RFabric: A Reconfigurable Network for the Rhythms of Disaggregated RL

Xin Tan[1], Yicheng Feng[1], Yu Zhou[2], Yimin Jiang[2], Yibo Zhu[2], Hong Xu[1]
[1]The Chinese University of Hong Kong, [2]StepFun

Abstract

Asynchronous reinforcement learning (RL) for post-training large language models induces highly dynamic and heterogeneous network traffic, with shifting communication hotspots driven by the disaggregated training and generation stages. Conventional static network fabrics are ill-suited to these evolving spatiotemporal patterns, leading to either substantial resource underutilization or persistent performance bottlenecks.

We present RFabric, a reconfigurable hierarchical network architecture that dynamically adapts to disaggregated RL demands. RFabric integrates conventional electrical packet switching (EPS) within racks for maximal flexibility, while employing reconfigurable optical circuit switching (OCS) at the aggregation and core layers to deliver high-bandwidth, direct optical paths for both intra- and inter-cluster communication. A workload-aware control plane continuously monitors traffic and proactively reconfigures the OCS fabric to match the application's evolving requirements. Large-scale packet-level simulations show that RFabric achieves performance comparable to non-blocking static fabrics while improving network cost-efficiency by 1.81×.

1 Introduction

Reinforcement Learning (RL) has emerged as the key post-training technique that unlocks the instruction-following and reasoning capabilities of premier large language models (LLMs), including OpenAI's GPT-5, Anthropic's Claude 4, and DeepSeek R1 [3, 4, 9]. RL pipelines inherently operate in two alternating, yet computationally distinct stages: inference-heavy sample generation and compute-intensive model training. During generation, the current model produces responses to large batches of prompts auto-regressively, characterizing it as a memory-bandwidth-bound workload. Conversely, training consumes these responses to update model parameters via algorithms like PPO or GRPO [10, 15, 16], demanding massive compute resource. To optimize hardware utilization for these divergent profiles, leading RL frameworks [8, 16, 20] adopt a disaggregated architecture. This design dedicates distinct GPU clusters exclusively to generation or training, enabling parallel execution.

While this disaggregated architecture boosts computational throughput, it introduces profound and multi-faceted networking challenges. Firstly, the generation and training stages impose hybrid, high-intensity traffic patterns on the network fabric. Training clusters require efficient collective communications, notably all-reduce for data parallelism and all-to-all for expert parallelism, demanding high bisection bandwidth. Concurrently, generation clusters exhibit bipartite communication patterns, including KVCache transfer traffic of Prefill-Decode (PD) disaggregation [19] and M2N traffic of Attention/FFN (AF) Disaggregation [17, 21]. Second, the disaggregated nature of RL necessitates frequent, high-volume synchronization of massive model parameters from the training cluster to all generation clusters. Optimizing the network fabric to meet these combined demands, *i.e.,* hybrid traffic patterns and massive parameter synchronization, is critical for scaling RL workloads across increasingly large GPU data centers.

While overprovisioning bisection bandwidth has been the conventional approach to scale with increasing GPU compute demands, this leads to significant bandwidth underutilization due to the spatiotemporal variation inherent in RL workloads (as revealed in §3). Recognizing the inherent mismatch between LLM communication patterns and fixed electrical network fabrics, recent advances [11, 13, 18] introduce Optical Circuit Switching (OCS) to dynamically reconfigure the datacenter network, achieving a better fit for training workloads. However, these designs focus primarily on optimizing the model training stage. They fail to adequately address the distinct communication requirements of the generation stage, characterized by bipartite traffic patterns like KVCache transfers and AF disaggregation–or the substantial model parameter synchronization demands between training and generation clusters. Consequently, existing OCS-based solutions deliver poor cost-efficiency or introduce performance bottlenecks when applied to the complete RL pipeline, highlighting a critical gap in support for end-to-end RL workloads.

To address this gap, we propose RFabric, a dedicated network fabric explicitly designed for the spatiotemporal dynamics of end-to-end RL workloads. RFabric logically partitions training and generation resources into distinct Points-of-Delivery (PoDs). Its core innovation lies in dynamically reconfiguring the available bisection bandwidth between training and generation PoDs based on real-time characterization of RL communication across both space and time.

RFabric comes with a hierarchical and reconfigurable hybrid optical-electrical design for intra-PoD and inter-PoD fabrics. The fabric has three layers, and the electronic Top-of-Rack switches and OCS-based aggregation compose the intra-PoD fabric, while PoDs are connected by an OCS-based

core layer. For training PoDs, RFabric uses aggregation-layer OCS to build a high-bisection-bandwidth fabric for intensive collectives. In parallel, RFabric creates a separate topology for the generation PoDs, tailored to their unique bipartite patterns. Crucially, during model parameter synchronization, RFabric orchestrates rapid, cross-PoD reconfiguration of both the core and aggregation OCS fabrics across all training and generation PoDs. It dynamically dismantles the existing topologies and constructs an optimized, high-bandwidth broadcast tree spanning training PoDs to all generation PoDs. This creates dedicated optical "express lanes" for efficient weight dissemination. RFabric transcends simple bimodal optimization. By precisely allocating optical resources only where and when needed, it simultaneously optimizes for concurrent, diverse RL workloads. This eliminates the resource contention and performance compromises inherent in static fabrics, unlocking near-maximum performance and efficiency for disaggregated RL pipelines.

Through large-scale, packet-level simulations, we demonstrate that RFabric achieves performance comparable to a non-blocking fat-tree fabric [7], while improving network cost-efficiency by 1.81×. Moreover, RFabric outperforms a representative OCS-based fabric [11], providing a scalable and cost-effective solution for future large-scale RL systems.

2 Background

The two-stage RL workflow. A standard RL workflow proceeds in two main stages for each data batch [9, 14]. The first stage, Generation (`Gen`), uses the current model to produce responses to prompts and then computes the corresponding rewards. The second stage, Training (`Train`), is compute-intensive: it consumes the generated trajectories to compute gradients and update model parameters. This workflow exhibits two salient properties. First, there is a strict data dependency: `Train` cannot begin until `Gen` for the batch completes. Second, the resource profiles are markedly different: `Gen` is memory-bound due to the autoregressive nature of LLM inference, whereas `Train` is compute-bound and benefits most from high-performance GPUs.

From co-location to asynchronous disaggregation. The architecture of RL systems has evolved from a co-located paradigm [10, 16] (Figure 1(a)), where `Train` and `Gen` share the same bundle of GPU resources, toward disaggregation. The co-located design suffers from inherent inefficiency, as the memory-bound `Gen` and compute-bound `Train` stages have conflicting resource profiles that preclude optimal hardware utilization and scalability.

To overcome this, recent systems [8, 10, 20] adopt a disaggregated architecture. While a naive disaggregated paradigm (Figure 1(b)) allows for specialized and independently scaled clusters, it introduces significant "pipeline bubbles" due to synchronous execution. To maximize resource utilization, this is refined into a one-step asynchronous disaggregated paradigm (Figure 1(c)). This model enables clusters to operate in parallel on different data batches, boosting system throughput by masking data dependency latency. The trade-off is the use of a one-step-stale model parameters for `Gen`, a minimal cost that is far outweighed by the substantial gains in efficiency. Crucially, this degree of staleness has been widely shown to preserve model convergence and final performance.

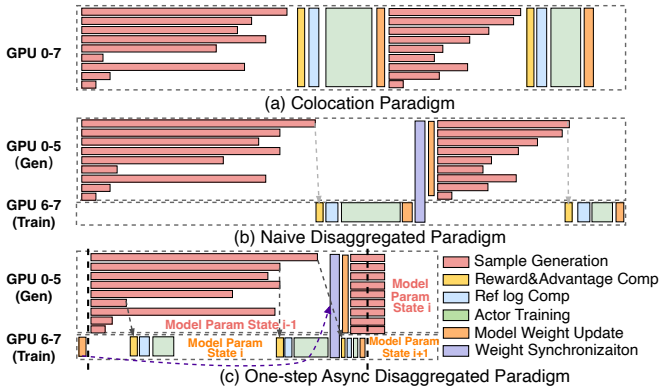

Figure 1. Overview of RL workflow in different paradigms with GRPO algorithm [15].

3 Heterogeneous Communication Profiles

This section characterizes the communication patterns of disaggregated RL. We analyze verl [16] under the common one-step off-policy setting, evaluating a Qwen-2.5 14B model on the openr1-math-220k dataset [5] using 64 NVIDIA H800 GPUs: a 24-GPU `Train` cluster and a 40-GPU `Gen` cluster, with GRPO as the RL algorithm. On the training side, Megatron [21] is configured with pipeline parallelism (PP) = 3 and tensor parallelism (TP) = 8; on the generation side, vLLM [12] is set to TP = 8 and data parallelism (DP) = 5.

Spatial heterogeneity in communication. Communication is highly asymmetric across roles during the parallel execution of `Gen` and `Train` phases (Figure 2). The `Train` cluster (Ranks 40-63) exhibits structured collectives combining TP, DP, and PP: frequent TP all-reduces within model-parallel groups, DP gradient synchronization, and PP send/recv across pipeline stages. In contrast, the `Gen` cluster (Ranks 0-39) communicates almost exclusively within small, disjoint TP groups, with negligible cross-group traffic. Cross-cluster communication is sparse and episodic, dominated by weight synchronization at phase boundaries.

Temporal and bimodal periodicity. The workload also exhibits pronounced temporal shifts in communication rate and regularity. To quantify this, we define the "traffic window duration" as the inter-arrival time between consecutive communication operations of the same type. As shown in Figure 3, the distribution of these windows is distinctly bimodal. One mode comprises high-frequency, latency-sensitive events

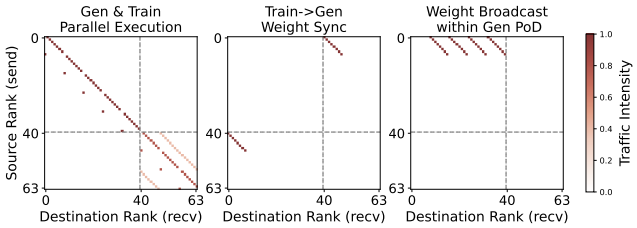

Figure 2. Spatial network traffic across RL phases. For weight synchronization, we use an optimized two-stage scheme: the `Train` DP group transmits weights once to the DP-0 group of each `Gen` pod, after which each `Gen` DP-0 broadcasts locally to its DP peers.

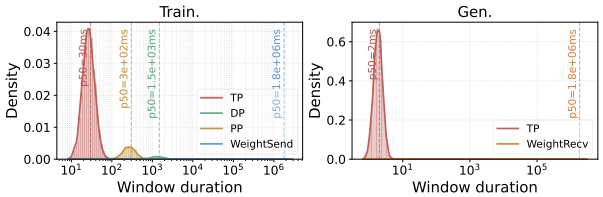

Figure 3. Distribution of traffic window durations (inter-arrival times) across communication types.

driven by model parallelism. During generation, TP operations have a median inter-arrival time of approximately 2 ms. During training, TP, PP, and DP operations show median intervals of roughly 30 ms, 300 ms, and 1.5 s, respectively. Consequently, `Gen` requires persistent, low-latency connectivity, whereas `Train` offers comparatively generous windows for network adaptation. The other mode consists of low-frequency, high-volume inter-cluster weight synchronization (WeightSend/Recv), with a median inter-arrival time of around 20 minutes. This multi-minute window presents a large, predictable opportunity for network reconfiguration. This stark contrast in timing means RL mixes sustained, mid- to high-rate collectives with rare, massive-volume transfers. It highlights the mismatch with static networks that are designed for a single, average-case scenario.

4 RFabric: An Adaptive Network for Disaggregated RL

The empirical analysis in Section 3 exposes a fundamental mismatch between the disaggregated RL workload and static network fabrics. Table 1 provides a comprehensive classification of these communication patterns, quantifying their profiles and their suitability for network adaptation. Our analysis identifies two key challenges: (1) pronounced spatial heterogeneity, with required network domains (see "Possible Domain" in Table 1) varying dramatically, and (2) a distinct temporal bimodality, resulting in "Small" to "Large" reconfiguration windows.

To address these challenges, we introduce RFabric—a hierarchical, reconfigurable hybrid network architecture that dynamically adapts the network topology to real-time workload demands as cataloged in Table 1.

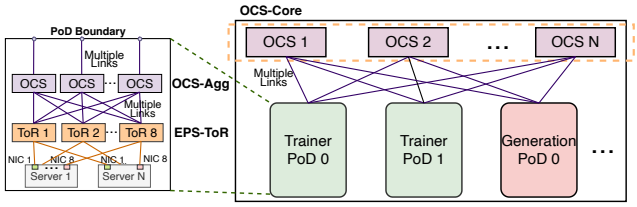

Figure 4. Network architecture overview in RFabric.

4.1 A Hybrid EPS-OCS Design

At the core of RFabric is a synergistic hybrid design that establishes a clear division of labor, as illustrated in Figure 4:

- **Static Electrical Packet-Switched (EPS) fabric:** The EPS fabric is deployed at the Top-of-Rack (ToR) layer—the base of the network. This layer experiences the most diverse and dynamic mix of traffic, characterized by frequent bursts and stringent latency requirements. For instance, operations such as EP and Attention-FFN Disaggregation in the `Gen` stage generate high-frequency flows with reconfiguration windows shorter than the typical OCS reconfiguration delays (see Table 2). The ToR must efficiently aggregate and forward all this traffic, even as patterns fluctuate rapidly. To address these demands, the EPS fabric delivers always-on, low-latency connectivity, making it ideally suited for latency-sensitive and highly variable workloads with tight reconfiguration constraints.
- **Dynamic Optical Circuit-Switched (OCS) fabric:** The aggregation and core layers provide high-bandwidth optical circuits that can be reconfigured on demand. This architecture is ideal for traffic characterized by medium or large reconfiguration windows, such as per-layer interleaved CP and EP collectives during training, as well as low-frequency, high-volume DP or weight synchronization operations. Even for generation workloads with predictable communication patterns—such as bipartite topologies—we can preconfigure the necessary links between servers in advance, despite the infeasibility of dynamic adaptation during inference due to the short configuration window. The OCS fabric enables the establishment of direct, contention-free optical paths, making it particularly well suited for predictable, bulk data transfers.

This hybrid approach allows RFabric to map the workload's bimodal temporal characteristics, quantified in Table 1, directly onto the optimal underlying network technology.

4.2 Dynamic Topology Materialization

The efficacy of RFabric arises from its core operational principle: dynamic topology materialization. The OCS fabric is continuously reconfigured to instantiate transient, purpose-built topologies that are precisely optimized for the ephemeral communication requirements of each RL phase.

	Comm Types	Profile			Fabrics	
		Volume	Frequency	Primitives	Possible Domain	Reconfiguration Window
Train. Stage	DP	High	Low	AllReduce	ToR-Agg-Core	Large
	TP	Medium	High	AllReduce	HBD	Medium
	PP	Low	Low	P2P	ToR-Agg	Large
	CP	Medium	High	P2P or All-to-All	ToR-Agg	Medium
	EP	Medium	High	All-to-All	ToR-Agg	Medium
Inter-Stage	Weight-Sync	High	Low	T2G	ToR-Agg-Core	Large
	Response-Stream	Low	Medium	G2T	ToR-Agg-Core	Large
Gen. Stage	TP	Medium	High	AllReduce	HBD	Small
	EP	Medium	High	All-to-All	ToR-Agg	Small
	P/D	Medium	Low	M2N (Bipartite)	ToR-Agg	Large
	A/F	Low	High	M2N (Bipartite)	ToR-Agg	Small

Table 1. Communication profiles for the disaggregated RL workflow from a fat-tree perspective. The **Reconfiguration Window** represents the opportunity space for network adaptation between operations, which dictates the mapping to either the static EPS or dynamic OCS fabric. "P/D" refers to Prefilling and Decoding disaggregation [19], while "A/F" stands for Attention-FFN disaggregation [17, 21]. "T2G" denotes transitions from `Train` to `Gen`, and vice versa.

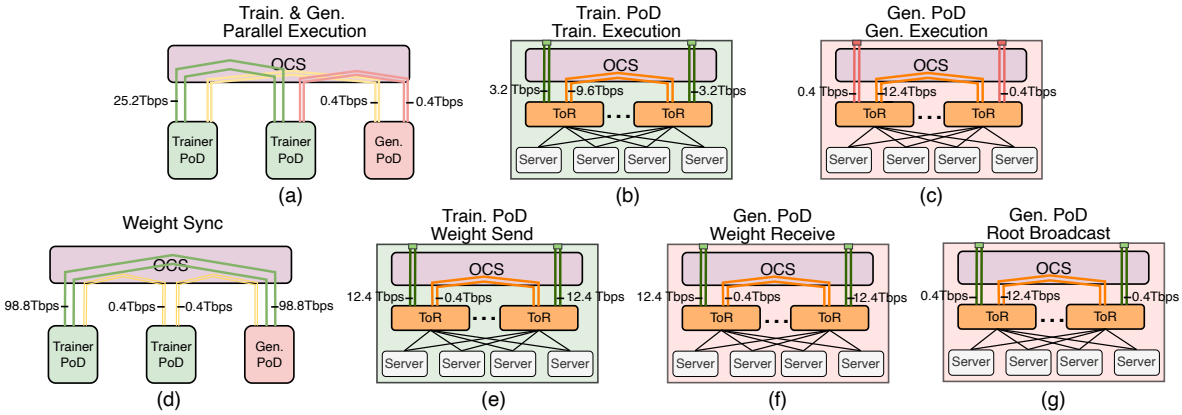

Figure 5. Dynamic topology materialization in action. The example illustrates a PoD containing 32 servers (each with 8 NICs) connected to 64-radix EPS ToR switches. For visual clarity, multiple OCS devices are represented as monolithic blocks at the aggregation and core layers.

OCS Type	Reconfig. delay (ms)	Radix ports
RotorNet (InFocus)	0.01	128
3D MEMS (Calient)	10	320
Piezo (Polatis)	25	576
Liquid crystal (Coherent)	100	512
Robotic (Telescent)	120000	1008

Table 2. Reconfiguration time for different OCS.

Figure 5 demonstrates how RFabric materializes distinct topologies to address the spatial heterogeneity characterized in Section 3, based on the requirements summarized in Table 1.

High-bisection fabric for `Train` collectives. To support the demanding Data Parallelism (DP) AllReduce operation—classified in Table 1 as a "ToR-Agg-Core" domain workflow with a "Large" communication window—RFabric configures the core-layer OCS to form a high-bisection-bandwidth inter-PoD mesh (Figure 5(a)). For intra-PoD traffic during forward and backward passes, appropriate bandwidth allocation is provisioned across different ToRs (Figure 5(b)).

Isolated intra-PoD fabrics for `Gen`. For highly localized traffic within the `Gen` cluster (e.g., EP All-to-All [9], M2N in Attention-FFN disaggregation [17, 21]), which Table 1 designates as occurring within the "ToR-Agg" domain, RFabric leverages the aggregation-layer OCS to carve out independent intra-PoD mesh topologies (Figure 5(c)). This strategy prevents interference between PoDs and avoids unnecessary over-provisioning of the network core. Notably, a small bandwidth allocation (0.4 Tbps) is reserved to connect `Gen` PoDs to the core, enabling streaming of generated responses back to the `Train` PoDs.

Purpose-built multicast tree for synchronization. To optimize the periodic T2G broadcast of model weights—a classic low-frequency task with a "Large Reconfiguration Window" per Table 1—RFabric reconfigures the core OCS to instantiate a dedicated optical multicast tree (Figure 5(d-f)). This provides a contention-free "express lane" for distributing model parameters from the DP group in `Train` PoDs to the DP 0 group in each `Gen` PoD. Subsequently, a local

broadcast within each Gen PoD is performed, materializing the topology to meet the required bandwidth (Figure 5(g)).

By dynamically materializing the right topology for the right job at the right time, RFabric transcends the limitations of static fabrics, ensuring that network resources are always aligned with application requirements.

4.3 Orchestration for Proactive Reconfiguration

To enable dynamic topology adaptation, we introduce a lightweight control proxy that translates RL workload intents into OCS configurations, bridging the application and network layers while hiding reconfiguration overhead. It works in two stages:

1. First-iteration profiling and caching. During the first complete pass through the RL pipeline, the control proxy enters profiling mode, tracks major communication phases (e.g., intra-PoD `Train`/`Gen` collectives, inter-PoD gradient aggregation, inter-cluster weight sync), and records the predictable schedule. For each phase and sub-interval, it selects the optimal OCS topology (e.g., hierarchical mesh for training, multicast tree for weight sync) and caches the circuit configuration per job. These profiles form reusable templates for later iterations.

2. Steady-state proactive reconfiguration. In later iterations, the system transitions to a proactive mode, leveraging the learned communication schedule to issue speculative reconfiguration requests ahead of time. For example, during model training, the control proxy can dynamically allocate bandwidth across different layer operations in both the forward and backward passes (e.g., CP in attention layers, EP in MoE layers), as well as during gradient synchronization. When `Gen` reaches the weight synchronization stage, the proxy retrieves the precomputed multicast-tree topology from its cache and configures the OCS fabric accordingly.

5 Preliminary Evaluation

To evaluate RFabric, we conduct a cost-performance analysis against four representative interconnects using packet-level simulations based on SimAI [6].

- **Non-blocking Fat-tree (FT) [7]**: A non-blocking fabric representing the upper bound on performance and cost.
- **Oversubscribed Fat-tree (FT-OS)**: A cost-effective 3:1 oversubscribed fat-tree.
- **Rail-optimized (RO) [1]**: A fat-tree-style topology that connects same-rank GPUs from each server to the same ToR, optimizing rail-local communication.
- **SIP-OCS [11]**: A direct-connect optical architecture where NICs connect through a flat optical patch panel.

Performance comparison. We evaluate end-to-end performance using Qwen2-MoE [2], with normalized throughput as the primary metric. All results are benchmarked against an ideal non-blocking Fat-tree (normalized to 1.0) at 2048- and 4096-GPU scales. The results (in Figure 6) show that both

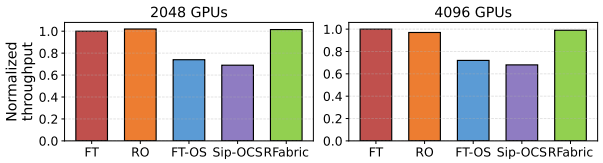

Figure 6. End-to-end performance comparison under 400 Gbps links across varying H800 GPU scales. We use 3D MEMS as the default OCS with 10ms reconfiguration delay.

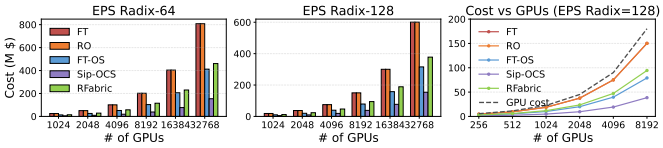

Figure 7. Network cost comparison for different fabrics with different H800 GPU numbers, with the cost model focusing on the primary expenditures of optical modules and switches.

the Oversubscribed Fat-tree and Sip-OCS architectures experience significant performance degradation at both scales, highlighting their inability to effectively mitigate network congestion—either due to oversubscription or inflexible OCS reconfiguration due to a centralized direct-connection design. In contrast, RFabric maintains performance close to that of the ideal Fat-tree and Rail-Optimized, owing to its flexible hybrid design and adaptive reconfiguration capabilities.

Cost analysis at scale. We further conduct a cost analysis from 1,024 to 32,768 GPUs, using both Radix-64 and Radix-128 EPS configurations. As shown in Figure 7, traditional electrical fabrics (Fat-tree, Rail-Optimized) exhibit unsustainable cost growth as scale increases, while higher-radix switches (Radix-128) provide only marginal relief. Although 3:1 oversubscribed Fat-tree and SIP-OCS architectures offer lower raw costs, they do so by sacrificing significant performance. When network costs are compared to total GPU expenditure, conventional designs often drive network costs to parity with, or even above, the compute hardware itself. In contrast, RFabric maintains networking as a modest fraction of total system cost, achieving a balanced trade-off between performance and efficiency at massive scale.

6 Conclusion

Disaggregated RL creates highly dynamic traffic that overwhelms static fabrics. We introduce RFabric, a hierarchical hybrid network that materializes optical topologies per RL phase. It couples a flexible intra-rack EPS fabric with a reconfigurable OCS at aggregation and core, adapting to evolving intra- and inter-cluster bandwidth. Large-scale simulations show RFabric matches non-blocking Fat-tree performance while significantly reducing cost.

Acknowledgments

This work is supported in part by funding from the Research Grants Council of Hong Kong (C7004-22G) and from CUHK (4937007, 4937008, 5501329, 5501517).

References

[1] 2022. Doubling all2all Performance with NVIDIA Collective Communication Library 2.12. https://developer.nvidia.com/blog/doubling-all2all-performance-with-nvidia-collective-communication-library-2-12/.

[2] 2024. Qwen2-57B-A14B. https://huggingface.co/Qwen/Qwen2-57B-A14B.

[3] 2025. Claude-4. https://www.anthropic.com/news/claude-4.

[4] 2025. GPT-5. https://openai.com/index/gpt-5-new-era-of-work/.

[5] 2025. OpenR1-Math-220k. https://huggingface.co/datasets/open-r1/OpenR1-Math-220k.

[6] 2025. SimAI. https://github.com/aliyun/SimAI.

[7] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review* (2008).

[8] 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. 2025. AReaL: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning. https://arxiv.org/abs/2505.24298

[9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[10] Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. 2024. OpenRLHF: An Easy-to-use, Scalable and High-performance RLHF Framework. *arXiv preprint arXiv:2405.11143* (2024).

[11] Mehrdad Khani, Manya Ghobadi, Mohammad Alizadeh, Ziyi Zhu, Madeleine Glick, Keren Bergman, Amin Vahdat, Benjamin Klenk, and Eiman Ebrahimi. 2021. SiP-ML: high-bandwidth optical network interconnects for machine learning training. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*.

[12] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*. 611–626.

[13] Xudong Liao, Yijun Sun, Han Tian, Xinchen Wan, Yilun Jin, Zilong Wang, Zhenghang Ren, Xinyang Huang, Wenxue Li, Kin Fai Tse, Zhizhen Zhong, Guyue Liu, Ying Zhang, Xiaofeng Ye, Yiming Zhang, and Kai Chen. 2025. MixNet: A Runtime Reconfigurable Optical-Electrical Fabric for Distributed Mixture-of-Experts Training. In *Proceedings of the ACM SIGCOMM 2025 Conference*. doi:10.1145/3718958.3750465

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[15] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

[16] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*. 1279–1297.

[17] Bin Wang, Bojun Wang, Changyi Wan, Guanzhe Huang, Hanpeng Hu, Haonan Jia, Hao Nie, Mingliang Li, Nuo Chen, Siyu Chen, et al. 2025. Step-3 is Large yet Affordable: Model-system Co-design for Cost-effective Decoding. *arXiv preprint arXiv:2507.19427* (2025).

[18] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. 2023. TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*.

[19] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*.

[20] Yinmin Zhong, Zili Zhang, Xiaoniu Song, Hanpeng Hu, Chao Jin, Bingyang Wu, Nuo Chen, Yukun Chen, Yu Zhou, Changyi Wan, et al. 2025. StreamRL: Scalable, Heterogeneous, and Elastic RL for LLMs with Disaggregated Stream Generation. *arXiv preprint arXiv:2504.15930* (2025).

[21] Ruidong Zhu, Ziheng Jiang, Chao Jin, Peng Wu, Cesar A Stuardo, Dongyang Wang, Xinlei Zhang, Huaping Zhou, Haoran Wei, Yang Cheng, et al. 2025. MegaScale-Infer: Serving Mixture-of-Experts at Scale with Disaggregated Expert Parallelism. *arXiv preprint arXiv:2504.02263* (2025).