MINDPRO: ACCELERATING LLM REINFORCEMENT LEARNING VIA ROBUST PARTIAL-ROLLOUT SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) typically waits for all responses in a rollout batch before each policy update, allowing a small number of long responses to delay the entire synchronous training step. We introduce **MindPRO**, a prompt-preserving partial-rollout scheduler that mitigates this long-tail barrier without continuously overlapping rollout generation and policy optimization. Building on quota-based partial-rollout scheduling, MindPRO over-provisions prompt groups, advances once a target quota becomes trainable, and manages never-launched prompts and reusable interrupted trajectories through two checkpointable buffers. Prompt preservation here refers to maintaining prompt identity and eligible partial work across cutoff and recovery, while all selected, terminal, fresh, and partial outcomes remain explicitly accounted for. Rollout collection and optimization remain phase-serialized, while optional stability safeguards mitigate the localized policy mismatch and completion-order bias introduced by prefix reuse. Experiments on mathematical reasoning tasks show verified mean-step reductions of 39.8%–48.6% across three MindRLHF workloads, while final-checkpoint evaluations generally retain or improve downstream reasoning quality. We will publicly release our code to facilitate further development of the MindSpore ecosystem.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has emerged as an important paradigm for enhancing mathematical reasoning, code generation, and other verifiable capabilities of large language models Shao et al. (2024); Yu et al. (2025); DeepSeek-AI et al. (2025). Modern training frameworks, including verl Sheng et al. (2025), OpenRLHF Hu et al. (2024), and MindRLHF MindSpore Lab (2024), have substantially improved large-scale reinforcement learning scalability by integrating distributed optimization, high-throughput inference engines, and flexible resource orchestration Huawei Technologies (2022); Kwon et al. (2023). However, as sampled responses, maximum generation lengths, and reasoning complexity continue to increase, rollout collection increasingly determines end-to-end training throughput.

Most existing RLVR systems follow a synchronous execution model in which each policy update waits for the complete rollout batch (Fu et al., 2025; Zhou et al., 2025; Xu et al., 2026). Because response lengths are highly variable and often long-tailed, many short generations finish early while a small number of long responses continue decoding, leaving the resources responsible for subsequent policy optimization idle at the synchronization boundary Zhou et al. (2025); Zhang et al. (2025); Gao et al. (2025); Xu et al. (2026); Khan et al. (2026). Consequently, the rollout latency of an entire step is governed by its slowest straggler responses rather than its typical responses. As reasoning models produce longer and more heterogeneous outputs, this rollout barrier increasingly limits the overall throughput and scalability of synchronous RLVR. Figure 1 illustrates the resulting systems timeline and a representative response-length distribution.

Existing approaches mitigate this barrier along two main directions. Fully asynchronous systems overlap rollout generation with policy optimization across successive updates, removing phase-level waiting but creating broader actor–learner policy lag when trajectories are consumed after the policy

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

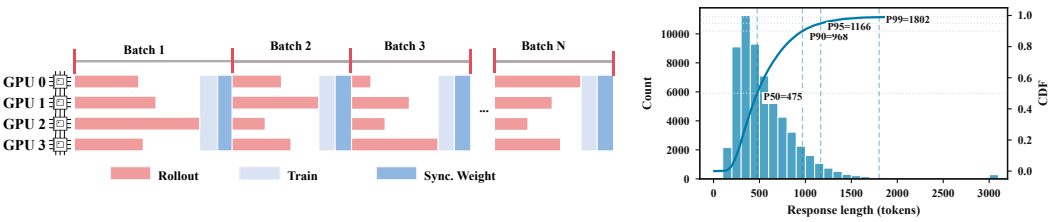

Figure 1: The synchronous rollout barrier. *Left:* a conceptual synchronous-training timeline shows rollout-length imbalance delaying the train and weight-synchronization phases. *Right:* a representative Qwen2.5-1.5B-Instruct/MATH rollout audit ($n = 8$) exhibits a right-skewed response-length distribution (P50 $= 475$, P90 $= 968$, P99 $= 1802$ tokens).

that generated them has changed Fu et al. (2025); Han et al. (2025); Zhong et al. (2025); Wu et al. (2025). Active partial-rollout sampling (APRS) provides an intermediate design: it executes rollout requests concurrently, over-provisions candidate groups, and advances once a target quota is trainable without continuously overlapping generation and optimization Zhou et al. (2025). Resuming an interrupted prefix after a policy update nevertheless introduces a localized cross-step mismatch, so this intermediate design is not strictly on-policy. Turning it into a reliable training system also requires more than an early cutoff: never-launched prompts must not be silently dropped, interrupted trajectories need an explicit and reusable lifecycle, and scheduler state must survive checkpoint recovery to preserve consistent sample accounting across restarts.

To address these challenges, we introduce **MindPRO**, where PRO denotes **P**artial-**R**ollout **O**rchestration. MindPRO is a prompt-preserving partial-rollout scheduler for phase-serialized RLVR training, implemented in verl (Figure 2). MindPRO builds on the established quota-cutoff scheduling paradigm and focuses on closing prompt and interrupted-trajectory lifecycles across cutoff, reuse, and recovery. A pending-fresh buffer retains never-launched or reset prompt groups, while a partial-trajectory buffer stores eligible interrupted prefixes with their behavior metadata and resume counts. Completed groups not admitted to the current update are marked terminal and remain explicitly visible in the accounting ledger. Both buffers and associated data positions, fetch orders, and counters are checkpointed with the actor state. Optional importance-ratio, length-reweighting, and salvaged-loss safeguards address training risks from prefix reuse, completion-order admission, and zero-variance groups. We further port MindPRO to MindRLHF to evaluate applicability across training frameworks MindSpore Lab (2024). MindPRO therefore turns partial-rollout scheduling into a reusable, recoverable, and auditable training workflow. Our contributions are threefold:

- We develop a prompt-preserving dual-buffer lifecycle that separately manages pending-fresh prompt groups and reusable interrupted trajectories, reducing rollout-tail waiting while maintaining a phase-serialized policy-update boundary.
- We build an auditable and recoverable scheduler with closed accounting over groups, requests, tokens, buffers, and data positions. Checkpointed queue states and forced-restart validation establish lifecycle continuity across training interruptions.
- We evaluate MindPRO across mathematical reasoning tasks, model scales, and training frameworks. Updated MindRLHF comparisons reduce matched-update mean step time by 39.8%–48.6%, with downstream sampled evaluations generally retained or improved; the 1.5B long-context experiment shows the same efficiency trend.

2 Related Work

RL Objectives and Training Architectures. RLVR advances mathematical and programmatic reasoning through verifiable feedback. PPO and GRPO provide policy-optimization foundations (Schulman et al., 2017; Shao et al., 2024), while RLOO, DAPO, GSPO, and ReMax refine advantage estimation and update objectives (Ahmadian et al., 2024; Yu et al., 2025; Zheng et al., 2025; Li et al., 2024). Complementary studies examine zero-style training dynamics, reward signals, and reasoning-length control (Hu et al., 2025a; Zeng et al., 2025; Aggarwal & Welleck, 2025; Liu et al., 2025b). Frameworks including DeepSpeed-Chat, NeMo-Aligner, OpenRLHF, verl, and ROLL

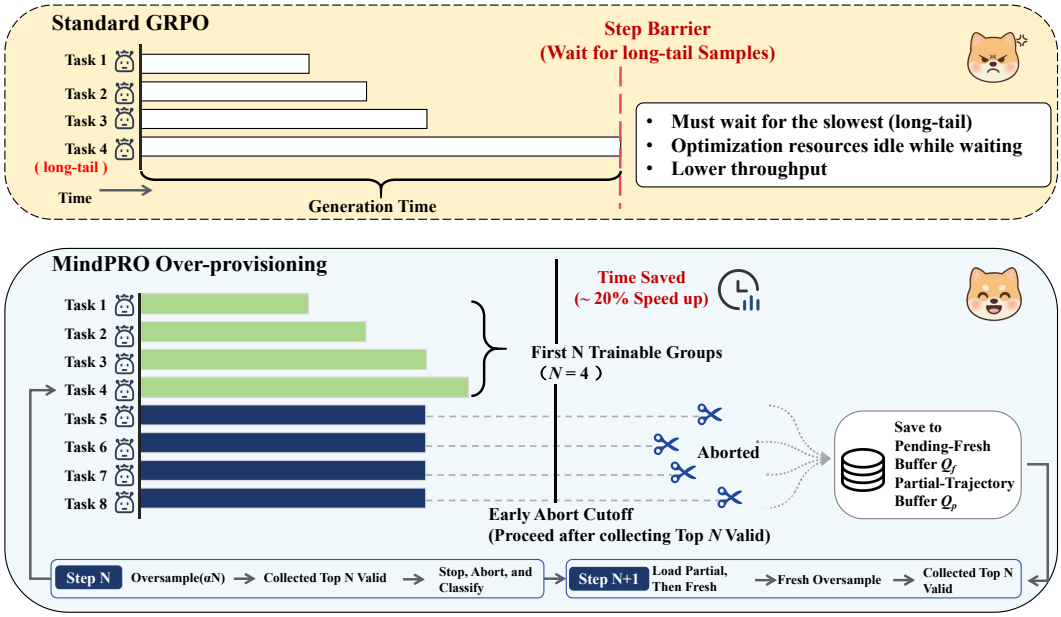

Figure 2: MindPRO workflow. Standard synchronous GRPO waits at the longest rollout, whereas MindPRO over-provisions candidate groups, advances after the first N trainable groups, stops the residual requests, and carries pending-fresh prompts and eligible partial prefixes into the next synchronous step.

support scalable rollout and policy optimization (Yao et al., 2023; Shen et al., 2024; Hu et al., 2024; Sheng et al., 2025; Wang et al., 2025), and ReaL optimizes resource allocation across stages (Mei et al., 2025). Their synchronous execution paths couple policy updates to full-batch rollout completion. AReaL, AsyncFlow, StreamRL, and LlamaRL instead overlap generation and optimization to improve utilization while managing cross-update policy lag (Fu et al., 2025; Han et al., 2025; Zhong et al., 2025; Wu et al., 2025). MindPRO complements these objective and architecture advances by accelerating batch collection within a phase-serialized workflow: quota-based admission advances the update boundary, while explicit behavior metadata tracks the policy history of reused prefixes.

Efficient Rollout Scheduling. Serving systems improve decoding throughput and latency through memory management, disaggregation, and scheduling (Kwon et al., 2023; Zhong et al., 2024; Holmes et al., 2024; Wu et al., 2024; Agrawal et al., 2024); rollout scheduling additionally addresses the batch-completion barrier in RL. Kimi k1.5 resumes budget-truncated trajectories (Kimi et al., 2025), and APRIL systematizes over-provisioning, quota cutoff, and resumption as active partial-rollout sampling (Zhou et al., 2025). RollPacker groups deferred long-tail work, while Co-PRIS combines concurrency-controlled partial rollouts with importance sampling (Gao et al., 2025; Qu et al., 2025). Speculative rollout methods reuse historical responses or adapt draft models (He et al., 2025; Liu et al., 2025a; Hu et al., 2025b); other systems explore group pipelining, idle-time speculation, adaptive group sizing, and agentic orchestration (Chen et al., 2026; Xu et al., 2026; Khan et al., 2026; Zhang et al., 2026). Building on quota-based partial rollouts, MindPRO unifies fresh-prompt retention, interrupted-trajectory reuse, checkpointed scheduler state, and closed lifecycle accounting across verl and MindRLHF. Configurable prefix correction, length reweighting, and salvage complement this execution layer; established importance-weighted and clipped policy objectives inform their integration (Wang et al., 2017; Munos et al., 2016; Espeholt et al., 2018; Ahmadian et al., 2024). MindPRO's focus is thus a reusable, recoverable, and auditable prompt lifecycle that carries partial-rollout execution across training updates and restarts.

Algorithm 1 One synchronous MindPRO training step.

Require: Actor π_θ, pending-fresh queue $\mathcal{Q}_f$, partial queue $\mathcal{Q}_p$, N_{train}, α, n
Ensure: Updated π_θ and checkpointed queues $\mathcal{Q}'_f$, $\mathcal{Q}'_p$
1: $N_{\text{over}} \leftarrow \lceil \alpha N_{\text{train}} \rceil$
2: Fetch whole data batches into $\mathcal{Q}_f$ until a plan can be formed
3: Fill surplus plan slots from $\mathcal{Q}_p$; fill the remainder from $\mathcal{Q}_f$
4: Build $N_{\text{over}} n$ request descriptors and launch up to the in-flight limit
5: **while** fewer than N_{train} trainable groups are complete **do**
6: Collect finished requests, score complete groups, and refill from the request queue
7: **end while**
8: Stop dispatch, abort in-flight requests, and collect returned prefixes
9: Requeue never-launched/reset groups in $\mathcal{Q}'_f$ and reusable prefixes in $\mathcal{Q}'_p$
10: Assemble N_{train} groups and apply enabled stability safeguards
11: Perform one synchronous PPO/GRPO update on π_θ
12: Persist the actor, queues, data position, and counters when a training checkpoint is requested

3 METHODOLOGY

MindPRO specifically targets a phase-serialized RLVR loop rather than a continuously asynchronous actor–learner pipeline. Algorithm 1 formalizes the scheduler logic, while Figure 2 illustrates its dual-queue lifecycle. We detail the barrier objective, prompt lifecycle, recoverable states, and practical implementation boundaries below.

At optimizer step t, actor π_t is synchronized to the rollout servers before dispatch. Optimization produces π_{t+1} only after the trainable quota is assembled and rollout servers are quiesced, so active generation and parameter updates do not proceed concurrently under changing weights. Trajectory freshness is nevertheless not strictly on-policy: a resumed prefix may originate from π_{t-k} and its suffix from π_t, where k is bounded by the configured resume depth.

3.1 SYNCHRONOUS ROLLOUT-BARRIER OBJECTIVE

Consider a training batch comprising N_{train} prompt groups, each requiring n sampled responses. Let T_g denote the time when all responses and rewards for group g are available. Standard synchronous GRPO hits its rollout barrier at the longest completion time:

$$T_{\text{roll}}^{\text{sync}} = \max_{g \le N_{\text{train}}} T_g. \tag{1}$$

To mitigate this, MindPRO provisions $N_{\text{over}} = \lceil \alpha N_{\text{train}} \rceil$ candidate groups and defines a fast-completion cutoff:

$$\tau = \min\{t : |\mathcal{C}_{\text{trainable}}(t)| \ge N_{\text{train}}\}, \tag{2}$$

where $\mathcal{C}_{\text{trainable}}(t)$ denotes the set of completed groups satisfying the admission rules by time t. A partial-rollout scheduler is effective only if the time saved from eliminating the straggler wait exceeds the system overheads (over-provisioning, aborts, buffering, prefix re-prefilling, and accounting). Therefore, we rigorously evaluate the end-to-end training step time:

$$T_{\text{step}} = T_{\text{data}} + T_{\text{roll}} + T_{\text{reward/logp}} + T_{\text{update}} + T_{\text{buffer}}, \tag{3}$$

ensuring true wall-clock acceleration rather than merely a reduced cutoff time.

A group enters $\mathcal{C}_{\text{trainable}}(t)$ only after all n responses and verifier rewards are available. Simultaneous candidates are ranked by reward standard deviation, while the admitted update remains fixed at N_{train} groups.

3.2 PROMPT-PRESERVING CUTOFF SCHEDULER

For clarity, a prompt group denotes one input prompt together with its n sampled response requests. MindPRO follows the partial-rollout paradigm by scheduling independent prompt groups, over-provisioning the candidate pool, and stopping collection once N_{train} groups become trainable. At cutoff, selected groups enter the current update, completed but unselected groups become terminal, and unfinished groups are evaluated for reuse. Eligible interrupted trajectories enter the partial

Table 1: Model quality and step-time comparisons. MindRLHF reports final logged validation reward (%); verl reports accuracy (%). The 7B GSM8K scores average three evaluation sampling seeds. Times average common optimizer updates; quality is evaluated at the selected final checkpoints. Ratios are baseline/MindPRO step times.

Train Dataset	Method	GSM8K ↑	MATH ↑	DAPO ↑	Time (s) ↓	Step-time Ratio
verl, Qwen2.5-1.5B-Instruct						
GSM8K	GRPO	76.95	**54.02**	**10.60**	804.94	1.000×
	MindPRO	**77.10**	53.16	5.80	**708.73**	**1.136**×
DAPO-Math-17K	GRPO	**42.99**	**53.42**	**12.80**	480.53	1.000×
	MindPRO	33.97	52.70	10.70	**154.81**	**3.104**×
MATH	GRPO	52.92	49.18	8.70	495.42	1.000×
	MindPRO	**72.40**	–	**13.40**	**243.51**	**2.034**×‡
verl, Qwen2.5-7B-Instruct						
GSM8K	GRPO	**89.46**	69.20	**28.97**	317.15	1.000×
	MindPRO	88.73	**69.31**	28.73	**182.05**	**1.742**×
MATH	GRPO	**35.9**†	58.2†	21.7	607	1.000×
	MindPRO	29.3†	**63.8**†	**26.4**	**348**	**1.74**×
MindRLHF, Qwen3-0.6B-Base						
DAPO-Math-17K	GRPO	3.34	**53.29**	**14.20**	281.50	1.000×
	MindPRO	**6.60**	53.11	13.70	**160.43**	**1.755**×
GSM8K	GRPO	60.88	53.52	3.18	110.90	1.000×
	MindPRO	**61.64**	**53.61**	**4.45**	**57.00**	**1.946**×
MATH	GRPO	5.08	**54.19**	**13.57**	257.69	1.000×
	MindPRO	**5.23**	53.48	10.92	**155.11**	**1.661**×

buffer, whereas never-launched groups and trajectories that do not satisfy the reuse constraints return to the fresh buffer. MindPRO thus preserves prompt identity and eligible partial work rather than every generated token.

Each group carries a stable prompt identifier, per-response prefixes, completion flags, behavior log-probabilities, and a resume count. At the group level, the scheduler enforces

$$N_{\text{scheduled}} = N_{\text{selected}} + N_{\text{terminal}} + N_{\text{fresh}} + N_{\text{partial}}, \tag{4}$$

where the four terms denote groups admitted to the current update, completed but not admitted, requeued as fresh, and requeued with reusable prefixes. Request-level counters separately track planned, launched, completed, aborted, failed, selected, and never-launched response requests. These checks provide closed lifecycle accounting through scheduler transitions and checkpoint recovery events and make both reusable and terminal outcomes explicit.

3.3 Dual Buffers and Checkpointed Resumption

MindPRO manages prompt states using two separate buffers. The pending-fresh buffer ($\mathcal{Q}_f$) stores untouched groups and groups reset because of capacity limits, while the partial-trajectory buffer ($\mathcal{Q}_p$) stores token prefixes, completion flags, behavior log-probabilities, and resume counts. To prevent stale partial trajectories from dominating a batch, partial groups are normally assigned only to the $N_{\text{over}} - N_{\text{train}}$ surplus slots. They are used to fill additional slots only when there is insufficient fresh data to meet the update quota. MindPRO resumes token sequences rather than memory-intensive KV caches. Reused prefixes therefore require re-prefill computation in the serving engine, and this overhead is included in the measured rollout time. For fault tolerance, both buffers, fetch orders, data positions, and accounting states are checkpointed together with the actor weights. During recovery, the system validates the queue shapes and associated states and rejects incompatible configurations. MindPRO preserves data states instead of requiring bitwise-identical generation replays, allowing consistent recovery and complete accounting after forced interruptions.

Table 2: Qwen2.5-1.5B-Instruct long-context MATH timing summary over 215 candidate matched updates. Non-rollout time is mean step time minus mean rollout time.

Method	Updates	Step (s)	Rollout (s)	Non-rollout (s)	Resp. Len.	Total / Speedup
Synchronous GRPO	215	495.42	320.75	174.67	475.69 tokens	29.59 h / 1.000×
MindPRO	215	**243.51**	**74.45**	**163.22**	549.35 tokens	**14.54 h / 2.034**×

3.4 Optional Stability Safeguards

MindPRO complements its cutoff scheduler with three independently configurable safeguards for prefix reuse, response-length weighting, and learning from zero-variance groups. They integrate into the policy-update objective while preserving the phase-serialized execution model.

Prefix Importance Sampling. To account for the behavior policy that generated a reused prefix, MindPRO restores its stored token log-probabilities as the old-policy values at prefix positions. For response i with prefix length p_i, the resulting ratio is

$$r_{i,t}^{\text{prefix}} = \exp\left(\ell_{i,t}^{\theta} - \ell_{i,t}^{b}\right), \qquad t \le p_i, \tag{5}$$

where ℓ^{θ} and ℓ^{b} denote current- and behavior-policy log-probabilities. This ratio enters the standard token-wise PPO/GRPO clipped surrogate, making prefix updates sensitive to their generation policy without changing the rollout lifecycle.

Length Reweighting. To increase the optimization weight of longer admitted responses, MindPRO scales the advantage as

$$\widetilde{A}_i = A_i \frac{L_i}{\text{mean}_j\, L_j}, \tag{6}$$

where L_i is the response length and the denominator is its batch mean. This rule preserves the advantage sign and adjusts relative loss weights without changing group admission.

Salvage. To extract a learning signal from groups with identical rewards, MindPRO reserves up to βN_{train} slots for zero-variance groups. It assigns surrogate advantages $+\lambda_{\text{SFT}}$ to all-correct groups and $-\lambda_{\text{unlik}}$ to all-wrong groups within the clipped objective, encouraging correct responses and discouraging incorrect ones. Each safeguard can be enabled separately; scheduler-only configurations retain the same quota cutoff and dual-buffer lifecycle without these objective modifications.

4 Experiments

4.1 Experimental Setup

Datasets and Models. We evaluate MindPRO on GSM8K Cobbe et al. (2021), MATH Hendrycks et al. (2021), and DAPO-Math-17K Yu et al. (2025), covering diverse mathematical reasoning workloads. Experiments use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct in **verl** Sheng et al. (2025), and Qwen3-0.6B-Base in **MindRLHF** MindSpore Lab (2024), enabling evaluation across model scales and training frameworks. GSM-Plus and the English text-only mathematics subset of OlympiadBench provide additional independent tests of downstream transfer.

Evaluation Metrics. Our primary efficiency metrics are mean training-step time and speedup over synchronous GRPO, computed over common optimizer updates and excluding validation time. Rollout and non-rollout latency, response length, and aggregate logged time characterize the execution costs. Table 1 reports benchmark accuracy for verl and logged validation reward for MindRLHF. Independent GSM-Plus and OlympiadBench evaluations use sampled pass@1 with identical decoding settings within each comparison; means and standard deviations are computed over three evaluation sampling seeds per checkpoint.

Table 3: Ablation study of stability safeguards on Qwen2.5-Math-1.5B-Instruct trained on DAPO-Math-17K (KL= 10^{-3}). "IS" denotes behavior log-probability correction, "Len" length reweighting, and "Salv" salvaged loss. (✗) indicates joint accuracy and response-length collapse; ‡ denotes final-step values after response length collapsed toward zero.

Group	Variant	IS	Len	Salv	DAPO ↑	Clip ↓	Resp. Len	Filter Ratio
Baseline	APRS	–	–	–	0.0 (✗)	1.85 %	11 (✗)	n/a‡
Single-Correction	+ IS only	✓	–	–	36.4	**0.012** %	**817**	49.1 %
	+ Len only	–	✓	–	35.7	0.015 %	649	52.1 %
	+ Salv only	–	–	✓	**36.4**	0.016 %	725	50.9 %
Full	**MindPRO**	✓	✓	✓	35.2	4.49 %	749	**48.4 %**

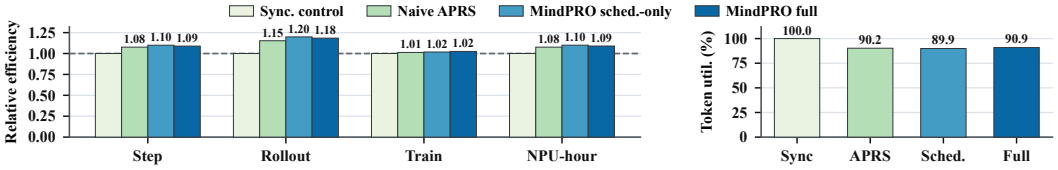

Figure 3: Controlled four-arm scheduler attribution on Qwen2.5-1.5B-Instruct/GSM8K over 12 matched updates. Left: relative efficiency (synchronous-control cost divided by each arm's cost). Right: token utilization (selected generated-new tokens divided by generated-new plus re-prefilled tokens). Higher is better.

MindSpore Backend Integration. We implement MindPRO's partial-rollout scheduling and safeguards in MindRLHF on the native MindSpore/MindFormers training path MindSpore Lab (2024); Huawei Technologies (2022). The integration registers the MindFormers actor and MindSpore policy loss, reuses MindRLHF's reward and policy-update interfaces, and keeps request admission, interrupted-prefix assembly, and selected-batch construction in the scheduler above backend-specific model operators. Hardware configurations, hyperparameters, and the four-arm comparison settings are detailed in Appendix A.

4.2 Main Results and Evaluation

For Q1: Does MindPRO reduce common-update training time? Table 1 shows that MindPRO's rollout optimization translates into lower training-step time across both frameworks and multiple model scales. The variation in speedup highlights the importance of the execution profile: end-to-end gains depend on how much of the synchronous step is spent collecting rollouts and how much work remains after cutoff, rather than on model size alone. The GSM8K timing audit illustrates this relationship. Rollout becomes faster, but policy optimization remains a substantial part of the step, so the overall gain is smaller than the generation-phase gain. In the MindRLHF comparisons, larger reductions in rollout time outweigh the added reuse and correction costs, yielding stronger end-to-end improvements. Figure 5 further shows savings throughout the paired GSM8K run, indicating a recurring execution benefit rather than an average dominated by a few slow baseline updates. Together, these observations establish rollout scheduling as a useful optimization across backends without requiring continuous rollout–training overlap.

For Q2: Where do MindPRO's execution gains originate? The four-arm comparison in Figure 3 separates the benefit of advancing the rollout boundary from the cost of managing interrupted work. The cutoff-only arm captures most time savings, consistent with avoiding the wait for residual requests once the training quota is ready. Scheduler-only MindPRO improves on this arm while retaining pending prompts and reusable prefixes; enabling safeguards preserves most of that efficiency advantage. The largest phase-level change occurs in rollout, while training time remains stable. This pattern supports cutoff as the main timing mechanism, with lifecycle management sustaining partial-rollout execution across updates. The MindRLHF phase profiles strengthen this interpretation: rollout savings exceed additional non-rollout costs, so the speedup is not explained by a

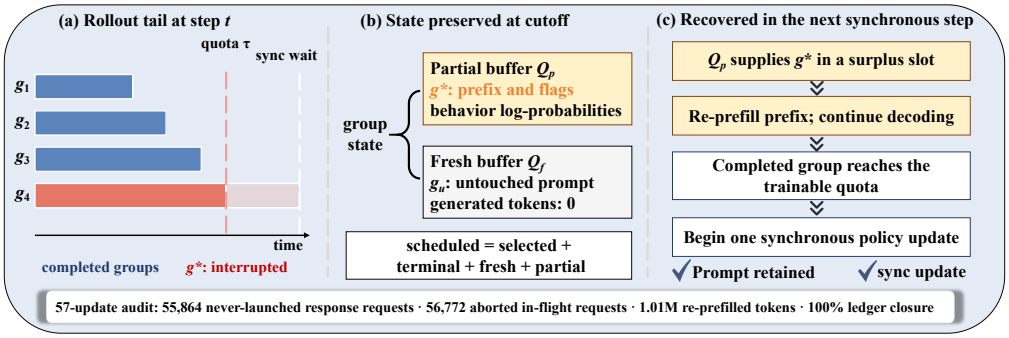

Figure 4: Representative prompt-lifecycle case across two synchronous optimizer steps. At the quota cutoff, completed groups proceed to training, an eligible interrupted group is stored in the partial buffer, and an untouched prompt remains in the fresh buffer. In the next step, the saved prefix is re-prefilled and completed before a synchronous policy update.

cheaper policy update. Table 2 provides a complementary generation-dominated timing breakdown for verl. These profiles capture combined workload effects of cutoff, prefix reuse, and response length. Closed lifecycle accounting guarantees traceability: selected, terminal, fresh, and partial outcomes remain explicit as work moves across updates. MindPRO thus couples earlier training admission with reusable and auditable rollout state, turning a local cutoff decision into a repeatable training workflow.

Recovery audit. We further evaluate checkpoint recovery of the scheduler through controlled failure injection. Training is interrupted after step 6, then resumed from the saved actor and scheduler states and continued through steps 7–10. As summarized in Table 4, both buffers remain free of duplicates, cumulative counters increase monotonically, and all 12 automated state and accounting checks pass after recovery. These results verify continuity of the scheduler state across a checkpoint boundary without requiring a bitwise-identical request completion order.

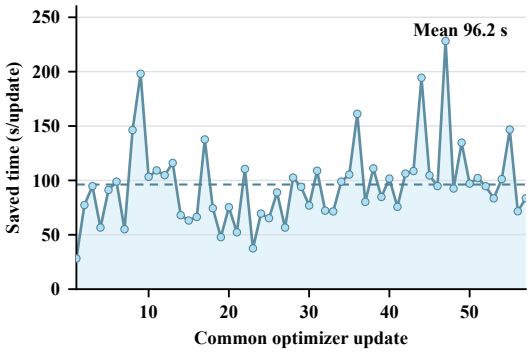

Figure 5: Historical paired GSM8K audit over 57 common optimizer updates.

For Q3: Does MindPRO retain downstream reasoning quality? Separate independent evaluations compare final checkpoints under the same sampled pass@1 protocol. MindPRO improves both GSM-Plus and OlympiadBench means after 1.5B MATH training and after the MindRLHF MATH and GSM8K runs. After MindRLHF DAPO training, OlympiadBench remains effectively unchanged (+0.10 percentage points), while GSM-Plus decreases by 0.66 points. Thus, the updated evidence does not show a consistent downstream degradation across tasks, but it also does not justify a training-level significance claim: the reported deviations come from three evaluation sampling seeds applied to one checkpoint, not independent training seeds.

4.3 Ablation Study

Table 3 and Figure 6 provide mechanism-level evidence under the historical DAPO-Math-17K configuration. Without safeguards, training with the evaluated APRS variant collapses, yielding 0.0% DAPO accuracy and a mean response length of 11 tokens. Enabling IS, length weighting, or salvage individually restores stable training, yielding 35.7–36.4% accuracy and mean response lengths of

Table 4: Controlled failure-injection audit at the interruption and final resumed checkpoints.

Audit item	**Step 6**	**Step 10**
Pending-fresh groups $\lvert\mathcal{Q}_f\rvert$	472	471
Partial groups $\lvert\mathcal{Q}_p\rvert$	24	22
Fetch sequence	2,048	3,072
Dataloader samples yielded	2,048	3,072
Scheduler attempts	6	10
Buffer PUID uniqueness	Pass	Pass
Scheduler-state size	0.99 MB	0.93 MB
Resumed updates	Steps 7–10 present	
Ledger closure	4/4 resumed updates	
Automated checks	12/12 passed	

649–817 tokens. The full MindPRO configuration also remains stable, achieving 35.2% accuracy, a mean response length of 749 tokens, and the lowest filter ratio among the stable variants. These results indicate that the three safeguards act as independently useful, risk-specific controls rather than as an inseparable stack. This ablation demonstrates training stability under the tested difficult configuration, whereas the scheduler-only DAPO and MATH comparisons isolate the timing contribution attributable to the prompt-preserving scheduler.

4.4 Case Analysis

Figure 4 follows a representative prompt lifecycle across two adjacent synchronous optimizer steps. In panel (a), groups g_1–g_3 satisfy the trainable quota at cutoff τ, so MindPRO can stop waiting for the long-running group g^* that would otherwise determine the synchronous rollout barrier. Panel (b) shows how reusable work remains traceable after cutoff: the prefix of g^* enters the partial buffer $\mathcal{Q}_p$, while the never-launched group g_u remains in the fresh buffer $\mathcal{Q}_f$. Completed but unselected groups follow the separately recorded terminal path. In panel (c), g^* is admitted to a surplus slot in step $t{+}1$, its prefix is re-prefilled into the serving engine, and decoding continues until the group becomes trainable; only then can the next synchronous policy update begin. The case therefore clarifies that MindPRO preserves prompt identity and eligible partial work rather than KV caches, while the audited footer connects this symbolic transition to the measured 57-update ledger closure.

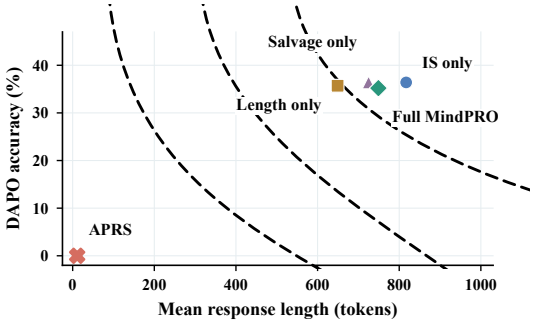

Figure 6: Historical DAPO-Math-17K safeguard ablation. Exact clip and filter ratios are reported in Table 3.

5 Conclusion and Future Outlook

In this paper, we presented **MindPRO**, a prompt-preserving partial-rollout scheduler that mitigates the rollout barrier in phase-serialized RLVR training. MindPRO combines quota-based cutoff, a dual-buffer prompt lifecycle, checkpointed scheduler states, closed lifecycle accounting, and configurable stability safeguards to make partial-rollout execution reusable, recoverable, and auditable. Future work will extend MindPRO to larger models and distributed training settings, supporting scalable reinforcement learning infrastructure and the MindSpore ecosystem.

AI USE STATEMENT

In this work, we used generative AI tools for coding assistance, log analysis, and language editing. We did not use generative AI tools to generate or fabricate experimental results, and all reported empirical results are derived from actual training logs and benchmark evaluations. We reviewed and verified all AI-assisted text, code, experimental configurations, analyses, and reported results. In particular, AI-assisted code was manually inspected and tested against the corresponding training and evaluation pipelines, and AI-assisted analysis was checked against the original experiment logs. We take responsibility for the final content of this work, including text, code, claims, and analyses produced with the aid of generative AI.

ETHICS STATEMENT

This work investigates systems methods for efficient reinforcement-learning training using public mathematical reasoning datasets and pretrained language models. It does not involve human subjects or the collection of private personal data. The models and datasets used in this work remain subject to their respective licenses and usage conditions. The proposed methods are intended to improve training efficiency and reduce computational overhead. As with other techniques that reduce the cost of large-scale model training, they may lower the computational barrier to both beneficial and potentially harmful model development. Responsible use and deployment therefore remain subject to the appropriate safeguards, policies, and usage restrictions of the resulting models and applications.

REPRODUCIBILITY STATEMENT

We have made efforts to support reproducibility throughout this work. Algorithm 1 and Section 3 describe the MindPRO scheduler, dual-buffer lifecycle, checkpointed scheduler state, and configurable training safeguards. Section 4.1 specifies the datasets, models, and evaluation protocol, while Appendix A provides the hardware settings, training hyperparameters, and comparison configurations. Section 4.2 reports the main efficiency and quality results, and Table 4 documents the checkpoint-recovery validation. The implementation, experiment recipes, and supporting configuration files will be released to facilitate reproduction of the reported results.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve, 2024. URL https://arxiv.org/abs/2403.02310.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.662. URL https://doi.org/10.18653/v1/2024.acl-long.662.

Rongjian Chen, Jianmin Hu, Kejiang Ye, and Minxian Xu. RolloutPipe: Overlapping pipelined rollout and training in disaggregated on-policy LLM reinforcement learning, 2026. URL https://arxiv.org/abs/2606.26997.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,

Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, Proceedings of Machine Learning Research, pp. 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

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, 2025. URL https://arxiv.org/abs/2505.24298.

Wei Gao, Yuheng Zhao, Dakai An, Tianyuan Wu, Lunxi Cao, Shaopan Xiong, Ju Huang, Weixun Wang, Siran Yang, Wenbo Su, Jiamang Wang, Lin Qu, Bo Zheng, and Wei Wang. RollPacker: Mitigating long-tail rollouts for fast, synchronous RL post-training, 2025. URL https://arxiv.org/abs/2509.21009.

Zhenyu Han, Ansheng You, Haibo Wang, Kui Luo, Guang Yang, Wenqi Shi, Menglong Chen, Sicheng Zhang, Zeshun Lan, Chunshi Deng, Huazhong Ji, Wenjie Liu, Yu Huang, Yixiang Zhang, Chenyi Pan, Jing Wang, Xin Huang, Chunsheng Li, and Jianping Wu. Asyncflow: An asynchronous streaming rl framework for efficient llm post-training, 2025. URL https://arxiv.org/abs/2507.01663.

Jingkai He, Tianjian Li, Erhu Feng, Dong Du, Qian Liu, Tao Liu, Yubin Xia, and Haibo Chen. History rhymes: Accelerating LLM reinforcement learning with RhymeRL, 2025. URL https://arxiv.org/abs/2508.18588.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, and Yuxiong He. DeepSpeed-FastGen: High-throughput text generation for LLMs via MII and DeepSpeed-Inference, 2024. URL https://arxiv.org/abs/2401.08671.

Jian Hu, Xibin Wu, Weixun Wang, Xianyu, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance RLHF framework. *CoRR*, abs/2405.11143, 2024. doi: 10.48550/ARXIV.2405.11143. URL https://doi.org/10.48550/arXiv.2405.11143.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-Reasoner-Zero: An open source approach to scaling up reinforcement learning on the base model, 2025a. URL https://arxiv.org/abs/2503.24290.

Qinghao Hu, Shang Yang, Junxian Guo, Xiaozhe Yao, Yujun Lin, Yuxian Gu, Han Cai, Chuang Gan, Ana Klimovic, and Song Han. Taming the long-tail: Efficient reasoning RL training with adaptive drafter, 2025b. URL https://arxiv.org/abs/2511.16665.

Huawei Technologies. *Huawei MindSpore AI Development Framework*, pp. 137–162. Springer, Singapore, 2022. ISBN 978-981-19-2879-6. https://www.mindspore.cn.

Azal Ahmad Khan, Ammar Ahmed, Zeshan Fayyaz, Sheng Di, Mingyi Hong, and Ali Anwar. Faster synchronous on-policy RL via straggler-aware group sizing, 2026. URL https://arxiv.org/abs/2606.02218.

Kimi, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Weixin Xu, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, Zonghan Yang, and Zongyu Lin. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. vLLM: Easy, fast, and cheap LLM serving with PagedAttention. https://github.com/vllm-project/vllm, 2023. arXiv:2309.06180.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. ReMax: A simple, effective, and efficient reinforcement learning method for aligning large language models, 2024. URL https://arxiv.org/abs/2310.10505.

Bingshuai Liu, Ante Wang, Zijun Min, Liang Yao, Haibo Zhang, Yang Liu, Xu Han, Peng Li, Anxiang Zeng, and Jinsong Su. SPEC-RL: Accelerating on-policy reinforcement learning with speculative rollouts, 2025a. URL https://arxiv.org/abs/2509.23232.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective, 2025b. URL https://arxiv.org/abs/2503.20783.

Zhiyu Mei, Wei Fu, Kaiwei Li, Guangju Wang, Huanchen Zhang, and Yi Wu. ReaL: Efficient RLHF training of large language models with parameter reallocation, 2025. URL https://arxiv.org/abs/2406.14088.

MindSpore Lab. MindRLHF: Reinforcement learning from human feedback on MindSpore. https://github.com/mindspore-lab/mindrlhf, 2024. Accessed: 2026-05-19.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016. URL https://arxiv.org/abs/1606.02647.

Zekai Qu, Yinxu Pan, Ao Sun, Chaojun Xiao, and Xu Han. CoPRIS: Efficient and stable reinforcement learning via concurrency-controlled partial rollout with importance sampling, 2025. URL https://arxiv.org/abs/2511.05589.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Gerald Shen, Zhilin Wang, Olivier Delalleau, Jiaqi Zeng, Yi Dong, Daniel Egert, Shengyang Sun, Jimmy Zhang, Sahil Jain, Ali Taghibakhshi, Markel Sanz Ausin, Ashwath Aithal, and Oleksii Kuchaiev. NeMo-Aligner: Scalable toolkit for efficient model alignment, 2024. URL https://arxiv.org/abs/2405.01481.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025*, pp. 1279–1297. ACM, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, Zichen Liu, Haizhou Zhao, Dakai An, Lunxi Cao, Qiyang Cao, Wanxi Deng, Feilei Du, Yiliang Gu, Jiahe Li, Xiang Li, Mingjie Liu, Yijia Luo, Zihe Liu, Yadao Wang, Pei Wang, Tianyuan Wu, Yanan Wu, Yuheng Zhao, Shuaibing Zhao, Jin Yang, Siran Yang, Yingshui Tan, Huimin Yi, Yuchi Xu, Yujin Yuan, Xingyao Zhang, Lin Qu, Wenbo Su, Wei Wang, Jiamang Wang, and Bo Zheng. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library, 2025. URL https://arxiv.org/abs/2506.06122.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay, 2017. URL https://arxiv.org/abs/1611.01224.

Bingyang Wu, Yinmin Zhong, Zili Zhang, Shengyu Liu, Fangyue Liu, Yuanhang Sun, Gang Huang, Xuanzhe Liu, and Xin Jin. Fast distributed inference serving for large language models, 2024. URL https://arxiv.org/abs/2305.05920.

Bo Wu, Sid Wang, Yunhao Tang, Jia Ding, Eryk Helenowski, Liang Tan, Tengyu Xu, Tushar Gowda, Zhengxing Chen, Chen Zhu, Xiaocheng Tang, Yundi Qian, Beibei Zhu, and Rui Hou. Llamarl: A distributed asynchronous reinforcement learning framework for efficient large-scale llm training, 2025. URL https://arxiv.org/abs/2505.24034.

Yuhang Xu, Kaibin Tian, Yang Tian, Zhice Yang, Yifeng Yu, Yan Li, Shengzhong Liu, Fan Wu, and Guihai Chen. BubbleSpec: Turning long-tail bubbles into speculative rollout drafts for synchronous reinforcement learning, 2026. URL https://arxiv.org/abs/2605.08862.

Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, Zhongzhu Zhou, Michael Wyatt, Molly Smith, Lev Kurilenko, Heyang Qin, Masahiro Tanaka, Shuai Che, Shuaiwen Leon Song, and Yuxiong He. DeepSpeed-Chat: Easy, fast and affordable RLHF training of ChatGPT-like models at all scales, 2023. URL https://arxiv.org/abs/2308.01320.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi

Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/ARXIV.2503.14476. URL https://doi.org/10.48550/arXiv.2503.14476.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL https://arxiv.org/abs/2503.18892.

Yiqi Zhang, Huiqiang Jiang, Xufang Luo, Zhihe Yang, Chengruidong Zhang, Yifei Shen, Dongsheng Li, Yuqing Yang, Lili Qiu, and Yang You. SortedRL: Accelerating RL training for LLMs through online length-aware scheduling. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. URL https://openreview.net/forum?id=YoV9lIZ827.

Zili Zhang, Yinmin Zhong, Chengxu Yang, Chao Jin, Bingyang Wu, Xinming Wei, Yuliang Liu, and Xin Jin. Heddle: A distributed orchestration system for agentic RL rollout, 2026. URL https://arxiv.org/abs/2603.28101.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving, 2024. URL https://arxiv.org/abs/2401.09670.

Yinmin Zhong, Zili Zhang, Xiaoniu Song, Hanpeng Hu, Chao Jin, Bingyang Wu, Nuo Chen, Yukun Chen, Yu Zhou, Changyi Wan, Hongyu Zhou, Yimin Jiang, Yibo Zhu, and Daxin Jiang. Streamrl: Scalable, heterogeneous, and elastic rl for llms with disaggregated stream generation, 2025. URL https://arxiv.org/abs/2504.15930.

Yuzhen Zhou, Jiajun Li, Yusheng Su, Gowtham Ramesh, Zilin Zhu, Xiang Long, Chenyang Zhao, Jin Pan, Xiaodong Yu, Ze Wang, Kangrui Du, Jialian Wu, Ximeng Sun, Jiang Liu, Qiaolin Yu, Hao Chen, Zicheng Liu, and Emad Barsoum. APRIL: active partial rollouts in reinforcement learning to tame long-tail generation. *CoRR*, abs/2509.18521, 2025. doi: 10.48550/ARXIV.2509.18521. URL https://doi.org/10.48550/arXiv.2509.18521.

A Implementation Details and Hardware

All experiments are conducted on Ascend 910-series NPUs, with hardware and batch configurations held identical within each comparison. The updated 1.5B long-context MATH block uses four NPUs, batch size 64, group size $n = 8$, maximum prompt/response lengths 2048/8192, tensor parallelism 2, and `max_num_seqs`=1024; MindPRO uses $\alpha = 2$, a 512-request per-replica admission limit, work stealing, and disables IS, length reweighting, and salvage. The updated MindRLHF blocks use four NPUs, Qwen3-0.6B-Base, batch size 64, group size $n = 8$, response cap 4096, and `max_num_seqs`=1024, with thinking disabled, temperature 0.7, top-p 0.8, top-k 20, KL coefficient 0.01, and zero entropy coefficient. The reported quality-aligned MindRLHF runs enable prefix IS. In the historical four-arm attribution study, the synchronous control waits for the full rollout batch; naive APRS applies only quota cutoff and request abortion; scheduler-only MindPRO adds the dual buffers, checkpointed scheduler state, and closed accounting; and full MindPRO additionally enables all three safeguards.