBind-τ diagnostic shifts the same coupled event by a deferral window τ: $t_i^{\text{vis}} = t_i^{\text{final}} = a_i + \tau$. The destination remains uncommitted before $a_i + \tau$, but the request's own prospective demand is also absent from the planning state during that interval. This diagnostic therefore preserves destination optionality while withholding already-arrived work from intervening capacity decisions. As Figure 1 illustrates, simply shifting the coupled event later does not resolve this tension.

RAVEL. RAVEL decouples the two events: $t_i^{\text{vis}} = a_i < t_i^{\text{final}}$, where successful engine handoff establishes finality. An arrived request therefore affects subsequent capacity decisions immediately while its own geographic destination remains revisable as additional causally available demand and serving state are observed.

This lifecycle preserves placement flexibility without hiding arrived demand, but realizing it safely requires four properties: request-specific feasibility to constrain valid destinations; exactly-once prospective visibility so that each Router-owned request appears at exactly one tentative destination; bounded observation time and replanning scope to limit waiting and computation; and ownership-defined finality to establish a failure-safe boundary after which ordinary geographic replanning no longer reopens placement. Section 3 realizes these requirements.

3 RAVEL: Immediate Visibility, Deferred Finality

3.1 Overview and Invariants

RAVEL keeps arrived demand visible and eligible requests revisable throughout their Router-owned lifetime, with a bounded per-pass replanning scope.

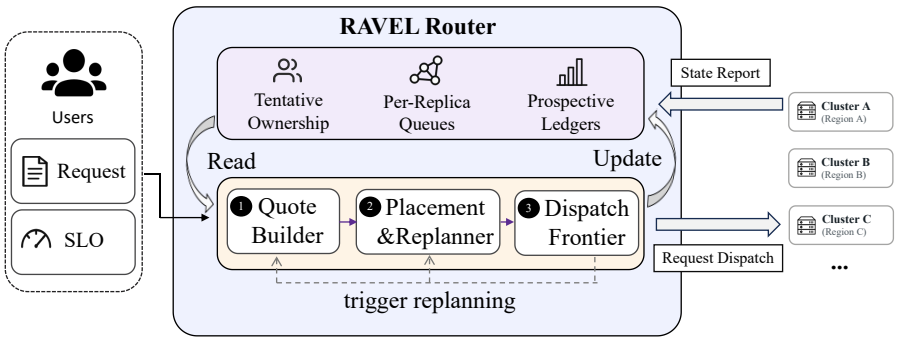


Figure 2: RAVEL routing architecture and ownership boundary. Router-owned requests are immediately visible to capacity planning, while eligible requests remain revisable until successful engine handoff transfers ownership and establishes geographic finality.

Figure 2 shows RAVEL's routing architecture and ownership boundary. On arrival, the Router selects a tentative destination and immediately accounts for the request's prospective demand there. We call this non-final, Router-owned assignment a virtual placement. While the request remains Router-owned, later arrivals and serving-state feedback may trigger replanning; successful engine handoff transfers ownership to the serving engine and establishes geographic finality.

Two invariants govern these transitions. Exactly-once prospective visibility requires each Router-owned request to have exactly one tentative destination and one prospective demand representation at every planning boundary. Ownership-defined finality makes geographic placement final only after successful engine handoff.

3.2 SLO-Aware Tentative Placement

RAVEL uses request- and replica-specific SLO quotes to identify viable destinations when placing or replanning a request (Section 2.1).

For request i and replica r, RAVEL constructs $Q_{i,r}(t) = (\Phi_{i,r}(t), \widehat{T}^{\mathrm{obj}}_{i,r}(t), \mathbf{z}_{i,r}(t))$, where $\Phi_{i,r}(t)$ indicates predicted SLO feasibility, $\widehat{T}^{\mathrm{obj}}_{i,r}(t)$ is the request-class-specific objective used to compare destinations, and $\mathbf{z}_{i,r}(t)$ contains secondary capacity-pressure and stability signals. Quotes combine calibrated Prefill/Decode profiles with WAN latency, observed engine state, and Router-owned prospective demand, using only information causally available at time t. For completion-oriented requests, the output-demand risk estimate is initialized from a 256-token hint and updated causally from completed requests of the same class; the current request's realized output length is never exposed to the planner.

The same placement rule applies to new arrivals and Router-owned requests reconsidered during replanning. If at least one replica is predicted feasible, RAVEL chooses among feasible targets. Otherwise, it first prefers lower pending-capacity overflow where the corresponding guard applies, and then lower normalized predicted SLO violation. Remaining ties are resolved lexicographically using SLO safety, capacity pressure, efficiency, and stability, so lower-priority efficiency or movement preferences cannot override an avoidable predicted SLO miss. Appendix A.1 gives the calibrated Prefill and Decode profiles, queue-ahead model, and request-class-specific feasibility predicates; Appendix A.2 gives deterministic ranking details.

3.3 Immediate and Transferable Demand Accounting

Exactly-once prospective visibility requires tentative work to affect later placement decisions before its destination becomes final. RAVEL therefore associates every Router-owned request with one tentative destination and a set of replica-dependent prospective charges.

When request i is tentatively placed at replica r, its prospective demand immediately enters r's planning state. Prompt demand contributes to the queue-ahead Prefill work observed by later SLO quotes, including quotes constructed later in the same planning pass. RAVEL also maintains

lightweight prospective service-pressure signals that complement the replica's observed pending and running engine state.

Reassignment transfers this representation together with the virtual placement. If request i moves from replica r to replica r', then for any transferable prospective ledger Z, $Z_r \leftarrow Z_r - z_{i,r}$ and $Z_{r'} \leftarrow Z_{r'} + z_{i,r'}$, where the destination-dependent charge is recomputed at r'. This remove–recompute–install transition occurs within one serialized Router accounting step, so every planning boundary observes request i at exactly one tentative destination.

Visibility also remains continuous across the ownership boundary. If a handoff attempt fails before finality, RAVEL restores the Router-owned prospective representation. After successful handoff, observed engine state tracks execution, while any retained service-ledger charge remains at the final replica until request termination. Appendix A.2 gives the exact charge definitions and accounting rule.

3.4 Slack-Bounded Observation and Pre-Handoff Replanning

A request's tentative destination can be revised while it remains Router-owned and eligible for replanning. New arrivals and serving-state updates may change which region an earlier request should use. Moving an earlier request to another feasible region can release capacity for a later request with fewer feasible destinations, as in the example in Section 1.

Slack-bounded active observation (Hold). RAVEL defers dispatch to observe new arrivals and serving-state updates. For an eligible non-LATENCY request, the observation interval is $h_i = \min\{\rho_i^{\min}, \sigma_i^{\text{arr}}\}$, where $\rho_i^{\min}$ is the smallest positive RTT from its origin to a remote serving region and σ_i^{arr} is its predicted arrival-time SLO slack. Eligibility requires at least two predicted-feasible regions and a recent interarrival gap no greater than the largest remote RTT; otherwise $h_i = 0$. LATENCY requests are never deliberately held.

A single revisable lifecycle. Prospective accounting begins at arrival and remains attached to the request throughout its Router-owned lifetime. At $a_i + h_i$, the request becomes dispatch-eligible; handoff proceeds as soon as capacity admission permits. Capacity admission requires the request's prompt-token charge to fit within a replica-specific dispatch allowance computed from engine-pending work, the batching budget, inter-cluster RTT, and calibrated Prefill cost. Active observation and subsequent waiting share the same replanning rules: an eligible request can be redirected while it awaits hold expiry, capacity admission, or completion of handoff. Becoming dispatch-eligible does not fix the destination: successful ownership transfer to the engine does.

Bounded replanning scope. RAVEL bounds deliberate observation time by h_i and per-pass replanning scope by K. Planning is event-driven, serialized, and coalesced. Each pass selects $K_{\text{eff}} \leq K$ movable Router-owned requests in deterministic deadline-first order, with a configured bound $K = 64$. Requests outside the cohort and current engine state remain fixed context. Before re-evaluating the cohort, RAVEL removes its old transferable representations from the pass-local planning state and then rebuilds the cohort sequentially. Each accepted virtual placement is installed before the next request is quoted, so later decisions in the same pass observe the prospective demand induced by earlier decisions.

With $R = |\mathcal{R}|$ candidate replicas, one pass performs $O(K_{\text{eff}} R)$ request–replica comparisons. Appendix A.3 gives the exact eligibility and cohort rules, while Appendix A.2 gives the deterministic placement key.

3.5 Ownership-Defined Finality

Successful Router-to-engine handoff establishes geographic finality. RAVEL initiates handoff as soon as dispatch eligibility and capacity admission permit. Until handoff succeeds, an eligible Router-owned request can still be redirected. Handoff comprises registration at the selected replica and creation and tracking of the corresponding engine-facing posting task.

If handoff does not complete, the request remains Router-owned and its prospective state is restored for subsequent planning. After successful handoff, queueing and chunked-prefill scheduling proceed

inside the serving engine with geographic placement already final. Attempt-indexed transitions ensure that asynchronous callbacks are applied only to the current ownership state. Appendix A.3 gives the exact ownership transitions.

3.6 Implementation Summary

RAVEL is implemented as an asynchronous Python routing layer in front of geo-distributed vLLM (Kwon et al., 2023) replicas without modifying the vLLM source tree. Router-side ownership, prospective accounting, replanning, and dispatch transitions share a serialized asynchronous control path, giving each planning pass a consistent Router-owned state while replica execution remains asynchronous. Replica-specific SLO quotes use deployment-calibrated service profiles.

4 Evaluation

We evaluate RAVEL around three questions. **Q1:** Does RAVEL improve request-level SLO attainment as contention increases? **Q2:** Do prospective accounting, pre-handoff redirection, and active observation each contribute to the gains? **Q3:** Are the gains stable across repeated executions, and are RAVEL's online feasibility signals and control-plane overhead practical at the evaluated deployment scale?

For each model, the load sweep uses one 500-request replay per policy–workload–load configuration. A separate five-run Qwen3-1.7B campaign at $16\times$ load measures run-to-run variation under paired repeated executions. Mechanism and prediction-quality analyses also use Qwen3-1.7B.

4.1 Experimental Methodology

Testbed. We evaluate all policies on a physical-WAN deployment with five serving replicas distributed across three clusters in a $2/2/1$ configuration. All replay clients originate in Region A. In the primary evaluation, each replica serves Qwen3-1.7B (Yang et al., 2025) on one NVIDIA RTX 4090 using vLLM 0.8.5.post1's V0 scheduler with bfloat16 weights. The Qwen3-8B extension retains the deployment, replay traces, and SLO definitions. Measured directed WAN RTTs between clusters range from 10.1 to 37.9 ms. Within each model-specific comparison, all policies use the same replicas, physical-WAN paths, base vLLM configuration, request traces, and per-request SLO metadata. Appendices B.1 and B.2 detail the engine, calibration, network, and replay settings.

Workloads and SLOs. We follow JITServe's workload and request-level SLO construction (Zhang et al., 2026), retaining request classes and per-request SLO annotations throughout replay. LMSYS represents heterogeneous conversational traffic (Zheng et al., 2024a). Burst retains the LMSYS requests and SLO annotations but uses a bursty arrival schedule, stressing placement under rapidly changing prospective queue state. DeepResearch represents multi-stage, completion-oriented traffic with heterogeneous request sizes and completion budgets. For each workload, we compress the replay timeline to increase offered load while preserving request order. LATENCY requests must satisfy their TTFT bound and every applicable time between tokens (TBT) bound; THROUGHPUT and COLLECTIVE requests must meet their request-specific end-to-end completion budgets. Appendix B.2 reports workload composition, absolute replay scale, request-size statistics, and SLO-budget summaries.

Baselines and metrics. We compare RAVEL against RoundRobin (RR), LeastLoad (LL), Random Dispatch (RD), DualMap, and SkyWalker. For each model–workload–load cell, "best baseline" denotes the non-RAVEL policy with the highest request-level SLO attainment in that cell. At Qwen3-1.7B and 16× load, DualMap is the best baseline for all three workloads, so we fix it as the comparator across all five paired repetitions of each workload. Appendix B.3 documents the exact baseline components; Appendix B.4 documents the comparison protocol used in the common serving harness.

Our primary metric is request-level SLO attainment. Secondary metrics include time to first token (TTFT), end-to-end latency (E2E), and time per output token (TPOT). TTFT and E2E are measured from arrival at the Router and therefore include Router-owned time, WAN delay, queueing, and engine execution.

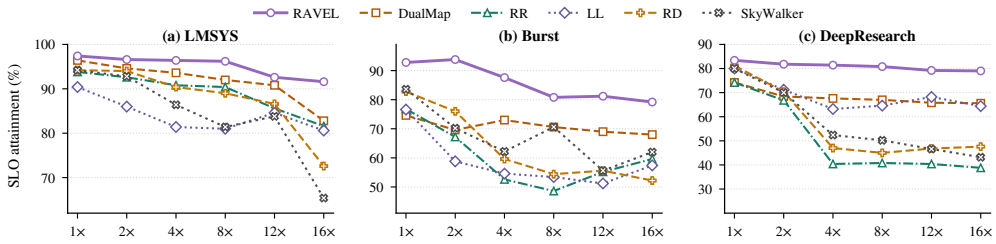


Figure 3: **Request-level SLO attainment versus offered load.** Results are shown for **(a)** LMSYS, **(b)** Burst, and **(c)** DeepResearch, using one 500-request replay per policy–workload–load configuration. RAVEL achieves the highest SLO attainment at every evaluated workload–load setting with Qwen3-1.7B.

4.2 End-to-End SLO Attainment

The repeated Qwen3-1.7B 16× runs provide our primary high-load comparison. Across five paired runs, RAVEL improves request-level SLO attainment over the fixed best-baseline comparator, DualMap, by an average of 4.08 percentage points on LMSYS (95% CI [0.49, 7.67]), 7.48 points on Burst ([3.78, 11.18]), and 17.40 points on DeepResearch ([10.55, 24.25]). All 15 paired run-level differences are positive (Appendix C.2).

Figure 3 shows SLO attainment across the full offered-load sweep. Across all three workloads, RAVEL maintains higher request-level SLO attainment as contention increases. The margin varies with workload and load, reflecting differences in contention and available placement opportunities.

In the Qwen3-1.7B single-run 16× cells of the load sweep, RAVEL exceeds the best baseline by 8.8 percentage points on LMSYS, 11.2 points on Burst, and 13.4 points on DeepResearch. At the same operating point, RAVEL reduces P95 TTFT by 49.1–80.5% and P95 end-to-end latency by 15.5–56.8% relative to the corresponding best baseline.

The Qwen3-8B extension also places RAVEL first in SLO attainment across all 18 workload–load settings. At 16×, RAVEL attains 85.4%, 73.8%, and 65.2% on LMSYS, Burst, and DeepResearch, exceeding the best competing policy by 8.2, 9.2, and 3.0 percentage points, respectively. Appendix C.1 reports the complete results for both models.

4.3 Why Does RAVEL Work?

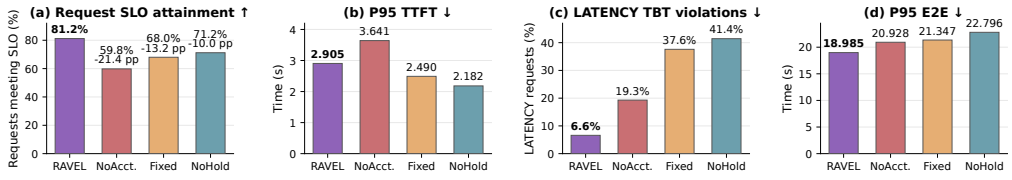


Figure 4: **Core lifecycle ablation on Burst at 12× load.** NoAccounting removes immediate demand visibility while preserving pre-handoff revisability. FixedPlacement disables reassignment while retaining accounting and the active observation window. NoHold sets only that observation window to zero.

We evaluate the three ablations defined in Figure 4 across all three workloads at 12× and 16× with Qwen3-1.7B. On Burst at 12×, RAVEL attains 81.2% request-level SLO attainment, versus 59.8% for NoAccounting, 68.0% for FixedPlacement, and 71.2% for NoHold, corresponding to drops of 21.4, 13.2, and 10.0 percentage points. Across all six workload–load cells, RAVEL exceeds NoHold by 2.4–10.0 percentage points, isolating the incremental benefit of deliberate observation beyond replanning during capacity-induced waiting (Appendix C.4).

The NoHold comparison also exposes the time–information tradeoff. On Burst at 12×, RAVEL has higher P95 TTFT than NoHold (2.905 versus 2.182 s), but fewer LATENCY-request TBT violations (6.6% versus 41.4%) and lower P95 E2E latency (18.985 versus 22.796 s). FixedPlacement exhibits

the same pattern: its P95 TTFT is lower (2.490 s), while its TBT violation incidence (37.6%) and P95 E2E latency (21.347 s) are both higher than RAVEL's. Across the six cells, RAVEL reduces P95 E2E relative to NoHold in every case, while its P95 TTFT is higher in three. The benefit therefore depends on the joint SLO objective rather than first-token latency alone.

Additional analyses probe this mechanism interpretation. In the same Burst 12× trace, FixedPlacement misses the request-level SLO for 160 of 500 requests; 66 (41.25%) satisfy our model-consistent pre-handoff reassignment criterion. Each such request is predicted infeasible under the frozen incumbent allocation, becomes predicted feasible after reassignment within the same planning episode, subsequently satisfies its realized SLO, and is accompanied by moving at least one earlier-arrived blocker away from a relevant region (Appendices B.5 and C.4.1). Separately, scaling inherited per-request SLO budgets by 0.8, 1.0, and 1.2 leaves RAVEL ahead of FixedPlacement by 5.0–13.2 percentage points and NoAccounting by 21.4–27.0 points in request-level SLO attainment (Appendix C.5). The mechanism gains therefore persist under both tighter and looser budgets, while retaining the first-token latency tradeoff.

4.4 Stability, Prediction Quality, and Control-Plane Cost

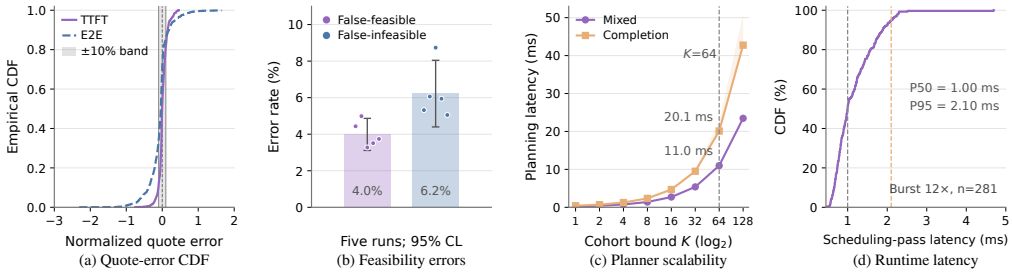


Figure 5: **Prediction quality and control-plane cost.** **(a)** Empirical CDFs of SLO-budget-normalized TTFT and E2E prediction errors; shading marks ±0.1 of the applicable SLO budget. **(b)** Selected-destination feasibility classification errors. **(c)** Median planner latency for mixed-request and workflow-completion scenarios as the cohort bound increases; shading shows P5–P95 across 50 repetitions per setting. **(d)** CDF of scheduling-pass latency on Burst at 12× load (281 passes).

Run-to-run stability. All 15 paired run-level SLO-attainment differences at 16× are positive (Appendix C.2). For each workload, the 95% Student-t confidence interval for the mean paired gain over DualMap lies entirely above zero.

Quote validity. Figure 5(a) shows the empirical CDFs of $(\widehat{T} - T)/D$, the prediction error normalized by the applicable SLO budget. Negative values indicate underprediction. TTFT errors are more tightly concentrated near zero than E2E errors, whose broader tails include both underprediction and overprediction (Appendix B.4). Figure 5(b) reports feasibility classification at selected destinations. In five DeepResearch 16× runs (2,500 final-handoff records), 80 of 2,004 predicted-feasible records miss their SLOs (five-run mean rate 4.0%), while 31 of 496 predicted-infeasible records meet them (five-run mean rate 6.2%). Both rates use the predicted class as the denominator; outcomes at unselected replicas are unavailable.

Control-plane cost. Across 281 actual planning passes on Burst at 12×, scheduling latency has a median of 0.998 ms and P95 of 2.101 ms (Figure 5(d)). Actual cohorts average 3.65 requests (P95 7, maximum 9), well below the configured bound $K = 64$.

Figure 5(c) separately measures planner latency as the cohort bound increases, showing increasing cost as more requests are processed per pass. At $K = 64$, median planning latency is 10.99 ms for mixed-request cohorts and 20.10 ms for workflow-completion cohorts. Appendix C.6 processes each specified cohort in full and increases the synthetic replica-state count to 160: planning latency grows approximately with $K_{\text{eff}}R$, and thus with R at fixed K_{eff}.

5 Related Work

Distributed and geo-aware LLM routing. Preble and DualMap balance reusable KV state against computation or queue load (Srivatsa et al., 2025; Yuan et al., 2026); SkyWalker, GORGO, and Solyx incorporate WAN and serving conditions into geographic placement (Xia et al., 2026; Toniolo et al., 2026; Bernhard & Katla, 2026). SGLang, CachedAttention, and Mooncake exploit KV reuse at the runtime, conversation, and distributed-serving levels (Zheng et al., 2024b; Gao et al., 2024; Qin et al., 2025). RAVEL addresses when geographic placement becomes final, using request-specific SLO quotes and prospective load signals to guide tentative decisions.

Serving capacity and scheduling. Orca and Sarathi-Serve study iteration-level batching and chunked-prefill scheduling (Yu et al., 2022; Agrawal et al., 2024b); Splitwise separates prefill and decoding resources (Patel et al., 2024). AlpaServe studies statistical multiplexing through model parallelism (Li et al., 2023), while Libra partitions and schedules unbalanced, dynamic workloads (Ruan et al., 2026). Other objectives include token-based fairness (Sheng et al., 2024), output-length ranking for scheduling (Fu et al., 2024b), and SLO-customized speculative decoding (Li et al., 2026). JITServe, QUARTZ, and Andes address request-level service objectives (Zhang et al., 2026; Liu et al., 2026; 2024). VIDUR uses profiling and predictive modeling to estimate inference performance (Agrawal et al., 2024a). These studies inform capacity and scheduling decisions; RAVEL focuses on prospective demand visibility and pre-handoff revisability.

Application-level coordination. Parrot and Agentix study serving at the level of LLM applications and agent programs (Lin et al., 2024; Luo et al., 2026). RAVEL operates at the request-routing boundary, using request-visible objectives rather than optimizing program execution.

Reassignment and migration. Serving systems use rescheduling, preemption, state movement, and migration (Sun et al., 2024; Miao et al., 2024; Fu et al., 2024a; Strati et al., 2024; Wu et al., 2024b). LoongServe dynamically adjusts sequence parallelism for long-context serving (Wu et al., 2024a). DualMap can also rebalance queued assignments (Yuan et al., 2026). RAVEL preserves geographic flexibility while a request remains Router-owned, before engine-managed execution state materializes; successful engine handoff fixes its geographic destination. This ownership boundary distinguishes RAVEL's pre-handoff replanning from runtime state migration.

Delayed and online scheduling. Delay scheduling postpones assignment (Zaharia et al., 2010); deadline-aware and online schedulers address finite slack, incomplete information, and scalable decisions (Liu & Layland, 1973; Ousterhout et al., 2013; Li et al., 2024; Schwarzkopf et al., 2013; Gog et al., 2016). RAVEL examines the visibility of demand during deferral. In our LateBind-τ diagnostic, waiting demand enters capacity planning only at final assignment. RAVEL instead counts demand immediately while keeping placement revisable.

6 Limitations

Our physical evaluation covers five replicas across three clusters with Qwen3-1.7B and Qwen3-8B on RTX 4090 GPUs; the supplementary microbenchmark measures Router-side planning cost over up to 160 synthetic replica states. End-to-end serving at larger physical scales and with model–hardware configurations beyond those evaluated here remains to be evaluated. RAVEL relies on online SLO quotes and request-visible output-demand estimates, whose prediction errors can affect placement quality under conditions beyond those evaluated here.

7 Conclusion

RAVEL separates immediate demand visibility from geographic placement finality. Its fixed-delay diagnostic identifies hidden demand as the main source of recoverable loss in that setting. Transferable prospective accounting, slack-bounded active observation, and bounded pre-handoff replanning preserve flexibility until successful engine handoff. On the evaluated physical-WAN deployment, this lifecycle improves request-level SLO attainment across conversational, bursty, and completion-oriented workloads with Qwen3-1.7B and Qwen3-8B, with low online planning overhead.

REFERENCES

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav S. Gulavani, Ramachandran Ramjee, and Alexey Tumanov. VIDUR: A large-scale simulation framework for LLM inference. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 351–366, 2024a. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/b74a8de47d2b3c928360e0a011f48351-Abstract-Conference.html.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 117–134, Santa Clara, CA, July 2024b. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/agrawal.

Aleks Bernhard and Nithin Katla. Solyx AI grid: Hardware-telemetry-aware routing across geographically distributed GPU clusters, 2026. URL https://arxiv.org/abs/2606.15050. arXiv:2606.15050v1.

Jeffrey Dean and Luiz André Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013. doi: 10.1145/2408776.2408794. URL https://doi.org/10.1145/2408776.2408794.

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. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 135–153, Santa Clara, CA, July 2024a. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/fu.

Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. Efficient LLM scheduling by learning to rank. In *Advances in Neural Information Processing Systems 37*, pp. 59006–59029. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024b. doi: 10.52202/079017-1882. URL https://doi.org/10.52202/079017-1882.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-Efficient large language model serving for multi-turn conversations with CachedAttention. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 111–126, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-41-0. URL https://www.usenix.org/conference/atc24/presentation/gao-bin-cost.

Ionel Gog, Malte Schwarzkopf, Adam Gleave, Robert N. M. Watson, and Steven Hand. Firmament: Fast, centralized cluster scheduling at scale. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 99–115, Savannah, GA, November 2016. USENIX Association. ISBN 978-1-931971-33-1. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/gog.

Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven WAN. *ACM SIGCOMM Computer Communication Review*, 43(4):15–26, 2013. doi: 10.1145/2534169.2486012. URL https://doi.org/10.1145/2534169.2486012.

Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013. doi: 10.1145/2534169.2486019. URL https://doi.org/10.1145/2534169.2486019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626. ACM, October 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Wenxin Li, Xin He, Yuan Liu, Keqiu Li, Kai Chen, Zhao Ge, Zewei Guan, Heng Qi, Song Zhang, and Guyue Liu. Flow scheduling with imprecise knowledge. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 95–111, Santa Clara, CA, April 2024. USENIX Association. ISBN 978-1-939133-39-7. URL https://www.usenix.org/conference/nsdi24/presentation/li-wenxin.

Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. AlpaServe: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pp. 663–679, Boston, MA, July 2023. USENIX Association. ISBN 978-1-939133-34-2. URL https://www.usenix.org/conference/osdi23/presentation/li-zhouhan.

Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, Yi-Hsiang Lai, Xinhao Cheng, Xupeng Miao, and Zhihao Jia. AdaServe: Accelerating multi-SLO LLM serving with SLO-customized speculative decoding. In *Proceedings of the 21st European Conference on Computer Systems*, EuroSys '26, pp. 36–54. ACM, April 2026. doi: 10.1145/3767295.3769315. URL https://doi.org/10.1145/3767295.3769315.

Gangmuk Lim, Wanyu Zhao, Brighten Godfrey, Jiaxin Shan, Le Xu, and Liguang Xie. Lodestar: An online-learning LLM inference router, 2026. URL https://arxiv.org/abs/2606.00946.

Chaofan Lin, Zhenhua Han, Chengruidong Zhang, Yuqing Yang, Fan Yang, Chen Chen, and Lili Qiu. Parrot: Efficient serving of LLM-based applications with semantic variable. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 929–945, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/lin-chaofan.

C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973. doi: 10.1145/321738.321743. URL https://doi.org/10.1145/321738.321743.

Jiachen Liu, Jae-Won Chung, Zhiyu Wu, Fan Lai, Myungjin Lee, and Mosharaf Chowdhury. Andes: Defining and enhancing quality-of-experience in LLM-based text streaming services, 2024. URL https://arxiv.org/abs/2404.16283.

Zhipeng Liu, Yifan Zheng, Fanqi Kong, and Ziming Zhao. QUARTZ: Quantile-aware routing and queueing for TTFT SLOs in LLM serving. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37886–37896, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1888. URL https://aclanthology.org/2026.findings-acl.1888/.

Michael Luo, Xiaoxiang Shi, Colin Cai, Tianjun Zhang, Justin Wong, Yichuan Wang, Chi Wang, Yanping Huang, Zhifeng Chen, Joseph E. Gonzalez, and Ion Stoica. Agentix: An efficient serving engine for LLM agents as general programs. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pp. 2443–2459, Renton, WA, May 2026. USENIX Association. ISBN 978-1-939133-54-0. URL https://www.usenix.org/conference/nsdi26/presentation/luo.

Yixuan Mei, Yonghao Zhuang, Xupeng Miao, Juncheng Yang, Zhihao Jia, and Rashmi Vinayak. Helix: Serving large language models over heterogeneous GPUs and network via max-flow. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 586–602. ACM, 2025. doi: 10.1145/3669940.3707215. URL https://doi.org/10.1145/3669940.3707215.

Xupeng Miao, Chunan Shi, Jiangfei Duan, Xiaoli Xi, Dahua Lin, Bin Cui, and Zhihao Jia. SpotServe: Serving generative large language models on preemptible instances. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1112–1127. ACM, 2024. doi: 10.1145/3620665.3640411. URL https://doi.org/10.1145/3620665.3640411.

Kay Ousterhout, Patrick Wendell, Matei Zaharia, and Ion Stoica. Sparrow: Distributed, low latency scheduling. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pp. 69–84. ACM, November 2013. doi: 10.1145/2517349.2522716. URL https://doi.org/10.1145/2517349.2522716.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132. IEEE, 2024. doi: 10.1109/isca59077.2024.00019. URL https://doi.org/10.1109/isca59077.2024.00019.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation — a KVCache-centric architecture for serving LLM chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*, pp. 155–170, Santa Clara, CA, February 2025. USENIX Association. ISBN 978-1-939133-45-8. URL https://www.usenix.org/conference/fast25/presentation/qin.

Chaoyi Ruan, Yinhe Chen, Dongqi Tian, Yandong Shi, Yongji Wu, Jialin Li, and Cheng Li. Libra: Flexible request partitioning and scheduling for serving unbalanced and dynamic LLM workloads. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pp. 1243–1258, Renton, WA, May 2026. USENIX Association. ISBN 978-1-939133-54-0. URL https://www.usenix.org/conference/nsdi26/presentation/ruan-libra.

Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. Omega: Flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pp. 351–364. ACM, 2013. doi: 10.1145/2465351.2465386. URL https://doi.org/10.1145/2465351.2465386.

Ying Sheng, Shiyi Cao, Dacheng Li, Banghua Zhu, Zhuohan Li, Danyang Zhuo, Joseph E. Gonzalez, and Ion Stoica. Fairness in serving large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 965–988, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/sheng.

Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. Preble: Efficient distributed prompt scheduling for LLM serving. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5bc342f48de8264779952fac378f96dc-Abstract-Conference.html.

Foteini Strati, Sara McAllister, Amar Phanishayee, Jakub Tarnawski, and Ana Klimovic. DéjàVu: KV-cache streaming for fast, fault-tolerant generative LLM serving. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 46745–46771. PMLR, July 2024. URL https://proceedings.mlr.press/v235/strati24a.html.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 173–191, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/sun-biao.

Alessio Ricci Toniolo, Rome Thorstenson, and Abinaya Dinesh. GORGO: Online tuning for cross-region network-aware LLM serving, 2026. URL https://arxiv.org/abs/2602.11688. arXiv:2602.11688v3.

Zhibin Wang, Shipeng Li, Yuhang Zhou, Xue Li, Zhonghui Zhang, Nguyen Cam-Tu, Rong Gu, Chen Tian, Guihai Chen, and Sheng Zhong. Revisiting service level objectives and system level metrics in large language model serving, 2025. URL https://arxiv.org/abs/2410.14257. arXiv:2410.14257v2.

Bingyang Wu, Shengyu Liu, Yinmin Zhong, Peng Sun, Xuanzhe Liu, and Xin Jin. LoongServe: Efficiently serving long-context large language models with elastic sequence parallelism. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pp. 640–654. ACM, 2024a. doi: 10.1145/3694715.3695948. URL https://doi.org/10.1145/3694715.3695948.

Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 911–927, Santa Clara, CA, July 2024b. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/wu-bingyang.

Tian Xia, Ziming Mao, Jamison Kerney, Ethan J. Jackson, Zhifei Li, Jiarong Xing, Scott Shenker, and Ion Stoica. SkyWalker: A locality-aware cross-region load balancer for LLM inference. In *Proceedings of the 21st European Conference on Computer Systems*, EuroSys '26, pp. 1349–1364. ACM, April 2026. doi: 10.1145/3767295.3769353. URL https://doi.org/10.1145/3767295.3769353.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 521–538, Carlsbad, CA, July 2022. USENIX Association. ISBN 978-1-939133-28-1. URL https://www.usenix.org/conference/osdi22/presentation/yu.

Ying Yuan, Pengfei Zuo, Bo Wang, Zhangyu Chen, Zhipeng Tan, and Zhou Yu. DualMap: Enabling both cache affinity and load balancing for distributed LLM serving. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/475f7a3a807cee549684e367d1fa13a0-Abstract-Conference.html.

Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pp. 265–278. ACM, April 2010. doi: 10.1145/1755913.1755940. URL https://doi.org/10.1145/1755913.1755940.

Wei Zhang, Zhiyu Wu, Yi Mu, Rui Ning, Banruo Liu, Nikhil Sarda, Myungjin Lee, and Fan Lai. JITServe: SLO-aware LLM serving with imprecise request information. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pp. 825–848, Renton, WA, May 2026. USENIX Association. ISBN 978-1-939133-54-0. URL https://www.usenix.org/conference/nsdi26/presentation/zhang-wei.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. LMSYS-Chat-1M: A large-scale real-world LLM conversation dataset. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5f9bfdfe3685e4ccdbc0e7fb29cccf2a-Abstract-Conference.html.

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. In *Advances in Neural Information*

Processing Systems 37, pp. 62557–62583. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024b. doi: 10.52202/079017-2000. URL https://doi.org/10.52202/079017-2000.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 193–210, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

AI USE STATEMENT

In preparing this paper, we used a large language model to polish the writing and identify relevant literature. Specifically, the LLM was employed to improve the grammar and readability of the text. AI tools were also used to assist with limited coding tasks and file organization. All AI-assisted outputs, including code and document-related materials, were manually reviewed and verified by the authors before use. No part of the research ideation, experimental design, data analysis, or results interpretation involved the use of LLMs. The authors take full responsibility for all content of the paper.

A RAVEL ALGORITHMIC DETAILS

This appendix provides the exact prediction, placement, accounting, and pre-handoff control rules underlying Section 3. All quantities used by the Router are computed from causally available request metadata, network measurements, calibrated service profiles, observed engine state, and Router-owned prospective state. Future arrivals and realized output lengths are unavailable to the planner.

A.1 SLO QUOTES AND FEASIBILITY

For request i and candidate replica r, RAVEL constructs $Q_{i,r}(t) = (\Phi_{i,r}(t), \widehat{T}^{\mathrm{obj}}_{i,r}(t), \mathbf{z}_{i,r}(t))$, where $\Phi_{i,r}(t)$ is predicted SLO feasibility, $\widehat{T}^{\mathrm{obj}}_{i,r}(t)$ is the request-class-specific placement objective, and $\mathbf{z}_{i,r}(t)$ contains secondary capacity-pressure and stability signals.

Let $c(r)$ denote replica r's cluster and n_r its observed engine-running sequence count at decision time. Let p_i be request i's full prompt-token count (clamped to at least one for accounting), and o_i its origin region; $\rho_{o_i,c(r)}$ is the RTT from that origin to replica r's cluster. RAVEL uses a calibrated Prefill profile $P_{c,n}(p) = b_{c,n} + a_{c,n}p$ and calibrated post-first-token Decode time $d_c(n)$. Profile values are linearly interpolated between measured concurrency levels and clamped to the nearest endpoint outside the calibrated range. With observed engine-pending prompt work E_r, Router-owned queue-ahead prompt work $B_{i,r}$, engine-pending request count N^{eng}_r, and Router-owned queue-ahead count $N^q_{i,r}$, the predicted Prefill service time and TTFT are

$$\widehat{S}^{\mathrm{pre}}_{i,r} = a_{c(r),n_r}(E_r + B_{i,r} + p_i) + b_{c(r),n_r}(N^{\mathrm{eng}}_r + N^q_{i,r} + 1), \tag{2}$$

$$\widehat{\mathrm{TTFT}}_{i,r}(t) = (t - a_i) + \rho_{o_i,c(r)} + \widehat{S}^{\mathrm{pre}}_{i,r}. \tag{3}$$

Queue-ahead terms include virtual placements accepted earlier in the same planning pass.

Let $\mathcal{O}_r$ denote the requests already represented at replica r in Router-owned, engine-pending, or engine-running state. Let N^q_r and N^{eng}_r denote the Router-owned tentative and engine-pending request counts, respectively; n_r is the engine-running sequence count defined above. Prospective Decode occupancy is

$$m_{i,r} = \max(1,\, N^q_r + N^{\mathrm{eng}}_r + n_r + \mathbf{1}[i \notin \mathcal{O}_r])\,. \tag{4}$$

The indicator adds request i only if it is not already represented at r; the lower bound of one keeps the Decode profile input positive. A LATENCY request is predicted feasible when $\widehat{\mathrm{TTFT}}_{i,r}(t) \le D^{\mathrm{TTFT}}_i$ and $d_{c(r)}(m_{i,r}) \le D^{\mathrm{TBT}}_i$.

For completion-oriented requests, RAVEL maintains a request-visible expected output-demand estimate L^{exp}_i and a conservative risk estimate L^{risk}_i. In our evaluation, L^{exp}_i is initialized from a fixed request-visible output-token hint of 256 tokens. RAVEL derives L^{risk}_i causally from completed requests of the same request class: after at least 20 observations, it takes the larger of the configured hint and the empirical 90th-percentile output length over a rolling window of up to 256 completed requests. Before sufficient history is available, the configured hint is used unchanged. The realized output length of the current request is never exposed to the planner. For $x \in \{\mathrm{exp}, \mathrm{risk}\}$,

$$\widehat{T}^x_{i,r}(t) = \widehat{\mathrm{TTFT}}_{i,r}(t) + \max(0, L^x_i - 1) d_{c(r)}(m_{i,r}). \tag{5}$$

Completion feasibility is $\Phi^{\mathrm{comp}}_{i,r}(t) = \mathbf{1}[\widehat{T}^{\mathrm{risk}}_{i,r}(t) \le D^{\mathrm{comp}}_i]$. The conservative estimate determines feasibility, while the expected estimate is used only as a secondary placement signal.

A.2 Prospective Accounting and Placement

Router-owned work contributes to planning state before geographic placement is final. For LATENCY request i, define $u_{i,r} = d_{c(r)}(1)/D_i^{\mathrm{TBT}}$ and $A_r = \sum_{j\in\mathcal{L}_r} u_{j,r}$, where $\mathcal{L}_r$ is the set of Router-owned LATENCY requests virtually placed at r. For profiled service demand, define $s_{i,r}^{\mathrm{own}} = \widehat{S}_{i,r}^{\mathrm{own,pre}} + \widehat{S}_{i,r}^{\mathrm{own,dec}}$ and $L_r = \sum_{j\in\mathcal{S}_r} s_{j,r}^{\mathrm{own}}$. Here $\mathcal{S}_r$ contains requests with an active service-ledger entry assigned to r in the current planning state, across request classes. Entries are installed at tentative placement and transferred on reassignment; when retained after handoff, they remain at the final replica until the request's terminal callback releases them. The own-service terms are $\widehat{S}_{i,r}^{\mathrm{own,pre}} = a_{c(r),n_r}p_i + b_{c(r),n_r}$ and, for completion-oriented requests, $\widehat{S}_{i,r}^{\mathrm{own,dec}} = \max(0, L_i^{\mathrm{exp}} - 1)d_{c(r)}(m_{i,r})$; the latter is zero for LATENCY requests. These terms exclude elapsed time, WAN delay, and queue-ahead work.

For any transferable prospective ledger Z, reassignment from r to r' performs

$$Z_r \leftarrow Z_r - z_{i,r}, \qquad Z_{r'} \leftarrow Z_{r'} + z_{i,r'}, \tag{6}$$

where the destination-dependent charge is recomputed at r'. The remove–recompute–install operation executes within one serialized Router transition, so every externally visible planning boundary represents a Router-owned request at exactly one virtual destination.

Initial placement and replanning use the same lexicographic key,

$$\begin{aligned} K_{i,r} = (&\neg\Phi_{i,r},\ O_{i,r},\ \ell_{i,r}^{\mathrm{obj}},\ \ell_{i,r}^{\mathrm{TTFT}},\\ &A_r + u_{i,r},\ L_r + s_{i,r}^{\mathrm{own}},\ \widehat{T}_{i,r}^{\mathrm{obj}}/D_i,\ M_{i,r},\ \mathrm{moved}_{i,r},\ id_r), \end{aligned} \tag{7}$$

with lower values preferred and inapplicable terms taking their neutral value. The key first prefers predicted-feasible targets, then lower applicable pending-capacity overflow and normalized SLO violation; prospective capacity-pressure terms follow, with efficiency and movement stability used only at lower priority.

Here D_i is the applicable request-class objective budget and $O_{i,r}$ is the pending-capacity overflow term. The normalized lateness terms are $\ell_{i,r}^{\mathrm{obj}} = \max(0, \widehat{T}_{i,r}^{\mathrm{obj}} - D_i)/\max(D_i, \varepsilon)$ and $\ell_{i,r}^{\mathrm{TTFT}} = \max(0, \widehat{\mathrm{TTFT}}_{i,r} - D_i^{\mathrm{TTFT}})/\max(D_i^{\mathrm{TTFT}}, \varepsilon)$. The marginal-service term is $M_{i,r} = ((W_r + s_{i,r}^{\mathrm{own}})^2 - W_r^2)/D_i^2$. To define W_r for a planning pass, let $\mathcal{F}_r$ contain fixed Router-owned requests at r outside the replanning cohort, and let $\mathcal{P}_r$ contain cohort requests already assigned to r earlier in the pass. Before evaluating i,

$$W_r = a_{c(r),n_r}\Big(E_r + \sum_{j\in\mathcal{F}_r} p_j\Big) + b_{c(r),n_r}\big(N_r^{\mathrm{eng}} + n_r + |\mathcal{F}_r|\big) + \sum_{j\in\mathcal{P}_r} s_{j,r}^{\mathrm{own}}.$$

Thus W_r combines the fixed state's Prefill service estimate with the own-service charges of earlier assignments; it is distinct from L_r. Let r_i^0 be request i's initial tentative replica. During replanning, $\mathrm{moved}_{i,r} = \mathbf{1}[r \neq r_i^0]$; before an initial placement exists, this term is zero for every candidate. The replica identifier id_r provides the deterministic final tie-break. For the overflow term, let $q_{i,r}^{+}$ be the engine-pending count plus fixed Router-owned requests, earlier cohort assignments to r, and request i itself. Then $O_{i,r} = \max(0, q_{i,r}^{+} - C)$, with pending-sequence limit $C = 64$; for a non-LATENCY request with $t < a_i + h_i$, $O_{i,r} = 0$, where h_i is the dispatch-eligibility offset defined in Appendix A.3. The sequence limit C and per-pass cohort bound K constrain different parts of the control path.

A.3 Exact Pre-Handoff Control Rules

Observation hold and dispatch eligibility. Let δ_i be the most recently observed interarrival gap and $\mathcal{D}_i$ the positive RTTs from request i's origin to remote serving clusters. If $\mathcal{D}_i \neq \emptyset$, define $\mathrm{Burst}_i = \mathbf{1}[\delta_i \leq \max \mathcal{D}_i]$ and $\rho_i^{\min} = \min \mathcal{D}_i$; otherwise $\mathrm{Burst}_i = 0$. The planning budget $\Delta_i = D_i$ is D_i^{TTFT} for LATENCY requests and D_i^{comp} for completion-oriented requests. Arrival-time slack is $\sigma_i^{\mathrm{arr}} = \max\{0, \Delta_i - \min_r \widehat{T}_{i,r}^{\mathrm{obj}}(a_i)\}$. The evaluated implementation sets the earliest dispatch-eligible time to $a_i + h_i$, where

$$h_i = \begin{cases} \min\{\rho_i^{\min}, \sigma_i^{\mathrm{arr}}\}, & \text{if } i \text{ is non-LATENCY, } \mathrm{Burst}_i = 1, \text{ and } |\mathcal{F}_i(a_i)| \geq 2,\\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

For LATENCY requests, $h_i = 0$. A positive hold delays dispatch eligibility within the predicted arrival-time slack; it does not create a separate ownership or replanning state. Before and after $a_i + h_i$, the request remains virtually placed, prospectively visible, and eligible for replanning until successful engine handoff. Dispatch also requires capacity admission below.

Replanning cohort. Movable Router-owned requests are ordered by $\kappa_i = (a_i + \Delta_i, a_i, id_i)$. Each pass processes $K_{\text{eff}} \leq K$ requests, where $K = 64$ is the configured cohort bound. Noncandidate Router-owned requests and current engine state remain fixed context. The cohort's old transferable charges are removed from the pass-local planning base before candidates are re-evaluated; candidates are then rebuilt sequentially, installing each accepted virtual placement before quoting the next request. For actual cohort size $N = K_{\text{eff}} \leq K$, R candidate replicas, and F fixed Router-owned requests used to construct queue-ahead prefixes, one pass costs $O(N \log N + F \log F + NR + F)$ and performs $O(K_{\text{eff}} R)$ request–replica comparisons.

Dispatch and finality. Let $H = \max_{c \neq c'} \rho_{c,c'}$ be the largest directed inter-cluster control RTT. Let B_{iter} be the engine's maximum batched-token budget per iteration (16,384 tokens in our evaluation). For replica r, the dispatch frontier uses $F_r = B_{\text{iter}} + \lceil H / a_{c(r),0} \rceil$ and $G_r = \max(0, F_r - E_r)$. A request's prompt charge is p_i tokens, with no estimated cache-hit deduction. A Router-owned request crosses the frontier only when $p_i \leq G_r$; otherwise it remains Router-owned, prospectively visible, and eligible for replanning.

A request becomes geographically final only after registration at its selected replica succeeds and the corresponding engine-facing posting task is successfully created and tracked. Until then, ownership remains with the Router; after successful handoff, ordinary geographic replanning no longer applies.

Lifecycle invariants. At every serialized Router planning boundary, each Router-owned request has exactly one virtual destination and one associated prospective demand representation. Geographic finality is established only by successful Router-to-engine ownership transfer. Serialized remove–recompute–install accounting maintains the former, while the handoff boundary maintains the latter.

B Experimental Details

This appendix summarizes the deployment, frozen workloads, baseline implementations, and experimental protocols used in Section 4.

B.1 Deployment, Models, and Calibration

Within each model-specific comparison, all policies share the model, replicas, physical WAN paths, engine configuration, request traces, and per-request SLO metadata. Table 1 summarizes the deployment and calibration settings. Replica-specific quotes use Prefill and Decode profiles calibrated separately from evaluation with token-controlled synthetic requests.

Table 1: **Evaluation and calibration configuration.**

Component	Configuration
Serving topology	5 replicas / 3 physical clusters $(2/2/1)$
Model / GPU	Qwen3-1.7B (primary); Qwen3-8B (extension); one NVIDIA RTX 4090 per replica
Serving engine	vLLM 0.8.5.post1 V0; bfloat16
Engine limits	16,384-token context; 64 sequences; 16,384 batched tokens
KV cache	16-token blocks; automatic prefix caching
Client origin	Region A
Directed RTT (A/B/C)	$\begin{pmatrix} 0 & 37.9 & 14.7 \\ 32.0 & 0 & 37.3 \\ 10.1 & 36.6 & 0 \end{pmatrix}$ ms
Replanning bound	$K = 64$
Output-demand estimate	256-token initial hint; causal same-class empirical Q90
Prefill calibration	$n = \{0, 8, 16, 32, 48, 63\}$, $p = \{256, 1024, 4096, 12000\}$
Decode calibration	$n = \{1, 9, 17, 33, 49, 64\}$; 5 repetitions per point

The primary evaluation uses Qwen3-1.7B. The Qwen3-8B extension retains the BF16 precision, vLLM version, topology, engine limits, first-500 traces, speedups, and SLO definitions. Experiment coverage is specified in Appendix B.4.

B.2 Workloads, Replay, and SLO Metrics

We evaluate the three frozen 500-request workloads characterized in Table 2. For each workload, offered-load factor $s \in \{1, 2, 4, 8, 12, 16\}$ compresses the same relative arrival timeline by $1/s$ without changing request order.

Table 2: **Frozen workload characteristics.** Token columns report median / P95; rate is the mean arrival rate on the $1\times$ timeline.

Workload	Input tokens median / P95	Output tokens median / P95	$1\times$ span (s)	$1\times$ rate (req/s)
LMSYS	56 / 550	126.5 / 898.95	200.27	2.49
Burst	56 / 550	126.5 / 898.95	38.97	12.81
DeepResearch	187 / 1476.8	280 / 1023.05	11.27	44.28

LMSYS uses the first 500 JITServe LMSYS requests in their original order and arrival timeline. Burst uses the same requests and SLOs with the first 500 arrival timestamps from the burst trace bundled with the JITServe artifact. DeepResearch uses the first 500 globally ordered JITServe workflow branches with frozen stage offsets. The token statistics in Table 2 characterize the original Qwen3-1.7B evaluation traces; they are not measurements of Qwen3-8B output lengths.

We follow JITServe's request-level SLO construction and use the base-budget configuration from `benchmark_e2e.sh` in its official artifact (Zhang et al., 2026). LMSYS and Burst contain 181 LATENCY, 301 THROUGHPUT, and 18 COLLECTIVE requests; DeepResearch contains 500 COLLECTIVE requests. The base budgets are 0.8 s for TTFT, 80 ms for TBT, and 8 s for completion, with a request-specific multiplier $m \in \{1, 2, 3, 4\}$. Thus, applicable TTFT, TBT, and completion budgets range over 0.8–3.2 s, 80–320 ms, and 8–32 s, respectively. A LATENCY request succeeds only if its TTFT and all measured TBT constraints are satisfied; THROUGHPUT and COLLECTIVE requests use the request-level completion budget. TTFT and E2E are measured from Router arrival.

Completion-oriented requests use the causal output-demand estimator in Appendix A.1, initialized with a 256-token hint.

B.3 Baselines, Diagnostics, and Ablations

All baselines run in the same serving harness, with policy-specific decision rules detailed in Table 3. RR, LL, and RD are direct policies implemented in the common serving harness. For DualMap, we adapt its routing logic to the common vLLM deployment. We integrate the authors' released SkyWalker codebase with the common evaluation harness.

Table 3: **Baseline implementations in the common serving harness.**

Policy	**Implemented rule**
RR	Cyclic assignment in configured replica order.
LL	Replica with the fewest pending input tokens; replica ID breaks ties.
RD	Uniform replica selection using a deterministic seed of 42.
DualMap	Prefix double-hash to two candidate clusters, with up to two candidate replicas per cluster; cache-affinity/load-aware selection and hotspot-triggered queued-request rebalancing. Rebalancing triggers above 32,768 queued tokens or 0.5 s queue wait, with 0.02 s hysteresis and at most eight moves per event.
SkyWalker	Prefer available local replicas, then available remote replicas; candidates are ordered by prefix locality and pending/running load, with a global least-loaded fallback when no replica is immediately available.

Diagnostic and ablation variants. The fixed-delay diagnostic varies whether waiting demand is visible to capacity planning and whether the physical delay is charged to the evaluated SLO (Table 4).

Table 4: **Fixed-delay diagnostic variants.**

Variant	Physical delay	Visible while waiting	SLO accounting
LateBind-τ	Yes	No	Standard
FreeWait	Yes	No	Delay excluded
ImmediateAccounting	Yes	Yes	Standard
Both	Yes	Yes	Delay excluded

LateBind-τ delays dispatch of an arrived request centrally and excludes it from prospective capacity state until one final placement after τ. FreeWait retains the physical delay and hidden-demand behavior but removes the waiting interval from SLO accounting. ImmediateAccounting retains the physical delay but makes waiting demand prospectively visible throughout the interval; Both applies both interventions.

For the RAVEL mechanism ablation, NoAccounting preserves SLO-aware virtual placement and pre-handoff replanning but removes Router-owned prospective demand from the state observed by subsequent placement decisions. FixedPlacement disables ordinary pre-handoff reassignment while preserving prospective accounting, RAVEL's initial placement rule, and the active observation window. NoHold only sets $h_i = 0$ in Equation (8); all other mechanisms remain unchanged, including prospective accounting, capacity admission, and ordinary replanning before handoff. It can therefore still incur queueing delay. The comparison evaluates the implemented hold rule, including its effect on dispatch eligibility, rather than an optimal hold duration.

B.4 Experiment Coverage and Statistical Protocols

Both models have complete load sweeps. The repeated-run, mechanism, latency-distribution, and SLO-sensitivity analyses use Qwen3-1.7B; planner scaling measures Router-side CPU execution without GPU inference.

Under the load-sweep protocol in Table 5, "best baseline" is the non-RAVEL policy with the highest request-level SLO attainment in the same workload–load cell. For Qwen3-1.7B at $16\times$, DualMap is the best baseline for all three workloads and is therefore held fixed as the comparator in the five paired runs.

Table 5: **Experimental protocols and comparison rules.**

Experiment	Setting	Protocol
Load sweep	1–16×	Qwen3-1.7B and Qwen3-8B: one 500-request run per policy–workload–load cell; best non-RAVEL policy selected within each cell.
Repeated runs	16×	Qwen3-1.7B: five paired executions per workload with one comparator fixed across all repetitions.
Mechanism ablation	All workloads, 12× and 16×	Qwen3-1.7B: RAVEL, NoAccounting, FixedPlacement, and NoHold. The main text presents Burst 12×; the appendix reports all cells.
SLO sensitivity	Burst 12×	Qwen3-1.7B: budget factors 0.8, 1.0, and 1.2; one 500-request run per scale–policy cell.
LateBind diagnostic	$\tau \in \{0, 10, 20, 40, 80\}$ ms	Qwen3-1.7B, DeepResearch 12×, five runs; fixed-delay sweep and waiting/visibility interventions.
Quote validation	DeepResearch 16×	Five runs, 2,500 final-handoff records.
Planner benchmark	$K \in \{1, 2, 4, 8, 16, 32, 64, 128\}$	50 repetitions per cohort bound; replica-scaling sampling is detailed in Appendix B.5.

For Figure 5(a), normalized quote error is $e_i = (\widehat{T}_i - T_i)/D_i$, where T_i is the observed latency, $\widehat{T}_i$ is its final-handoff prediction, and D_i is the applicable SLO budget. LATENCY requests use TTFT and its budget; completion-oriented requests use the point E2E prediction and completion budget. Positive errors indicate overprediction, and the shaded $\pm 10\%$ band denotes errors within one tenth of the applicable SLO budget.

Each model's load sweep contains $3 \times 6 \times 6 = 108$ cells: three workloads, six speedups, and six policies. These are individual runs, not repeated-run averages. Best-baseline comparisons are made within each model; the fixed DualMap comparator for the repeated Qwen3-1.7B runs is not imposed on the Qwen3-8B results.

For SLO sensitivity, every request's inherited budget is multiplied by the same factor, preserving request classes and relative budget assignments. Each Burst replay includes 181 LATENCY requests; TBT violation rates use these requests as the denominator.

B.5 Specialized Measurement Procedures

Pre-handoff opportunity reconstruction. We match requests between the Qwen3-1.7B Burst $12\times$ Full and FixedPlacement runs to identify model-consistent reassignment opportunities. A FixedPlacement miss is included only if (i) the request is predicted infeasible under the frozen incumbent allocation, (ii) it becomes predicted feasible after reassignment within the corresponding planning episode, (iii) the matched request subsequently satisfies its realized request-level SLO under RAVEL, and (iv) at least one earlier-arrived Router-owned request is moved away from the selected region by the same plan. Requests that fail any of these conditions remain unclassified.

For the reconstruction, materialized engine state is held fixed and the current Router cohort is removed from prospective state before the affected request is re-evaluated across candidate regions. Candidate quotes and feasibility decisions use the same causal prediction model as RAVEL, and each accepted reassignment is installed into the pass-local prospective state before subsequent requests are evaluated.

Planning path and timing. We time RAVEL's Router-side replanning procedure, including SLO quotes, prospective accounting, candidate scoring, and ledger updates. The total outstanding count across Router queues and engine pending/running states is computed once per synchronous pass and cached until that pass exits, including on failure. Timing excludes topology/request construction, file reads, plotting, and candidate-edge audits. For every sample, all $K_{\text{eff}}R$ candidate edges are audited separately on a state constructed from identical inputs.

Synthetic state and execution environment. Requests use Burst-trace request-visible lengths, types, SLOs, and relative arrival times. The three regions keep their service profiles and RTTs, with replicas cloned 2:2:1. Engine pending/running states start empty, caches start cold, and K_{eff} Router-owned requests are distributed across replicas. One process runs on an Intel Xeon Gold 6430, pinned to logical CPU 0 under a 16-core CPU quota without exclusive physical-core access. No quota throttling occurred. No GPU or network inference runs.

Sampling and statistics. Each configuration uses 50 samples: 300 in total for $K_{\text{eff}} = 8$ and 900 for $K_{\text{eff}} \in \{16, 32, 64\}$. Cell order is shuffled with seed 42. P95 latency is computed by linear interpolation. Throughput divides total comparisons by total timed planning duration.

C Extended Evaluation

This appendix reports full model-specific SLO results, run-to-run stability, latency distributions, mechanism analyses, SLO sensitivity, and planning cost. Experimental configuration, replay construction, baseline definitions, and ablation protocols are given in Appendix B.

C.1 SLO Attainment Across Model Sizes

Table 6 reports request-level SLO attainment across two model sizes, three datasets, and six offered-load multipliers. Model configurations and replay protocols are specified in Appendices B.1 and B.4.

The strongest competing policy changes in the larger-model evaluation. For Qwen3-8B at $16\times$, it is LeastLoad on LMSYS, DualMap and Random Dispatch jointly on Burst, and SkyWalker on

DeepResearch. RAVEL achieves higher SLO attainment in each case, so its advantage persists across these changes in the strongest baseline.

Table 6: **SLO attainment across model sizes and offered loads (%).** Bold marks the best policy for each model, dataset, and speedup, including ties.

Model	Dataset	Policy	Trace-replay speedup					
			1×	2×	4×	8×	12×	16×
Qwen3-1.7B	LMSYS	RAVEL	**97.4**	**96.6**	**96.4**	**96.2**	**92.6**	**91.6**
		DualMap	96.4	94.6	93.6	92.0	90.8	82.8
		RR	93.8	92.6	90.8	90.4	85.6	81.6
		LL	90.4	86.0	81.4	81.0	84.6	80.6
		RD	94.2	94.0	90.4	89.0	86.6	72.6
		SkyWalker	94.2	92.8	86.4	81.4	83.8	65.4
	Burst	RAVEL	**92.8**	**93.8**	**87.6**	**80.8**	**81.2**	**79.2**
		DualMap	74.6	69.6	73.0	70.6	69.0	68.0
		RR	76.8	67.2	52.6	48.6	55.2	59.6
		LL	76.6	58.8	54.6	53.4	51.2	57.4
		RD	82.8	76.0	59.6	54.4	55.6	52.2
		SkyWalker	83.6	70.2	62.2	70.6	55.6	62.0
	DeepResearch	RAVEL	**83.4**	**81.8**	**81.4**	**80.8**	**79.2**	**79.0**
		DualMap	74.2	68.4	67.6	67.0	65.8	65.6
		RR	74.2	66.8	40.4	40.8	40.4	38.8
		LL	80.0	71.2	48.8	46.4	45.0	43.4
		RD	81.2	70.4	47.0	45.0	46.8	47.6
		SkyWalker	80.0	70.0	52.4	50.2	46.6	43.2
Qwen3-8B	LMSYS	RAVEL	**91.6**	**90.2**	**88.2**	**87.4**	**82.0**	**85.4**
		DualMap	90.8	89.6	83.6	77.0	77.6	71.2
		RR	91.0	89.0	84.0	78.0	74.0	72.2
		LL	81.2	71.6	72.2	72.4	72.8	77.2
		RD	89.8	86.6	83.4	77.6	73.2	74.2
		SkyWalker	86.8	81.0	71.0	69.2	70.8	73.0
	Burst	RAVEL	**84.4**	**84.0**	**82.2**	**79.6**	**76.0**	**73.8**
		DualMap	78.2	73.0	70.4	69.8	61.4	64.6
		RR	78.4	74.2	72.4	71.2	66.2	64.0
		LL	71.8	71.6	66.6	64.8	61.6	60.4
		RD	74.8	73.0	65.6	67.6	61.0	64.6
		SkyWalker	75.2	71.8	69.4	62.8	65.6	59.4
	DeepResearch	RAVEL	**70.0**	**67.2**	**68.4**	**66.2**	**63.6**	**65.2**
		DualMap	67.4	59.8	56.2	46.8	52.2	42.8
		RR	66.2	65.2	64.4	61.6	61.6	61.2
		LL	69.0	64.2	63.8	63.2	61.4	61.0
		RD	67.8	66.0	63.2	61.8	61.2	61.0
		SkyWalker	67.6	65.0	63.8	61.6	62.6	62.2

Results. RAVEL attains the highest reported SLO attainment in all 18 dataset–speedup settings for each model. At 16×, its gains over the best competing policy are 8.8, 11.2, and 13.4 percentage points on LMSYS, Burst, and DeepResearch with Qwen3-1.7B, and 8.2, 9.2, and 3.0 points with Qwen3-8B. These results support the SLO benefit across the two evaluated model sizes.

C.2 Repeated-Run Robustness

To assess Qwen3-1.7B run-to-run variability, we repeat the 16× comparison five times for each workload using the fixed workload-specific comparator defined in Appendix B.4. Table 7 reports the paired difference in request-level SLO attainment between RAVEL and that comparator for every run.

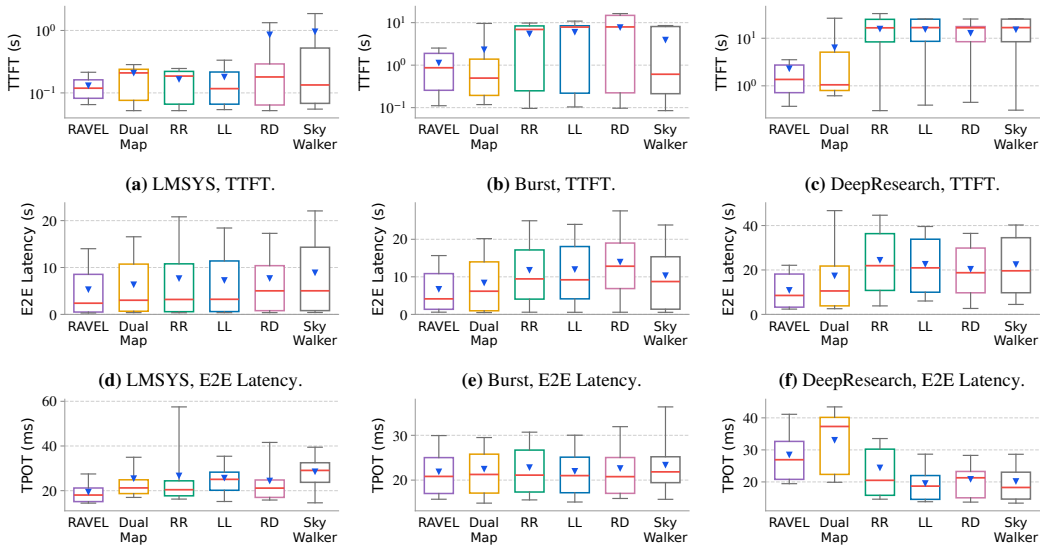
Table 7: **Repeated-run SLO-attainment gains at 16× load.** R1–R5 are paired RAVEL-minus-comparator differences in percentage points. The final column gives a paired 95% Student-t confidence interval over the five run-level differences.

	Paired run-level gain (pp)					Summary (pp)	
Workload	**R1**	**R2**	**R3**	**R4**	**R5**	**Mean**	**95% CI**
LMSYS	+3.2	+2.4	+8.0	+0.8	+6.0	+4.08	[0.49, 7.67]
Burst	+2.4	+7.2	+9.2	+9.4	+9.2	+7.48	[3.78, 11.18]
DeepResearch	+19.4	+18.6	+11.8	+25.0	+12.2	+17.40	[10.55, 24.25]

All fifteen paired run-level differences are positive. The mean gains are 4.08, 7.48, and 17.40 percentage points for LMSYS, Burst, and DeepResearch, respectively, matching the aggregate values reported in Section 4. All three 95% confidence intervals are above zero. DeepResearch exhibits the largest run-to-run variation in absolute gain, whereas LMSYS shows the smallest mean effect.

C.3 Per-Request Latency Distributions at 16×

Figure 6 complements the aggregate Qwen3-1.7B 16× results with the corresponding per-request TTFT, E2E, and TPOT distributions for all evaluated policies. Table 8 gives the absolute tail latencies and reductions against the best baseline.



(a) LMSYS, TTFT. **(b)** Burst, TTFT. **(c)** DeepResearch, TTFT.

(d) LMSYS, E2E Latency. **(e)** Burst, E2E Latency. **(f)** DeepResearch, E2E Latency.

(g) LMSYS, TPOT. **(h)** Burst, TPOT. **(i)** DeepResearch, TPOT.

Figure 6: **Per-request latency distributions at 16× load.** Columns show LMSYS, Burst, and DeepResearch; rows report **(top)** time to first token (TTFT), **(middle)** end-to-end latency (E2E), and **(bottom)** time per output token (TPOT). All distributions use the same 500-request runs as the corresponding 16× cells in Figure 3. TTFT and E2E are measured from Router arrival. Box plots show the median, interquartile range, 10th–90th percentile whiskers, and mean.

TTFT and E2E include the effects of geographic routing, Router-side waiting, and downstream queueing, whereas TPOT is accumulated primarily after geographic placement has become final. Accordingly, policy differences are generally more pronounced in TTFT and E2E than in TPOT.

Table 8: **Absolute tail latencies and reductions at $16\times$.** Values use the Qwen3-1.7B single-run load sweep. Latencies are in seconds; reduction is $100(1 - \text{RAVEL}/\text{best baseline})\%$. Bold marks lower latency.

	P95 TTFT			P95 E2E		
Workload	RAVEL	Best baseline	Reduction (%)	RAVEL	Best baseline	Reduction (%)
LMSYS	**0.234**	0.460	49.1	**16.376**	19.391	15.5
Burst	**2.684**	13.767	80.5	**18.780**	25.098	25.2
DeepResearch	**9.048**	34.888	74.1	**24.895**	57.619	56.8

C.4 Mechanism Ablation and Reassignment Opportunities

Section 4.3 uses Burst at $12\times$ load to isolate the effects of prospective demand visibility, pre-handoff revisability, and active observation. Table 9 extends the same ablation to all three workloads at both $12\times$ and $16\times$ load. Each policy cell contains 500 requests from one run; these ablations do not provide run-to-run confidence intervals.

Table 9: **Mechanism ablation across high-load settings.** Values are request-level SLO attainment (%). Full, NoAcct, and Fixed abbreviate RAVEL, NoAccounting, and FixedPlacement, respectively. Δ_{acct} denotes Full minus NoAccounting and Δ_{rev} denotes Full minus FixedPlacement, in percentage points. Δ_{hold} denotes Full minus NoHold.

Workload	Load	Full	NoAcct	Fixed	NoHold	Δ_{acct}	Δ_{rev}	Δ_{hold}
LMSYS	$12\times$	**92.6**	88.0	81.6	86.6	+4.6	+11.0	+6.0
LMSYS	$16\times$	**91.6**	84.8	84.8	84.6	+6.8	+6.8	+7.0
Burst	$12\times$	**81.2**	59.8	68.0	71.2	+21.4	+13.2	+10.0
Burst	$16\times$	**79.2**	60.8	67.4	70.6	+18.4	+11.8	+8.6
DeepResearch	$12\times$	**79.2**	73.6	73.4	74.6	+5.6	+5.8	+4.6
DeepResearch	$16\times$	**79.0**	77.8	75.6	76.6	+1.2	+3.4	+2.4

Table 10 reports P95 TTFT and P95 E2E for the same workload–load cells.

Table 10: **Tail latency for the extended mechanism ablation.** The two panels report P95 TTFT and P95 E2E in seconds.

Workload	Load	Full	NoAcct	Fixed	NoHold
P95 TTFT (s)					
LMSYS	$12\times$	0.209	0.215	0.278	0.336
LMSYS	$16\times$	0.234	0.232	0.242	0.833
Burst	$12\times$	2.905	3.641	2.490	2.182
Burst	$16\times$	2.684	4.276	4.718	4.250
DeepResearch	$12\times$	5.019	9.139	16.640	4.869
DeepResearch	$16\times$	9.048	4.710	15.199	4.910
P95 E2E (s)					
LMSYS	$12\times$	17.448	17.301	19.293	19.095
LMSYS	$16\times$	16.376	17.641	16.746	16.686
Burst	$12\times$	18.985	20.928	21.347	22.796
Burst	$16\times$	18.780	21.069	20.825	19.102
DeepResearch	$12\times$	21.988	27.368	31.612	26.953
DeepResearch	$16\times$	24.895	25.623	33.057	25.079

Full RAVEL achieves the highest request-level SLO attainment in all six high-load cells. Removing prospective accounting reduces attainment by 1.2–21.4 percentage points, while disabling pre-handoff reassignment reduces it by 3.4–13.2 points. The relative contribution is workload-dependent: the visibility effect is largest under the Burst trace, whereas its incremental effect is smaller for DeepResearch at $16\times$. These results extend the main-text ablation beyond a single workload–load setting and support all three mechanisms. Removing active observation reduces SLO attainment by 2.4–10.0 percentage points across the six cells.

At Burst $12\times$, FixedPlacement has lower P95 TTFT than Full (2.490 versus 2.905 s) but lower SLO attainment (68.0% versus 81.2%). At DeepResearch $16\times$, NoAccounting shows the same tradeoff: 4.710 versus 9.048 s P95 TTFT and 77.8% versus 79.0% attainment. Full reduces P95 E2E relative to FixedPlacement and NoHold in all six cells, and to NoAccounting in five. Full has higher P95 TTFT than NoHold on Burst $12\times$ and both DeepResearch loads, supporting an observation-time/SLO tradeoff rather than uniform latency gains.

C.4.1 Model-Consistent Pre-Handoff Opportunity Analysis

The FixedPlacement ablation establishes that preserving pre-handoff revisability can improve request-level SLO attainment. We further inspect the Burst $12\times$ run to identify FixedPlacement misses with a model-consistent pre-handoff reassignment opportunity in the logged RAVEL planning episode.

The classification and reconstruction procedure is specified in Appendix B.5.

FixedPlacement attains 68.0% SLO attainment on the 500-request Burst $12\times$ run, corresponding to 160 request-level misses. Of these, 66 satisfy all four conditions. Thus, the reconstruction identifies a model-consistent pre-handoff reassignment opportunity for $66/160 = 41.25\%$ of FixedPlacement misses.

C.5 Sensitivity to SLO-Budget Scaling

We evaluate SLO-budget sensitivity using the Qwen3-1.7B protocol in Appendix B.4.

Table 11 compares SLO attainment and latency under the three budget scales.

Table 11: **SLO-budget sensitivity on Burst at $12\times$.** SLO attainment is over all 500 requests; TBT violations are the percentage of LATENCY requests with at least one violating token interval. Latencies are in seconds.

SLO scale	Policy	SLO attainment (%)	P95 TTFT	P95 E2E	TBT violations (%)
0.8	RAVEL	**78.6**	2.396	17.952	**15.5**
	NoAccounting	55.8	5.166	19.711	30.9
	FixedPlacement	69.6	**2.001**	**16.839**	40.3
1.0	RAVEL	**81.2**	2.905	**18.985**	**6.6**
	NoAccounting	59.8	3.641	20.928	19.3
	FixedPlacement	68.0	**2.490**	21.347	37.6
1.2	RAVEL	**89.2**	2.267	**16.831**	**8.8**
	NoAccounting	62.2	5.420	20.648	23.8
	FixedPlacement	84.2	**1.872**	17.802	19.9

RAVEL achieves higher SLO attainment than both ablations at every tested scale. Its gains over FixedPlacement are 9.0, 13.2, and 5.0 percentage points at budget factors 0.8, 1.0, and 1.2, respectively; the corresponding gains over NoAccounting are 22.8, 21.4, and 27.0 points. Thus, the benefits of prospective accounting and pre-handoff redirection persist across all three tested budget settings, rather than depending on the nominal budgets alone. FixedPlacement retains the active observation window, so its gap to Full reflects the value of using that interval and subsequent waiting to revise placement.

The redirection gain is not monotonic in budget scale: it is largest at the nominal setting and smaller when budgets are either tightened or relaxed. This pattern is consistent with two competing effects. Tight budgets leave less room for alternative placements to become feasible, whereas relaxed budgets allow more requests to meet their SLOs without redirection. The measured gains therefore depend on both available slack and the placement opportunities it can make usable.

The latency results expose the associated tradeoff. FixedPlacement has lower P95 TTFT throughout, by 0.395–0.415 s, and lower P95 E2E at $0.8\times$. Nevertheless, RAVEL reduces TBT violation rates relative to FixedPlacement by 24.8, 31.0, and 11.1 percentage points across the three scales. Higher request-level SLO attainment thus coexists with a first-token tail-latency cost and fewer streaming violations.

C.6 Router-Side Planner Scaling

Figure 7 extends Figure 5(c) to synthetic replica counts $R \in \{5, 10, 20, 40, 80, 160\}$ and $K_{\text{eff}} \in \{8, 16, 32, 64\}$ requests processed per pass, evaluating all $K_{\text{eff}} R$ pairs. Here K_{eff} is the actual cohort size, whereas K is the configured bound. Measurement procedures are specified in Appendix B.5.

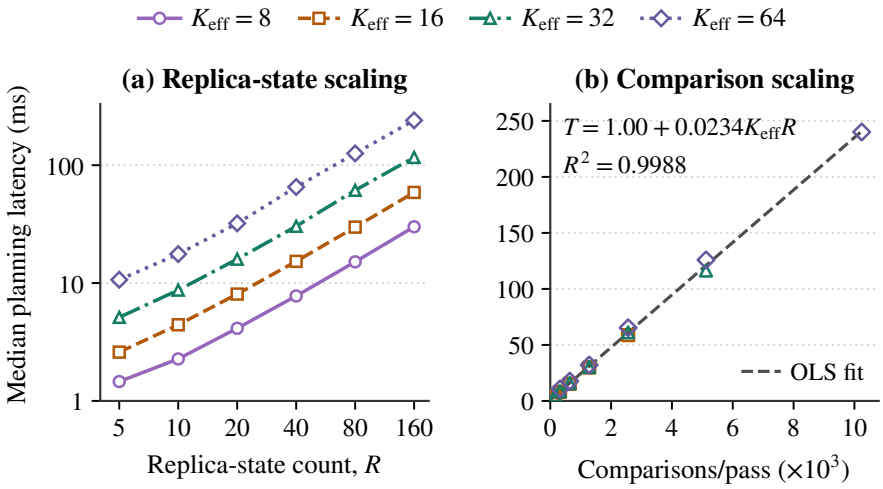


Figure 7: **Router-side planning cost tracks request–replica comparisons.** **(a)** Median latency versus synthetic replica-state count, with logarithmic axes, for four actual cohort sizes K_{eff}. **(b)** The same 24 cell medians against $K_{\text{eff}} R$ on linear axes, with an unweighted least-squares fit; T is measured in ms. Colors and markers identify cohort sizes in both panels.

Scaling and the cohort bound. An unweighted linear fit to the 24 cell medians gives $T_{\text{median}} \approx 1.00 + 0.0234(K_{\text{eff}} R)$ ms, with $R^2 = 0.9988$ over the measured range. This empirical relationship is consistent with the $O(K_{\text{eff}} R)$ request–replica comparison count in Section 3.4. For $R \geq 40$, measured comparison throughput ranges from 39,621 to 43,395 comparisons/s. The cohort bound therefore controls how many requests incur the cost of evaluating a growing replica candidate set in each pass.

Across the configurations in Table 12, increasing R from 40 to 160 at $K_{\text{eff}} = 8$ raises median latency from 7.770 to 30.027 ms and P95 from 9.113 to 33.920 ms. At $K_{\text{eff}} = 64$, the corresponding medians are 65.309 and 240.099 ms, exposing the higher cost of large replanning cohorts.

At $K_{\text{eff}} R = 1{,}280$, the four cohort–replica configurations have median latencies between 29.817 and 32.088 ms (Table 12). Their similar latency at a fixed comparison count illustrates the common scaling trend across different cohort sizes in Figure 7(b).

The cohort bound determines how many requests can be reconsidered in each pass and how much comparison work the Router may perform. A larger bound permits broader replanning, while a smaller bound limits computation as the replica set grows. The measured curves can guide selection of this bound for a target planning latency.

Table 12: **Synthetic planner latency across replica counts and cohort sizes.** Columns give replica-state count R; rows give requests actually processed per pass, K_{eff}. Panels report median and P95 latency in ms.

	Replica-state count R					
K_{eff}	5	10	20	40	80	160
Median latency (ms)						
8	1.460	2.272	4.133	7.770	15.141	30.027
16	2.594	4.425	8.061	15.280	29.817	58.732
32	5.118	8.711	15.950	30.320	61.230	116.367
64	10.666	17.617	32.088	65.309	125.877	240.099
P95 latency (ms)						
8	1.759	2.722	4.479	9.113	17.124	33.920
16	2.748	4.747	8.581	16.243	31.430	63.764
32	5.594	9.275	17.073	32.243	76.624	125.453
64	15.657	18.293	33.318	68.295	139.065	248.944

For context, the five-replica Burst $12\times$ serving run in Section 4.4 has a mean movable cohort of 3.65 requests, P95 7, and maximum 9 across 281 passes, with median/P95 scheduling latencies of 0.998/2.101 ms.