MetroRLHF: Enabling Memory-Effective Training for On-Policy RLHF via Adaptive Sequence Streaming

Wei Cui
Microsoft Research Asia
Vancouver, Canada
weicu@microsoft.com

Peng Cheng
Microsoft Research Asia
Redmond, USA
pengc@microsoft.com

Abstract

Reinforcement learning from human feedback (RLHF [1, 2]) has become the standard post-training technique for endowing large language models (LLMs) with helpful, harmless, and intent-consistent behavior. In practice, however, its adoption is hampered by prohibitive memory consumption during the phase of the policy-model update, especially when training on long-form generation tasks. In this paper, we propose MetroRLHF, a memory-efficient, on-policy RLHF approach that exploits the inference-time computations to reduce the training-time memory budget and to skip unnecessary work. By re-using the inference-phase materialized K, V context, the inter-token dependencies are freely removed that normally force the entire sequence to train in parallel. Building upon fine-grained subsequence streaming, RLHF can train the productive tokens in an effective manner. This yields a training pipeline that matches the exact behavior of conventional full-sequence RLHF while using less memory and incurring no arithmetic recomputation. Experiments on the Qwen-3 models demonstrate that MetroRLHF rescheduled algorithm reduces peak training memory usage to $\frac{1}{3.8} \sim \frac{1}{5.9}$, enabling not only memory-effective but also semantic-reliable fine-tuning for LLM.

1 Introduction

LLMs have demonstrated remarkable capabilities across a wide range of natural language processing (NLP) tasks. To better align these models with desired human preferences, RLHF has emerged as the prevailing paradigm for post-training, and it is widely used in modern systems such as ChatGPT [3], DeepSeek [4], and Gemini [5]. Despite its effectiveness, RLHF is substantially expensive.

Generally, the RLHF pipeline consists of three phases: *1. inference phase* (for experience resampling) $\rightarrow$ *2. rewarding phase* (scope-specific scoring for the experience) $\rightarrow$ *3. training phase* (updating the policy model). Among these, the training phase consumes a huge amount of memory. While the memory required during inference [6, 7, 8] is typically on the order of the model-parameter size, the training-phase memory footprint grows additionally with the length of the entire trajectory sequence L. In scenarios such as multi-turn dialogues or long-form generation (e.g., Long Chain-of-Thought) [9], L can easily reach tens or even hundreds of thousands of tokens, causing the memory demand to spike dramatically. Under such conditions, practitioners are compelled to make difficult trade-offs (e.g., truncating the trajectory [10, 11, 12, 13], resampling sparsification [14], or gradient checkpoint [15]). Heavily relying on various trade-offs may reduce the effective context available to the learner and increase the risk of degrading sampling quality and performance, unless multiple GPUs are introduced to mitigate memory consumption [16, 17, 18, 19], though this is uneconomical.

In this work, we introduce ***MetroRLHF*** to address the stated challenge: it eliminates the training-phase memory blow-up at low cost while preserving the exact computational semantics of the full-trajectory training, and further enables reliable, fine-grained, and adaptive scheduling in RLHF.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

2 Memory dependency and observation for RLHF post-training

Unlike inference where tokens are generated auto-regressively one at a time and past hidden states are discarded immediately, the training phase must traverse and back-propagate through *"the entire sequence in a single pass"* to avoid recomputation. This difficulty occurs only during training.

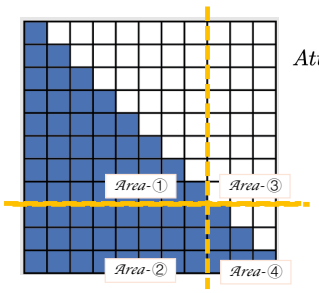
Specifically, when a full training sequence $\mathcal{S}_{eq} = (s_0, \ldots, s_{L-1})$ is processed, the training scheduler must respect 3 constraints: ***(a). Forward-pass context propagation:*** unless all tokens are computed simultaneously, token s_i (the "effect") must be computed after token s_j (the "cause") with $j < i$ to avoid increased context computation complexity. ***(b). Backward-pass causality feedback:*** the gradient of token s_j (the "cause") depends reversely on token s_i (the "effect"). ***(c). Inherent mutual dependencies between forward-pass and backward-pass:*** activation states have to be produced first by the forward pass, and the backward pass must be completed before those states are released from memory. Combining (a) $\sim$ (c) yields a closed loop of dependencies, as depicted in Eq. 1.

$$\begin{array}{ccccccc} \mathcal{FW}(s_0) & \xrightarrow{(a)} & \mathcal{FW}(s_1) & \xrightarrow{(a)} & \mathcal{FW}(s_2) & \xrightarrow{(a)} \cdots \xrightarrow{(a)} & \mathcal{FW}(s_{L-1}) \\ \Updownarrow (c) & & \Updownarrow (c) & & \Updownarrow (c) & \cdots & \Updownarrow (c) \\ \mathcal{BW}(s_0) & \xleftarrow[(b)]{} & \mathcal{BW}(s_1) & \xleftarrow[(b)]{} & \mathcal{BW}(s_2) & \xleftarrow[(b)]{} \cdots \xleftarrow[(b)]{} & \mathcal{FW}(s_{L-1}) \end{array} \tag{1}$$

Notably in on-policy RLHF, both inference phase and training phase operate on exactly the same resampled sequence, while the former phase already materializes a K, V cache that stores the LLM context, and can be reused during the training phase to decouple inter-token dependencies by eliminating all edges (a) in Eq. 1, thereby turning the memory dependency into a *"directed linked list"*. Guided by the new topology, we reschedule the RLHF training phase by streaming subsequences from s_{high} to s_{low}, which unties the training under constrained memories.

3 Rescheduling algorithm

3.1 Decomposing attention to enable semantically identical subsequence training



$$Atten(L) = \begin{cases} Area\text{-}① : Recursive\ Atten(L-T) \\ Area\text{-}② : K, V\ cache\ memorized \\ Area\text{-}③ : No\text{-}grads \begin{cases} \frac{\partial \text{softmax}(masked(QK'))V}{\partial K} = \frac{\partial -\infty}{\partial (QK')} \cdot (..) = 0 \\ \frac{\partial \text{softmax}(masked(QK'))V}{\partial V} = softmax(-\infty) = 0 \end{cases} \\ Area\text{-}④ : Local\ partial\ Atten(\Delta T) \end{cases}$$

Figure 1: Decomposition of causal-based attention into separated subsequence training stages

Figure 1 visualizes the $L \times L$ causal attention [20] matrix, where the vertical and horizontal axes enumerate the query and K, V positions respectively. To enable subsequence training, we perform a single training stage that computes attention over a subsequence $s_{L-T}, \ldots, s_{L-1}$ of length T. During this stage, the attention outputs receive contributions from both *Area-2* and *Area-4*, whereas *Area-3* contributes neither attention outputs nor productive gradients. In *Area-2*, all key-value entries lie in the indices $s_0, \ldots, s_{L-T-1}$, which can be loaded in one batch from the K, V cache. Together with the $T \times T$ sub-grids in *Area-4*, they form an eventual attention computation with a "right-angled trapezoid" pattern. Crucially in the backward pass, the "effect-to-cause" feedback from this training stage generates not only local gradients but also cross-block gradients. All cross-block gradients produced by training stages that solve high-dimensional indices $s_{x+1}, \ldots, s_{L-1}$ are accumulated and subsequently constitute the final local gradient for s_x. Therefore, we maintain a history-gradient accumulator that successively adds the cross-block gradient from one training stage to the next.

Building on the decomposed structure, we present the *Stream-able Attention Layer* in Figure 2, which consists of "Instance $\mathcal{FW}$" to compute position-corrected local attention and "Stream-able $\mathcal{BW}$" that propagates local gradients vertically while accumulating cross-block gradients horizontally.

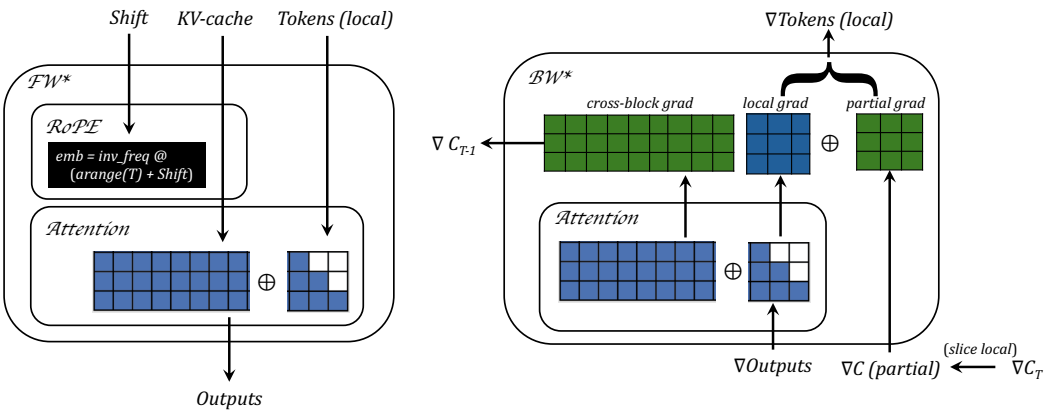


Figure 2: "Instant $\mathcal{FW}$ (left)" & "Stream-able $\mathcal{BW}$ (right)" for *Stream-able Attention Layer*.

3.2 Omitting unproductive loss with "zero-rewarding shortcut"

"Zero-rewarding shortcut" is built on top of the fine-grained sequence streaming for the possible elimination of a block-wise training stage. According to the post-training standard, the rewarding loss in RLHF is incurred only for the answer tokens (those identified by $\texttt{mask}_A(s_i)$), while the question tokens do not contribute to the loss. This yields two consequences for a block-wise training stage at the fine-grained level: (a) the local loss contributes no gradient to either the intra-block or cross-block updates if the entire span lies within a question region, so its computation can be omitted; (b) the cross-block loss can contribute gradients with respect to the local context, which must be retained. If neither of the two conditions is satisfied, this fine-grained training stage can be omitted entirely. Therefore, we establish the block-wise training algorithm 1, which processes the sequence in reversed order while appropriately maintaining both local gradients and cross-block gradients.

Algorithm 1 Memory-effective training of on-policy RLHF

Require: Model parameters θ, reward function $R(\cdot)$, block size T, prompt Q, attention window size W
Ensure: Updated model θ'

$\mathcal{S}eq_{0..k-1}, C_{0..k-1} \leftarrow \texttt{ResamplingAsync}(\theta, Q)$ ▷ resampling k experiences with KV cache as C
$adv_{0..k-1} \leftarrow R(\mathcal{S}eq_{0..k-1})$ ▷ REINFORCE advantages by reward function
$grads\_final \leftarrow 0$

for $x \leftarrow 0$ **to** $k-1$ **do**
  $L_{ctx} \leftarrow +\infty$
  $\nabla C \leftarrow zeros\_like(C_x)$ ▷ initialize history-gradient accumulator

  for $i \leftarrow \lceil len(\mathcal{S}eq_x) \,/\, T \rceil - 1$ **downto** 0 **do** ▷ block-wise training stages (train mode)
    $seq\_local \leftarrow \mathcal{S}eq_x[i \times T : (i+1) \times T]$
    if $\texttt{masks}_A(seq\_local)\texttt{.any()}$ **or** $(i+1) \times T > L_{ctx}$ **then**
      $C\_history \leftarrow C_x[0 : (i+1) \times T]$
      $shift \leftarrow i \times T$ ▷ RoPE correction
      $logits \leftarrow \mathcal{FW}(seq\_local; C\_history, shift)$ ▷ "instant forward"
      $loss \leftarrow compute\_rl\_loss(logits, adv_x) \cdot \texttt{masks}_A(seq\_local)$
      $\nabla C\_local \leftarrow \nabla C[i \times T : (i+1) \times T]$
      $grads,\ \nabla C\_history \leftarrow \mathcal{BW}(loss; \nabla C\_local)$ ▷ "stream-able backward"
      $\nabla C[0 : i \times T]\texttt{.add\_}(\nabla C\_history)$ ▷ shift cross-block grads to ∇C for future stages
      $grads\_final\texttt{.add\_}(grads)$ ▷ merge current-stage grads to grads_final
      $L_{ctx} \leftarrow shift - W$
    end if
  end for
end for

$\theta' \leftarrow \texttt{Optimizer}(\theta,\ grads\_final)\texttt{.step()}$ ▷ policy model update with θ'

4 Experiments

All experiments were conducted using a single AMD MI300x GPU [21] (192 GB device memory) paired with a 48-core Intel Xeon Platinum 8480C CPU. The software stack comprised ROCm 6.3.0 and PyTorch 2.7.0. We use the open-source Qwen-3 models [22, 23] (Apache-2.0) for performance evaluation. Both parameters and the optimizer RMSprop are `bfloat16` precision, and LoRA [24] adapters are applied with a rank of 4. For the $W = 1024$ (attention window size) configuration, each of the input (question) and output (answer) token sequences is half of the total context length, depicting a text translation scenario, while W set to $+\infty$ corresponds to full attention. Notably, the memory consumption of the K, V cache is included in all MetroRLHF evaluations.

The evaluation results from Table 1 show that MetroRLHF achieves a drastic reduction in memory usage compared with full-sequence training, while different block sizes T affect the expected trade-off between peak memory consumption and training throughput, since the overhead for small block sizes mainly stems from more frequent gradient accumulation under limited memory. "Zero-rewarding shortcut" enabled at $W = 1024$, where the attention complexity has been inherently reduced, yields an average computational saving of $\frac{\mathcal{C}tx + min(2\ T,\ \mathcal{C}tx)}{2\ \mathcal{C}tx}$ compared with not using the shortcut.

Qwen3 Family (BFloat16) *RMSProp, kSamples=8, LoRank=4, MI300 × 1*	**FullSeq** *(baseline)*	**MetroRLHF** *T = 1K*	**MetroRLHF** *T = 2K*	**MetroRLHF** *T = 4K*	**MetroRLHF** *T = 8K*	**MetroRLHF** *T = 16K*
PM (Qwen3-0.6B, $\mathcal{C}tx$=32K, W= $+\infty$)	72.3	**12.2**	14.7	19.5	29.1	48.4
TC (Qwen3-0.6B, $\mathcal{C}tx$=32K, W= $+\infty$)	**2.48**	4.82	3.43	2.83	2.57	2.51
PM (Qwen3-8B, $\mathcal{C}tx$=32K, W= $+\infty$)	OoM	**37.2**	43.7	57.5	85.4	141.1
TC (Qwen3-8B, $\mathcal{C}tx$=32K, W= $+\infty$)	OoM	9.96	8.91	8.25	**7.81**	7.84
PM (Qwen3-32B, $\mathcal{C}tx$=32K, W= $+\infty$)	OoM	**106.4**	126.5	167.2	OoM	OoM
TC (Qwen3-32B, $\mathcal{C}tx$=32K, W= $+\infty$)	OoM	33.05	30.23	**27.79**	OoM	OoM
PM (Qwen3-32B, $\mathcal{C}tx$=64K, W= $+\infty$)	OoM	**122.9**	142.8	183.5	OoM	OoM
TC (Qwen3-32B, $\mathcal{C}tx$=64K, W= $+\infty$)	OoM	96.68	89.02	**82.78**	OoM	OoM
PM (Qwen3-32B, $\mathcal{C}tx$=64K, W=1024)	OoM	**122.9**	142.8	183.5	OoM	OoM
TC (Qwen3-32B, $\mathcal{C}tx$=64K, W=1024)	OoM	23.26	18.94	**18.01**	OoM	OoM

Table 1: Single-device performance for Qwen3 at various model and context scales. In the table, T indicates the subsequence length scheduled to train per stage, PM represents the peak training memory usage (GiB), and TC denotes the total training time (sec) required to process a full sequence.

5 Conclusion

MetroRLHF[1] demonstrates that the computation already performed during RLHF inference can be turned into a memory-saving asset for the later training phase. By materializing the inference-phase K, V context, it freely removes the inter-token causal constraints. Therefore, RLHF training phase can be carried out on memory-affordable subsequences while preserving the semantics of the full-sequence training and further allow fine-grained "zero-reward shortcut". Moreover, MetroRLHF can also be seamlessly complementary with weight-focused optimization techniques to further reduce memory footprint, such as mixed precision, QLoRA [25]. This synergy paves the way for adapting to a wide range of resource-constrained environments while delivering reliable and efficient post-training.

MetroRLHF offers memory-effective training of RLHF with several important limitations:

1. **Zero recomputation for on-policy RLHF.** The fine-grain sequence streaming that MetroRLHF enables relies on the context materialized in RLHF inference phase. In pre-training or supervised fine-tuning (SFT), such a pre-existing cache is unavailable, which incurs recomputation overhead.
2. **Requires an auto-regressive LLM.** The "token dependency loop" is eliminated upon the causal (unidirectional) attention pattern that is used in most LLMs. Architectures that employ non-causal attention still preserve inter-token dependencies after the inference phase of RLHF, and these dependencies cannot be stripped away without compromising computational semantics.
3. **Cuts down activation memory only.** MetroRLHF targets the instant sequence-memory reduction, but it does not shrink the weight memory required by model parameters and optimizers.

[1]Details of MetroRLHF: https://github.com/MetroRLHF/MetroRLHF

References

[1] Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[2] Karim Galliamov, Ivan Titov, and Ilya Pershin. Enhancing RLHF with human gaze modeling. *CoRR*, abs/2507.09016, 2025.

[3] Partha Pratim Ray. Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, 3:121–154, 2023.

[4] 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, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

[5] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023.

[6] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *CoRR*, abs/2404.07143, 2024.

[7] Junyang Zhang, Tianyi Zhu, Cheng Luo, and Anima Anandkumar. MOM: memory-efficient offloaded mini-sequence inference for long context language models. *CoRR*, abs/2504.12526, 2025.

[8] Cheng Luo, Zefan Cai, Hanshi Sun, Jinqi Xiao, Bo Yuan, Wen Xiao, Junjie Hu, Jiawei Zhao, Beidi Chen, and Anima Anandkumar. Headinfer: Memory-efficient LLM inference by head-wise offloading. *CoRR*, abs/2502.12574, 2025.

[9] Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. Qwenlong-l1: Towards long-context large reasoning models with reinforcement learning. *CoRR*, abs/2505.17667, 2025.

[10] Tiantian Fan, Lingjun Liu, Yu Yue, Jiaze Chen, Chengyi Wang, Qiying Yu, Chi Zhang, Zhiqi Lin, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Bole Ma, Mofan Zhang, Gaohong Liu, Ru Zhang, Haotian Zhou, Cong Xie, Ruidong Zhu, Zhi Zhang, Xin Liu, Mingxuan Wang, Lin Yan, and Yonghui Wu. Truncated proximal policy optimization. *CoRR*, abs/2506.15050, 2025.

[11] Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. *CoRR*, abs/2505.23564, 2025.

[12] Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *CoRR*, abs/2507.07969, 2025.

[13] Wenhao Li, Yuxin Zhang, Gen Luo, Daohai Yu, and Rongrong Ji. Training long-context llms efficiently via chunk-wise optimization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 2691–2700. Association for Computational Linguistics, 2025.

[14] Linda He, Jue Wang, Maurice Weber, Shang Zhu, Ben Athiwaratkun, and Ce Zhang. Scaling instruction-tuned llms to million-token contexts via hierarchical synthetic data generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[15] Wadjih Bencheikh, Jan Finkbeiner, and Emre Neftci. Optimal gradient checkpointing for sparse and recurrent architectures using off-chip memory. *CoRR*, abs/2412.11810, 2024.

[16] Youshao Xiao, Zhenglei Zhou, Fagui Mao, Weichang Wu, Shangchun Zhao, Lin Ju, Lei Liang, Xiaolu Zhang, and Jun Zhou. Flexrlhf: A flexible placement and parallelism framework for efficient RLHF training. In *IEEE International Parallel and Distributed Processing Symposium, IPDPS 2025, Milano, Italy, June 3-7, 2025*, pages 358–369. IEEE, 2025.

[17] Zhuohan Li, Siyuan Zhuang, Shiyuan Guo, Danyang Zhuo, Hao Zhang, Dawn Song, and Ion Stoica. Terapipe: Token-level pipeline parallelism for training large-scale language models. *CoRR*, abs/2102.07988, 2021.

[18] 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. *CoRR*, abs/2409.19256, 2024.

[19] Yinmin Zhong, Zili Zhang, Bingyang 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. In Theophilus A. Benson and Radhika Niranjan Mysore, editors, *22nd USENIX Symposium on Networked Systems Design and Implementation, NSDI 2025, Philadelphia, PA, USA, April 28-30, 2025*, pages 489–503. USENIX Association, 2025.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[21] Alan Smith, Eric Chapman, Chintan Patel, Raja Swaminathan, John J. Wuu, Tyrone Huang, Wonjun Jung, Alexander Kaganov, Hugh McIntyre, and Ramon Mangaser. 11.1 AMD instincttm MI300 series modular chiplet package - HPC and AI accelerator for exa-class systems. In *IEEE International Solid-State Circuits Conference, ISSCC 2024, San Francisco, CA, USA, February 18-22, 2024*, pages 490–492. IEEE, 2024.

[22] 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, Jian Yang, Jiaxi Yang, Jingren 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. *CoRR*, abs/2505.09388, 2025.

[23] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176, 2025.

[24] Michael Santacroce, Yadong Lu, Han Yu, Yuanzhi Li, and Yelong Shen. Efficient RLHF: reducing the memory usage of PPO. *CoRR*, abs/2309.00754, 2023.

[25] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.