A-3PO: ACCELERATING ASYNCHRONOUS LLM TRAINING WITH STALENESS-AWARE PROXIMAL POLICY APPROXIMATION

Xiao-Can (Bruce) Li*, Shi-Liang (Bruce) Wu*& Zheng Shen
Huawei Canada
hsiaotsan.li@alumni.utoronto.ca
{okwsl201210, zhengshencn}@gmail.com

ABSTRACT

Decoupled PPO has been a successful reinforcement learning (RL) algorithm to deal with the high data staleness under the asynchronous RL setting. Decoupled loss used in decoupled PPO improves coupled-loss style of algorithms' (e.g., standard PPO, GRPO) learning stability by introducing a proximal policy to decouple the off-policy correction (importance weight) from the policy update constraint (trust region). However, the proximal policy requires an extra forward pass through the model at each training step, creating a computational overhead for large language models training. We observe that since the proximal policy only serves as a trust region anchor between the behavior and target policies, we can approximate it through simple interpolation without explicit computation. We call this approach A-3PO (APproximated Proximal Policy Optimization). A-3PO eliminates this overhead, accelerating training by 1.8$\times$ speedup while maintaining comparable performance. Code & off-the-shelf example are contributed to the open-source RL training system AReaL at: https://github.com/inclusionAI/AReaL/blob/v1.0.0.rc1/docs/algorithms/prox_approx.md

1 INTRODUCTION

Reinforcement learning (RL) has become a central approach to improve the reasoning capabilities of large language models (LLMs) Shao et al. (2024); Yang et al. (2025); Zheng et al. (2025); Yu et al. (2025); Ouyang et al. (2022), with extensive surveys covering RL from human feedback methods and workflows Kaufmann et al. (2025); Dong et al. (2024); Wang et al. (2024) and various alternative approaches including AI feedback Lee et al. (2024) and safety considerations Dai et al. (2024). Among RL algorithms, Proximal Policy Optimization (PPO) Schulman et al. (2017) has emerged as the dominant method due to its stable trust-region constraints, building upon earlier trust region methods like TRPO Schulman et al. (2015). However, standard PPO performs rollout-then-training loop, i.e., the training stage must wait until the rollout stage collects predefined number of episodes, limiting throughput (measured by the number of environment steps per unit of time) and underutilizes computational resources.

To improve the throughput and computational resources utilization, asynchronous RL Fu et al. (2025); Wang et al. (2025); Sheng et al. (2024); Feng et al. (2025); Noukhovitch et al. (2025); Wu et al. (2025); Espeholt et al. (2018); Lan et al. (2025); Duan et al. (2024); Cohen et al. (2025) treats rollout and training as two independent engines, which can be executed in parallel. Nevertheless, the target policy on the training engine can be several updates ahead of the behavior policy on the rollout engine. Such staleness (off-policyness) caused by asynchronous RL setting could lead to severe learning instability in standard PPO. To mitigate this, decoupled PPO Hilton et al. (2022) improves the learning stability by introducing a proximal policy that decouples the off-policy correction (importance weight) from the policy update constraint (trust region). Decoupled loss empirically demonstrates improved learning stability in Atari games when high off-policyness exists.

*The first and second authors have equal contribution.

Apart from Atari games, AReaL Fu et al. (2025), an LLM post training framework, demonstrated superior learning stability of decoupled loss on LLM reasoning tasks under high off-policyness setting. Thanks to asynchronous RL setup, AReaL also achieved up to 2.77× training speedup. However, the proximal policy in decoupled loss requires an extra forward pass through the neural network at each training step, which is expensive for autoregressive LLMs. This overhead limits the potential speedups from asynchronous training.

This raises a natural question: do we really need to compute the proximal policy explicitly? Looking at the objective from first principles, the proximal policy simply serves as a trust region anchor: it does not need to be computed from the network, but it just needs to lie somewhere between the behavior and target policies to prevent extreme importance weights. This insight leads to our solution: instead of computing the proximal policy through a forward pass, we approximate it by interpolating between the behavior policy and the target policy in log-probability space. Our staleness-aware interpolation weighs fresher data more heavily, maintaining the stabilizing effect of decoupled loss while eliminating the computational overhead.

Our contributions are threefold:

1. A staleness-aware proximal probability interpolation method that eliminates the computation cost of proximal policies in decoupled loss while retaining PPO's trust-region structure.
2. Empirical evaluation across two model scales (1.5B and 8B parameters) demonstrates that our method achieves up to 1.8× speedup in training time while maintaining comparable task performance and superior training stability compared to both the standard decoupled PPO and synchronous training baselines, with particular advantages at larger scales.
3. Open-source implementation, providing an efficient large-scale asynchronous RL-based LLM post training algorithm.

2 PRELIMINARY

2.1 FROM COUPLED LOSS TO DECOUPLED LOSS

Among coupled-loss RL algorithms, the standard Proximal Policy Optimization (PPO) Schulman et al. (2017) has become one of the most dominant methods, thanks to its stable trust-region constraints that prevent destructive policy updates. In standard PPO, the clipped objective is:

$$L^{\text{CLIP}}_{\text{coupled}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{1}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ is the importance weight and $\hat{A}_t$ is the advantage estimate. Here, the old policy π_{old} serves two purposes simultaneously: it provides importance sampling weights Papini et al. (2024); De Asis et al. (2023) to correct for off-policy data, and it defines the trust region that constrains how far the new policy can deviate.

However, this coupled role becomes problematic in asynchronous RL settings, where the behavior policy used for data collection may be several updates behind the current policy. When training on stale data, using the behavior policy as the trust region anchor pulls the current policy towards outdated, potentially lower-quality policies, leading to learning instability.

Decoupled loss Hilton et al. (2022) addresses this issue with a key insight: these two roles of π_{old} can and should be separated. For importance sampling, we must use the actual behavior policy π_{behav} that generated the data. But for trust region control, we can use some recent policy π_{prox} as the proximal anchor. The decoupled clipped objective becomes:

$$L^{\text{CLIP}}_{\text{decoupled}}(\theta) = \mathbb{E}_t \Big[\underbrace{\frac{\pi_{\text{prox}}(a_t|s_t)}{\pi_{\text{behav}}(a_t|s_t)}}_{\text{Importance Weight}} \min \Big(\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{prox}}(a_t|s_t)}\hat{A}_t,\ \text{clip}\Big(\underbrace{\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{prox}}(a_t|s_t)}}_{\text{Trust Region Anchor}},\ 1-\epsilon,\ 1+\epsilon \Big)\hat{A}_t \Big) \Big] \tag{2}$$

By using a more recent policy π_{prox} as trust region anchor, the trust region now constrains updates around a higher-quality policy, while still correctly accounting for the off-policy nature of the data

through π_{behav}. This decoupling is crucial for asynchronous RL systems like AReaL Fu et al. (2025), which achieves significant training speedup by completely separating generation and training phases.

2.2 The Price of Decoupled Loss

While decoupled loss improves stability, it comes with a computational price. At the start of each training step, we must perform a forward pass through the model to compute π_{prox}. This proximal policy is then frozen (detached from gradient updates) and used as the trust region anchor for all subsequent mini-batch updates within that training step. This overhead becomes significant for large language models where a single forward pass can take 10 seconds or longer, depending on the input size, model size, and hardware specification. The timing is demonstrated in Fig. 1.

3 Methodology: A-3PO

In our proposed method, the objective function is still Eq. 2, but the proximal policy is interpolated between the behavior policy and target policy at log scale, weighted by the staleness-aware coefficient, as defined in Eq. 3. The more the target policy is ahead of the behavior policy (high staleness), the closer the approximated proximal policy is to the target policy:

$$\log \pi_{\text{prox}} = \alpha \log \pi_{\text{behav}} + (1 - \alpha) \log \pi_\theta, \tag{3}$$

Input arguments $(a|s)$ are omitted for brevity. The main reason to perform interpolation in log-probability space rather than probability space is that it maintains numerical stability: log-probabilities avoid underflow issues that arise when dealing with very small probability values common in large action spaces. Practically, the inference engine, e.g., SGLang Zheng et al. (2024) and vLLM Kwon et al. (2023), provides token log-probabilities by default.

Here, α is a staleness-aware coefficient depending on the staleness d (the training step difference between target and behavior policies). $v(\pi)$ denotes the training step of policy π:

$$d = v(\pi_\theta) - v(\pi_{\text{behav}}), \quad \alpha = \begin{cases} 0, & d = 0, \\ \dfrac{1}{d}, & d \geq 1. \end{cases} \tag{4}$$

When $d = 0$, it recovers the standard PPO where the target policy equals the behavior policy, hence the approximated proximal policy is exactly the behavior policy.

When $d \geq 1$, the coefficient α monotonically decreases as the staleness d increases, approximating the proximal policy more with the target policy, and give less weights to the behavior policy.

3.1 Theoretical Stability Analysis

3.1.1 Sandwich Property

Thanks to the interpolation, $\pi_{\text{prox}}(a|s)$ is bounded:

$$\min\{\pi_{\text{behav}}(a|s), \pi_\theta(a|s)\} \leq \pi_{\text{prox}}(a|s) \leq \max\{\pi_{\text{behav}}(a|s), \pi_\theta(a|s)\}. \tag{5}$$

This property guarantees that π_{prox} serves as a valid trust-region anchor under policy staleness.

3.1.2 Contractive Stability

The staleness-aware proximal policy defined by log-linear interpolation admits a simple closed-form importance ratio:

$$r(a|s) = \left(\frac{\pi_\theta(a|s)}{\pi_{\text{behav}}(a|s)}\right)^\alpha \tag{6}$$

This form implies that importance weights are contractively scaled as staleness increases, preventing extreme ratios and ensuring stable trust-region updates. Moreover, raising importance weights to a power $\alpha < 1$ provably vanishes their variance, leading to more stable and fewer clipped updates in PPO-style objectives. A unified theoretical analysis establishing bounded updates and contractive stability is provided in Appendix A.

3.2 IMPLEMENTATION

The core implementation of our staleness-aware proximal policy approximation is remarkably simple. In Listing 1 we show the key computation in PyTorch, extracted from our open-source implementation in the AReaL framework.

```python
def compute_prox_logp_approximation(
    old_logp: torch.Tensor,       # log π_behav
    logprobs: torch.Tensor,       # log π_θ
    versions: torch.Tensor,       # v(π_behav) per token
    current_version: int,         # v(π_θ)
) -> torch.Tensor:
    """Approximate proximal policy log-probabilities."""
    # Extract version information
    v_behave = versions.float()
    v_theta = float(current_version)

    # Compute staleness: d = v(π_θ) − v(π_behav)
    staleness = v_theta - v_behave

    # Compute staleness-aware coefficient α
    # α = 0 when d = 0, α = 1/d when d ≥ 1
    alpha = torch.where(
        staleness >= 1,
        1.0 / staleness,
        torch.zeros_like(v_behave),
    )

    # Log-linear interpolation: log π_prox = α log π_behav + (1 − α) log π_θ
    prox_logp = alpha * old_logp + (1 - alpha) * logprobs

    return prox_logp
```

Listing 1: Core implementation of staleness-aware proximal policy approximation.

The implementation highlights the efficiency of our approach: computing π_{prox} requires only element-wise arithmetic operations on tensors that are already available from the training loop. No additional forward pass through the neural network is needed, making this approximation essentially free compared to the 10-second forward pass required for explicit computation.

4 EXPERIMENTS

4.1 TRAINING DETAILS

In this work, we focus on mathematical reasoning tasks to evaluate our algorithm, which can be readily applied to other tasks. We adopt the AReaL framework Fu et al. (2025) for training. We use decoupled PPO Hilton et al. (2022) and vanilla GRPO Shao et al. (2024) as our baseline algorithms and estimate advantages using group reward normalization.

We conduct experiments with two different model-dataset configurations:

Setup 1: Qwen2.5-1.5B-Instruct on GSM8K. We train and evaluate on GSM8K Cobbe et al. (2021), a dataset of 8.5K grade school math problems that require 2-8 steps of multi-step reasoning using basic arithmetic operations. For rollout, the prompt batch size is 256 and we sample 4 responses for each prompt. We set the maximum response length as 1,024 tokens, and max tokens per mini batch is 10,240.

Setup 2: Qwen3-8B on DAPO-Math-17k. We also use the base model Qwen3-8B Yang et al. (2025) trained and evaluated on the DAPO-Math-17k dataset Yu et al. (2025). For rollout, the prompt batch size is 128 and we sample 4 responses for each prompt. We set the maximum response length as 2,048 tokens, and max tokens per mini batch is 10,240.

Common hyperparameters. For both setups, we use the Adam optimizer Kingma & Ba (2015) with a constant learning rate of 8.5×10^{-6}. For training, the number of mini batches is set to 4, i.e., 4 gradient updates for each training step. For sampling parameters, the temperature is set to 1.0, and top-p is set 1, and top-k is set to use all vocabularies.

4.2 Results & Discussion

We evaluate our A-3PO method (labeled as "loglinear") against two baselines: the asynchronous decoupled GRPO with proximal policy recomputing Hilton et al. (2022) (labeled as "recompute") and synchronous GRPO (labeled as "sync"). The synchronous baseline represents the standard coupled-loss approach without asynchronous training, serving as a reference for comparing both decoupled methods. We present results across both experimental setups that demonstrate consistent improvements in computational efficiency and training stability.

4.2.1 Computational Efficiency

Our staleness-aware approximation method dramatically reduces the computational cost of proximal policy evaluation. Fig. 1 shows the wall-clock time required to compute log probabilities of the proximal policy at each training step. The loglinear approximation method achieves near-zero computation time (mean: 0.0012 seconds), while the full recompute approach requires approximately 4 to 8 seconds per step depending on the setting, and the sync method does not require proximal policy computation. This represents at least 3,000$\times$ speedup in proximal policy's log probability computation for decoupled methods, translating directly to reduced training time.

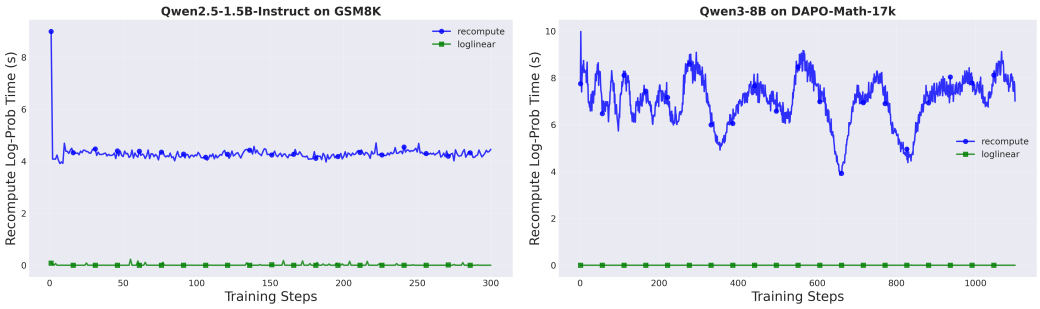

Figure 1: Log probability computation time comparison. The loglinear method achieves near-instantaneous computation compared to the 10-second forward pass required by recomputing. Sync does not require this computation.

As shown in Fig. 2, the loglinear method consistently achieves faster training while maintaining comparable task performance to both baselines. For a fair comparison, the same training epochs are used for all methods. For both setups, our A-3PO method converges to similar final task rewards, demonstrating that our approximation maintains task performance while significantly reducing the training time across different model scales and datasets.

To further validate the training progress and model quality, we evaluate the policies on held-out test prompts during training. Fig. 3 shows the evaluation reward trajectories over training steps. In Setup 1, all three methods achieve comparable final evaluation rewards (gap $< 1\%$), demonstrating that the loglinear approximation maintains task performance. In Setup 2, the asynchronous methods (loglinear and recompute) significantly outperform the baseline "sync", converging to evaluation rewards around 0.63 compared to 0.44 for sync method. This substantial gap highlights the benefits of asynchronous training with decoupled loss at larger model scales. Notably, the loglinear method achieves this performance without the computational overhead of explicit proximal policy computation, making it the most efficient approach overall.

4.2.2 Training Stability Analysis

Beyond computational efficiency, we analyze whether our A-3PO maintains the stabilizing properties of decoupled loss across both experimental setups. Fig. 4 shows that all three methods exhibit

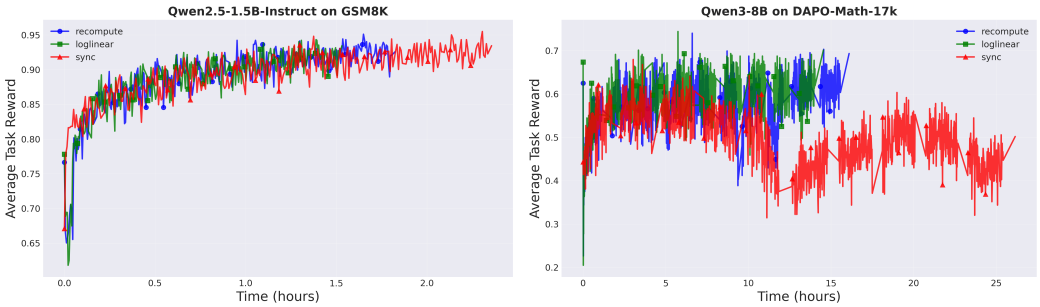

Figure 2: Training progress (average task reward vs. wall-clock time). All curves use the same training epochs. Asynchronous training with loglinear approximation is fastest.

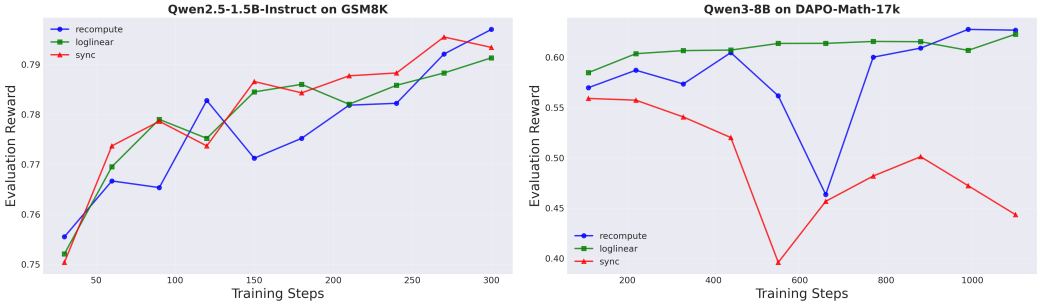

Figure 3: Evaluation reward on held-out test prompts. Setup 1: all methods converge similarly. Setup 2: asynchronous methods substantially outperform sync, demonstrating decoupled loss effectiveness at larger scales.

similar entropy decay patterns. The baseline "sync" and both decoupled methods show healthy entropy decay, indicating stable policy optimization. This demonstrates that our approximation preserves healthy exploration dynamics across different model scales and datasets, comparable to both the recompute method and the synchronous baseline.

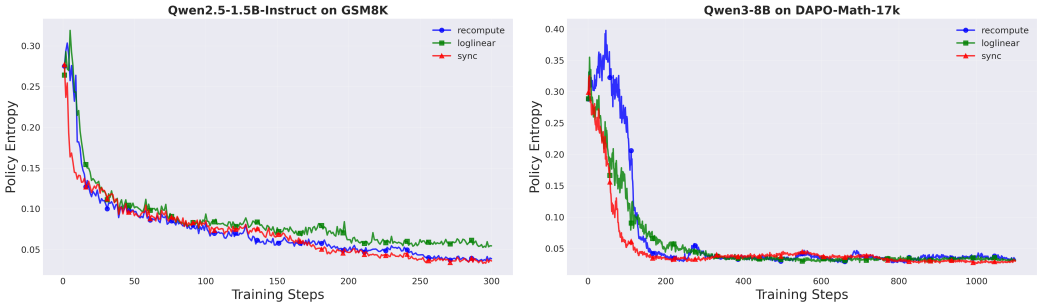

Figure 4: Policy entropy over training steps. All methods show healthy entropy decay, indicating stable policy optimization.

Importance weight statistics provide further evidence of stability across both setups. Fig. 5 displays the maximum and minimum importance weights during training for the two decoupled methods (note that the sync method uses coupled loss and does not compute separate importance weights). Notably, the recompute approach exhibits very high importance weights, indicating that the recomputed proximal policy becomes unreliable at larger model scales. In contrast, the loglinear approximation consistently maintains more balanced importance sampling that avoids extreme weight values which can destabilize training.

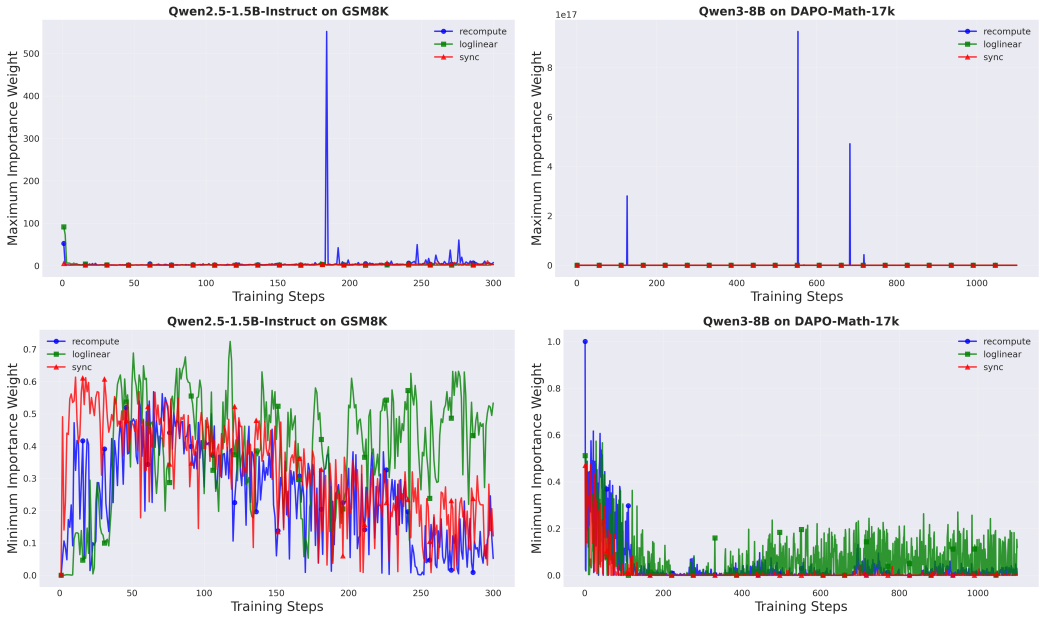

Figure 5: Importance weight statistics (top: max; bottom: min). Loglinear shows more controlled weights. Setup 2: recompute produces very high weights, indicating instability at larger scales.

Finally, Fig. 6 compares the number of clipped tokens across all three methods. The recomputing and sync methods clip significantly more tokens than the loglinear method (A-3PO), indicating they trigger trust region constraints more frequently. The loglinear method shows least clipping behavior, suggesting smoother, more stable policy updates that naturally stay within trust region bounds, indicating higher sample-efficiency.

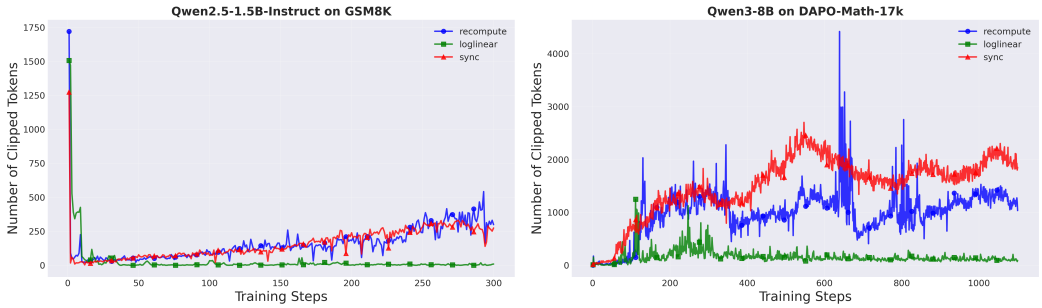

Figure 6: Number of clipped tokens per training step. Loglinear clips the least, indicating less token waste.

4.2.3 Benchmark Evaluation

Table 1 summarizes the final evaluation rewards and total training times across all methods and setups. In Setup 1, all three methods achieve comparable final evaluation rewards (0.791–0.797) on GSM8K test dataset, with loglinear completing training in 1.53 hours compared to 1.82 hours for recompute (1.2× speedup) and 2.36 hours for sync (1.5× speedup). In Setup 2, the asynchronous methods (recompute and loglinear) significantly outperform the sync baseline in both final reward on DAPO-Math-17k test dataset (0.623–0.627 vs 0.443) and training time. Notably, loglinear method completes training in 14.54 hours compared to 16.10 hours for recompute (1.1× speedup) and 26.15 hours for sync (1.8× speedup), while maintaining comparable performance to recompute. These results demonstrate that our approximation method consistently delivers efficiency benefits across model scales while maintaining or improving task performance.

Table 1: Final evaluation reward and training time. Setup 1: Qwen2.5-1.5B-Instruct on GSM8K. Setup 2: Qwen3-8B on DAPO-Math-17k.

Setup	Method	Final Eval Reward	Training Time (hours)
Setup 1	Sync GRPO	0.793	2.36
	Recompute	0.797	1.82
	Loglinear (A-3PO)	0.791	**1.53**
Setup 2	Sync GRPO	0.443	26.15
	Recompute	0.627	16.10
	Loglinear (A-3PO)	0.623	**14.54**

Table 2: Benchmark evaluation for Setup 2: Qwen3-8B on DAPO-Math-17k.

Method	AIME24 pass@1	MATH500 pass@1	Average
Sync GRPO	$40.00 \pm 9.10\%$	$46.80 \pm 2.23\%$	43.40%
Recompute	$66.67 \pm 8.75\%$	$62.80 \pm 2.16\%$	64.74%
Loglinear (A-3PO)	$\mathbf{66.67 \pm 8.75\%}$	$\mathbf{66.60 \pm 2.11\%}$	$\mathbf{66.64\%}$

Furthermore, we evaluate the trained model from Setup 2 on two challenging mathematical reasoning benchmarks: AIME2024[1] and MATH500[2]. Table 2 presents the pass@1 accuracy for each method on these benchmarks. The results demonstrate that our proposed A-3PO method (loglinear) achieves the best performance over the "recompute" and synchronous baseline.

5 Conclusion

Decoupled policy optimization algorithms have proven effective for handling data staleness in asynchronous RL systems, but their reliance on explicit proximal policy computation creates a computational bottleneck that limits their practical benefits. In this work, we addressed this limitation through a simple yet principled observation: the proximal policy serves merely as a trust region anchor between the behavior and target policies, and thus does not require expensive neural network evaluation. Instead, it just needs to lie somewhere in between.

Our staleness-aware approximation method implements this insight by interpolating between behavior and target policies in log-probability space, with fresher data weighted more heavily. We evaluated our approach across two experimental setups spanning different model scales: Qwen2.5-1.5B-Instruct on GSM8K and Qwen3-8B on DAPO-Math-17k. Comparing against both the explicit recompute baseline and synchronous GRPO baseline, our results demonstrate substantial practical benefits: up to $1.8\times$ speedup in training time while maintaining comparable task performance across all three methods. More importantly, our approximation exhibits improved training stability compared to explicit computation, with more controlled importance weights and fewer clipped tokens. Notably, at larger model scales (Qwen3-8B), the recompute method exhibits very high importance weights indicating instability, while our approximation maintains stable behavior, suggesting that simpler can indeed be better and more reliable.

Beyond computational efficiency, this work highlights a broader principle: when designing RL algorithms for large-scale systems, we should question which components truly require expensive computation and which can be approximated from first principles. Our method applies to any decoupled policy optimization approach, not just PPO, making asynchronous RL more practical for training large language models and other computationally demanding domains.

Acknowledgments

We acknowledge the use of Huawei Ascend NPUs for training the model reported in this paper.

[1] https://huggingface.co/datasets/math-ai/aime24

[2] https://huggingface.co/datasets/math-ai/math500

REFERENCES

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. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168.

Taco Cohen, David W. Zhang, Kunhao Zheng, Yunhao Tang, Remi Munos, and Gabriel Synnaeve. Soft policy optimization: Online off-policy RL for sequence models, 2025.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe RLHF: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations (ICLR)*, 2024.

Kristopher De Asis, Eric Graves, and Richard S. Sutton. Value-aware importance weighting for off-policy reinforcement learning. In *Conference on Lifelong Learning Agents (CoLLAs)*, volume 232, pp. 2086–2112. PMLR, 2023.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF workflow: From reward modeling to online RLHF. *Transactions on Machine Learning Research*, 2024.

Jiangfei Duan, Shuo Zhang, Zerui Wang, Lijuan Jiang, Wenwen Qu, Qi Hu, Guoteng Wang, Qizhen Weng, Hang Yan, Xingcheng Zhang, Xipeng Qiu, Dahua Lin, Yonggang Wen, Xin Jin, Tianwei Zhang, and Peng Sun. Efficient training of large language models on distributed infrastructures: A survey, 2024.

Lasse Espeholt, Hubert Soyer, Remi 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 *International Conference on Machine Learning (ICML)*, volume 80, pp. 1407–1416. PMLR, 2018.

Laingjun Feng, Chenyi Pan, Xinjie Guo, Fei Mei, Benzhe Ning, Jianxiang Zhang, Xinyang Liu, Beirong Zhou, Zeng Shu, Chang Liu, Guang Yang, Zhenyu Han, Jiangben Wang, and Bo Wang. Mindspeed rl: Distributed dataflow for scalable and efficient rl training on ascend npu cluster, 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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems*, 2025. doi: 10.48550/arXiv.2505.24298.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17086–17098. Curran Associates, Inc., 2022.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. doi: 10.48550/arXiv.1412.6980.

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 paged attention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. doi: 10.1145/3600006.3613165.

Guangchen Lan, Dong-Jun Han, Abolfazl Hashemi, Vaneet Aggarwal, and Christopher G. Brinton. Asynchronous federated reinforcement learning with policy gradient updates: Algorithm design and convergence analysis. In *International Conference on Learning Representations (ICLR)*, 2025.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Machine Learning (ICML)*, 2024.

Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. Asynchronous RLHF: Faster and more efficient off-policy RL for language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Matteo Papini, Giorgio Manganini, Alberto Maria Metelli, and Marcello Restelli. Policy gradient with active importance sampling. *Reinforcement Learning Journal*, 1, 2024.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, volume 37, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.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, 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. *arXiv preprint arXiv: 2409.19256*, 2024. doi: 10.48550/arXiv.2409.19256.

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Jin, Senjie Xiong, Mengyu Fang, Zhiheng Zhou, Yuhao Qiao, Xuanjing Xie, Zhongyu Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. Secrets of RLHF in large language models part II: Reward modeling, 2024.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025. doi: 10.48550/arXiv.2506.06122.

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.

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, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, 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, 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, 2025.

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.

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, 2024.

A Theoretical Analysis

Theorem 1 (Sandwich Property and Contractive Stability of Staleness-Aware Proximal Policy Approximation)**.** *Let $\pi_{\text{behav}}(\cdot|s)$ be the behavior policy, $\pi_\theta(\cdot|s)$ be the target policy, and define the staleness-aware proximal policy*

$$\log \pi_{\text{prox}}(a|s) = \alpha \log \pi_{\text{behav}}(a|s) + (1-\alpha)\log \pi_\theta(a|s), \quad \alpha = \begin{cases} 0, & d = 0, \\ \frac{1}{d}, & d \geq 1, \end{cases} \tag{7}$$

where d denotes policy staleness.

Define the importance ratios

$$w(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\text{behav}}(a|s)}, \qquad r(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\text{prox}}(a|s)}. \tag{8}$$

Assume $w(a|s)$ has finite second moment under $\pi_{\text{behav}}(\cdot|s)$. Then the following properties hold for all states s and actions a:

1. ***Trust-region sandwich property***

$$\min\{\pi_{\text{behav}}(a|s), \pi_\theta(a|s)\} \leq \pi_{\text{prox}}(a|s) \leq \max\{\pi_{\text{behav}}(a|s), \pi_\theta(a|s)\}. \tag{9}$$

2. ***Contractive stability***

$$r(a|s) = w(a|s)^\alpha \tag{10}$$

In particular, $\lim_{d\to\infty} r(a|s) = \lim_{\alpha\to 0} r(a|s) = 1$.

For variance, it vanishes as the staleness increases:

$$\lim_{d\to\infty} \text{Var}_{a\sim\pi_{\text{behav}}}\big[r(a|s)\big] = 0. \tag{11}$$

Sandwich property. For any $x, y > 0$ and $\alpha \in [0, 1]$,

$$\min(x, y) \leq x^\alpha y^{1-\alpha} \leq \max(x, y), \tag{12}$$

substituting $x = \pi_{\text{behav}}(a|s)$ and $y = \pi_{\text{theta}}(a|s)$ establishes the sandwich property of π_{prox}.

Q.E.D. □

Contractive stability. Exponentiating the definition of π_{prox} yields

$$\pi_{\text{prox}}(a|s) = \pi_{\text{behav}}(a|s)^\alpha \pi_\theta(a|s)^{1-\alpha}. \tag{13}$$

Therefore,

$$r(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\text{prox}}(a|s)} = \left(\frac{\pi_\theta(a|s)}{\pi_{\text{behav}}(a|s)}\right)^\alpha = w(a|s)^\alpha, \tag{14}$$

which proves the contractive form of the update and $\lim_{d\to\infty} r(a|s) = 1$.

To prove the vanishing variance, first, observe the pointwise convergence. For any fixed $w > 0$, as $\alpha \to 0$, we have $w^\alpha \to 1$. Consequently,

$$\lim_{\alpha\to 0} (w^\alpha - 1)^2 = 0. \tag{15}$$

To interchange the limit and the expectation, we apply the Dominated Convergence Theorem (DCT). We must find an integrable random variable Y such that $(w^\alpha - 1)^2 \leq Y$ for all $\alpha \in (0, 1]$. We propose the dominating function $Y = (w - 1)^2$. We verify this bound in two cases:

- **Case 1:** $w \geq 1$**.** Since $\alpha \in (0, 1]$, the function $f(x) = x^\alpha$ is concave and bounded by the identity for $x \geq 1$. Thus, $1 \leq w^\alpha \leq w$. Subtracting 1 yields $0 \leq w^\alpha - 1 \leq w - 1$. Squaring both sides, we obtain $(w^\alpha - 1)^2 \leq (w - 1)^2$.
- **Case 2:** $0 \leq w < 1$**.** Since $\alpha \in (0, 1]$, the root brings the value closer to 1, i.e., $w \leq w^\alpha \leq 1$. Thus, the distance to 1 satisfies $|w^\alpha - 1| = 1 - w^\alpha \leq 1 - w = |w - 1|$. Squaring both sides yields $(w^\alpha - 1)^2 \leq (w - 1)^2$.

Therefore, for all $w \geq 0$, we have $(w^\alpha - 1)^2 \leq (w - 1)^2$.

Next, we check the integrability of the dominating function. Expanding the term, we have $\mathbb{E}[(w - 1)^2] = \mathbb{E}[w^2] - 2\mathbb{E}[w] + 1$. By assumption, the second moment $\mathbb{E}[w^2]$ is finite, which implies $\mathbb{E}[w]$ is also finite. Thus, $\mathbb{E}[(w - 1)^2] < \infty$.

Since the sequence converges pointwise to 0 and is dominated by an integrable variable, the DCT implies:

$$\lim_{\alpha \to 0} \mathbb{E}[(w^\alpha - 1)^2] = \mathbb{E}\left[\lim_{\alpha \to 0} (w^\alpha - 1)^2\right] = 0. \tag{16}$$

This implies the variance vanishes as $\alpha \to 0$.

Q.E.D. □