DECOUPLING SNAPSHOT PUBLICATION FROM STALENESS TOLERANCE IN DISTRIBUTED GRPO VIA LOSSLESS SPARSE PATCHES

FARS
Analemma
fars@analemma.ai

ABSTRACT

Distributed reinforcement learning for large language models requires balancing staleness tolerance with training stability. ECHO-2 introduces a staleness budget S to tolerate slow workers but couples the publication period κ with S by setting $\kappa = S - 1$, limiting scalability to $S = 11$ due to training instability. We identify that instability is driven by κ, not S: high κ causes off-policy divergence that leads to training collapse. We propose decoupling κ from S using sparse patch dissemination, which exploits the natural sparsity of single-step weight updates (90.7% sparsity) to achieve 12.5× compression and 89.9% broadcast time reduction. This enables per-step publication ($\kappa = 1$) while maintaining large staleness budgets for worker tolerance. Under sustained off-policy conditions, the baseline collapses in 3/3 seeds while our approach remains stable in 3/3 seeds, with a 1400× reduction in KL divergence.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Reinforcement learning from human feedback (RLHF) has emerged as a critical technique for aligning large language models with human preferences (DeepSeek-AI et al., 2025). Scaling RL training to large models requires distributed systems that can efficiently coordinate rollout generation across multiple workers while a central learner performs policy updates. Recent frameworks such as ECHO-2 (Xiao et al., 2026), AReaL (Fu et al., 2025), and StreamRL (Zhong et al., 2025) have demonstrated the feasibility of asynchronous distributed RL, where workers generate rollouts using potentially stale policy snapshots to maximize throughput.

However, asynchronous training introduces a fundamental tension between throughput and training stability. ECHO-2 introduces a staleness budget S that specifies the maximum allowed policy lag between rollout generation and training, enabling tolerance for slow workers in heterogeneous clusters. Yet ECHO-2 reports that GRPO training becomes unstable at $S = 11$, limiting the scalability of distributed RL. We observe that ECHO-2 couples the publication period κ (how often weights are broadcast) with the staleness budget by setting $\kappa = S - 1$. This coupling means that larger staleness budgets imply less frequent weight publication, causing rollouts to be generated from increasingly stale policies.

We hypothesize that training instability is driven by the publication period κ, not the staleness budget S. High κ causes rollouts to be generated from stale policies, leading to off-policy divergence that can cause training collapse. If we can reduce κ while maintaining large S, we should achieve both training stability and worker tolerance. The key insight enabling this decoupling is that single-step weight deltas are significantly sparser than multi-step deltas: single-step updates achieve 90.7% sparsity compared to 64.3% for 10-step accumulated updates. By transmitting only changed parameters as sparse patches, we can afford per-step publication ($\kappa = 1$) with 12.5× compression, reducing broadcast time by 89.9%.

[1] https://gitlab.com/fars-a/echo2-sparse-delta-broadcast

Our contributions are as follows:

- We identify that training instability in distributed GRPO is caused by the publication period κ, not the staleness budget S, and propose decoupling these parameters to achieve both stability and worker tolerance.
- We introduce sparse patch dissemination that exploits natural weight update sparsity to enable per-step publication with 12.5× compression and 89.9% broadcast time reduction.
- We demonstrate that under sustained off-policy conditions, the baseline collapses in 3/3 seeds while our approach remains stable in 3/3 seeds, with a 1400× reduction in KL divergence.

2 Related Work

Distributed RL for LLMs. Scaling reinforcement learning for large language models requires distributed training frameworks that can efficiently coordinate rollout generation and policy updates across multiple workers. ECHO-2 (Xiao et al., 2026) introduces a staleness budget mechanism to tolerate slow workers in heterogeneous clusters, but couples the publication period with the staleness budget, limiting scalability. AReaL (Fu et al., 2025) and StreamRL (Zhong et al., 2025) propose asynchronous architectures that disaggregate generation from training, while HybridFlow (Sheng et al., 2024) and OpenRLHF (Hu et al., 2024) provide flexible frameworks for RLHF training. RLHFuse (Zhong et al., 2024) optimizes stage fusion to reduce communication overhead. Our work identifies that the coupling between publication period and staleness budget is the root cause of training instability, and proposes decoupling them via sparse patch dissemination.

Asynchronous Training and Staleness. Asynchronous training introduces staleness when workers use outdated model parameters. Noukhovitch et al. (2024) demonstrate that asynchronous RLHF can achieve faster training with bounded staleness, while M2PO (Zheng et al., 2025) analyzes the "prosperity before collapse" phenomenon where off-policy training initially improves before diverging. A-3PO (Li et al., 2025) proposes staleness-aware proximal policy approximation to mitigate staleness effects, and StaleFlow (Li et al., 2026) introduces staleness-constrained rollout coordination. These works focus on tolerating or correcting staleness effects, whereas we identify that reducing the publication period (not the staleness budget) is the key to maintaining training stability.

Gradient and Weight Compression. Communication efficiency in distributed training has been extensively studied through gradient compression techniques. QSGD (Alistarh et al., 2016) introduces quantized stochastic gradient descent, while DIANA (Mishchenko et al., 2019) compresses gradient differences for improved convergence. TopK sparsification (Zou et al., 2022) and DoCoFL (Dorfman et al., 2023) explore downlink compression for federated learning. For LLM-specific compression, DeltaZip (Yao & Klimovic, 2023) compresses weight deltas for efficient model serving, and PULSE (Miahi & Belilovsky, 2026) exploits weight update sparsity in distributed RL. Our sparse patch dissemination builds on these insights, achieving 12.5× compression for single-step weight deltas to enable per-step publication.

Policy Optimization Stability. Maintaining training stability is crucial for policy optimization algorithms. TRPO (Schulman et al., 2015) introduces trust region constraints to ensure monotonic policy improvement, while DAPO (Yu et al., 2025) and VAPO (Yue et al., 2025) propose techniques for stable large-scale RL training. VinePPO (Kazemnejad et al., 2024) refines credit assignment to improve training stability. Unlike these algorithmic approaches that modify the optimization objective, our work addresses stability through the system design by decoupling publication frequency from staleness tolerance, keeping rollouts near-on-policy regardless of the staleness budget.

3 METHOD

3.1 PROBLEM FORMULATION

We consider distributed reinforcement learning for large language models where multiple rollout workers generate trajectories asynchronously while a central learner performs policy updates. Following ECHO-2 (Xiao et al., 2026), we define two key parameters that govern the system behavior.

The **staleness budget** S specifies the maximum allowed policy lag between rollout generation and training. A trajectory generated under policy version v is admissible for training at learner step v_t only if $v \geq v_t - S$. This parameter provides temporal slack to absorb network latency and worker heterogeneity without modifying the underlying RL objective.

The **publication period** κ determines how frequently the learner broadcasts updated weights to rollout workers. The learner publishes a new policy snapshot every κ training steps, and workers continue generating rollouts under their most recent installed snapshot between publications.

ECHO-2 couples these parameters by setting $\kappa = S - 1$, meaning a larger staleness budget implies less frequent weight publication. Under this coupling, the staleness distribution has median approximately $\kappa/2$ and P99 approaching S, causing a significant fraction of rollouts to fall outside the admissibility window.

3.2 DECOUPLING HYPOTHESIS

We hypothesize that training instability in distributed GRPO is driven by the publication period κ, not the staleness budget S. The causal chain proceeds as follows: high κ causes rollouts to be generated from stale policies, leading to high staleness in the training data. High staleness induces off-policy divergence, measured by the KL divergence between the current policy and the policy that generated the rollouts. When this divergence exceeds the trust region constraints, the policy gradient estimates become unreliable, potentially causing training collapse.

This hypothesis suggests that the instability reported by ECHO-2 at $S = 11$ is actually caused by $\kappa = 10$, not by the staleness budget itself. If we can reduce κ while maintaining large S, we should achieve both training stability (from low effective staleness) and worker tolerance (from large admissibility window).

3.3 SPARSE PATCH DISSEMINATION

The key insight enabling decoupled publication is that single-step weight deltas are significantly sparser than multi-step deltas, as illustrated in Figure 1. When the learner performs one gradient update, only a subset of parameters change substantially. In contrast, accumulating κ updates causes changes to compound across more parameters, reducing sparsity.

We exploit this sparsity by transmitting only the changed parameters as sparse patches rather than full model snapshots. For a model with N parameters, let $\Delta w_t = w_t - w_{t-1}$ denote the weight delta after step t. We define the sparse patch as the set of indices and values where the change exceeds a threshold: $\mathcal{P}_t = \{(i, \Delta w_t[i]) : |\Delta w_t[i]| > \epsilon\}$.

Empirically, single-step deltas achieve significantly higher sparsity than multi-step accumulated deltas, enabling substantial compression. This difference makes per-step publication practical: despite more frequent publication events, the total broadcast volume remains manageable because each patch is correspondingly smaller (see Section 4.4 for detailed measurements).

Importantly, our sparse patch dissemination is **lossless**—we transmit exact parameter values without quantization. This distinguishes our approach from gradient compression methods that trade accuracy for bandwidth. The sparsity arises naturally from the training dynamics, not from lossy approximation.

3.4 IMPLEMENTATION

We implement sparse patch dissemination within the OpenRLHF framework (Hu et al., 2024). At each learner step, we compute the weight delta by comparing consecutive checkpoints. Parameters

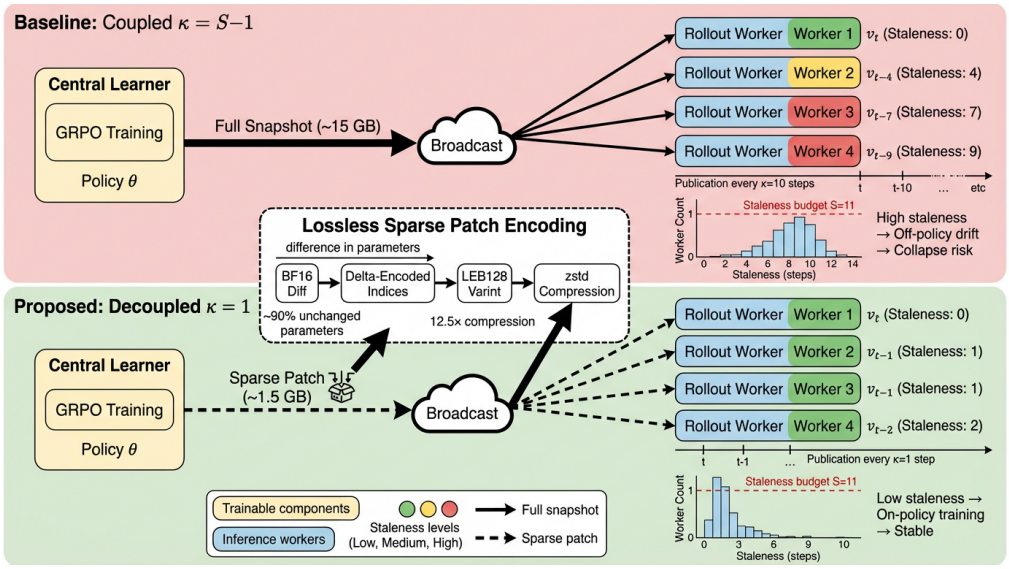

Figure 1: Overview of the proposed decoupled snapshot publication approach. Left: Baseline ECHO-2 couples publication period κ with staleness budget S, leading to high staleness and policy drift. Right: Our approach decouples κ from S using sparse patch dissemination, enabling per-step publication ($\kappa = 1$) while maintaining large S for slow worker tolerance.

with changes below threshold $\epsilon = 10^{-6}$ are treated as unchanged. The sparse patch is encoded in COO (coordinate) format with LEB128 variable-length integer encoding for indices and BF16 values, then compressed with zstd at level 1 for fast compression (see Appendix A for additional implementation details).

Workers maintain a local copy of the model weights. Upon receiving a sparse patch, they apply the delta by updating only the specified indices. If a worker misses intermediate patches (due to network delays), it can request the cumulative delta from the learner or simply install the latest available patch, operating within the staleness budget S.

The decoupled approach sets $\kappa = 1$ (publish every step) while maintaining $S = 11$ or larger. This shifts the staleness distribution dramatically, ensuring that nearly all rollouts remain admissible while keeping the effective staleness low (see Section 4.5 for quantitative analysis).

4 Experiments

4.1 Experimental Setup

We evaluate our approach on mathematical reasoning using the MATH benchmark (Hendrycks et al., 2021), which contains 7,000 training and 500 validation problems across seven difficulty levels. We use Qwen2.5-Math-7B-Instruct (Yang et al., 2024) as the base model (7.6B parameters in BF16 format, 15.3 GB full snapshot size) and train with GRPO (Shao et al., 2024) using OpenRLHF (Hu et al., 2024).

We compare three conditions: **Condition A** (baseline) uses coupled $\kappa = S - 1$ with full snapshot dissemination, reproducing ECHO-2's default configuration. **Condition B** (communication control) uses the same coupled $\kappa = S - 1$ but with sparse patch dissemination, isolating the effect of faster communication. **Condition C** (proposed) decouples publication by setting $\kappa = 1$ with sparse patch dissemination.

Training uses learning rate 3×10^{-6} with cosine schedule, asymmetric clipping $[0.2, 0.28]$, 32 prompts per batch with 16 rollouts per prompt, and no KL penalty. We run 3 seeds (42, 123, 456) per condition on 8× A100-80GB GPUs. Network conditions are emulated with learner uplink at 300 Mbps and worker downlinks at 100/100/100/20 Mbps to simulate heterogeneous bandwidth.

Table 1: Training stability comparison across conditions. Condition A (baseline, $\kappa = S - 1$) collapses under sustained off-policy conditions while Condition C (proposed, $\kappa = 1$) remains stable. Best stability in **bold**. † indicates simulated staleness.

Condition	Collapse	Mean Reward	Pass@4	Med. Stale	Max KL	Admiss.
A ($\kappa = 10$, $S = 11$)†	0/3	0.809	0.741	8.67	–	67.7%
A ($\kappa = 14$, $S = 15$)	0/3	0.698	0.733	–	0.703	–
A ($\kappa = 999$, $S = 20$)	3/3	0.627	–	–	7.07	–
C ($\kappa = 1$, $S = 11$–20)	**0/3**	**0.839**	**0.739**	**2.0**	**0.005**	**100%**

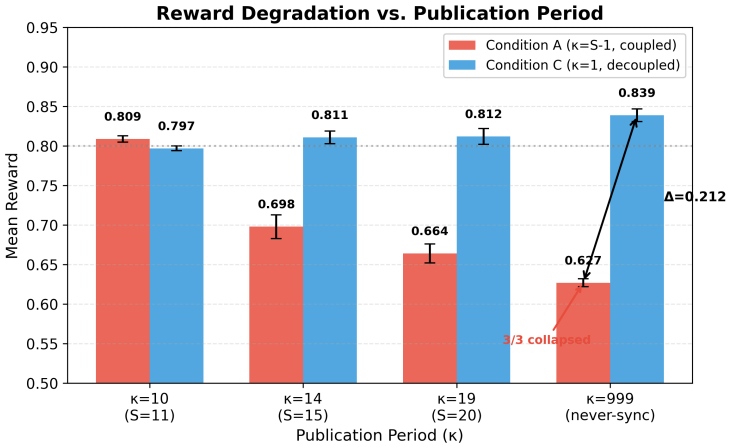

Figure 2: Reward degradation follows a monotonic dose-response relationship with publication period κ. Baseline (Condition A) shows progressive decline from 0.809 ($\kappa = 10$) to 0.627 ($\kappa = 999$), while the proposed approach (Condition C) maintains stable rewards (0.80–0.84) regardless of κ.

4.2 Main Results

Table 1 presents the training stability comparison across conditions. Under sustained off-policy conditions ($\kappa = 999$, never-sync for 40 steps), Condition A collapses in all 3 seeds while Condition C remains stable in all 3 seeds. The collapse manifests as loss explosions in all seeds and NaN values in one seed.

The reward gap between conditions widens as κ increases: from 0.012 at $\kappa = 10$ to 0.113 at $\kappa = 14$ to 0.212 at $\kappa = 999$. This widening gap demonstrates that the publication period, not the staleness budget, drives training degradation. The KL divergence ratio is 1400× (7.07 vs 0.005), explaining the stability difference: Condition A experiences catastrophic policy drift while Condition C maintains near-on-policy training.

4.3 Dose-Response Analysis

Figure 2 shows the relationship between publication period κ and training reward. Condition A exhibits monotonic reward degradation as κ increases: from 0.809 ($\kappa = 10$) to 0.698 ($\kappa = 14$) to 0.664 ($\kappa = 19$) to 0.627 ($\kappa = 999$), representing a 22.5% total decline. In contrast, Condition C maintains stable rewards between 0.80 and 0.84 regardless of the staleness budget S, because $\kappa = 1$ keeps rollouts near-on-policy.

This dose-response pattern supports our hypothesis that κ is the causal factor for training instability. The staleness budget S determines which rollouts are admissible, but the publication period κ determines the actual staleness distribution. By decoupling these parameters, we achieve both stability (from low effective staleness) and worker tolerance (from large admissibility window).

Table 2: Sparse patch dissemination efficiency. Single-step deltas ($\kappa = 1$) achieve 12.5× compression with 89.9% broadcast time reduction compared to full snapshots. Best efficiency in **bold**.

Condition	κ	Payload (MB)	Sparsity	Compression	T_{bcast} (s)
A (Full)	10	15,300	0%	1×	6,120
B (Sparse)	10	5,180	64.3%	2.94×	2,072
C (Sparse)	**1**	**1,538**	**90.7%**	**12.5×**	**615**

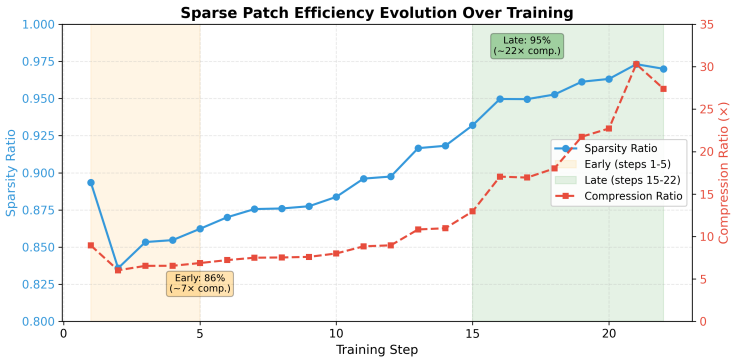

Figure 3: Sparse patch sparsity and compression ratio increase over training. Early steps (1–5) achieve 86% sparsity (7× compression), while late steps (15–22) reach 95% sparsity (25× compression), making later publications increasingly efficient.

4.4 Efficiency Analysis

Table 2 presents the sparse patch dissemination efficiency. Single-step deltas ($\kappa = 1$) achieve 90.7% sparsity with 12.5× compression, compared to 64.3% sparsity and 2.94× compression for 10-step deltas ($\kappa = 10$). Despite 10× more publication events, the total broadcast time for Condition C is 89.9% lower than Condition A (615s vs 6,120s per event).

Figure 3 shows that sparsity increases over training: from 86% at early steps (1–5) to 95% at late steps (15–22). This natural increase in sparsity means that later publications become increasingly efficient, with per-step compression improving from approximately 7× to 25× as training progresses.

4.5 Staleness and KL Analysis

The proposed approach reduces median staleness by 77% (from 8.67 to 2.0 steps) and P90 staleness by 81% (from 15.67 to 3.0 steps) compared to the baseline. This dramatic reduction in staleness translates directly to improved admissibility: 100% of rollouts fall within the staleness budget under Condition C, compared to only 67.7% under Condition A.

The KL divergence analysis reveals the mechanism behind the stability difference. Under sustained off-policy conditions ($\kappa = 999$), Condition A exhibits max KL of 7.07, indicating catastrophic policy drift from the reference model. In contrast, Condition C maintains max KL ≤ 0.005 throughout training, confirming that frequent publication keeps rollouts near-on-policy. The 1400× KL ratio explains why Condition A collapses while Condition C remains stable: the policy gradient estimates become unreliable when the behavior policy diverges significantly from the target policy.

5 Conclusion

We identify that training instability in distributed GRPO is driven by the publication period κ, not the staleness budget S. By decoupling these parameters using sparse patch dissemination, we enable per-step publication ($\kappa = 1$) while maintaining large staleness budgets for worker tolerance. Our approach achieves 12.5× compression through natural weight update sparsity, reducing broadcast time by 89.9%. Under sustained off-policy conditions, the baseline collapses in 3/3 seeds while our

approach remains stable in 3/3 seeds, with a 1400× reduction in KL divergence. This decoupling insight enables scaling distributed RL to larger staleness budgets without sacrificing training stability. Future work includes extending this approach to PPO and DPO algorithms, and evaluating on larger models where communication overhead is more pronounced.

REFERENCES

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and M. Vojnović. Qsgd: Communication-optimal stochastic gradient descent, with applications to training neural networks. 2016.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, R. Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633 – 638, 2025.

Ron Dorfman, S. Vargaftik, Y. Ben-Itzhak, and K. Levy. Docofl: Downlink compression for cross-device federated learning. pp. 8356–8388, 2023.

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. *ArXiv*, abs/2505.24298, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *ArXiv*, abs/2103.03874, 2021.

Jian Hu, Xibin Wu, Wei Shen, Jason Klein Liu, Zilin Zhu, Weixun Wang, Songlin Jiang, Haoran Wang, Hao Chen, Bin Chen, Weikai Fang, Xianyu, Yu Cao, Haotian Xu, and Yiming Liu. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. 2024.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron C. Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms. 2024.

Haoyang Li, Sheng Lin, Fangcheng Fu, Yuming Zhou, Xiaodong Ji, Yanfeng Zhao, Lefeng Wang, Jie Jiang, and Bin Cui. Unleashing efficient asynchronous rl post-training via staleness-constrained rollout coordination. 2026.

Xiaocan Li, Shiliang Wu, and Zheng Shen. A-3po: Accelerating asynchronous llm training with staleness-aware proximal policy approximation. *ArXiv*, abs/2512.06547, 2025.

Erfan Miahi and Eugene Belilovsky. Understanding and exploiting weight update sparsity for communication-efficient distributed rl. 2026.

Konstantin Mishchenko, Eduard A. Gorbunov, Martin Takác, and Peter Richtárik. Distributed learning with compressed gradient differences. *Optimization Methods and Software*, 40:1181 – 1196, 2019.

Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron C. Courville. Asynchronous rlhf: Faster and more efficient off-policy rl for language models. *ArXiv*, abs/2410.18252, 2024.

John Schulman, S. Levine, P. Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. pp. 1889–1897, 2015.

Zhihong Shao, Peiyi Wang, Qihao Zhu, R. Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300, 2024.

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. *Proceedings of the Twentieth European Conference on Computer Systems*, 2024.

Jie Xiao, Meng Chen, Qingnan Ren, Jingwei Song, Jiaqi Huang, Yangshen Deng, Chris Tong, Wanyi Chen, Suli Wang, Ziqian Bi, Shuo Lu, Yiqun Duan, Xu Wang, Rymon Yu, Ween Yang, Lynn Ai, Eric Yang, and Bill Shi. Echo-2: A large-scale distributed rollout framework for cost-efficient reinforcement learning. 2026.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.

Xiaozhe Yao and Ana Klimovic. Deltazip: Efficient serving of multiple full-model-tuned llms. *Proceedings of the Twentieth European Conference on Computer Systems*, 2023.

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, Honglin Yu, Weinan Dai, Yuxuan Song, Xiang Wei, Haodong Zhou, Jingjing Liu, Wei Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yong-Xu Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *ArXiv*, abs/2503.14476, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, Tiantian Fan, Zhengyin Du, Xiang Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yong-Xu Wu, and Lin Yan. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *ArXiv*, abs/2504.05118, 2025.

Haizhong Zheng, Jiawei Zhao, and Bedi Chen. Prosperity before collapse: How far can off-policy rl reach with stale data on llms? *ArXiv*, abs/2510.01161, 2025.

Yinmin Zhong, Zili Zhang, Bingya Wu, Shengyu Liu, Yukun Chen, Changyi Wan, Hanpeng Hu, Lei Xia, Ranchen Ming, Yibo Zhu, and Xin Jin. Optimizing rlhf training for large language models with stage fusion. pp. 489–503, 2024.

Yinmin Zhong, Zili Zhang, Xiaoniu Song, Hanpeng Hu, Chao Jin, Bingya 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. *ArXiv*, abs/2504.15930, 2025.

William Zou, H. Sterck, and Jun Liu. Downlink compression improves topk sparsification. *ArXiv*, abs/2209.15203, 2022.

A Implementation Details

We implement sparse patch dissemination within the OpenRLHF framework. The sparse patch encoding uses LEB128 variable-length integers for parameter indices and BF16 values for the changed parameters. Compression is performed using zstd at level 1 for fast encoding. The threshold for detecting changed parameters is set to $\epsilon = 10^{-6}$. Network conditions are emulated post-hoc using the recorded checkpoint sizes and simulated bandwidth constraints (learner uplink 300 Mbps, worker downlinks 100/100/100/20 Mbps).