Experiential Fairness: Bridging the Gap between User Experience and Resource-Centric Fairness in Online LLM Services

Jiahua Huang[1,2], Wentai Wu[*3], Yongheng Liu[2], Guozhi Liu[1], Yang Wang[4], Weiwei Lin[*1,2],

[1]South China University of Technology, Guangzhou, China
[2]Pengcheng Laboratory, Shenzhen, China
[3]Jinan University, Guangzhou, China
[4]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
wentaiwu@jnu.edu.cn, linww@scut.edu.cn

Abstract

Conventional fairness in multi-tenant Large Language Model (LLM) inference services is typically defined by system-centric metrics such as equitable resource allocation. We argue that this is unilateral and it creates a gap between measured system performance and actual user-perceived quality. We challenge this notion by introducing and formalizing Experiential Fairness, a user-centric paradigm that shifts the objective from equality of opportunity (resource access) to equity of outcome (user experience). With this motivation we propose ExFairS, a lightweight scheduling framework that perceives each user's satisfaction as a composite measure of Service Level Objective (SLO) compliance and resource consumption, and dynamically re-orders the serving queue guided by a credit-based priority mechanism. Extensive experiments on an 8-GPU NVIDIA V100 node show that ExFairS reduces the SLO violation rate by up to 100% and improves system throughput by 14-21.9%, outperforming state-of-the-art schedulers and delivering a demonstrably higher degree of Experiential Fairness.

Code — https://github.com/dadiaokua/ExFairS

Introduction

As LLMs are widely adopted in areas such as intelligent customer service, content creation, and automated software development, achieving efficient and stable inference in production has become a critical challenge. To address this, next-generation inference frameworks such as vLLM (Kwon et al. 2023) leverage key techniques like block-wise KV caching and dynamic shared memory management to reduce memory fragmentation and enhance multi-request concurrency, thereby significantly improving runtime efficiency and resource utilization.

However, beyond performance metrics such as throughput and latency, the fairness of serving under constrained computational resources presents a unique challenge when it comes to online serving. As the common rationale, First-Come, First-Served (FCFS) potentially gives advantage to users with high request rate, degrading the experience for others. While more advanced schedulers like VTC (Sheng et al. 2024) attempt to enforce fairness by prioritizing users with lower historical resource consumption, they suffer from a critical limitation that they are experience-blind. These system-centric approaches ignore the subjective aspects of user requests such as differing service tiers or latency sensitivities. This leads to a "deceptive fairness", where equalizing a system metric like average latency across clients with different needs can result in a profoundly unequal user experience.

For instance, consider a math-solving request and a simple daily question both served with 3s latency. While this appears to be "fair" on the system side, the delay is acceptable for chained reasoning for the first but creates poor experience for the second, leading to a huge gap of satisfaction between different users.

In this paper, we propose ExFairS, a request scheduling method that jointly considers resource usage and user experience. We introduce the SLO-Aware Fairness Index (SAFI), which reflects each client's status based on these two factors. To further improve system-wide fairness, we incorporate a credit mechanism: clients with divergent SAFI engage in resource exchanges mediated by credits—borrowing resources earns credits, while consuming resources spends them. Clients with higher credits are prioritized in future exchanges. This dynamic adjustment technique operates externally to the inference engine, providing a lightweight and efficient approach to fairness optimization.

In summary, this paper makes the following contributions:

- We propose Experiential Fairness for LLM inference services, shifting from Equality of Opportunity (uniform resource allocation) to **Equity of Outcome** (equitable user experience). To our knowledge, this is the first work to formally define and optimize this user-centric fairness concept in LLM serving.
- We design ExFairS, an efficient scheduling framework that operationalizes Experiential Fairness. ExFairS uses a composite metric unifying user SLO compliance with resource consumption to guide scheduling decisions.
- We introduce a proportional priority enqueueing mechanism that translates client fairness state into scheduling advantage, accelerating requests for underserved users

[*]Corresponding author.

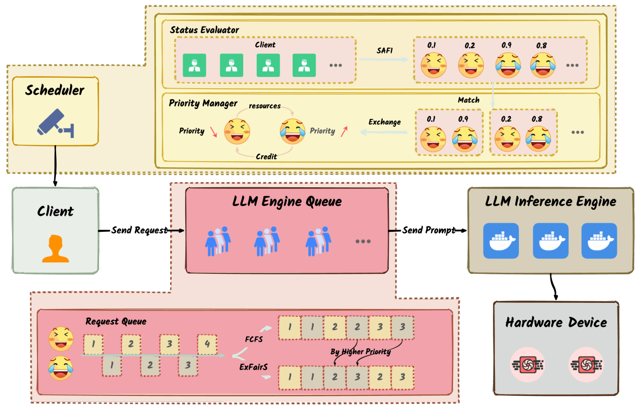

Figure 1: Overview of ExFairS.

while preventing low-priority request starvation.

- We conducted extensive experiments across diverse and challenging workloads. Results demonstrate that ExFairS outperforms state-of-the-art schedulers in both throughput and SLO compliance, validating the practical viability of the Experiential Fairness paradigm.

Background

This section introduces LLM serving background and reviews existing fairness work, highlighting the limitations and challenges of prior approaches.

Related Work

SLO Optimization in LLM Inference LLM inference services must meet strict SLOs to ensure user experience. Existing methods address SLO management at various layers: QLM (Patke et al. 2024) predicts queueing delays for workload control, PLANCK (Lin et al. 2024) allocates fine-grained SLO budgets across pipeline stages, SCOOT (Cheng et al. 2025) tunes engine parameters via Bayesian optimization, the Past-Future Scheduler (Gong et al. 2025) balances batching and memory usage, T-Vaccine (Liu et al. 2024) improves safety alignment efficiency, and ServerlessLLM (Fu et al. 2024) mitigates cold-start latency through multi-tier checkpoint loading.

Fairness Optimization in Online Services Fairness optimization spans multiple domains. In LLM inference, VTC (Sheng et al. 2024) and DLPM (Cao et al. 2025) approximate max-min fairness by prioritizing low-service clients, with DLPM adding flexibility for higher cache efficiency. In recommendation systems, P-MMF (Xu et al. 2023) and (Singh, Kempe, and Joachims 2021) enhance exposure fairness and model uncertainty-aware value fairness. For online scheduling, PFAPPO (Sun et al. 2024) employs fairness-aware reinforcement learning, and FAIR k-FOOD (Singh, Kumar, and Chakraborty 2024) ensures equitable provider income. These works balance fairness, efficiency, and utility across contexts.

Challenge

Existing research on LLM inference services primarily focuses on system-level metrics—fairness, SLO compliance, throughput (Sheng et al. 2023; Park and Egger 2024; Yang et al. 2024b; Patel et al. 2024), cost (Xia et al. 2023; Pan et al. 2024, 2025), and resource utilization (Oh et al. 2024; He et al. 2024)—treating them as core optimization targets. However, this paradigm creates a critical gap between measured performance and actual user-perceived quality. Bridging this gap by creating a truly user-centric scheduling methodology presents three fundamental challenges:

- **Quantifying User Experience**: Standard metrics like average latency are insufficient for streaming LLM services. A robust metric must be sensitive to multi-dimensional factors (e.g., Time-To-First-Token) and, most importantly, to tail latency, which greatly affects user satisfaction.
- **Unifying Conflicting Objectives**: Potential conflicts exist between user-centric indicators (e.g., low latency) and system-centric goals (e.g., high throughput). This requires designing a unified utility function that explicitly encodes the trade-offs between them.
- **Designing an Efficient Online Algorithm**: Finally, a robust online scheduling algorithm is needed to optimize this unified function, with its efficacy proven through both rigorous experimentation and theoretical analysis.

Experiential Fairness

In this section we elaborate on why we need a new criterion of fairness and how it is derived from resource usage and user experience measures.

Necessity for a New Criterion

Fairness is a critical principle for multi-tenant LLM inference services, yet conventional approaches suffer from an ”experience-blindness” flaw. We argue that true fairness is relative; the quality of a user’s experience can only be judged against their specific and often diverse SLO. This principle reframes the scheduling objective: instead of enforcing a single performance standard for all, a fair system must strive to provide a uniformly high degree of SLO fulfillment to each individual user. Conventional methods, however, remain system-centric, mistakenly equating fairness with the uniform allocation of resources while critically ignoring this relativity. For instance, an urgent interactive query and a background batch job might consume identical resources, but the outcome is a failure for the latency-sensitive user while being a success for the other. This leads to a ”deceptive fairness,” where system metrics appear balanced, yet the actual, relative user experience remains profoundly unequal.

Measuring the Equity of Outcome

To address the aforementioned shortcoming, we propose a new fairness standard based on a composite metric that unifies user experience and resource consumption.

Algorithm 1: Resource Exchange by Credit

Input: A set of active clients $\mathcal{C}$
Parameter: Fairness threshold β
Output: Updated credit scores and resource quotas for all clients $\mathcal{C}$

```
1:  $\mathcal{L} \leftarrow$ Sort($\mathcal{C}$) by $SAFI$ (desc) and Credit (desc)
2:  $head \leftarrow 0$
3:  $tail \leftarrow |\mathcal{L}| - 1$
4:  while $head < tail$ do
5:      $client_H \leftarrow \mathcal{L}[head]$
6:      $client_L \leftarrow \mathcal{L}[tail]$
7:      if $client_H.SAFI - client_L.SAFI \geq \beta$ then
8:          $\Delta \leftarrow client_H.SAFI - client_L.SAFI$
9:          $R \leftarrow$ round($\Delta \times 5$)
10:         $client_H.credit \leftarrow client_H.credit - R$
11:         $client_H.resource \leftarrow client_H.resource + R$
12:         $client_L.credit \leftarrow client_L.credit + R$
13:         $client_L.resource \leftarrow client_L.resource - R$
14:         $head \leftarrow head + 1$
15:         $tail \leftarrow tail - 1$
16:     else
17:         break
18:     end if
19: end while
20: return Updated $\mathcal{C}$
```

Resources Usage Quantifying per-request resource usage in LLM inference services presents a significant challenge, as fine-grained attribution of GPU consumption is considerably more complex than monitoring conventional metrics such as CPU utilization. To address this complexity, we adopt total token volume as a practical and robust proxy for computational load. This approach is well-grounded in the near-constant per-token processing cost inherent to Transformer architectures. Following the VTC (Sheng et al. 2024) cost model, we assign greater weight to output tokens to reflect their higher computational overhead—an asymmetry that is also reflected in commercial API pricing schemes such as OpenAI's:

$$Service_i = Token_{Input_i} + 2 \times Token_{Output_i} \tag{1}$$

To derive a normalized measure of resource usage, we divide each client's service value by the system-wide maximum:

$$Service_{Usage_i} = \frac{Service_i}{\max_j Service_j} \tag{2}$$

This formulation provides a lightweight and scalable way to approximate relative resource consumption across clients.

User Experience For online services, latency serves as a core determinant of user experience, with SLOs typically expressed as latency thresholds. Rather than relying on average-based metrics, we adopt a more sensitive measure of user dissatisfaction: the SLO violation rate. This rate represents the proportion of requests that are deemed "SLO-violating"—meaning their total response latency exceeds the predefined threshold—as formally defined in Eq.3.

$$SLO_i = \frac{Count_{SLOV}}{Count_{ALL}} \tag{3}$$

The fundamental rationale for this choice is that the latter metric exhibits greater sensitivity to tail latency events, which are particularly detrimental to user experience. In essence, a user's ultimate satisfaction is often determined by their worst interactions rather than overall average performance.

$$SAFI_i = \alpha(SLO_i) + (1 - \alpha)(Service_{Usage_i}) \tag{4}$$

SLO-Aware Fairness Index With explicit definitions for user experience and resource usage established, we unify them into a linear weighted metric termed the SLO-Aware Fairness Index (SAFI), as defined in Eq. 4. This design is driven by two key considerations. First, the linear formulation ensures simplicity and interpretability, expressing the final score as a transparent combination of two independent factors—experience and consumption—without complex nonlinear interactions. Second, given that the scheduler operates online at high frequency, the lightweight computation involving only two multiplications and one addition enables real-time decision-making with minimal computational overhead.

SAFI thus provides a holistic and quantitative assessment of a user's service state. A higher SAFI value indicates a less favorable condition, attributed to either degraded user experience or disproportionate resource consumption.

Definition of Experiential Fairness

With SAFI established as a tailored metric for user status, we can now formally define Experiential Fairness.

By design, SAFI values approaching zero signify more desirable user states (in terms of both experience and resource consumption), whereas values approaching one indicate deteriorating conditions. We therefore assert that true Experiential Fairness is achieved not by equalizing any single metric, but by minimizing the disparity in SAFI values across the entire user population.

Based on SAFI, we can now establish a formal guarantee for Experiential Fairness. Rather than merely describing an idealized state, we present the following theorem, which establishes a key theoretical property of our scheduling policy.

Theorem 1. *A scheduling system governed by the ExFairS ensures that for any small, predefined fairness threshold $\beta > 0$, the state of the system will converge such that for any two active clients, i and j, in the set of clients $\mathcal{C}$.*

$$|SAFI_i - SAFI_j| < \beta, \quad \forall i, j \in \mathcal{C} \tag{5}$$

Theorem 1 provides a theoretical guarantee that the variance of user states is bounded, ensuring that the service is experientially fair. Its empirical validation will be presented in the evaluation section.

Method

This section details ExFairS's core scheduling mechanism, which uses a dynamic credit system to achieve experiential fairness. The mechanism translates client real-time status into scheduling priority, enabling dynamic resource allocation as shown in Figure 1.

Credit-based Resource Exchange

To rectify fairness violations, ExFairS employs a dynamic control mechanism designed to actively regulate and minimize the disparity between client states. The mechanism operates on the principle of disparity reduction, targeting the two clients representing the extremes of the fairness distribution: the one with the highest SAFI (worst status) and the one with the lowest (best status). A corrective resource exchange is triggered only when the gap between these two extremes exceeds the predefined fairness threshold, β. This targeted exchange boosts the priority of the underserved client while moderating the over-served, creating a negative feedback loop that continuously drives the system towards a globally equitable state.

Given a matched pair ($client_H$, $client_L$), where $client_H$ has a higher SAFI, we compute an exchange value R to guide the adjustment. Initially, all clients have zero resource quota, and the exchange value is determined by the SAFI gap between the two clients. The value is computed as follows:

$$R = \lfloor \frac{SAFI_H - SAFI_L}{2} \times 10 \rfloor \tag{6}$$

Subsequently, a bidirectional exchange is conducted based on the computed value R. The disadvantaged client ($client_H$) spends R credits to acquire additional resources, while the advantaged client ($client_L$) transfers R units of resources in exchange for credits. This mechanism establishes a micro-economy where clients trade resource surplus for future credits or vice versa, enabling continuous self-balancing of fairness in the system. The full procedure is detailed in Algorithm 1.

Request Serving by Priority

Following the specifics of the resource exchange, we employ a priority-based queue management strategy to rearrange the serving order of requests on the fly.

ExFairS translates a client's resource advantage into service priority via a dynamic reordering mechanism within the waiting queue. The priority of each request is determined by the client's accumulated resources as calculated in Algorithm 1. Clients with higher resource allocations receive correspondingly higher scheduling priorities. A critical limitation of naive priority queuing, however, is its propensity for low-priority request starvation. Our approach is explicitly designed to circumvent this issue. Instead of advancing high-priority requests to the absolute front of the queue, we introduce a proportional priority insertion strategy that guarantees forward progress for all requests, with the full procedure detailed in Algorithm 2.

The core of this strategy is a multi-stage calculation to decide where a new request should stand in the queue. First, the algorithm assesses the request's relative standing by computing a normalized priority rank based on the current distribution of priorities in the queue. The number of positions the request advances is then calculated as a fraction of the total overtakable, lower-priority requests, where this fraction is determined by its relative priority rank. This ensures that higher-priority requests gain a significant—yet not absolute—advantage. Finally, this advancement is constrained by tunable parameters (a multiplier and a maximum positional limit) to provide fine-grained control over the policy's aggressiveness and ensure system stability.

Algorithm 2: Proportional Priority Enqueueing Strategy

Input: A new request req_{new}, the current queue $\mathcal{Q}$, the priority distribution cache $\mathcal{C}$
Parameter: Insert multiplier M, Max forward positions P_{max}
Output: Updated $\mathcal{Q}'$, $\mathcal{C}'$

```
1:  if $\mathcal{C}$ is empty then
2:      Insert $req_{new}$ at the end of $\mathcal{Q}$.
3:  else
4:      $p_{new} \leftarrow req_{new}.priority$
5:      $\mathcal{P}_{unique} \leftarrow$ sorted($\mathcal{C}$.keys())
6:      $N_h \leftarrow |\{p \in \mathcal{P}_{unique} \mid p < p_{new}\}|$
7:      $N_{total} \leftarrow |\mathcal{P}_{unique}|$
8:      if $p_{new} \notin \mathcal{P}_{unique}$ then
9:          $N_{total} \leftarrow N_{total} + 1$
10:     end if
11:     $r \leftarrow N_h / N_{total}$ {Calculate relative priority rank}
12:     $N_o \leftarrow \sum_{p,count \in \mathcal{C}.\text{items() if } p > p_{new}} count$
13:     $P_f \leftarrow 0$
14:     if $N_o > 0$ then
15:         $A \leftarrow 1.0 - r$
16:         $P_b \leftarrow \text{floor}(N_o \times A)$
17:         $P_f \leftarrow \min(P_b \times M, P_{max}, N_o, |\mathcal{Q}|)$
18:     end if
19:     $insert\_pos \leftarrow \max(0, |\mathcal{Q}| - P_f)$
20:     Insert $req_{new}$ into $\mathcal{Q}$ at position $insert\_pos$.
21: end if
22: Update cache $\mathcal{C}$ with $p_{new}$.
23: return Updated $\mathcal{Q}$ and $\mathcal{C}$.
```

A key property of this strategy is its formal guarantee against starvation, which we state as the following theorem:

Theorem 2 (Starvation Avoidance Guarantee). *For any request req in the queue $\mathcal{Q}$ at position k, Algorithm 1 guarantees that req will be dequeued after a finite number of subsequent enqueue and dequeue operations, provided the system continues to process requests.*

In addition to this formal guarantee, Algorithm 2 offers several other key advantages:

- **Starvation Avoidance** As established in Theorem 2, the bounded nature of positional advancement ensures all requests are eventually served.
- **Dynamic and Context-Aware Prioritization** A request's scheduling advantage, A_i, is determined not by its absolute priority value P_i, but by its relative rank

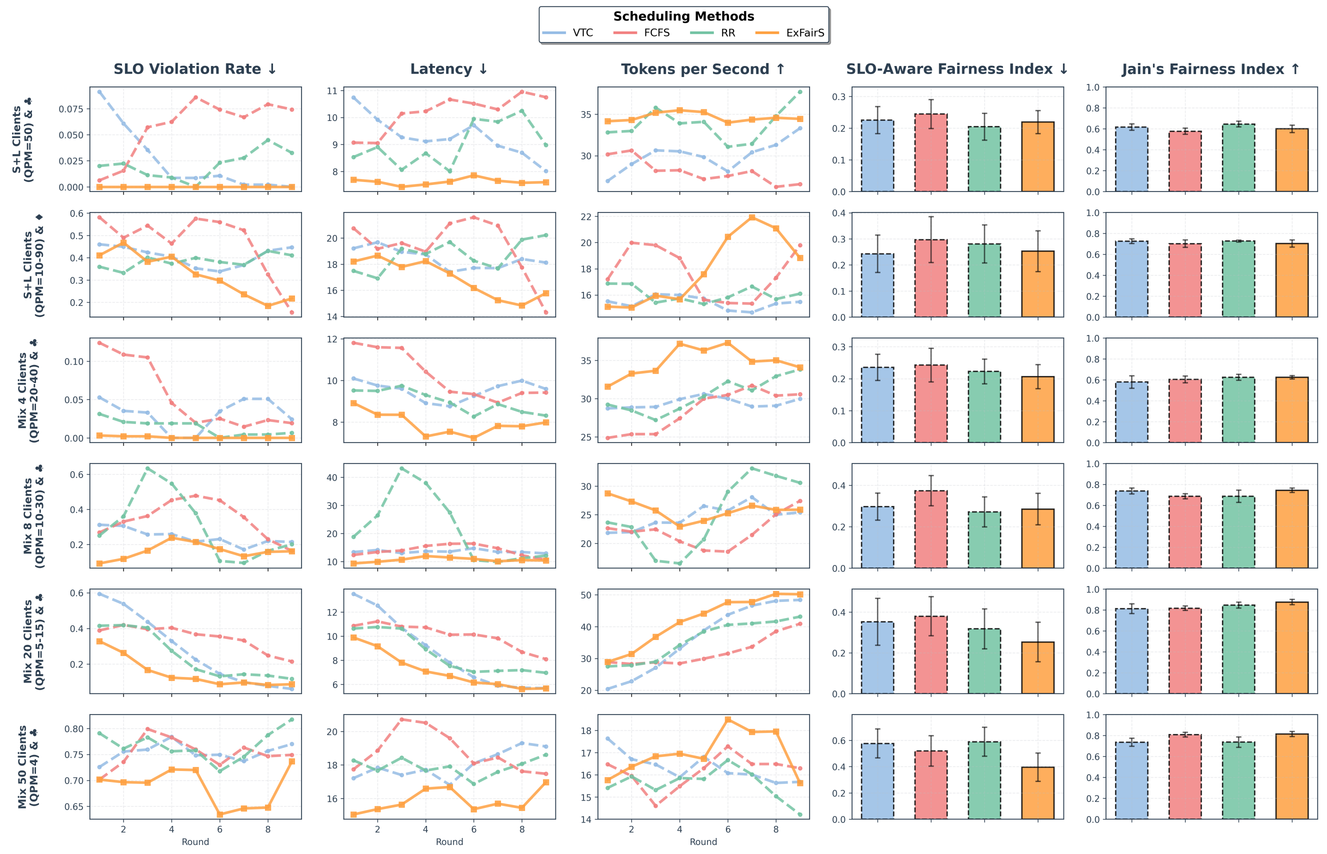

Figure 2: Performance Comparison of Scheduling Algorithms Across Different Workloads (α = 0.7 in Eq. 4). Line plots smoothed with 3-point moving average. **S**(Short Input) **L**(Long Input) **Mix**(S & L Input) **QPM**(Querys Per Minute) ↑(Higher is Better) ↓(Lower is Better) ♣(Inconsistent SLO) ♦(Consistent SLO)

$Rank(P_i)$ within the current queue state Q, ensuring adaptivity to workload fluctuations.

- **Proportional Fairness** Provides nuanced fairness where the positional advancement ΔPos_i is proportional to the product of a request's relative priority rank R_i and the number of overtakable lower-priority requests $N_{<P_i}$.
- **Configurable Granularity of Control** Offers fine-grained control via tunable hyperparameters. The final positional jump is the minimum of the calculated proportional advancement (scaled by a multiplier M) and a hard cap $P_{\max}$, allowing administrators to balance policy aggressiveness versus fairness.

Evaluation

In this section, we evaluate ExFairS against baseline methods across multiple scenarios to validate its effectiveness in optimizing user experience and system efficiency.

Experimental Setup

Implementation We implement our ExFairS in Python and tested in real testbed on top of `vLLM` 0.8.1 (Kwon et al. 2023), an industrial-grade LLM inference engine. Note that our ExFairS is pluggable, allowing seamless compatibility across all LLM inference frameworks.

Tunable Parameters To validate ExFairS robustness, we evaluate its performance across six experimental scenarios designed to simulate diverse conditions, from balanced workloads to complex high-concurrency heterogeneous environments. Configuration details are presented in Table **??**. For the fairness threshold β in Theorem 1, we set β = 0.1, empirically determined to balance scheduling stability and responsiveness to fairness imbalances.

Models and Hardware We conduct our experiments using Llama-3.1-8B (Patterson et al. 2022) and Qwen2.5-32B (Yang et al. 2024a) as the inference models. The hardware platform consists of a single machine equipped with eight NVIDIA V100 GPUs, each with 32 GB of memory.

Workloads We construct our request dataset using ShareGPT-4o (OpenGVLab 2024) for short requests and LongBench (Bai et al. 2024; Trivedi et al. 2022; Kočiský et al. 2017) for long requests, creating a mixed workload with varying lengths.

Baselines We compare ExFairS with three baseline scheduling algorithms:

Scenario	Configuration	Description	# Clients	Task Sizes (QPM)	SLO (sec.)
I	S+L Clients (QPM=50)	Balanced Load	4 (2S+2L)	50, 50, 50, 50	20, 30
II	S+L Clients (QPM=10-90)	Imbalanced Load	4 (2S+2L)	10, 90 (Polarized)	20
III	Mix 4 Clients (QPM=20-40)	Heterogeneous	4 Mix	20, 40	15, 20
IV	Mix 8 Clients (QPM=10-30)	Heterogeneous	8 Mix	10, 20, 30	10, 15, 20, 25
V	Mix 20 Clients (QPM=5-15)	High Concurrency	20 Mix	5, 8, 12, 15	5, 8, 10, 12, 15, 20
VI	Mix 50 Clients (QPM=4)	High Concurrency	50 Mix	4 (Uniform)	8, 10, 12, 15, 20

Table 1: Overview of the Six Experimental Scenarios and Their Configurations

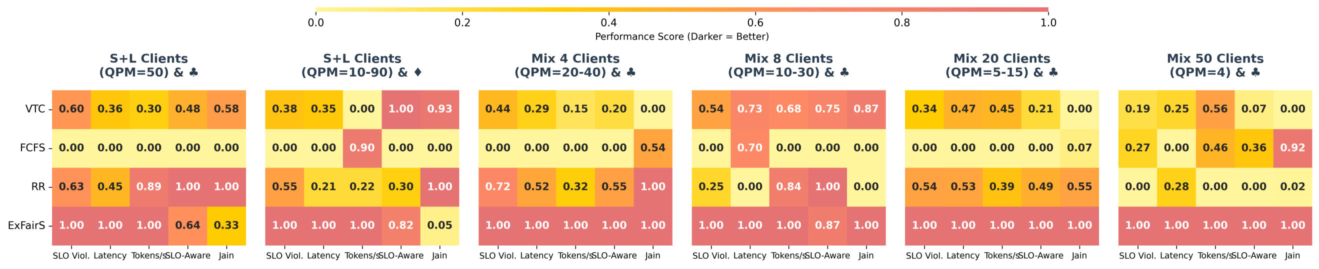

Figure 3: Normalized Performance Score Matrix: Quantitative Assessment of Scheduling Algorithm Effectiveness Across Varied Client Configurations and Request Patterns. **S**(Short Input) **L**(Long Input) **Mix**(S & L Input) **QPM**(Querys Per Minute) ♣(Inconsistent SLO) ♦(Consistent SLO) **SLO Viol.**(Slo Violation Rate) **Jain**(Jain Index)

- **FCFS** A scheduling policy that processes requests strictly in the order of their arrival.
- **Round Robin (RR)** A strategy that distributes requests sequentially across available servers.
- **VTC (Sheng et al. 2024)** A scheduling algorithm that maintains a service counter for each client and prioritizes requests from the client with the lowest counter value.
- **Adapted QoE-MinComp** Adapted from max-min fairness principles in (Feng et al. 2023).

These baselines span different scheduling paradigms: FCFS as the naive default, Round Robin for simple fairness, VTC for state-of-the-art resource-aware scheduling, Adapted QoE-MinComp for theoretical comparison (used primarily for convergence comparison due to utility metric adaptations and implementation complexity), and QoE for QoE-driven resource adjustment. Most existing scheduling approaches emphasize system performance, making them less suitable for our user-centric evaluation.

Metrics To validate the effectiveness of our results, we employ the following key metrics for comparison.

- **SLO Violation Rate**: The proportion of requests that violate SLO constraints within a given time window, normalized by the number of completed requests. This metric reflects client-side user experience.
- **Latency**: The p99th percentile of request response time. It is used in conjunction with the SLO Violation Rate as a core standard for measuring user experience.
- **Tokens Per Second**: Measures system-level throughput and overall performance.
- **SAFI**: As defined earlier, this metric captures the service status of individual clients.
- **Jain's Index (Jain et al. 1984)**: Calculated based on SAFI across clients to quantify system-wide fairness.

Performance under Workload Imbalance

Scenarios I and II evaluate robustness against varying workload structures. In Scenario I (balanced load), ExFairS demonstrates superior performance, achieving zero SLO violations as shown in Figure 2.

ExFairS's robustness is most prominently highlighted in Scenario II (imbalanced load), which is specifically designed to create high contention. FCFS suffers from severe head-of-line blocking, resulting in high and erratic SLO violation rates. In contrast, ExFairS's SAFI metric immediately identifies the deteriorating experience of starved clients, while its credit-exchange mechanism dynamically elevates their priority. This proactive intervention enables ExFairS to maintain a perfect normalized score of 1.00 for both SLO Violation and Latency metrics in the heatmap and exhibit the lowest performance curves in the time-series analysis.

Performance under Heterogeneous Service Levels

Scenarios III and IV introduce client heterogeneity to simulate multi-tenant environments with diverse SLOs. Results in Figure 3 and Figure 2 clearly demonstrate that ExFairS's superiority becomes increasingly pronounced as heterogeneity and contention intensify from Scenario III to the more complex Scenario IV.

Figure 3 reveals RR's catastrophic latency performance failure (score of 0.00) in Scenario IV. Figure 2 explains this failure: RR's latency spikes dramatically as its simplistic equal-turn policy cannot differentiate between clients with varying SLO requirements. In stark contrast, ExFairS main-

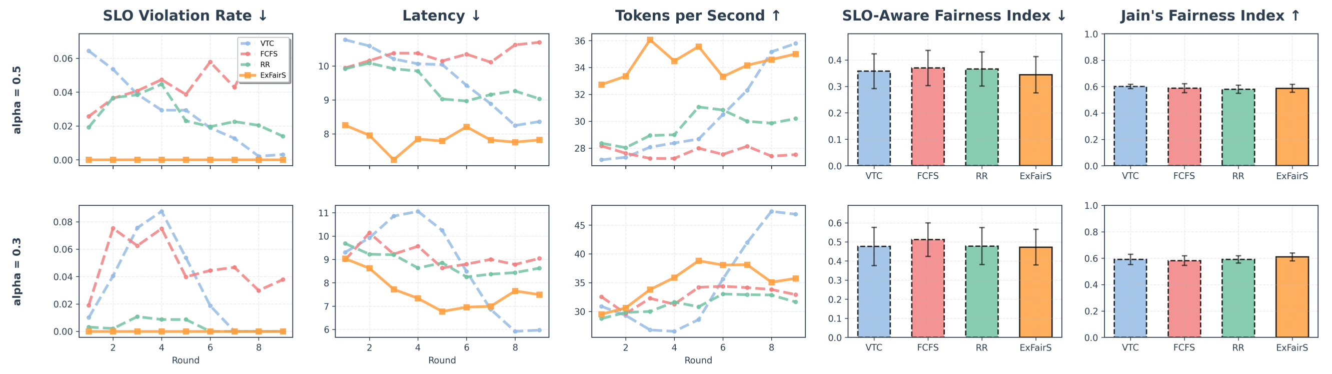

Figure 4: Sensitivity Analysis of Weighting Factor α in Eq. 4. Line plots smoothed with 3-point moving average.

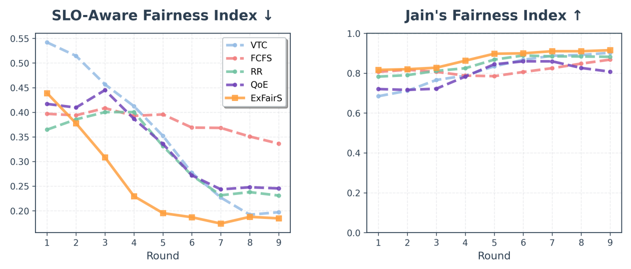

Figure 5: SAFI Convergence. All time-series data is smoothed with a moving average (window size=3).

tains the lowest and most stable latency. This directly validates its core design philosophy, where SAFI's explicit incorporation of SLO constraints enables intelligent prioritization of clients with stricter service requirements.

Scalability and Stability under High Concurrency

The final experiments, Scenarios V and VI, assess system scalability and stability under extreme concurrent loads. Under Scenario V's 20-client diverse workload, the time-series data in Figure 2 showcases ExFairS's rapid convergence, providing empirical validation of our theoretical guarantees. The corresponding SAFI bar chart demonstrates ExFairS achieving a significantly superior (lower) score.

Even under Scenario VI's extreme 50-client "thundering herd" configuration, where all systems experience immense stress, ExFairS maintains its performance advantage. The heatmap reveals that while FCFS achieves a high Jain's Index score, its core performance metrics approach zero, indicating catastrophic system failure. ExFairS, conversely, sustains the highest possible scores for both SLO Violation and Latency metrics. This demonstrates that its proportional and bounded priority mechanisms effectively prevent system oscillations, ensuring stability even under the most severe workload conditions.

Convergence Analysis

To demonstrate dynamic convergence, we analyze the 20-client high-contention Scenario V, including QoE, a strong baseline adapted from max-min QoE principles. As shown in Figure 5, ExFairS exhibits rapid convergence to the lowest SAFI value of approximately 0.17, outperforming all baselines. FCFS shows minimal convergence, while RR and VTC converge slowly to inferior states. Notably, QoE also shows downward trends but stabilizes at a significantly higher SAFI value than ExFairS, highlighting our superior fairness formulation. ExFairS achieves this optimal experiential fairness while maintaining one of the highest Jain's Fairness Index scores, confirming its holistic fairness capability.

Analysis of the Weighting Factor α

Evaluating the robustness of ExFairS with respect to its weighting factor α, we conducted a comprehensive sensitivity analysis specifically within the Scenario III. Figure 4 presents the results, which demonstrate that ExFairS consistently outperforms all baseline methods across different α configurations in this challenging scenario.

The analysis reveals a clear trade-off: lower α values reduce emphasis on user experience, resulting in moderated yet superior SLO compliance compared to baselines. This finding validates our selection of α=0.7 as the default setting, which effectively aligns with our user-centric objective of optimizing experiential fairness.

Conclusion

In conclusion, this paper challenges the conventional, system-centric definition of fairness in the context of LLM inference serving. We introduce and formularize a new notion, Experiential Fairness, which fundamentally shifts the scheduling objective from equality of resource allocation to equity of user-perceived experience. We developed ExFairS, a novel scheduling framework that uses a unified metric to co-optimize for user SLO compliance and system efficiency. Extensive experiments on real testbed demonstrate that our approach is highly effective, reducing SLO violations by up to 100% while simultaneously increasing system throughput by 14–21.9%. These results not only validate ExFairS as a high-performance scheduler but, more importantly, establish the practical viability of prioritizing experiential equity, proving that user-centric fairness and system efficiency can be achieved in concert.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (62402198), the Guangdong Provincial Natural Science Foundation (2025A1515010113), the Shandong Provincial Natural Science Foundation (ZR2024LZH012), the Guangxi Key Research and Development Project (2024AB02018), and the Major Key Project of PCL (PCL2025AS11).

References

Bai, Y.; Lv, X.; Zhang, J.; Lyu, H.; Tang, J.; Huang, Z.; Du, Z.; Liu, X.; Zeng, A.; Hou, L.; Dong, Y.; Tang, J.; and Li, J. 2024. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. arXiv:2308.14508.

Cao, S.; Wang, Y.; Mao, Z.; Hsu, P.-L.; Yin, L.; Xia, T.; Li, D.; Liu, S.; Zhang, Y.; Zhou, Y.; et al. 2025. Locality-aware Fair Scheduling in LLM Serving. *arXiv preprint arXiv:2501.14312*.

Cheng, K.; Wang, Z.; Hu, W.; Yang, T.; Li, J.; and Zhang, S. 2025. SCOOT: SLO-Oriented Performance Tuning for LLM Inference Engines. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, 829–839. New York, NY, USA: Association for Computing Machinery. ISBN 9798400712746.

Feng, J.; Liu, L.; Hou, X.; Pei, Q.; and Wu, C. 2023. QoE Fairness Resource Allocation in Digital Twin-Enabled Wireless Virtual Reality Systems. *IEEE Journal on Selected Areas in Communications*, 41(11): 3355–3368.

Fu, Y.; Xue, L.; Huang, Y.; Brabete, A.-O.; Ustiugov, D.; Patel, Y.; and Mai, L. 2024. ServerlessLLM: Low-Latency Serverless Inference for Large Language Models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 135–153. Santa Clara, CA: USENIX Association. ISBN 978-1-939133-40-3.

Gong, R.; Bai, S.; Wu, S.; Fan, Y.; Wang, Z.; Li, X.; Yang, H.; and Liu, X. 2025. Past-Future Scheduler for LLM Serving under SLA Guarantees. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '25, 798–813. New York, NY, USA: Association for Computing Machinery. ISBN 9798400710797.

He, Y.; Fang, J.; Yu, F. R.; and Leung, V. C. 2024. Large Language Models (LLMs) Inference Offloading and Resource Allocation in Cloud-Edge Computing: An Active Inference Approach. *IEEE Transactions on Mobile Computing*, 23(12): 11253–11264.

Jain, R. K.; Chiu, D.-M. W.; Hawe, W. R.; et al. 1984. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 21(1): 2022–2023.

Kočiský, T.; Schwarz, J.; Blunsom, P.; Dyer, C.; Hermann, K. M.; Melis, G.; and Grefenstette, E. 2017. The NarrativeQA Reading Comprehension Challenge. arXiv:1712.07040.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, 611–626. New York, NY, USA: Association for Computing Machinery. ISBN 9798400702297.

Lin, Y.; Peng, S.; Wu, S.; Li, Y.; Lu, C.; Xu, C.; and Ye, K. 2024. Planck: Optimizing LLM Inference Performance in Pipeline Parallelism with Fine-Grained SLO Constraint. In *2024 IEEE International Conference on Web Services (ICWS)*, 1306–1313.

Liu, G.; Lin, W.; Huang, T.; Mo, R.; Mu, Q.; and Shen, L. 2024. Targeted vaccine: Safety alignment for large language models against harmful fine-tuning via layer-wise perturbation. *arXiv preprint arXiv:2410.09760*.

Oh, H.; Kim, K.; Kim, J.; Kim, S.; Lee, J.; Chang, D.-s.; and Seo, J. 2024. ExeGPT: Constraint-Aware Resource Scheduling for LLM Inference. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, 369–384. New York, NY, USA: Association for Computing Machinery. ISBN 9798400703850.

OpenGVLab. 2024. ShareGPT-4o. https://huggingface.co/datasets/OpenGVLab/ShareGPT-4o.

Pan, X.; Li, E.; Li, Q.; Liang, S.; Shan, Y.; Zhou, K.; Luo, Y.; Wang, X.; and Zhang, J. 2024. InstInfer: In-Storage Attention Offloading for Cost-Effective Long-Context LLM Inference. arXiv:2409.04992.

Pan, X.; Li, E.; Li, Q.; Liang, S.; Shan, Y.; Zhou, K.; Luo, Y.; Wang, X.; and Zhang, J. 2025. InstAttention: In-Storage Attention Offloading for Cost-Effective Long-Context LLM Inference. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 1510–1525.

Park, D.; and Egger, B. 2024. Improving Throughput-oriented LLM Inference with CPU Computations. In *Proceedings of the 2024 International Conference on Parallel Architectures and Compilation Techniques*, PACT '24, 233–245. New York, NY, USA: Association for Computing Machinery. ISBN 9798400706318.

Patel, P.; Choukse, E.; Zhang, C.; Shah, A.; Goiri, Í.; Maleki, S.; and Bianchini, R. 2024. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 118–132. IEEE.

Patke, A.; Reddy, D.; Jha, S.; Qiu, H.; Pinto, C.; Narayanaswami, C.; Kalbarczyk, Z.; and Iyer, R. 2024. Queue Management for SLO-Oriented Large Language Model Serving. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, SoCC '24, 18–35. New York, NY, USA: Association for Computing Machinery. ISBN 9798400712869.

Patterson, D.; Gonzalez, J.; Hölzle, U.; Le, Q.; Liang, C.; Munguia, L.-M.; Rothchild, D.; So, D.; Texier, M.; and Dean, J. 2022. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. arXiv:2204.05149.

Sheng, Y.; Cao, S.; Li, D.; Zhu, B.; Li, Z.; Zhuo, D.; Gonzalez, J. E.; and Stoica, I. 2024. Fairness in serving large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 965–988.

Sheng, Y.; Zheng, L.; Yuan, B.; Li, Z.; Ryabinin, M.; Chen, B.; Liang, P.; Re, C.; Stoica, I.; and Zhang, C. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 31094–31116. PMLR.

Singh, A.; Kempe, D.; and Joachims, T. 2021. Fairness in Ranking under Uncertainty. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 11896–11908. Curran Associates, Inc.

Singh, D. D.; Kumar, A.; and Chakraborty, A. 2024. Towards Fairness in Online Service with K Servers and Its Application on Fair Food Delivery. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18): 19929–19936.

Sun, G.; Wang, Y.; Yu, H.; and Guizani, M. 2024. Proportional Fairness-Aware Task Scheduling in Space-Air-Ground Integrated Networks. *IEEE Transactions on Services Computing*, 17(6): 4125–4137.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. arXiv:2108.00573.

Xia, H.; Zheng, Z.; Li, Y.; Zhuang, D.; Zhou, Z.; Qiu, X.; Li, Y.; Lin, W.; and Song, S. L. 2023. Flash-LLM: Enabling Cost-Effective and Highly-Efficient Large Generative Model Inference with Unstructured Sparsity. arXiv:2309.10285.

Xu, C.; Chen, S.; Xu, J.; Shen, W.; Zhang, X.; Wang, G.; and Dong, Z. 2023. P-MMF: Provider Max-min Fairness Re-ranking in Recommender System. In *Proceedings of the ACM Web Conference 2023*, WWW '23, 3701–3711. New York, NY, USA: Association for Computing Machinery. ISBN 9781450394161.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; Lin, H.; Tang, J.; Wang, J.; Yang, J.; Tu, J.; Zhang, J.; Ma, J.; Yang, J.; Xu, J.; Zhou, J.; Bai, J.; He, J.; Lin, J.; Dang, K.; Lu, K.; Chen, K.; Yang, K.; Li, M.; Xue, M.; Ni, N.; Zhang, P.; Wang, P.; Peng, R.; Men, R.; Gao, R.; Lin, R.; Wang, S.; Bai, S.; Tan, S.; Zhu, T.; Li, T.; Liu, T.; Ge, W.; Deng, X.; Zhou, X.; Ren, X.; Zhang, X.; Wei, X.; Ren, X.; Liu, X.; Fan, Y.; Yao, Y.; Zhang, Y.; Wan, Y.; Chu, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; Guo, Z.; and Fan, Z. 2024a. Qwen2 Technical Report. arXiv:2407.10671.

Yang, D.; Han, X.; Gao, Y.; Hu, Y.; Zhang, S.; and Zhao, H. 2024b. PyramidInfer: Pyramid KV Cache Compression for High-throughput LLM Inference. arXiv:2405.12532.