SCHEDULING MIXED RL ROLLOUTS BEYOND PREFIX LOCALITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern reinforcement learning (RL) post-training pipelines for large language models (LLMs) increasingly combine rollout workloads across multiple domains and feedback paradigms. Prefix-aware routing improves inference efficiency through cache reuse and load balancing, but it does not control how heterogeneous rollout sessions compete for KV-cache capacity. When reinforcement learning with verifiable rewards (RLVR), reinforcement learning from human feedback (RLHF), and agentic rollouts share an asynchronous inference service, their distinct sequence structures, interaction patterns, and KV-residency times create substantially different serving demands. Rollout scheduling must account for this heterogeneity without distorting the workload mixture specified by the trainer. We present **MISA-T**, a routing-layer admission policy for mixed rollout serving. MISA-T combines adaptive session admission, workload-aware KV-capacity allocation, and residency-time-aware KV accounting. In rollout-only ablations on Step3.7 and Qwen3.6-35B-A3B, MISA-T improves rollout throughput over a sweep-tuned cache-aware vLLM Router by 53.3% and 43.6%, respectively, while maintaining high prefix-cache hit rates. In a matched 50-iteration Step3.7 experiment, it increases rollout throughput by 35.6% and reduces mean iteration time by 22.8%, while keeping the consumed workload mixture close to the trainer target and achieving comparable task scores.

1 INTRODUCTION

Modern RL post-training for large language models increasingly treats rollout generation as a distributed serving workload. A trainer releases rollout tasks, and a router assigns their model requests to inference instances. Prefix reuse reduces redundant prefill, especially for agentic trajectories whose later turns contain long accumulated interaction histories (Kang et al., 2026; Zhu et al., 2026).

The scheduling problem becomes more complex when a rollout stream contains heterogeneous workloads. Such streams arise in multi-domain RL (Yang et al., 2025a; Liang et al., 2025) and in multi-teacher on-policy distillation stages that integrate domain specialists (Ma et al., 2026; Yang et al., 2026a; Li et al., 2026; Kimi Team, 2026; Yang et al., 2026b). These workloads differ in decode length, prefix reuse, and KV residency: in our trace, RLVR has long decode tails, RLHF has shorter responses, and agentic sessions retain prefixes across model turns and tool/environment interactions. These are serving classes, not mutually exclusive reward mechanisms. Prompt fanout can also introduce completion dependencies among samples from the same prompt (Yu et al., 2026).

The trainer releases rollout tasks subject to staleness and maximum-concurrency limits, whereas the router controls serving-side admission and instance selection. Prefix-aware placement alone does not bound the number of sessions competing for KV capacity or determine how that capacity should be shared across workloads. Admission must account for these differences without changing the trainer's target workload composition.

We formulate this setting as mixed-rollout routing under trainer-side mixture constraints and propose MISA-T (**M**ix-aware **S**ession **A**dmission with a **T**ime factor). The contributions are:

1. **Overload-aware session admission.** We adapt admission to observed KV demand and cache pressure, protecting continuations and holding excess demand to reduce repeated static-concurrency tuning.

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

2. **Workload-aware capacity allocation.** We allocate protected capacity according to class demand and KV footprint, accounting for heterogeneous sessions without changing their workload identities.
3. **Residency-time-aware KV accounting.** We weight demand by estimated KV residency across model execution and tool/environment interactions to account for how long sessions require protected capacity.

Across two rollout-only model families, MISA-T improves rollout throughput by up to 53.3% over a sweep-tuned vLLM Router baseline. In matched end-to-end training, it improves rollout throughput by 35.6% and reduces mean iteration time by 22.8%.

2 Background and Motivation

2.1 Mixed rollout serving

Figure 1 illustrates a common architecture for distributed RL rollout generation. In fully asynchronous systems, rollout generation overlaps model training, subject to trainer-side maximum-concurrency and policy-version staleness limits (Fu et al., 2025; Yan et al., 2025; Hu et al., 2026). Rollout workers issue model requests; agent servers additionally maintain interaction loops and execute tool actions in isolated sandboxes. Between model turns, an agent session may be executing a tool rather than issuing an inference request, yet its cached prefix remains useful for the next turn. The routing layer controls admission and instance selection, leaving model execution, physical KV allocation, and the agent runtime unchanged.

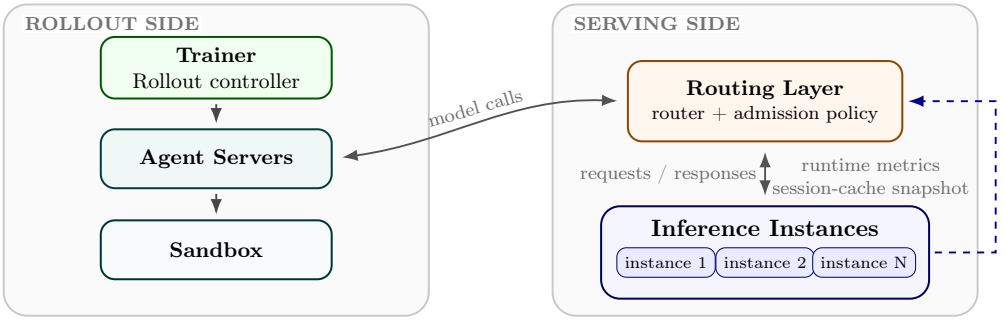

Figure 1: System overview. MISA-T is a routing-layer policy. It can be deployed with the router and scheduler as separate services or as a gateway plugin, and depends only on request metadata, runtime metrics, and session-cache snapshots from the serving layer.

Joint multi-domain RL samples heterogeneous prompts within one on-policy process (Yang et al., 2025a; Liang et al., 2025). Integration through distillation can produce a similar stream: MOPD generates student rollouts from multi-domain prompts and routes trajectories to matching domain teachers (Ma et al., 2026). Related integration stages appear in MiMo-V2-Flash (Xiaomi LLM-Core Team, 2026), Kimi K3 (Kimi Team, 2026), Nemotron 3 Ultra (NVIDIA, 2026), and Nemotron-Cascade 2 (Yang et al., 2026b). Across these settings, the trainer defines the sampling distribution, while the serving layer responds to request-time cache state and queueing pressure.

2.2 Prefix-aware routing

PagedAttention manages KV memory efficiently (Kwon et al., 2023), and RadixAttention enables prefix reuse (Zheng et al., 2024). Scheduling policies balance reuse against load (Srivatsa et al., 2025), fairness (Cao et al., 2025), or latency (Dexter et al., 2025). The cache-aware policy in vLLM Router favors prefix matches under balanced load and lower-load instances under imbalance (vLLM Project, 2026). It controls placement, not how many new sessions enter an instance. These routing policies do not allocate admission capacity by workload-class KV block-time demand, leaving heterogeneous sessions to compete for reusable cache capacity. Program-aware scheduling goes beyond request placement: ThunderAgent uses a space-time cost model to guide program pausing,

restoration, and cross-instance balancing (Kang et al., 2026). Our focus is allocating protected admission capacity across mixed rollout classes using their outstanding demand, KV footprint, and KV-residency time.

2.3 Workload evidence and design insight

Table 1 quantifies request-level token differences in the online mixed-rollout trace with a 128K-token context limit. Agent sessions reuse long prefixes across tool-interleaved turns, consistent with studies of agentic serving (Kang et al., 2026; Zhu et al., 2026), whereas RLVR samples are decode-heavy, as in long-output reasoning workloads (Du et al., 2025). These measurements characterize the observed workload, not the training algorithms in general.

Table 1: Request-level token statistics under a 128K context limit.

Workload	Requests	Sessions	Input p50	Input p95	Input p99	Output p50	Output p95	Output p99
Agent	73,659	1,058	37,757	103,209	123,207	148	1,612	3,930
RLHF	599	599	60	2,564	2,564	7,344	24,447	37,101
RLVR	809	809	156	1,129	1,277	47,583	130,815	130,958

Tool/environment interaction intervals outside inference account for 6.6% and 22.7% of agent task time on Step3.7 and Qwen3.6-35B-A3B, respectively, so model execution alone understates session residency. Appendix A gives the agent task-state statistics.

Limits of routing alone. On Qwen3.6-35B-A3B, vLLM Router at a concurrency limit of 96 achieves 52.5% lower request throughput than its sweep-tuned configuration at 48, while prefix-cache hit rate falls from 92.4% to 4.5%. New sessions evict reusable prefixes, forcing later turns to repeat prefill; the resulting queueing and session overlap create further eviction pressure. A global session cap can interrupt this loop, but a class-agnostic cap does not account for differences in KV footprint and residency time. Under the same mixed stream and permissive concurrency ceiling, MISA-T improves rollout throughput over class-agnostic Session Admission by 37.3% on Step3.7 and 20.4% on Qwen3.6-35B-A3B. We separate the contributions of workload-specific allocation and residency-time weighting in Section 5.

Admission is a KV commitment. Admitting a session commits capacity for future KV growth and reuse, including intervals between model turns. Larger footprints require more space, while longer residency delays its release. Allocating class quotas by both factors can protect capacity for other workloads; dividing each quota by its reference footprint yields a session cap without changing trainer sampling.

This observation leads to three systems challenges:

1. **Online adaptation:** regulate admission as sequence lengths and demand change, reducing repeated concurrency tuning. Final sequence growth and residency are unknown at admission time, and cache observations arrive asynchronously.
2. **Workload heterogeneity:** account for class-specific footprint and residency without changing the trainer's workload composition.
3. **Composition:** coexist with cache-aware placement and KV offloading without taking over their responsibilities.

3 Problem Formulation

We formalize request placement and session admission under the trainer's workload mixture and protected KV capacity.

3.1 REQUESTS AND PROTECTED CAPACITY

We consider a trainer that releases rollout units from workload classes $\mathcal{B}$ to a shared pool of inference instances $\mathcal{W}$. A rollout unit is one trainer-counted sample; a prompt may fan out into a group of units, and an agentic unit comprises multiple model turns and tool/environment interactions. Each request r inherits a workload class $b_r \in \mathcal{B}$, rollout-unit identity u_r, session identity s_r, and optional group identity g_r from the trainer. Let $A_b^\pi(T)$ be the class-b units released by time T under policy π, and $A^\pi(T) = \sum_b A_b^\pi(T)$. Trainer-side in-flight limits can make release times depend on completion progress; the target mixture remains fixed, with $A_b^\pi(T)/A^\pi(T) \to \rho_b$. The router chooses $\pi_t(r) \in \mathcal{W} \cup \{\text{HOLD}\}$: admit to an instance or delay admission for periodic re-evaluation. It observes request metadata and serving state, but not final sequence growth or residency, and does not change rollout identities or labels.

For session s, let $k_s(t)$ be its retained KV blocks and T_s its effective KV-residency time across model execution and tool/environment interactions. Its KV occupation is naturally measured in block-time,

$$Z_s = \int k_s(t)\, dt \;\approx\; \bar{k}_s T_s, \tag{1}$$

where $\bar{k}_s$ is a reference footprint and agentic residency includes the tool/environment interaction intervals between model turns.

For protected sessions $\mathcal{P}_w(t)$, reserved-footprint estimates $\widetilde{k}_s(t)$, and budget C_w, the admission target is

$$\sum_{s \in \mathcal{P}_w(t)} \widetilde{k}_s(t) \le C_w, \qquad \forall w \in \mathcal{W}. \tag{2}$$

Session caps approximate this target, not a physical allocation bound. If updated estimates exceed the budget, correction restricts new admissions without preempting counted sessions.

3.2 SCHEDULING OBJECTIVE UNDER THE TRAINER CONTRACT

Let $N_b^\pi(T)$ be the number of complete class-b rollout samples delivered to the trainer by time T under policy π. We define rollout throughput as

$$S_\pi(T) = \frac{1}{T} \sum_{b \in \mathcal{B}} N_b^\pi(T). \tag{3}$$

An agentic trajectory counts once after all turns and tool/environment interactions complete, so $S_\pi(T)$ differs from request throughput. Define the class backlog as $Q_b^\pi(T) = A_b^\pi(T) - N_b^\pi(T)$. For a fixed trainer configuration with continuing rollout demand, our design objective is

$$\max_\pi \; \liminf_{T \to \infty} S_\pi(T) \quad \text{subject to} \quad Q_b^\pi(T) = o(A^\pi(T)), \qquad \forall b \in \mathcal{B}. \tag{4}$$

Equation (2) supplies the admission budget target. The backlog condition limits aggregate class imbalance, not the waiting time of every session. Because $N_b^\pi = A_b^\pi - Q_b^\pi$, if $A_b^\pi/A^\pi \to \rho_b$ and $Q_b^\pi/A^\pi \to 0$, then the completed mixture also converges to ρ. This is a conditional mixture property; MISA-T uses backlog-dependent quotas as a heuristic, without a stability or throughput-optimality guarantee. In end-to-end evaluation, we measure $D_{\text{TV}}(p, q) = \frac{1}{2}\|p - q\|_1$ between the trainer-consumed mixture p and target mixture q.

4 MIX-AWARE SESSION ADMISSION

Figure 2 summarizes the three controls: adaptive session admission, workload-aware capacity allocation, and residency-time weighting.

4.1 ADAPTIVE SESSION ADMISSION

For inference instance w, let $K_w(t)$ denote the protected session cap, O_w the active protected sessions, and P_w the pending sessions that have been admitted but are not yet visible in the latest cache

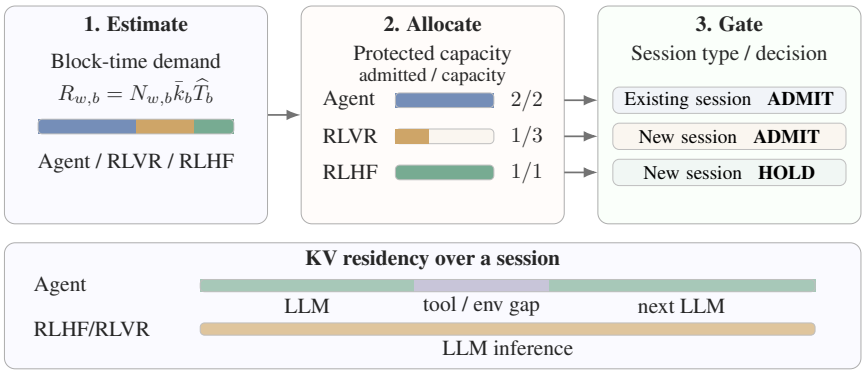

Figure 2: MISA-T scheduling overview. Existing admitted sessions preserve locality, while each additional admitted session commits capacity for future KV growth and reuse.

snapshot:

$$\textsc{Admit}(r, w) = \begin{cases} 1, & s_r \in O_w \cup P_w, \\ 1, & |O_w| + |P_w| < K_w(t), \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Requests whose sessions are already in O_w or P_w continue without additional cap headroom, including after a cap contraction. A request whose session is in neither set must acquire cap headroom; otherwise, the router holds and periodically re-evaluates it. Pending accounting includes admissions not yet in snapshots; held sessions remain in the demand ledger. For grouped rollouts, the recheck delay decreases as more siblings from the same prompt group become visible to the scheduler.

Appendix B.1 analyzes the trade-off between admission delay and prefix reuse.

Let $\bar{k}_w(t)$ be a reference KV footprint in blocks, derived from recent completed sessions and current observations. If C_w KV blocks are reserved for protected locality, the base cap is $K_w^{\text{base}}(t) = \lfloor C_w/\bar{k}_w(t) \rfloor$. The operational cap satisfies $K_w(t) \le K_w^{\text{base}}(t)$. A confirmed prefix-hit deficit or persistent waiting requests without declining prefill backlog trigger overload pressure, temporarily contracting the operational cap before gradual recovery. This feedback corrects observed overload that the reference footprint alone may not capture.

Prefix pressure compares observed hit rate $\hat{h}_w(t)$ against a reference $h_w^\star(t)$ learned from recent healthy windows, rather than a fixed threshold:

$$G_w(t) = \max\{0, h_w^\star(t) - \hat{h}_w(t)\}, \tag{6}$$

Confirmation and slope checks filter transient changes before cap reduction or recovery.

4.2 Workload-aware session caps

A global cap bounds session count but does not distinguish class-specific KV demand, allowing a dominant class to occupy most protected capacity. Adding class-specific soft quotas yields MISA, the variant without residency-time weighting. For inference instance w and workload class b, $N_{w,b}$ is the instantaneous number of distinct unfinished session demands visible when evaluating w, rather than a time-averaged resident population. Sessions are deduplicated across active, pending, held, and candidate state; each contributes one demand count. Held and candidate sessions consume cap headroom only upon admission, when the selected instance records them as pending. $\bar{k}_b$ is the class reference footprint in blocks, estimated from length priors, session observations, and bounded histories.

MISA allocates block quotas in proportion to spatial demand $S_{w,b} = N_{w,b}\bar{k}_b$ and divides each quota by $\bar{k}_b$ to obtain a class cap:

$$M_{w,b}^{\text{space}} = C_w \frac{S_{w,b}}{\sum_{b'} S_{w,b'}}, \qquad K_{w,b}^{\text{space}} = \max\left\{1, \left\lfloor \frac{M_{w,b}^{\text{space}}}{\bar{k}_b} \right\rfloor\right\}. \tag{7}$$

The resulting class caps are soft limits, not exact block partitions after rounding. A minimum cap of one reserves no slot: new or rebound sessions require headroom under both class and global limits.

4.3 KV-CACHE RESIDENCY-TIME WEIGHTING

Footprint-only allocation treats equal KV footprints as equal demand regardless of how long reusable state remains relevant. Let $\widehat{T}_b$ estimate the recent KV-residency time of class b across model execution and, for agentic sessions, tool/environment interactions between turns.

MISA-T replaces spatial demand with the block-time demand proxy

$$R_{w,b} = S_{w,b}\widehat{T}_b = N_{w,b}\bar{k}_b\widehat{T}_b. \tag{8}$$

For block quota $M_{w,b}$, the estimated backlog drain time is

$$\tau_{w,b} \approx \frac{N_{w,b}\bar{k}_b\widehat{T}_b}{M_{w,b}}. \tag{9}$$

Before cap rounding and global-cap enforcement, allocation in proportion to $R_{w,b}$ balances these approximate drain times:

$$M^{\text{time}}_{w,b} = C_w \frac{R_{w,b}}{\sum_{b'} R_{w,b'}}, \qquad K^{\text{time}}_{w,b} = \max\left\{1, \left\lfloor \frac{M^{\text{time}}_{w,b}}{\bar{k}_b} \right\rfloor\right\}. \tag{10}$$

Before rounding, session-cap ratios follow $N_{w,b}\widehat{T}_b$; footprints set block shares and the common capacity scale (Appendix B.2). For example, consider two classes with the same number of unfinished sessions, where class B uses twice the KV capacity and retains it for twice as long as class A. Equation (10) assigns block quotas in a 1:4 ratio and session caps in a 1:2 ratio; over two A-session durations, one A slot completes two A sessions while two B slots complete two B sessions. This illustrative allocation balances approximate class drain times rather than establishing throughput optimality.

The controller estimates $\widehat{T}_b$ from recently completed sessions using a per-class exponentially weighted moving average (EWMA). A shared neutral prior supplies initial weights until each class has sufficient observations. Both footprint and residency estimates follow recent sessions rather than fixed per-label costs.

4.4 SESSION-LEVEL CACHE ACCOUNTING

A session snapshot records running and recently completed sessions, estimated total and cached KV blocks, lifecycle timestamps, and instance capacity. Constructed from session-tagged request events and observed token lengths, it supports continuation detection, occupancy estimates, and admission caps without reading the engine's physical KV-block map.

In our deployment, engine-reported prefix-hash additions and removals separately maintain the router's hash-to-instance mapping. The longest matching prompt-block hash estimates prefix availability from the latest reports; reported eviction or reset invalidates its location. The admission extension filters eligible instances, after which the existing cache-aware policy selects among them. Appendix C summarizes the control sequence and key settings.

5 EVALUATION

We evaluate end-to-end training performance and the contributions of the three scheduling mechanisms.

Serving rates are aggregated over each run's measurement window. RPM counts successful inference requests per minute, while rollout throughput counts complete rollout samples returned to the trainer per minute. Prefix hit rate is the fraction of queried prompt tokens found in the prefix cache. Prompt TPS denotes effective prompt-token throughput, including prefix-hit tokens, rather than tokens processed by prefill kernels alone. Relative changes use the matched vLLM Router run for end-to-end training and sweep-tuned vLLM Router for rollout-only ablations.

5.1 EXPERIMENTAL SETUP

End-to-end RL training. We compare MISA-T with vLLM Router on Step3.7 for 50 iterations under the same model initialization, sampling stream, target mixture, and verifier/reward configuration. Step3.7 is a 196B-A11B sparse-MoE model. The deployment uses six NVIDIA H200 nodes for training and four for rollout inference; each H200 GPU has 141 GB of HBM3e memory, and each inference node runs one TP=8 replica. The vLLM Router baseline uses the best static per-instance concurrency identified by a coarse-grained sweep. MISA-T uses the same outer ceiling and dynamically regulates admitted sessions within it.

Rollout-only ablation. The rollout-only benchmark generates online mixed rollouts at a fixed model checkpoint, isolating serving-side scheduling from weight updates and cache resets. Table 3 reports relative changes computed from arithmetic means over three independent runs per configuration. The admission-controlled variants use the same controller parameters across the two evaluated model and hardware configurations. Step3.7 uses two NVIDIA H200 inference nodes, each running one TP=8 replica; Qwen3.6-35B-A3B uses 16 NVIDIA H100 GPUs with 80 GB of HBM3 memory each, organized as four TP=4 replicas. For the vLLM Router baseline, we perform a coarse static-concurrency sweep with increments of 32 for Step3.7 and 16 for Qwen3.6-35B-A3B; the best observed static concurrency is 128 and 48, respectively. We additionally report a capacity-safe configuration based on the KV-cache capacity required by a 128K-token context. The high-load Qwen3.6-35B-A3B diagnostic gives vLLM Router the same permissive ceiling as the admission-controlled variants, which regulate admitted session counts dynamically within that ceiling.

The ablation compares cache-aware routing with dynamic class-agnostic admission, footprint-based class allocation, and residency-weighted allocation:

1. **vLLM Router:** cache-aware placement without admission control.
2. **Session Admission:** the dynamic global controller in Section 4.1, including prefix/waiting-pressure feedback, without class quotas.
3. **MISA:** session admission that allocates per-class capacity according to session demand and estimated KV-cache footprint.
4. **MISA-T:** MISA with class-level KV-residency-time weighting.

5.2 END-TO-END TRAINING RESULTS

Table 2 reports the matched 50-iteration Step3.7 comparison.

Table 2: End-to-end Step3.7 results over 50 iterations.

	Change vs. baseline (%)						50-iteration mix (%)		
Router	Iter. time	RPM	Sample rate	Prompt TPS	Decode TPS	Prefix hit	Agent	RLVR	RLHF
vLLM Router	0.0	0.0	0.0	0.0	0.0	74.5%	48.3	29.4	22.4
MISA-T	**-22.8**	**+32.5**	**+35.6**	**+33.6**	**+38.0**	**96.2%**	45.9	33.3	20.8

MISA-T increases rollout throughput by 35.6%, reduces mean iteration time by 22.8%, and raises prefix-cache hit rate from 74.5% to 96.2%. These end-to-end rates include periodic weight synchronization: inference replicas reset prefix caches after loading updated weights, and subsequent requests perform cold prefill until reusable state accumulates again.

The target Agent/RLVR/RLHF mixture is $48.6/33.1/18.2\%$. The consumed mixture has a total-variation distance of 2.71 percentage points under MISA-T, compared with 4.14 under vLLM Router.

At the same training horizon, the absolute pass@4 difference between MISA-T and vLLM Router is below 0.5 percentage points on SWE-Bench Pro (Deng et al., 2026), SWE-Verified (Chowdhury et al., 2024), and SWE-bench Multilingual (Yang et al., 2025b). Figure 3 shows smoothed iteration

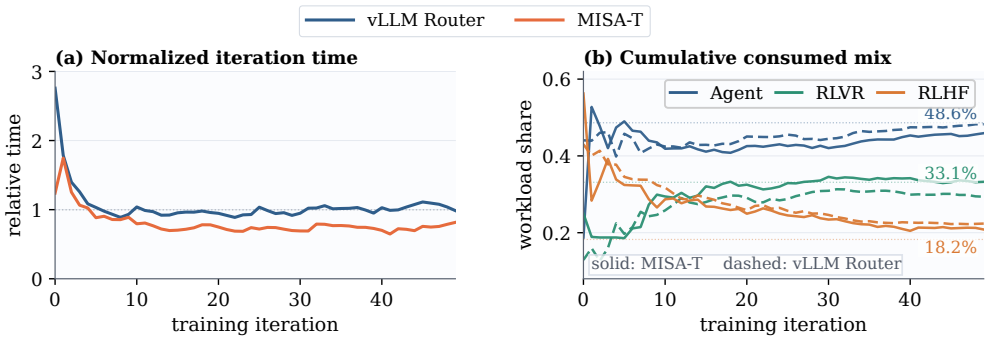

Figure 3: End-to-end progress over 50 iterations.

Table 3: Rollout-only serving results.

Model	Router	Change vs. baseline (%) Sample rate	RPM	Prompt TPS	Decode TPS	Prefix hit
Step3.7	vLLM Router (128K-safe)	-21.7%	-27.8%	-29.5%	-22.1%	97.1%
	vLLM Router (sweep)	0.0%	0.0%	0.0%	0.0%	95.9%
	Session Admission	+11.7%	+22.7%	+20.5%	+16.8%	97.1%
	MISA	+23.3%	+28.8%	+25.4%	+19.8%	97.3%
	MISA-T	**+53.3%**	**+45.5%**	**+40.4%**	**+26.5%**	**97.8%**
Qwen3.6-35B-A3B	vLLM Router (128K-safe)	-12.2%	-10.2%	-9.0%	-6.8%	95.9%
	vLLM Router (sweep)	0.0%	0.0%	0.0%	0.0%	92.4%
	Session Admission	+19.3%	+23.7%	+23.4%	+7.8%	**96.3%**
	MISA	+30.9%	+28.9%	+30.2%	+12.3%	79.2%
	MISA-T	**+43.6%**	**+37.5%**	**+36.5%**	**+18.3%**	95.3%
	vLLM Router (high load)	-55.2%	-52.5%	-57.3%	-38.5%	4.5%

time normalized by the mean vLLM Router value and cumulative consumed mixture, with dotted lines marking trainer targets.

5.3 Rollout-only ablation

Table 3 compares each policy with sweep-tuned vLLM Router on the same fixed model checkpoint.

Relative to sweep-tuned vLLM Router, MISA-T improves rollout throughput by 53.3% on Step3.7 and 43.6% on Qwen3.6-35B-A3B, with prefix-cache hit rates of 97.8% and 95.3%, respectively.

Relative to Session Admission, MISA improves rollout throughput by 10.4% on Step3.7 and 9.7% on Qwen3.6-35B-A3B. Adding residency-time weighting improves it by a further 24.3% and 9.7%, respectively. Together, these controls improve rollout throughput over dynamic Session Admission by 37.3% and 20.4%, respectively.

On Qwen3.6-35B-A3B, all admission variants prevent the prefix-cache collapse observed under high-load vLLM Router. Once prefix locality is preserved, rollout throughput also depends on per-workload admission allocation: MISA outperforms Session Admission despite a lower but still substantial hit rate, while MISA-T achieves the highest throughput and restores high prefix reuse.

5.4 Controller behavior

Figure 4 shows admission-cap adaptation under prefix-cache pressure. The base cap follows the moving session-length reference; the current cap gates admission and contracts during confirmed prefix-hit degradation, marked by red bands. More pressure-triggered corrections occur early in the run because the initial length reference underestimates the KV demand of long-tailed sessions. As

completed-session observations accumulate, the reference adapts and the admission cap approaches a steady state.

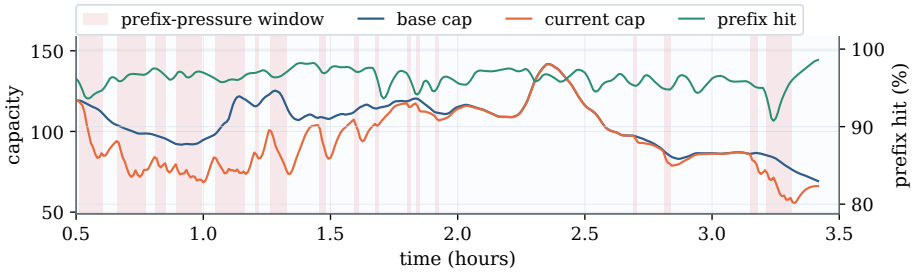

Figure 4: MISA-T admission control on Qwen3.6-35B-A3B.

5.5 Compatibility with CPU KV offloading

Our Step3.7 rollout-only comparison enables CPU KV-cache offloading (Lee et al., 2024; Sun et al., 2025) with unchanged MISA-T concurrency settings. The 1.3 TB CPU tier backs up all GPU-resident KV state, while routing-layer admission remains unchanged.

Figure 5 shows higher GPU KV occupancy with offloading, close to or above 90% through most of the steady-state interval, while retaining high prefix-cache hit rates and improving mean per-replica RPM by 35.6%.

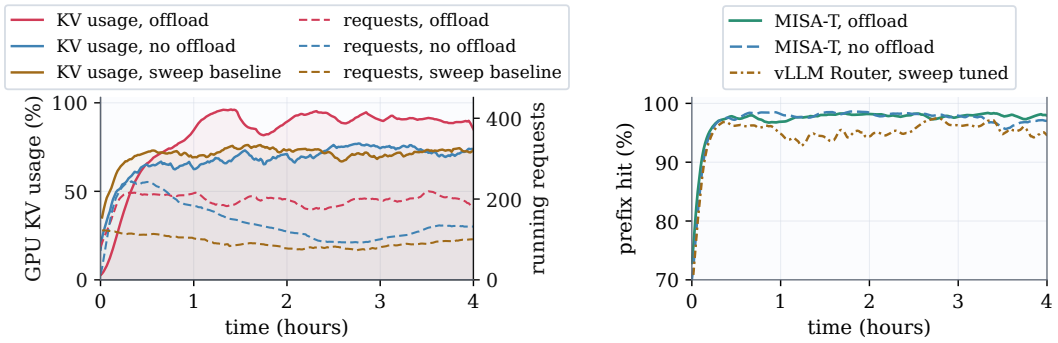

Figure 5: MISA-T with CPU KV-cache offloading on Step3.7.

6 Discussion and Conclusion

MISA-T combines adaptive session admission with workload-aware KV allocation and residency-time accounting. It improves rollout throughput by 53.3% and 43.6% over sweep-tuned routing in the two rollout-only models, and by 35.6% in matched end-to-end training, with a 22.8% reduction in iteration time and comparable task scores.

Within the trainer's outer concurrency limit, admission adapts to session lengths, cache state, and workload demand. This reduces reliance on repeated static-concurrency searches across algorithms, workloads, and checkpoints, without identifying a globally optimal limit.

Limitations. MISA-T assumes that requests carry workload labels and relies on timely serving-state reports from inference instances. Delayed or incomplete snapshots can temporarily reduce the accuracy of KV-demand estimates and admission caps. The evaluated configurations do not establish robustness to arbitrary mixtures or feedback delays. The 50-iteration run does not fully characterize long-horizon training behavior.

AI USE STATEMENT

We used OpenAI Codex for literature search, manuscript polishing, plotting code, and LaTeX formatting. AI tools were not used to design or run experiments, generate experimental data, or produce the reported measurements. The authors reviewed the AI-assisted material and take responsibility for the final manuscript, code, and results.

REFERENCES

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

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified. OpenAI, 2024. https://openai.com/index/introducing-swe-bench-verified/.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI Agents Solve Long-Horizon Software Engineering Tasks? In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Gregory Dexter, Shao Tang, Ata Fatahi, Qingquan Song, Tejas Dharamsi, and Aman Gupta. LLM query scheduling with prefix reuse and latency constraints. In *Advances in Neural Information Processing Systems 38*, 2025.

Dong Du, Shulin Liu, Tao Yang, Shaohua Chen, and Yang Li. UloRL: An ultra-long output reinforcement learning approach for advancing large language models' reasoning abilities. *arXiv preprint arXiv:2507.19766*, 2025.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning. In *Advances in Neural Information Processing Systems 38*, 2025.

Tianhao Hu, Xiangcheng Liu, Yuchun Miao, Youshao Xiao, Hongyu Zang, Yang Zheng, Xuan Huang, Jinrui Ding, Yufei Zhang, Yu Yang, Yi-Kai Zhang, Yueqing Sun, Chengcheng Han, Xiandi Ma, Wei Wang, Qi Gu, Yerui Sun, Yuchen Xie, and Xunliang Cai. DORA: A scalable asynchronous reinforcement learning system for language model training. *arXiv preprint arXiv:2604.26256*, 2026.

Hao Kang, Ziyang Li, Weili Xu, Xinyu Yang, Yinfang Chen, Junxiong Wang, Beidi Chen, Tushar Krishna, Chenfeng Xu, and Simran Arora. ThunderAgent: A fast, simple, and program-aware agentic inference system. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Kimi Team. Kimi K3: Open Frontier Intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

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 ACM SIGOPS Symposium on Operating Systems Principles*, 2023.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation*, pages 155–172, 2024.

Fengxiang Li, Han Zhang, Haoyang Huang, Jinghui Wang, Jinhua Hao, Kun Yuan, Mengtong Li, Minglei Zhang, Pengcheng Xu, Wenhao Zhuang, Yizhen Shao, Zongxian Feng, Can Tang, Chao Wang, Chengxiao Tong, Fan Yang, Gang Xiong, Haixuan Gao, Han Gao, Hao Wang, Haochen Liu, Hongliang Sun, Jiabao Li, Jingwen Chang, Jun Du, Junyi Peng, Leizhen Cui, Meimei Jing, Mingqi Wu, Shangpeng Yan, Shaotong Qi, Suzhe Xu, Wenxuan Zhao, Xianda Sun, Xuan Xie, Yanbo Wang, Yao Xia, Yinghan Cui, Yingpeng Chen, Yong Wang, Yuze Shi, Zhiwei Shen, Ziyu Wang, Ming Sun, Lin Ye, and Bin Chen. KAT-Coder-V2 Technical Report. *arXiv preprint arXiv:2603.27703*, 2026.

Yiqing Liang, Jielin Qiu, Wenhao Ding, Zuxin Liu, James Tompkin, Mengdi Xu, Mengzhou Xia, Zhengzhong Tu, Laixi Shi, and Jiacheng Zhu. MoDoMoDo: Multi-domain data mixtures for multimodal LLM reinforcement learning. *arXiv preprint arXiv:2505.24871*, 2025.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, Jinhao Dong, Zhifang Sui, and Fuli Luo. MOPD: Multi-Teacher On-Policy Distillation for Capability Integration in LLM Post-Training. *arXiv preprint arXiv:2606.30406*, 2026.

NVIDIA. Nemotron 3 Ultra: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning. *arXiv preprint arXiv:2606.15007*, 2026.

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.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 57355–57373, 2025.

vLLM Project. vLLM Router. Software repository and documentation, 2026. https://github.com/vllm-project/router.

Xiaomi LLM-Core Team. MiMo-V2-Flash Technical Report. *arXiv preprint arXiv:2601.02780*, 2026.

Ran Yan, Youhe Jiang, Tianyuan Wu, Jiaxuan Gao, Zhiyu Mei, Wei Fu, Haohui Mai, Wei Wang, Yi Wu, and Binhang Yuan. AReaL-Hex: Accommodating asynchronous RL training over heterogeneous GPUs. *arXiv preprint arXiv:2511.00796*, 2025.

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. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling Data for Software Engineering Agents. In *Advances in Neural Information Processing Systems 38*, 2025.

Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond Teacher: Generalized On-Policy Distillation with Reward Extrapolation. *arXiv preprint arXiv:2602.12125*, 2026a.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, Renjie Pi, Grace Lam, Nayeon Lee, Alexander Bukharin, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-Cascade 2: Post-training LLMs with cascade RL and multi-domain on-policy distillation. *arXiv preprint arXiv:2603.19220*, 2026b.

Weichen Yu, Xiaomin Li, Yizhou Zhao, Xiaoze Liu, Ruowang Zhang, Haixin Wang, Yinyi Luo, Chen Henry Wu, Gaurav Mittal, Matt Fredrikson, and Yu Hu. Multi-rollout on-policy distillation via peer successes and failures. *arXiv preprint arXiv:2605.12652*, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 37*, 2024.

Kan Zhu, Mathew Jacob, Chenxi Ma, Yi Pan, Stephanie Wang, Arvind Krishnamurthy, and Baris Kasikci. TraceLab: Characterizing coding agent workloads for LLM serving. *arXiv preprint arXiv:2606.30560*, 2026.

A AGENT TASK-STATE STATISTICS

Table 4 gives the agent task-state statistics discussed in Section 2.3.

Table 4: Agent task-state occupancy.

Model	Model request	Tool/environment interaction
Step3.7	93.4%	6.6%
Qwen3.6-35B-A3B	77.3%	22.7%

Preliminary stress test. In a preliminary unbounded-admission stress test, prefix hit rate fell below 5%, repeated prefill dominated the queue, and both prefill and decode throughput decreased markedly.

B AUXILIARY SCHEDULING ANALYSIS

B.1 PREFIX REUSE AND ADMISSION DELAY

For prompt length L_r and reusable prefix $H_{r,w}(t)$, estimated prefill cost on instance w is

$$\widehat{c}_{r,w}(t) = d_w(t) + \frac{L_r - H_{r,w}(t)}{\mu_w(t)}. \tag{11}$$

Here, $d_w(t)$ is prefill backlog delay in seconds and $\mu_w(t)$ is the service rate for uncached prompt tokens, distinct from the effective Prompt TPS reported in evaluation. This cost model balances prefix reuse against load, while admission controls how many sessions enter the cache.

For a continuation that must reacquire admission, assume a common uncached-token prefill rate F on both routes. Let H_r be its reusable prefix, q_{cold} and q_{hit} the cold-route and locality-preserving queueing delays, and Δ_r the hold delay. Holding for the cached instance reduces prefill completion time relative to immediate cold routing when

$$\Delta_r + q_{\text{hit}} - q_{\text{cold}} < \frac{H_r}{F}. \tag{12}$$

At prefill saturation, request rate is bounded by $F/\mathbb{E}[L_r - H_r]$, where L_r is prompt length; admitting additional sessions cannot improve this bound if KV churn reduces H_r.

B.2 BACKLOG DRAIN-TIME APPROXIMATION

Under MISA-T, estimated drain time $\tau_{w,b} \approx N_{w,b}\bar{k}_b\widehat{T}_b/M_{w,b}$ is approximately balanced by Eq. (10). Before cap rounding and minimum-slot constraints, the block-to-session conversion gives

$$\frac{M^{\text{time}}_{w,b}}{\bar{k}_b} = \frac{C_w N_{w,b}\widehat{T}_b}{\sum_j N_{w,j}\bar{k}_j\widehat{T}_j}.$$

Thus slot ratios follow $N_{w,b}\widehat{T}_b$, while footprints set block shares and the common normalization. For fixed demand and residency estimates, increasing a footprint reduces the affordable slot counts through this denominator. Without residency weighting, Eq. (7) instead gives slot ratios proportional to $N_{w,b}$. These are unrounded quota ratios, not guarantees of realized concurrency or completion rates under the global cap.

The footprint-residency product is a reference-based proxy for the integral in Eq. (1), not a measurement of physical KV occupancy over time. Recent residency estimates reflect model execution and tool/environment interactions, and unfinished sessions need not have their full residency demand remaining. Longer observed residency can increase a class's quota, but only after affected sessions complete and enter the history. The EWMA limits each observation's influence; it does not establish stability of this coupled feedback.

C CONTROLLER SETTINGS

The decision sequence and completion updates are:

1. Refresh session snapshots, pending admissions, class histories, and runtime metrics.
2. Update the base cap from the reference footprint; contract the operational cap on confirmed prefix degradation or persistent waiting pressure, and recover it as pressure clears.
3. Compute class block-time demand and soft caps using Eq. (10).
4. Retain instances where the session is already counted, or where admitting it satisfies both global and class limits. Apply cache-aware placement to eligible instances, recording a new admission as pending; otherwise hold and recheck.
5. Update per-class footprint and residency histories as sessions complete.

Table 5 lists the key settings shared by the two evaluated rollout-only configurations.

Table 5: Key controller settings.

Component	Setting
Protected KV budget	90% of the total KV-block capacity visible to the scheduler, including GPU and CPU blocks, leaving a 10% safety margin for estimation error and reporting delays.
Global length reference	Reference KV footprint estimated from recent completed sessions and current session observations.
Per-class reference	Class-specific reference KV footprint estimated from recent session observations.
Residency estimate	Per-class EWMA of model-execution and tool/environment interaction time, with new-sample weight 0.01 and a shared neutral prior during warm-up.
Prefix reference	Token-weighted hit observations in recent healthy 10-second windows.
Prefix-pressure entry	Hit rate at least 2 percentage points below the healthy reference for at least 3 seconds.
Cap response	Reduce the current cap by approximately 10% under pressure; after pressure clears, increase it by 1% each healthy decision cycle until it reaches the base cap.
Waiting pressure	Increasing queued prefill tokens for at least 3 seconds while KV-cache usage is at least 75%, filtering transient long-prefill bursts under low cache occupancy.
Control cadence	Session snapshots and scheduler decisions updated every 1 second.