QUARTZ: Quantile-Aware Routing and Queueing for TTFT SLOs in LLM Serving

Zhipeng Liu*
Zhejiang University
Hangzhou, China
zhipengliu@zju.edu.cn

Yifan Zheng*
Zhejiang University
Hangzhou, China
zhengyifan1016@gmail.com

Fanqi Kong
Guangxi University
Nanning, China
fanqikong@st.gxu.edu.cn

Ziming Zhao†
Zhejiang University
Hangzhou, China
zimingzhao@zju.edu.cn

Abstract

Large Language Model (LLM) serving systems increasingly face strict time-to-first-token (TTFT) service-level objectives (SLOs), yet TTFT remains highly sensitive to router-side queueing effects. Prefill costs scale with prompt length, decode lengths are uncertain, and prefix locality creates strong performance skew across requests. Despite major advances in continuous batching and KV-cache management, today's routers are often agnostic to request size, which makes them vulnerable to head-of-line blocking and tail-latency amplification under mixed workloads. We propose **QUARTZ**, a quantile-aware routing and queueing layer for LLM serving that predicts service-time quantiles, rather than point estimates, using lightweight, router-visible features such as prompt length, cache-hit signals, and decoding parameters. QUARTZ uses these quantiles to route each request to the worker that minimizes predicted tail completion, and to prioritize admission and queueing decisions to satisfy TTFT SLOs while preserving fairness. We implement QUARTZ as a router upgrade for SGLang and evaluate it on representative interactive and retrieval-augmented workloads. The results show consistent reductions in TTFT tail latency and SLO violations across heterogeneous workloads.

1 Introduction

Interactive LLM applications, including chat assistants, coding agents, and document question answering, are governed by time to first token (TTFT) constraints. Users abandon or retry when the first token arrives too late, and product SLOs are typically specified at high percentiles such as P95 or P99 TTFT. Meeting tail SLOs is difficult because variability is amplified at scale (Dean and Barroso, 2013). In LLM serving, variability is structural: each request includes a prefill phase that processes the prompt and a decode phase that generates tokens autoregressively. Prefill cost grows with prompt length and prefix locality, while decode length is unknown until completion and benefits from batching (Agrawal et al., 2024). As a result, the same system must handle short interactive prompts and long retrieval augmented prompts, with heterogeneous and partially uncertain costs.

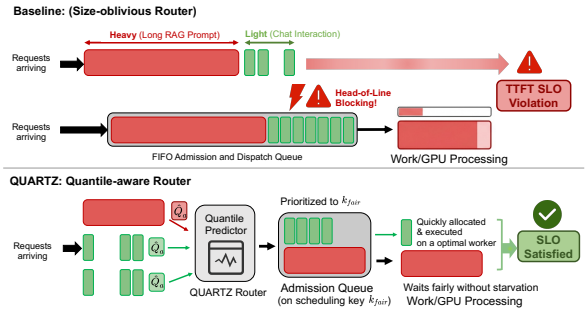

Figure 1: HOL blocking and TTFT tails.

Modern engines improve GPU efficiency via continuous batching and KV-cache management, for example ORCA, vLLM, and SGLang (Yu et al., 2022; Kwon et al., 2023; Zheng et al., 2024). However, a significant portion of tail TTFT is determined before GPU execution, at the router queue, where requests are admitted, assigned to workers, and prioritized under load. Figure 1 illustrates how size unaware admission induces head of line blocking and inflates TTFT tails.

Motivation. When the router ignores request cost, heavy requests, such as long prompts with low locality, can block light ones in admission and dispatch queues, amplifying tail latency. This phenomenon is well known in service systems, where size aware policies can reduce waiting but require a notion of job size (Jeon et al., 2014). In LLM serving, exact sizes are hard to obtain because decode

*These authors contributed equally.
†Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 37886–37896

length is uncertain and locality depends on cache state. Recent work shows that approximate ordering can already approach shortest job first behavior (Fu et al., 2024), while token based accounting can prevent starvation under mixed prompt lengths (Sheng et al., 2024; Cao et al., 2025). Yet existing routers still lack an objective that directly targets TTFT tail percentiles under SLOs.

Key insight. Tail SLOs are quantile targets, so optimizing only a mean or a single point estimate is misaligned with the goal. Underestimation is especially harmful because it admits heavy requests early and triggers severe head of line blocking. We therefore predict conservative service-time quantiles at levels $\alpha \in \{0.9, 0.95\}$ from router-visible features, and use them to make routing and admission decisions that minimize predicted tail completion while preserving fairness.

QUARTZ. We present **QUARTZ** (Quantile-Aware Routing and Queueing for TTFT SLOs), a router level design with four components. First, a lightweight quantile predictor estimates per request service time quantiles from prompt features, decoding parameters, and prefix locality signals. Second, a quantile aware routing rule assigns each request to the worker that minimizes predicted tail completion, using a small candidate set motivated by the power of two choices principle (Mitzenmacher, 2001). Third, a quantile aware admission policy prioritizes requests using conservative quantile risk, with aging and token based fairness to prevent starvation. Fourth, QUARTZ monitors residuals, applies lightweight calibration, and falls back to safe baselines under workload shifts. We implement QUARTZ by modifying the SGLang router queue, and it is compatible with existing GPU execution engines.

Contributions. This paper makes three contributions:

- We identify router side head of line blocking as a primary driver of TTFT SLO violations under mixed prompt lengths and prefix locality, motivating quantile aligned router objectives.
- We design QUARTZ, a quantile-aware routing and queueing framework that targets TTFT tail percentiles while preserving fairness.
- We implement QUARTZ in the SGLang router and evaluate it on interactive and retrieval-augmented workloads, showing consistent reductions in TTFT tails and SLO violations.

2 Background and Problem Setting

2.1 LLM serving and heterogeneity

We study online serving of transformer LLMs (Vaswani et al., 2017; Brown et al., 2020; Touvron et al., 2023). Requests execute in two phases: *prefill* processes the prompt to populate the KV cache, and *decode* generates tokens autoregressively while reusing cached states. Modern engines improve efficiency via continuous batching and KV-cache management (Yu et al., 2022; Kwon et al., 2023), and may exploit prefix reuse through cache structures such as RadixAttention (Zheng et al., 2024).

Request costs remain heterogeneous and partially uncertain. Prefill time scales with prompt length and prefix locality, while decode length is unknown until completion. Therefore routing and admission must operate under uncertainty.

2.2 TTFT and tail SLOs

We focus on *time to first token* (TTFT). A request r arrives at time a_r and emits its first token at $t_r^{(1)}$:

$$\mathrm{TTFT}(r) = t_r^{(1)} - a_r. \tag{1}$$

We decompose TTFT as

$$\mathrm{TTFT}(r) = T_{\mathrm{queue}}(r) + T_{\mathrm{prefill}}(r) + T_{\mathrm{first}}(r), \tag{2}$$

where T_{queue} includes router and worker waiting, T_{prefill} is prompt processing, and T_{first} covers the time from prefill completion to the first decoded token. Production SLOs are typically specified at high percentiles (e.g., P95 or P99), motivating designs that directly target tail behavior (Dean and Barroso, 2013).

2.3 Router role and observability

We consider a router and W workers hosting the same model. The router decides (i) *routing* to a worker and (ii) *admission ordering* among pending requests. It observes request metadata and lightweight worker signals such as coarse backlog summaries and KV-cache pressure (Yu et al., 2022; Kwon et al., 2023). When prefix reuse is available, it can also obtain a bounded prefix-locality proxy from cache structures (Zheng et al., 2024).

Heterogeneous request costs make size-oblivious policies vulnerable to head-of-line

blocking. Classical results suggest that size-aware scheduling mitigates blocking (Schrage, 1968; Harchol-Balter and Scully, 2022), and low-overhead load balancing (e.g., power of two choices) reduces imbalance across workers (Azar et al., 1999). These insights motivate using a router-visible *size proxy* to guide both routing and admission, even when true service time is not directly observable (Jeon et al., 2014; Fu et al., 2024). Fairness further constrains the design, as token-based accounting can prevent starvation under mixed prompt lengths (Sheng et al., 2024; Cao et al., 2025).

2.4 Objective and constraints

Let τ be a TTFT SLO threshold. We measure violations via

$$\mathbb{I}[\mathrm{TTFT}(r) > \tau]\,. \tag{3}$$

Our objective is to reduce tail TTFT and the violation rate under mixed workloads, while ensuring fairness across request classes (Sheng et al., 2024; Cao et al., 2025). We summarize notation in Table 1.

2.5 Why quantile-aligned prediction

A single point estimate of service time is misaligned with tail SLOs. Underestimation is especially harmful because it admits heavy requests early and amplifies head-of-line blocking. We therefore predict *service-time quantiles* and use them directly in routing and admission decisions. Quantile regression estimates conditional quantiles (Bauer et al., 2024), and calibration methods such as conformalized quantile regression can improve coverage under distribution shift (Romano et al., 2019). This motivates QUARTZ: conservative quantile prediction from router-visible signals, coupled with quantile-aware routing and queueing.

3 Method

Overview. QUARTZ is a router-layer framework that targets TTFT tail percentiles by combining quantile-aligned prediction with routing and admission decisions using only router-visible signals. Figure 2 illustrates the dataflow. For each incoming request, the router extracts lightweight features, predicts conservative service-time quantiles, selects a worker, and enqueues the request with a priority that reflects TTFT risk and fairness constraints. After the request completes prefill and emits the first token, the router logs outcomes and updates a lightweight calibration state.

Symbol	Meaning
r / w / c	request / worker / client
TTFT	time to first token
τ	TTFT SLO threshold
prefill / decode	prompt processing / autoregressive generation
p_r	prompt length (tokens)
m_r	generation budget (e.g., max new tokens)
ℓ_r / hit_proxy	prefix-locality proxy for request r
$\widehat{W}(w)$	estimated expected workload on worker w
$\widehat{Q}_\alpha^{\mathrm{prefill}}(r,w)$	predicted α-quantile prefill time on worker w
$\widehat{Q}_\alpha^{\mathrm{exec}}(r,w)$	predicted α-quantile end-to-end service time
$S_\alpha(r,w)$	routing score for worker w
$\widehat{s}_\alpha(r)$	size proxy for admission ordering
$k(r)$ / $k_{\mathrm{fair}}(r)$	(aged) priority key / fairness-adjusted key
λ / β	aging strength / fairness weight
δ	fallback trigger (under-coverage threshold)

Table 1: Notation.

Design principles. **Quantile alignment:** since TTFT SLOs are quantile targets, QUARTZ predicts conditional quantiles of service times rather than only predicting means (Bauer et al., 2024), thereby avoiding optimistic decisions that can amplify head-of-line blocking under mixed workloads (Dean and Barroso, 2013). **Router visibility:** all routing and admission decisions rely only on router-visible signals, including request metadata, coarse worker backlogs, and prefix-locality proxies exposed by the serving engine (Yu et al., 2022; Kwon et al., 2023; Zheng et al., 2024). **Fairness and robustness:** size-aware scheduling can reduce latency but may starve long requests without safeguards (Harchol-Balter et al., 2003; Harchol-Balter and Scully, 2022). QUARTZ therefore integrates token-based fairness inspired by recent LLM serving work (Sheng et al., 2024; Cao et al., 2025) and falls back to safe baselines under drift, motivated by calibration results for quantile models (Romano et al., 2019).

3.1 Problem formulation and signals

A router connects to W workers serving the same model with continuous batching and KV caching. A request r arrives at time a_r with metadata (p_r, m_r, θ_r), is assigned to a worker $w(r)$, and is dispatched by an admission policy. The router observes a lightweight feature vector from request

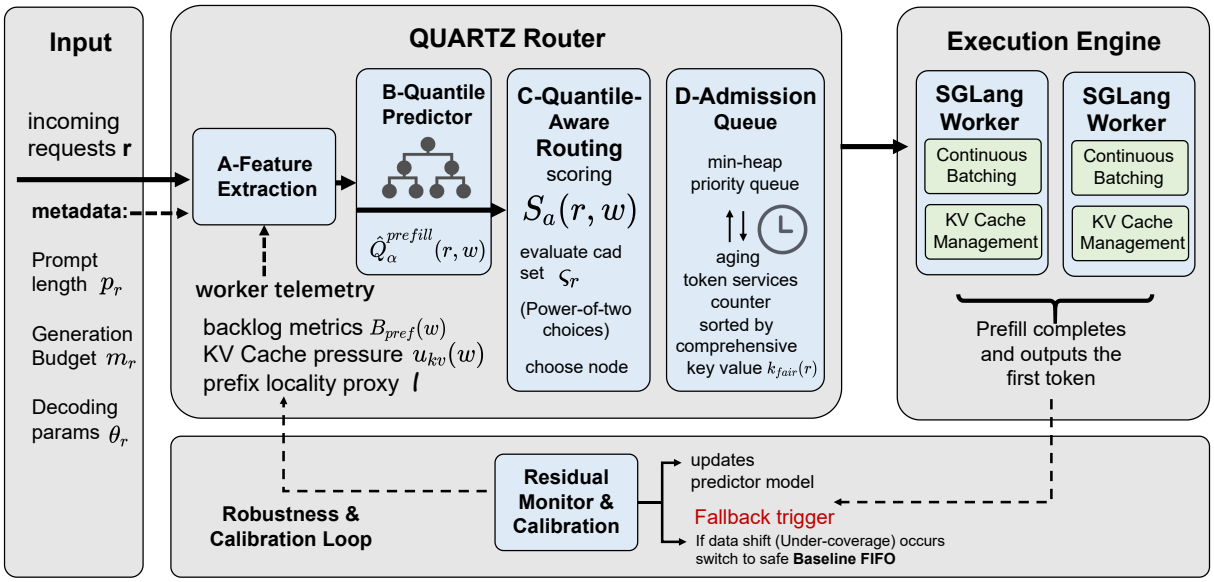

Figure 2: Overview of QUARTZ architecture

Group	Feature	Definition (router-visible)
Request	p_r	prompt length in tokens (or tokenizer proxy)
Request	m_r	max new tokens / generation budget
Request	θ_r	decoding params (e.g., temperature, top-p)
Request	type, stream	request class and streaming flag
Locality	ℓ_r	prefix-locality proxy (e.g., LPM length in tokens)
Locality	bucket(ℓ_r, p_r)	coarse locality class (none / partial / high)
Worker	$n_{\text{seq}}(w)$	active sequences on worker w
Worker	$B_{\text{pref}}(w)$	prefill backlog proxy (tokens or ms)
Worker	$u_{\text{kv}}(w)$	KV-cache pressure / utilization proxy

Table 2: Router-visible features used by QUARTZ.

metadata, worker telemetry, and a prefix-locality proxy exposed by the engine (Zheng et al., 2024; Yu et al., 2022; Kwon et al., 2023); see Table 2. Because the locality proxy is captured at arrival, it may experience *cache state drift* by the time of actual dispatch due to intervening evictions or insertions in the admission queue. QUARTZ explicitly tolerates this temporal uncertainty as an engineering trade-off: it relies on the macro-stability of coarse locality classes (e.g., none / partial / high) over short queueing windows, and absorbs micro-deviations through online residual calibration.

3.2 Quantile prediction

QUARTZ predicts conditional service-time quantiles aligned with tail objectives. Given router-visible features, a compact tree predictor estimates the α-quantile prefill time $\widehat{Q}_\alpha^{\text{prefill}}(r,w)$ (with $\alpha \in \{0.9, 0.95\}$) using pinball loss (Bauer et al., 2024). To improve reliability under drift, QUARTZ applies lightweight bucketed calibration via online residual correction, optionally conformalized for coverage (Romano et al., 2019).

3.3 Quantile-aware routing

QUARTZ maintains a work-based backlog proxy $B(w)$ and derives an estimated expected waiting time $\widehat{W}(w)$, avoiding queue-length-only load signals (Harchol-Balter and Scully, 2022). Crucially, to avoid the statistical fallacy of summing quantiles, $\widehat{W}(w)$ aggregates the *expected* (mean) work of queued requests rather than their tail percentiles. For request r and candidate worker w, it computes the routing score:

$$S_\alpha(r,w) = \widehat{W}(w) + \widehat{Q}_\alpha^{\text{prefill}}(r,w) + \psi(r,w), \quad (4)$$

where S_α acts as a conservative tail-completion proxy formed by adding the risk-aware prefill quantile of the *current* request to the expected baseline delay, and $\psi(r,w)$ conservatively penalizes undesirable states (e.g., high KV pressure). For each request, QUARTZ evaluates a small candidate set $\mathcal{C}(r)$ (default $K=2$) and selects

$$w(r) = \arg\min_{w \in \mathcal{C}(r)} S_\alpha(r,w), \quad (5)$$

following the power-of-two choices principle (Azar et al., 1999).

3.4 Quantile-aware admission queue

To mitigate router-side head-of-line blocking, QUARTZ orders admission using a conservative

Algorithm 1 QUARTZ: quantile-aware routing and admission

On arrival of request r:
Construct candidate set $\mathcal{C}(r)$ of size K
for all $w \in \mathcal{C}(r)$ **do**
    Compute $\widehat{W}(w)$ from backlog proxy $B(w)$
    Predict $\widehat{Q}_\alpha^{\text{prefill}}(r, w)$ (and optional terms)
    Compute $S_\alpha(r, w)$ by Eq. 4
end for
Choose $w(r) = \arg\min_{w \in \mathcal{C}(r)} S_\alpha(r, w)$ by Eq. 5
Compute $\widehat{s}_\alpha(r)$ (Eq. 6), $k(r)$ (Eq. 7), and $k_{\text{fair}}(r)$ (Eq. 8)
Push $(k_{\text{fair}}(r), r)$ into router heap; update $B(w(r))$
On dispatch capacity: pop the minimum-key request and dispatch it
On first-token completion: update residual monitor and update $u(c(r))$

size proxy

$$\widehat{s}_\alpha(r) = \widehat{Q}_\alpha^{\text{prefill}}(r, w(r)). \tag{6}$$

It implements aging without priority updates via an arrival-time-adjusted key

$$k(r) = \widehat{s}_\alpha(r) + \lambda \cdot a_r, \tag{7}$$

and preserves fairness using token-based service counters $u(c)$:

$$k_{\text{fair}}(r) = k(r) + \beta \cdot u(c(r)). \tag{8}$$

The router maintains a priority queue ordered by $k_{\text{fair}}(r)$ and dispatches the minimum-key request when capacity is available (Sheng et al., 2024; Cao et al., 2025).

3.5 Robustness via calibration and fallback

QUARTZ monitors residuals between realized prefill times and predicted quantiles to detect under-coverage. When under-coverage exceeds a threshold, QUARTZ reduces reliance on prediction by shifting traffic to a safe baseline (e.g., FIFO admission with least-loaded routing), preserving stability under workload shifts (Dean and Barroso, 2013).

Algorithm sketch. Algorithm 1 summarizes QUARTZ routing and admission.

4 Implementation

We implement QUARTZ as a router-level upgrade in SGLang with minimal disruption to the execution engine. QUARTZ changes only the router admission queue and dispatch logic, and uses lightweight signals already available at workers, including load summaries and a prefix-locality proxy exposed by RadixAttention (Zheng et al., 2024). Continuous batching and KV cache management remain unchanged (Yu et al., 2022; Kwon et al., 2023).

4.1 Integration points in SGLang

Router queue and dispatcher. We replace the default admission ordering with QUARTZ's priority queue and insert quantile-aware worker selection in the dispatch path. The router pipeline is: extract features, predict q_α, score a small set of candidates, enqueue by $k_{\text{fair}}(r)$, and dispatch when capacity is available.

Worker telemetry and prefix locality. Workers periodically report coarse load signals (e.g., active sequences, prefill/decode backlog, KV pressure) that the router consumes opportunistically without blocking (Kwon et al., 2023; Agrawal et al., 2024). To capture prefix reuse, we expose a bounded locality proxy from RadixAttention (e.g., longest-match length or a coarse class), used as a predictor feature and an optional routing penalty (Zheng et al., 2024).

4.2 Quantile predictor and calibration

We implement the quantile predictor as a compact decision tree evaluator on the router critical path. Features combine request metadata (prompt/generation lengths and decoding parameters), the latest telemetry snapshot, and the locality proxy. To improve coverage under workload heterogeneity, QUARTZ maintains a small regime-level calibration table (discretized by prompt/locality/load) and applies an online correction via lightweight residual tracking (Romano et al., 2019).

4.3 Admission queueing and fairness

The admission queue is a binary heap keyed by $k_{\text{fair}}(r)$; the aging formulation in Equation 7 avoids dynamic priority updates. We implement token-based fairness using per-client exponentially decayed service counters (e.g., via an exponentially weighted moving average, EWMA) updated at first-token completion. This crucially prevents long-running active clients from being permanently starved by unbounded counter growth. We retain state only for active clients with time-based eviction (Sheng et al., 2024).

4.4 Routing state and scoring

QUARTZ maintains a work-based per-worker backlog proxy $B(w)$ updated on assignment and decayed using recent service observations, yielding $\widehat{W}(w)$ that reflects predicted *expected* work rather than request counts or aggregated quantiles (Harchol-Balter and Scully, 2022). For each request, the router evaluates a small candidate set (default $K{=}2$) and selects the lower score, following the power-of-two choices principle (Azar et al., 1999; Mitzenmacher, 2001). Scores use cached per-worker state, request features, and predicted quantiles, without synchronous communication.

4.5 Robustness

QUARTZ logs prediction residuals to monitor regime-level under-coverage and triggers a safe fallback (e.g., FIFO admission with least-loaded routing) when confidence degrades under shift (Dean and Barroso, 2013).

5 Evaluation

5.1 Experimental Setup

All experiments are conducted on a single node with 4×RTX 6000 GPUs, using one SGLang worker per GPU. Unless otherwise specified, we enable continuous batching, prefix reuse, and chunked prefill in the backend serving engine, and only modify the router queue and dispatch logic for QUARTZ. We evaluate three models from the same family, namely 7B, 14B, and 32B, in order to test whether the router-level gains of QUARTZ remain stable as model size increases.

QUARTZ uses a lightweight quantile predictor trained offline from router-visible traces. The predictor takes request metadata, prefix-locality signals, and coarse worker telemetry as input, and predicts conservative high-quantile prefill costs. During online serving, we keep the predictor fixed and only allow lightweight bucketed calibration and residual correction. Unless otherwise specified, the default QUARTZ parameters are $\alpha = 0.95$, $K = 2$, $\lambda = 0.02$, $\beta = 0.10$, and $\delta = 0.05$.

For all workloads, we report the average across three runs with different random seeds. We compare all methods under identical request traces and offered loads.

5.2 Workloads and Evaluation Protocol

We use four workload families. **Chat** is constructed from a public ShareGPT-style multi-turn conversation corpus and represents short interactive prompts with moderate outputs. **RAG/Long-context** is built from a public retrieval-style text corpus and contains substantially longer prompts due to injected retrieved passages. **Structured Prefix-Reuse** is a synthetic workload with controlled shared prefixes, used to isolate the effect of locality-aware routing signals. **Mixed** combines Chat and RAG requests to create the heterogeneous prompt-length distribution that most strongly induces head-of-line blocking. We further construct a **Mixed Bursty** regime by injecting short arrival bursts on top of the Mixed workload.

Table 3: Default QUARTZ parameters.

Parameter	Value
Quantile level α	0.95
Candidate workers K	2
Aging coefficient λ	0.02
Fairness weight β	0.10
Fallback threshold δ	0.05

We assign workload-specific TTFT SLOs. Short interactive requests use a 2.0 s TTFT target, while long-context requests use a 4.0 s TTFT target. For mixed workloads, each request inherits the SLO of its request class. Unless otherwise noted, each trace contains 3,000 requests and we sweep offered load from under-utilized to near-saturation regimes. For the main cross-model comparison, we report a representative high-load operating point close to the saturation knee of each model.

5.3 Baselines

We compare QUARTZ against the following baselines: **FIFO-LL**, which uses FIFO admission with least-loaded routing; **FIFO-P2C**, which replaces least-loaded routing with power-of-two choices; **Locality-aware**, which routes using a prefix-locality signal but keeps size-oblivious admission; **TokenFair**, which emphasizes token-based fairness in admission; **PointPred**, which uses the same predictor features as QUARTZ but replaces quantile prediction with a point estimate; and **Oracle**, which is an offline upper bound using observed prefill costs.

5.4 Main Results Across Model Scales

Table 4 summarizes the main results on the Mixed workload at a representative high-load operating point for each model scale. Across all three models, QUARTZ consistently reduces tail TTFT and SLO violations relative to both FIFO-LL and PointPred. The gains become larger as model size increases,

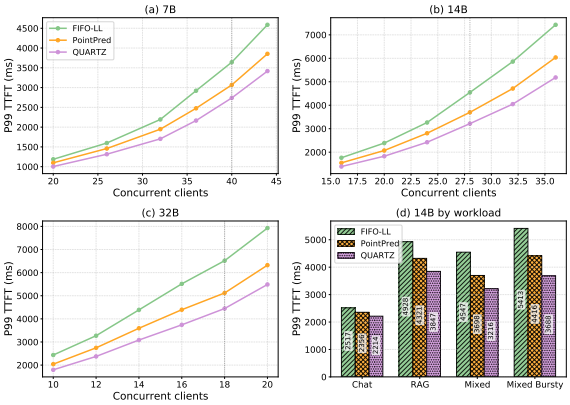

Figure 3: Main results across model scales.

which is consistent with the intuition that router-side queueing decisions become increasingly important as the serving regime approaches saturation.

On the 14B model, QUARTZ reduces P99 TTFT from 2517 ms to 2214 ms on Chat, from 4928 ms to 3847 ms on RAG, from 4547 ms to 3216 ms on Mixed, and from 5413 ms to 3688 ms on Mixed Bursty, relative to FIFO-LL. The relative gains are therefore smallest on homogeneous chat traffic and largest on mixed and bursty workloads, where head-of-line blocking is most severe.

5.5 Ablation Study

Table 5 reports the ablation results on the 14B Mixed workload. Replacing quantile prediction with a point estimate significantly hurts tail performance, which confirms that QUARTZ benefits specifically from quantile-aligned prediction rather than merely from having any predictor. Routing-only and queueing-only variants both recover part of the gain, but neither matches the full method, suggesting that the two components are complementary. Removing locality features also leads to a measurable degradation, especially in the mixed workload where prefix reuse is uneven across requests.

Removing fairness slightly improves tail latency on this nominal workload, but substantially harms fairness metrics reported later. This pattern is expected: pure size-biased ordering can be more aggressive on short requests, but it does so by penalizing long requests and increasing cross-client imbalance.

5.6 Predictor Quality and Calibration

The predictor quality results are shown in Table 6. On the nominal 14B Mixed workload, the raw $q_{0.95}$

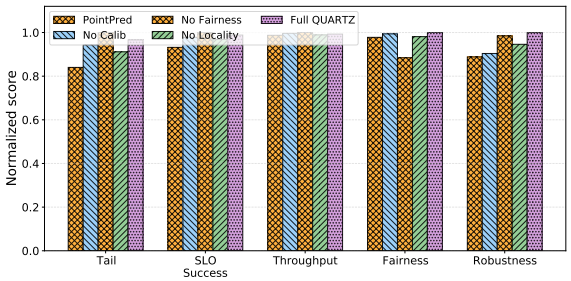

Figure 4: Ablation summary.

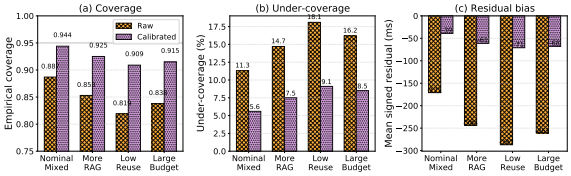

Figure 5: Predictor quality and calibration.

predictor achieves empirical coverage of 0.887, indicating that it underestimates the target quantile too often. After lightweight online calibration, coverage improves to 0.944, much closer to the desired operating point. The same pattern holds under distribution shifts, where calibration reduces under-coverage by roughly a factor of two.

The signed residuals further indicate that calibration mainly corrects optimistic bias rather than reducing all noise uniformly. For example, under the low-reuse shift, the mean signed residual improves from -287 ms to -71 ms, which is consistent with the goal of reducing harmful underestimation in routing and queueing decisions.

5.7 Robustness Under Workload Shifts

Table 7 reports robustness results on the 14B model. QUARTZ consistently outperforms FIFO-LL and PointPred under all three shifts. The largest gains appear under the low-reuse and large-budget settings, where queueing errors are amplified because long and expensive requests are harder to predict and more disruptive to dispatch order. Calibration and fallback are especially important in these cases: removing them reduces SLO success from 73.1% to 65.7% under low reuse and from 69.8% to 63.4% under large generation budgets.

On the 32B model, the absolute SLO success rates are lower because the system operates closer to saturation, but the relative pattern is similar. Under the same three shifts, QUARTZ maintains 70.4%, 66.5%, and 63.9% SLO success, compared with 57.8%, 53.1%, and 49.2% for FIFO-LL.

Table 4: Main results on the Mixed workload at representative high-load operating points.

Model	Load	Method	P95 TTFT (ms)	P99 TTFT (ms)	SLO violation (%)	Throughput (tok/s)
7B	40 clients	FIFO-LL	2374	3641	18.8	92,413
7B	40 clients	PointPred	2116	3067	14.6	92,857
7B	40 clients	QUARTZ	1869	2738	11.9	93,084
14B	28 clients	FIFO-LL	2861	4547	26.4	61,742
14B	28 clients	PointPred	2418	3698	19.7	62,118
14B	28 clients	QUARTZ	2137	3216	15.8	62,463
32B	18 clients	FIFO-LL	4038	6516	35.1	28,641
32B	18 clients	PointPred	3394	5119	28.9	28,917
32B	18 clients	QUARTZ	2912	4446	22.1	29,168

Table 5: Ablation results on the 14B Mixed workload.

Variant	P99 TTFT (ms)	SLO violation (%)	Throughput (tok/s)	Jain fairness
PointPred	3698	19.7	62,118	0.931
Routing-only	3576	18.4	62,771	0.938
Queueing-only	3471	17.8	61,954	0.942
No Calibration	3294	16.5	62,681	0.946
No Fairness	3107	15.0	62,914	0.842
No Locality Feature	3409	17.2	62,207	0.934
Full QUARTZ	3216	15.8	62,463	0.952

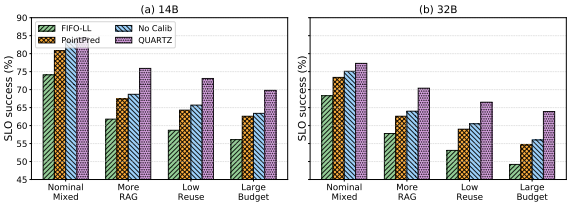

Figure 6: Robustness under workload shifts.

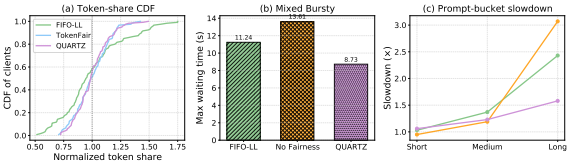

Figure 7: Fairness analysis.

5.8 Fairness Analysis

Table 8 shows the fairness results on the 14B model. On the Mixed workload, QUARTZ improves Jain's fairness index from 0.919 to 0.952 and reduces the maximum waiting time from 9.57 s to 7.43 s compared with FIFO-LL. The improvement is more pronounced under Mixed Bursty, where size-oblivious admission causes long waits for some clients. QUARTZ maintains high fairness because its aging and token-based fairness terms counteract the bias introduced by conservative size-aware ordering.

The `No Fairness` ablation confirms this trade-off. Although it attains slightly better tail latency on the nominal workload, Jain's index drops to 0.842 and long-prompt slowdown increases to 3.07×, indicating that fairness safeguards are necessary for stable multi-client behavior.

5.9 Summary

Overall, the results support three main conclusions. First, QUARTZ consistently improves TTFT tails and SLO attainment across 7B, 14B, and 32B models. Second, both quantile-aware routing and quantile-aware queueing are necessary to obtain the full gain, while calibration is especially important under workload shift. Third, the fairness terms help prevent severe starvation for long or under-served clients.

6 Related Work

LLM inference engines improve throughput and latency via cache-aware execution and batching, but prompt heterogeneity and mixed SLO classes make router-side queueing a dominant source of tail TTFT. Recent system designs address these issues using cluster-level rescheduling and queue management. Llumnix performs runtime rescheduling across model instances via request migration to improve isolation and tail latency under unpredictable request sizes (Sun et al., 2024). QLM targets end-to-end SLO attainment by orchestrating queue management operations that mitigate head-of-line blocking in shared LLM serving (Patke et al., 2024). Complementary to queue management, QLLM explores fine-grained preemption for mixed-priority workloads (in MoE inference) to protect latency-sensitive requests (Siavashi et al., 2025). More broadly, model serving systems such as TensorFlow Serving and Clipper establish practical deployment and batching abstractions (Olston et al., 2017; Crankshaw et al., 2017). SLO-aware

Table 6: Predictor quality on 14B. Coverage is measured for the target quantile level $\alpha = 0.95$.

Scenario	Raw cov.	Calib. cov.	Raw under-cov. (%)	Calib. under-cov. (%)	Signed residual (ms)
Nominal Mixed	0.887	0.944	11.3	5.6	$-171 \rightarrow -39$
More RAG	0.853	0.925	14.7	7.5	$-244 \rightarrow -61$
Low Reuse	0.819	0.909	18.1	9.1	$-287 \rightarrow -71$
Large Budget	0.838	0.915	16.2	8.5	$-261 \rightarrow -68$

Table 7: Robustness results on the 14B model.

Scenario	FIFO-LL	PointPred	No Calib.	QUARTZ
Nominal Mixed	74.1	80.8	83.5	84.4
More RAG	61.8	67.4	68.7	75.9
Low Reuse	58.7	64.3	65.7	73.1
Large Budget	56.1	62.6	63.4	69.8

Table 8: Fairness results on the 14B model.

Method	Jain ↑	Max wait (s) ↓	Long slow. ↓
Mixed			
FIFO-LL	0.919	9.57	2.11
TokenFair	0.944	8.08	1.66
QUARTZ	**0.952**	**7.43**	**1.49**
Mixed Bursty			
FIFO-LL	0.907	11.24	2.43
No Fairness	0.842	13.61	3.07
QUARTZ	**0.946**	**8.73**	**1.58**

and cost-aware inference serving has been studied in MArk and INFaaS (Zhang et al., 2022; Romero et al., 2021), while pipeline-level tail latency objectives were addressed by InferLine and later pipeline serving frameworks such as Loki (Crankshaw et al., 2020; Ahmad et al., 2024). QUARTZ differs by operating entirely at the router with router-visible signals, using conservative service-time quantiles to align routing and admission decisions with tail TTFT SLOs; we also report fairness using the standard Jain index (Jain et al., 1998).

7 Conclusion

We presented QUARTZ, a router-only framework for TTFT tail SLOs that aligns routing and admission with conservative service-time quantiles predicted from router-visible signals. By combining quantile prediction, quantile-aware routing, quantile-aware admission, and lightweight calibration and fallback, QUARTZ reduces head-of-line blocking while preserving fairness under heterogeneous prompt lengths and prefix locality. Experiments on an SGLang testbed show that QUARTZ consistently lowers high-percentile TTFT and SLO violation rates compared with load-based and point-estimate baselines, while also improving robustness under workload shifts and maintaining a simple deployment path that modifies only router-side logic. Future work includes extending QUARTZ to decode-phase objectives and to multi-node heterogeneous clusters.

Limitations

QUARTZ operates entirely at the router and relies on router-visible features, so it cannot directly observe per-request execution variability inside a worker (e.g., kernel-level stalls, KV-cache fragmentation, or transient GPU contention). Its benefits therefore depend on the stability and calibration of the quantile predictor; under abrupt workload shifts or unseen prompt patterns, prediction errors can reduce routing quality and may require more frequent recalibration. Our study focuses on TTFT tail SLOs and does not optimize other objectives such as token-level decode latency or end-to-end user-perceived latency beyond the first token. The evaluation is limited to the model and serving stack we tested; different engines, parallelism strategies, and heterogeneous clusters may change both the dominant bottlenecks and the best routing features. Finally, while our admission queue includes fairness safeguards, extreme overload can still induce trade-offs between strict tail SLO protection and per-tenant fairness guarantees.

References

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. 2024. Taming throughput-latency tradeoff in LLM inference with sarathi-serve. In *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, pages 117–134. USENIX Association.

Sohaib Ahmad, Hui Guan, and Ramesh K. Sitaraman. 2024. Loki: A system for serving ML inference pipelines with hardware and accuracy scaling. In *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing, HPDC 2024, Pisa, Italy, June 3-7, 2024*, pages 267–280. ACM.

Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. 1999. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200.

Ida Bauer, Harry Haupt, and Stefan Linner. 2024. Pin-

ball boosting of regression quantiles. *Comput. Stat. Data Anal.*, 200:108027.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shiyi Cao, Yichuan Wang, Ziming Mao, Pin-Lun Hsu, Liangsheng Yin, Tian Xia, Dacheng Li, Shu Liu, Yineng Zhang, Yang Zhou, Ying Sheng, Joseph Gonzalez, and Ion Stoica. 2025. Locality-aware fair scheduling in LLM serving. *CoRR*, abs/2501.14312.

Daniel Crankshaw, Gur-Eyal Sela, Xiangxi Mo, Corey Zumar, Ion Stoica, Joseph Gonzalez, and Alexey Tumanov. 2020. Inferline: latency-aware provisioning and scaling for prediction serving pipelines. In *SoCC '20: ACM Symposium on Cloud Computing, Virtual Event, USA, October 19-21, 2020*, pages 477–491. ACM.

Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. 2017. Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 613–627. USENIX Association.

Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM*, 56(2):74–80.

Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. 2024. Efficient LLM scheduling by learning to rank. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Mor Harchol-Balter, Bianca Schroeder, Nikhil Bansal, and Mukesh Agrawal. 2003. Size-based scheduling to improve web performance. *ACM Trans. Comput. Syst.*, 21(2):207–233.

Mor Harchol-Balter and Ziv Scully. 2022. The most common queueing theory questions asked by computer systems practitioners. *SIGMETRICS Perform. Evaluation Rev.*, 49(4):3–7.

Raj Jain, Dah-Ming Chiu, and W. Hawe. 1998. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *CoRR*, cs.NI/9809099.

Myeongjae Jeon, Saehoon Kim, Seung-won Hwang, Yuxiong He, Sameh Elnikety, Alan L. Cox, and Scott Rixner. 2014. Predictive parallelization: taming tail latencies in web search. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 253–262. ACM.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Michael Mitzenmacher. 2001. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distributed Syst.*, 12(10):1094–1104.

Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. 2017. Tensorflow-serving: Flexible, high-performance ML serving. *CoRR*, abs/1712.06139.

Archit Patke, Dhemath Reddy, Saurabh Jha, Haoran Qiu, Christian Pinto, Chandra Narayanaswami, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. 2024. Queue management for slo-oriented large language model serving. In *Proceedings of the 2024 ACM Symposium on Cloud Computing, SoCC 2024, Redmond, WA, USA, November 20-22, 2024*, pages 18–35. ACM.

Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. 2019. Conformalized quantile regression. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3538–3548.

Francisco Romero, Qian Li, Neeraja J. Yadwadkar, and Christos Kozyrakis. 2021. Infaas: Automated model-less inference serving. In *Proceedings of the 2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, pages 397–411. USENIX Association.

Linus Schrage. 1968. Letter to the editor - A proof of the optimality of the shortest remaining processing time discipline. *Oper. Res.*, 16(3):687–690.

Ying Sheng, Shiyi Cao, Dacheng Li, Banghua Zhu, Zhuohan Li, Danyang Zhuo, Joseph E. Gonzalez, and Ion Stoica. 2024. Fairness in serving large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, pages 965–988. USENIX Association.

Mohammad Siavashi, Faezeh Keshmiri Dindarloo, Dejan Kostic, and Marco Chiesa. 2025. Priority-aware preemptive scheduling for mixed-priority workloads in moe inference. In *Proceedings of the 5th Workshop on Machine Learning and Systems, EuroMLSys 2025, World Trade Center, Rotterdam, The Netherlands, 30 March 2025- 3 April 2025*, pages 132–138. ACM.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. 2024. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, pages 173–191. USENIX Association.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022*, pages 521–538. USENIX Association.

Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. 2022. Enabling cost-effective, slo-aware machine learning inference serving on public cloud. *IEEE Trans. Cloud Comput.*, 10(3):1765–1779.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. 2024. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.