Photonic Rails in ML Datacenters

Eric Ding
Cornell University
Ithaca, NY, USA

Chuhan Ouyang
Cornell University
Ithaca, NY, USA

Rachee Singh
Cornell University
Ithaca, NY, USA

Abstract

Rail-optimized network fabrics have become the de facto datacenter scale-out fabric for large-scale ML training. However, the use of high-radix electrical switches to provide all-to-all connectivity in rails imposes massive power, cost, and complexity overheads. We propose a rethinking of the rail abstraction by retaining its communication semantics, but realizing it using optical circuit switches. The key challenge is that optical switches support only one-to-one connectivity at a time, limiting the fan-out of traffic in ML workloads using hybrid parallelisms. We introduce *parallelism-driven rail reconfiguration* as a solution that leverages the sequential ordering between traffic from different parallelisms. We design a control plane, Opus, to enable time-multiplexed emulation of electrical rail switches using optical switches. More broadly, our work discusses a new research agenda: datacenter fabrics that co-evolve with the model parallelism dimensions *within* each job, as opposed to the prevailing mindset of reconfiguring networks before a job begins.

CCS Concepts

• **Networks** → **Data center networks**; **Network architectures**; **Application layer protocols**; • **Hardware** → **Networking hardware**.

Keywords

Rail-optimized topology; Optical circuit switching; Hybrid parallelism in ML workloads

ACM Reference Format:
Eric Ding, Chuhan Ouyang, and Rachee Singh. 2025. Photonic Rails in ML Datacenters. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25), November 17–18, 2025, College Park, MD, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3772356.3772414

HotNets '25, College Park, MD, USA

ACM ISBN 979-8-4007-2280-6/25/11
https://doi.org/10.1145/3772356.3772414

1 Introduction

The design of datacenter interconnect fabrics has a significant impact on the scalability and efficiency of large-scale machine learning (ML) systems. A wide range of general datacenter fabric designs have been proposed in the past decade, including electrical [1, 24, 27, 70, 71] and photonic networks [11, 15, 22, 26, 77]. More recently, the rapid growth of ML training workloads has driven a shift toward ML-centric datacenter fabric designs [31, 32, 34, 35, 79, 80, 85]. Among the many proposals, one datacenter topology has seen broad adoption: the rail-optimized fabric [21, 58, 78]. The rail fabric explicitly aligns with the communication patterns of hybrid parallelisms in ML workloads by wiring together "rails" of GPUs—sets of GPUs with identical ranks across multiple high-bandwidth (or *scale-up*) domains, forming the *scale-out* domain (Fig. 1). This design can achieve congestion-free communication for common collectives like AllReduce and AllGather in distributed ML pipelines [58].

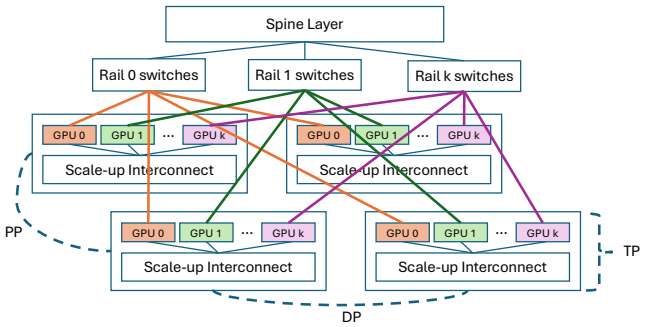

Figure 1: Rail-optimized fabrics. We propose to replace packet switches (shown as Rail 0, Rail 1 *etc.*) with optical circuit switches. We make the case for retaining the *illusion of full connectivity* between GPU ranks connected to the same optical rail switch using in-job reconfiguration.

But this performance comes at a steep cost. To ensure contention-free communication across rails, rail fabrics are significantly over-provisioned in electrical switch bandwidth [78]. Each rail switch connects GPUs of the same rank in all scale-up domains, resulting in networks built from high-radix packet switches. These switches are not only costly, but also power-hungry. Switch ASIC processing, transceiver electrical-optical conversions, and multi-tier Clos, all contribute to the energy and complexity burden of the fabric [9, 53, 61]. Ironically, the electrical rail design achieves congestion freedom by brute-force overcapacity, even though ML communication patterns are structured and predictable.

This paper asks whether it is possible to retain the desirable properties of rail-optimized fabrics while dramatically improving their energy and cost efficiency. Rather than redesigning the datacenter topology [6, 34, 35, 39, 72, 79], we pursue a different direction to answer the question: we propose to replace electrical packet switches in the rail with reconfigurable optical circuit switches (OCSes) which consume a magnitude lower power than their electrical counterparts [80]. We call the resulting design a *photonic rail-optimized fabric*. Our proposal draws inspiration from recent successes in optical ML fabrics [31, 85] but departs from them by preserving the already-successful rail design, while fundamentally changing how data is switched within it.

However, the shift from packet switching to circuit switching in rails is not straightforward. Electrical rail switches enable *all-to-all* packet-level connectivity among GPUs in the same rail, whereas OCSes only offer one-to-one circuit connectivity (one GPU to another GPU) at a given time. This limits the node degree of each GPU rank and breaks a key invariant of rail-optimized designs–full connectivity among the same-rank GPUs across all scale-ups (Fig. 1). Without full connectivity, hybrid ML parallelisms will become inefficient or even infeasible on rail fabrics. The core technical challenge, then, is: *how to retain the abstraction of seamless rail communication despite the limitations of photonic switching?*

One key observation is that most communication in distributed ML is predictable and structured [20, 79]. Collectives are issued in known sequences, organized by parallelism type (*e.g.,* tensor, data, pipeline) [54]. We exploit this structure to break the illusion of requiring full connectivity by *reconfiguring the photonic fabric between collectives within the job.* To our knowledge, this is the first proposal to reconfigure fabrics between ML collectives, while other proposals reconfigure once prior to the job start [31, 79, 85], or reconfigure during workloads specifically for one type of parallelism [39].

We consider the feasibility of in-job reconfiguration that adapts the fabric to each parallelism phase, without disrupting the ML stack or requiring bespoke OCSes. To realize the vision of photonic rails, we propose a novel control layer in ML software stacks, Opus. The collective communication libraries will act as clients to Opus and issue provisional intents to communicate. Opus will interface with the traditional network controller to orchestrate rail reconfiguration in response to such intents. Our early experiments show that Opus can reduce networking infrastructure cost by 70% and power consumption by 96% while only incurring 3% increase in iteration time over a network with electrical rails.

2 Communication across Parallelisms

Training ML models at scale requires a combination of parallelism strategies across thousands of GPUs (Tbl. 1) [12, 41, 64, 68]. To make this feasible, ML systems leverage multiple, co-existing dimensions of parallelism. These include data parallelism (DP, and variants like fully sharded data parallelism or FSDP), pipeline parallelism (PP), tensor parallelism (TP, often combined with sequence parallelism or SP), context parallelism (CP), and expert parallelism (EP) [16, 21, 33, 41, 43, 64, 68, 82, 84]. As shown in Tbl. 2, each parallelism axis incurs communication that differs in: (1) data volume–ranging from full model weights in DP to per-layer activations in TP; (2) start time–some collectives occur during the forward pass, others only during backpropagation; (3) frequency–some fire once per layer, others once per micro-batch; and (4) communication pattern–from ring-based AllReduce to high-fanout AllToAll. Importantly, the communication operations from different parallelisms are not ordered arbitrarily: they follow strict dependencies defined by the model's execution graph (DAG). Fig. 2 illustrates these dependencies in a 3D-parallel training step.

Model size	Compute (N GPUs)	Practices
Small (< 10B)	$N \leq 8$	TP or DP
Large (> 10B)	$8 < N \leq 512$	TP & PP, TP & DP, or DP
Large (> 10B)	$512 < N \leq 1024$	DP & PP, or DP & TP
Large (> 10B)	$N > 1024$	TP, DP & PP

Table 1: Rule-of-thumb LLM parallelism strategies [74].

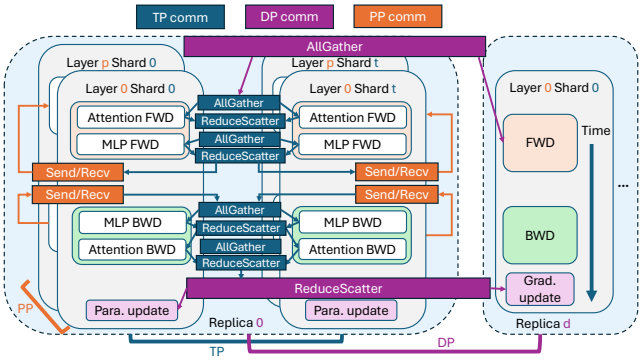

Figure 2: Traffic in a training iteration with 3D parallelism.

2.1 Our Proposal: Electrical → Optical Rails

Rail-optimized topologies are gaining traction for scaling ML workloads [58]. These topologies organize the network into multiple independent "rails," where each rail connects GPUs of the same local rank across different scale-up domains, *e.g.,* Nvidia DGX [56] or HGX [57] nodes. The number of rails equals the number of GPUs in each scale-up domain.

Fig. 1 illustrates how a 3D parallelism strategy maps naturally onto a rail-optimized network. Frequent and latency-sensitive TP collectives are confined within the high-bandwidth scale-ups, whereas PP and DP collectives often traverse the slower scale-out network because they occur less frequently and can be overlapped with compute [21, 63]. The rail abstraction allows these scale-out collectives to occur without

Parallelism	Memory reduction	Compute reduction	Communication type and frequency
DP	gbs/dp	gbs/dp	bwd AR per layer/per model
FSDP	gbs/dp, params/dp	gbs/dp	fwd AG, bwd RS per layer/model
TP	params/tp, grads/tp, optims/tp	params/tp	fwd bwd AR per operator
TP & SP	params/tp, grads/tp, optims/tp, activs/tp	params/tp, activs/tp	fwd bwd AG&RS per operator
CP	kv_cache/cp, seq/cp	seq/cp	fwd AG bwd RS per layer
PP	params/pp, grads/pp, optims/pp, activs/pp	params/pp	fwd bwd Send/Recv per microbatch
EP	experts/ep	experts/ep	fwd bwd AllToAll per layer

Table 2: Characteristics of different parallelism strategies [38]. gbs: global batch size. dp: data parallel degree. seq: sequence length. fwd: forward pass. bwd: backward pass. AR: AllReduce. AG: AllGather. RS: ReduceScatter. params: model parameter size. grads: gradients size. optims: optimizer states. activs: activation states.

oversubscription: every rail provides a dedicated, congestion-free path across domains for a specific GPU rank.

Limitations of packet-switched rails. Despite their performance benefits, today's rail fabrics suffer from scalability challenges. Optical fibers connect scale-up domains to the network, terminating at transceivers that interface with server NICs and packet switches. Each packet switch introduces optical-electrical-optical (OEO) conversions, adding energy and latency overhead to the data path. These conversions, coupled with the switch ASIC's work—packet queueing, header parsing, and TCAM lookups—consume significant energy [83]. Moreover, while link speeds (*e.g.,* 400 Gbps) continue to scale, ASIC processing speed has not kept pace. As a result, packet-switched fabrics now represent a bottleneck in both power efficiency and bandwidth scalability, especially in the face of LLM training jobs that require hundreds of rails and thousands of endpoints [8].

Replacing rail packet switches with OCSes. We propose a re-imagined rail-optimized fabric in which each rail is implemented not with electrical packet switches, but with OCSes. These photonic switches can form end-to-end optical paths without OEO conversions, eliminating switch ASICs entirely from the datapath. The result is a dramatic reduction in energy consumption, near-zero datapath latency, and the ability to scale bandwidth without incurring ASIC bottlenecks [44]. Importantly, our proposal retains the logical structure of the existing rail-optimized topology: the scale-up domains, cabling, and GPU-to-rail mapping, all remain unchanged. There is no multi-tier electrical rail or spine. Instead, each rail becomes a flat, photonic point-to-point fabric. Cross-rank communication can still be supported via forwarding through the high-bandwidth interconnect in scale-up (*e.g.,* PXN [50]), as explored in prior work [78]. The control plane remains electrical and host-driven.

Mature technologies like MEMS-based OCSes already have characteristics required by this design: millisecond reconfiguration and switch radix in the hundreds [3, 37, 39, 44, 60]. We note that our proposal differs from recent silicon photonic architectures proposed by Nvidia and Broadcom [7, 52], which still rely on electrical switching ASICs but use co-packaged optics (CPO) instead of pluggable optical transceivers.

3 Challenges in Realizing Photonic Rails

Replacing electrical rail switches with OCSes is non-trivial. In ML jobs with hybrid parallelisms, each GPU is a member of multiple communication groups — logical constructs managed by collective communication libraries like NCCL [49]. A single GPU belongs to several groups, each associated with a different parallelism axis. This leads to a high communication degree per GPU. For example, the degree requirement is 6 in a 3D–parallel job using ring-based AllReduce, where one GPU has two neighbors per ring. Packet-switched rails support this naturally due to their full connectivity. However, the number of simultaneous optical circuits per GPU is bounded by its physical degree, *i.e.,* the number of network ports. This degree limitation leads to three constraints:

C1: Collective algorithm. Low degree restricts collectives to ring algorithms that are bandwidth-efficient but incur high latency [67]. Latency-optimized strategies like tree-based or recursive-doubling collectives cannot be used [65, 76].

C2: Parallelism dimensionality. The number of parallelisms are constrained as some parallelisms (*e.g.,* CP) need to be implemented independently [38].

C3: Bandwidth fragmentation. Statically partitioning NIC ports across communication groups allocates only a fraction of NIC bandwidth to each collective.

Examples. Consider training a model with 3D parallelism (TP, DP, and PP) on DGX H200 nodes, where TP is within the scale-up domain. Each GPU is mapped to one ConnectX-7 NIC with three port configuration options, *e.g.,* one logical 400Gbps port, two logical 200Gbps ports, and four 100Gbps ports [51, 55]. The DP and PP collectives must share the scale-out optical rail network. If 4-port configuration is used by the NIC, two ports must be allocated to each parallelism dimension (for two neighbors in a ring), effectively halving the available bandwidth per group (C3). The low node degree forces ring-based collectives (C1), and adding CP would be infeasible without additional NICs or switching hardware (C2). One workaround is to multiplex parallelisms over shared

physical links, but this introduces a new set of problems: forwarding traffic via intermediate GPUs inflates latency and incurs a bandwidth tax [45]. Prior OCS-based ML fabrics have sidestepped these issues by adding multiple NICs per GPU [79, 80] or by placing one parallelism (*e.g.,* EP) in the OCS fabric while relying on packet-switched network for other traffic [39] — costly solutions that increase network complexity and fail to support higher-dimensional parallelisms cleanly. Google's TPU cluster [31] uses OCSes to construct a 3D torus before the job starts, mapping parallelisms to x,y,z dimensions but suffers from C1 and C3 [34, 35].

Key Insight. This naturally raises a provocative question: can we reconfigure the OCSes *during a job* to enable 5D parallelisms? Prior systems have explored microsecond- or nanosecond-scale reconfiguration to enable traffic-aware or traffic-oblivious scheduling in general datacenter networks [2, 3, 15, 22, 37, 45, 69, 77]. However, they are poorly suited to the repetitive and high-volume collective communication patterns of ML workloads [79]. We propose to embrace a different unit of adaptation: the collectives themselves. ML collectives occur in well-defined patterns dictated by the model's computational graph. These patterns naturally create brief *windows* of communication inactivity between phases of parallelism — ideal time to reshape the rail network topology to match the upcoming collective's demands.

3.1 Time Window between Parallelisms

To study the window sizes, we run an LLM training workload using TorchTitan [38] on the Perlmutter supercomputer [47]. We use 4 nodes connected by Slingshot 11 interconnect fabric [28]. Each node has 4 A100 GPUs interconnected via NVlink 3.0. We train Llama3-8B with TP=4 (intra-node), FSDP=2, and PP=2 [23]. The PP schedule is 1-forward-1-backward [68], and the micro-batch size is 2.

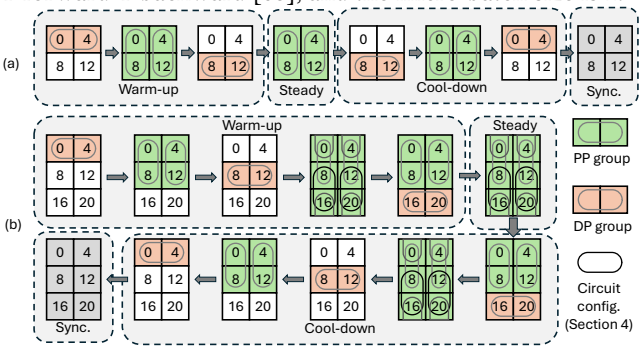

Figure 3: Communication pattern for PP and FSDP in one iteration, split based on the warm-up, steady, and cool-down stages of the pipeline (4 rails in total, only showing rail 0, TP is hidden). (a) PP=2, FSDP=2. (b) PP=3, FSDP=2.

Fig. 3(a) shows the communication pattern in rail 0 (same pattern for other rails). Rank 0 first performs stage 0 micro-batch 0 forward pass (overlapped with per-layer AllGather calls to collect the next layer's parameters), and sends the activation to stage 1 hosted by rank 8 through a Send/Recv call along the pipeline dimension. Once the Send/Recv call is finished, rank 8 computes the forward pass, while doing AllGather. Then, it performs backpropagation for micro-batch 0, followed by pipeline Send/Recv. ReduceScatter calls are issued after partial gradients are updated. During the optimizer step, several short AllReduce calls are issued for synchronization and numerical robustness [46, 59]. We observe that the DP traffic do not overlap with TP traffic. Fig. 3(b) shows the pattern for PP=3. The data dependency between operations, and PyTorch's lazy DTensor operation (*e.g.,* the first AllGather call for stage 1 only starts when it receives the activation from stage 0), dictate the sequential order between PP and DP traffic, though collectives from two dimensions are issued in different CUDA streams.

We find that the windows (the arrows in Fig. 3) are on the order of milliseconds. We define the window as the idle time between two consecutive parallelism phases $P1$ and $P2$, which are two distinctive sets of communication groups:

$$T_{window} = \min_{comm_j \in P2} T_{comm_j\_start} - \max_{comm_i \in P1} T_{comm_i\_end},$$

where $comm_j \neq comm_i$ for all $comm_i \in P1$. In addition,

$$T_{comm_j\_start} = \max_{rank_x \in comm_j} T_{rank_x\_comm_j\_start},$$

where $rank_x$ participates in $comm_j$, since the collective starts only when the slowest rank joins. $T_{comm_i\_end}$ is the end time of the $comm_i$, the same for all participating ranks.

Based on the definition, we plot the CDF of the window sizes and categorize the windows according to the total traffic volume in $comm_j$ (communication after the window) in Fig. 4 for the Llama3-8B workload. Our observation is that more than 75% of the windows are over 1ms long and are similar in size across rails. And the biggest traffic volume (ReduceScatter) is preceded by the largest window (1000ms in average). For general workloads, the number of windows in one iteration can be determined by Eq. 1 (assuming FSDP is used, and TP domain does not exceed scale-up). Using the training configurations reported by [12], there are 127 windows over one Llama3.1-405B training iteration, approximately 20 seconds with 1k H100s (≈ 6 windows/second) [48].

Our findings indicate that topology reconfiguration can have a minimal impact on the application performance if the reconfiguration delay is on the order of milliseconds, allowing it to be hidden in the windows between parallelisms.

4 Opus: Parallelism-driven Reconfiguration

We summarize several objectives for performing in-job topology reconfiguration: **Objective 1:** reconfiguring for new traffic demand with minimal delay; **Objective 2:** reducing reconfiguration frequency to increase circuit up-time

$$\text{Window count} \leq \underbrace{4 \cdot (PP-1)}_{\substack{\text{PP and FSDP}\\ \text{fwd/bwd}\\ \text{interleave}}} + \underbrace{2 \cdot \frac{n_{\text{layer}}}{PP} - 1}_{\substack{\text{CP/EP and FSDP}\\ \text{1st microbatch}\\ \text{fwd interleave}}} + \underbrace{4 \cdot n_{\text{microbatch}}}_{\substack{\text{CP/EP and PP}\\ \text{fwd/bwd}\\ \text{interleave}}} + \underbrace{2 \cdot n_{\text{microbatch}} \cdot \left(2 \cdot \frac{n_{\text{layer}}}{PP} - 1\right)}_{\substack{\text{CP and EP}\\ \text{fwd/bwd interleave}}} + \underbrace{4}_{\substack{\text{PP warm-up, steady,}\\ \text{cool-down, and sync.}\\ \text{state transition}}} \quad (1)$$

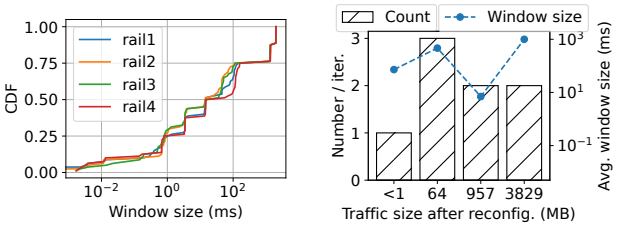

Figure 4: (a) CDF of window size from 10 iterations. (b) Rail 0 window break-down based on traffic volume after the window and before the next window, in one iteration. <1MB: AllReduce synchronization calls, 64MB: PP Send/Recv, 957MB: DP AllGather, 3829MB: DP ReduceScatter).

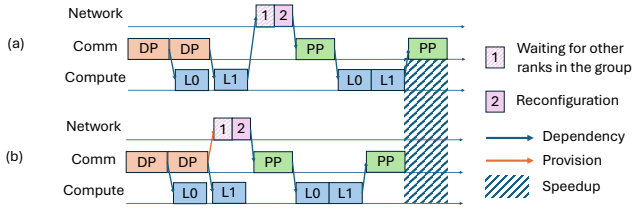

Figure 5: Reconfiguration during the warm-up stage of rank 0 and 4, (a) without provisioning, (b) with provisioning.

and prevent blocking traffic; **Objective 3:** avoiding circuit conflicts (*e.g.,* new circuit interferes with ongoing traffic, demand from two ranks result in conflicting topology, *etc.*).

Meeting Objective (1) and (2) requires the control layer to accurately interpret the traffic patterns defined by the ML frameworks. Overlapping the reconfiguration delay with the traffic window is necessary to reduce overhead. To achieve this, the control layer can perform *provisioning*, initiating the reconfiguration immediately once the previous communication kernel finishes (Fig. 5), if it identifies that the next communication belongs to a different parallelism. The control layer can also reduce reconfiguration frequency by only reconfiguring when there is a shift in parallelism. Fig. 3 shows all circuit configurations for the 3D-parallel workload.

To prevent circuit-level conflict with the workload DAG, a first-come-first-serve (FC-FS) scheduling policy is necessary in the control layer. A communication kernel which is issued first by the application should be served first within its communication group domain. The control layer should also prevent undesired control divergence across rails when collectives span multiple rails. Additionally, the reconfiguration can only be performed after the completion of the previous kernel, avoiding disrupting the ongoing traffic.

To achieve those objectives, we believe *the network control logic should be implemented in the application layer* where the parallelism-level hints and traffic patterns across iterations could be easily obtained. This design decision differs from previous control systems for reconfigurable networks in that they perform traffic and topology engineering on the packet or flow level in layer 2 and 3 [2, 3, 15, 22, 37, 45, 69, 77].

4.1 Design Sketch

We present a high-level design sketch of the control plane, Opus, for photonic rails, illustrated by Fig. 6. **Opus shim** runtime sits between the application and the collective communication layer in every scale-up domain. By "intercepting" communication calls from the application layer, the shim gets the information of the traffic volume and group. The shim CLI profiles, interprets, and predicts traffic demand for individual ranks. The shim manager coalesces demand across rails and issues reconfiguration requests. The manager also selects suitable networks based on traffic types, *e.g.,* the optical GPU-backend rail network for bulky data transfer, and high-bandwidth interconnect for intra-server traffic. **Opus controller** orchestrates each rail's OCSes to perform reconfiguration upon receiving requests from all ranks of one communication group. Together, the shim and the controller provide an illusion of an all-to-all GPU backend network, *treating network connectivity as an allocatable resource.*

During the first iteration of the training process, Opus shim profiles the traffic pattern and issues reconfiguration request only if the demand matrix of the parallelism changes, reducing reconfiguration frequency. Upon receiving the requests, Opus controller populates per-communication-group metadata in a job-specific table, generates circuit configurations, and configures the OCSes. The shim waits for the controller's acknowledgment, and calls the collective communication library to execute the communication.

During later iterations, Opus reduces reconfiguration overhead by *provisioning*. The shim issues speculative requests based on the profiled schedules immediately after the completion of the previous traffic from a different communication group. The controller fetches the cached circuit configurations and programs the switches, allowing the subsequent communication being executed as soon as possible.

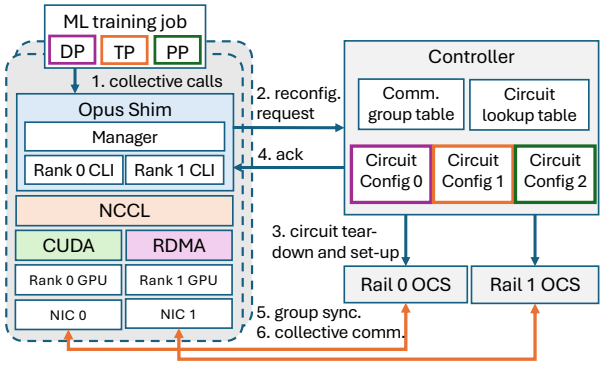

Figure 6: Opus control plane.

OCS Tech	Reconfig. time (ms)	Radix (ports)	# GPUs (GB200)	# GPUs (H200)
PLZT (EpiPhotonics)	0.00001	16	576	64
SiP (Lightmatter)	0.007	32	1152	128
RotorNet (InFocus)	0.01	128	4608	512
3D MEMS (Calient)	15	320	11520	1280
Piezo (Polatis)	25	576	20736	2304
Liquid crystal (Coherent)	100	512	18432	2048
Robotic (Telescent)	120000	1008	36288	4032

Table 3: Opus scalability-latency tradeoff. # GPUs = # (GPUs in scale-up) × radix/2 (using 2-port NIC configuration and bi-directional transceivers [42, 61]).

4.2 Opus Analysis

Scalability and energy efficiency. Using commodity switches, Opus GPU-backend network can scale up to 36K GPUs, and has a much lower cost and energy consumption compared to the state-of-the-art GPU networks. Tbl. 3 shows the scalability–reconfiguration latency tradeoff for two scale-up domains using specifications from OCS vendors and prior works [10, 13, 14, 39, 40, 45, 60, 73, 75]. We believe Piezo or 3D MEMS OCS is ideal for our design. To show Opus's cost and power efficiency, we use the methodology from [78, 79]. Opus saves cost by up to 70.5% and power by up to 95.84% (Fig. 7) thanks to the flat topology based on power-efficient OCSes and end-to-end optical data paths between GPUs. For further scalability, Opus can adopt a recursive BCube topology [25] with multiple levels of OCSes, similar to the design in [81], at the cost of requiring more server ports ($n_{level} \times n_{GPU/server}$).

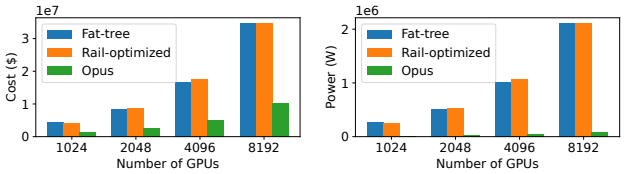

Figure 7: GPU-backend network cost and power comparison using DGX H200, 400Gbps transceivers and switches [17, 18, 51, 60], excluding fiber cable cost and power.

Reconfiguration provisioning. We simulate the Llama3-8B training workload with optical rails using the trace in §3.1. Fig. 8 shows the impact of reconfiguration delay on application when the rail reconfigures on the critical path (without provisioning), and proactively (with provisioning). The case of reconfiguration latency 0 stands for the baseline, a fully-connected network. Since Opus only reconfigures if the traffic pattern changes across parallelisms, the number of reconfiguration is kept small and the impact is minimal (6.5% iteration time increase with a 100ms switching delay over the baseline). The overhead could be minimized through provisioning. At 100ms switching delay, the network only has a 3.5% longer iteration than the baseline.

While our simulation assumes equal bandwidth between optical and electrical rails, optical interconnects have the potential to deliver significantly higher bandwidth, further reducing communication overhead in ML workloads [15, 77].

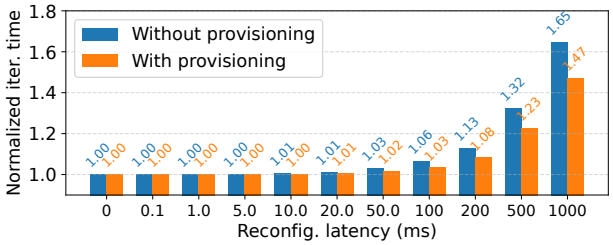

Figure 8: Iteration time for varying network reconfiguration delay (Llama3-8B with TorchTitan, TP=4, DP=PP=2).

5 Discussion

Control plane and synchronization. Scheduling and time-synchronization have been major challenges in reconfigurable networks [2, 3, 22, 37]. Network elements need to be tightly synchronized to prevent packet loss. Centralized and distributed algorithms such as Edmonds and VLB [15, 45, 77] are used for packet delivery and load balancing. Our design achieves time-synchronization implicitly using the application-driven reconfiguration scheme and the collective barriers of collective communication libraries. The simple FC-FS scheduling is sufficient as the bandwidth resource is not shared across jobs, and there is a sequential ordering of traffic demands defined by the job framework.

Supporting any communication patterns. Optical rails form a physical ring connecting GPUs of the same rank in scale-out. This is suitable for AllReduce, AllGather, ReduceScatter traffic, but is not optimal for certain traffic matrices that could not be easily implemented with a ring algorithm, such as AllToAll traffic in EP. Frequent switching among multiple communication groups (short AllReduce calls towards the end of a training iteration along PP and DP), leading to smaller windows, is also challenging. A possible solution is to perform multi-hopping through connected GPUs

in the same rail, and forward traffic through scale-up interconnect. The optical circuits can be configured to prioritize serving bottleneck traffic [39]. Different OCS configurations may diverge across rails to improve connectivity, and optimized collective algorithms for heterogeneous topologies could be used [67]. Small, bursty traffic with high-incast could also be offloaded to the host-based packet-switched network [66]. The control plane should adaptively select the appropriate solution based on the traffic volume, fan-out degree, window size, and the switching frequency of parallelism groups.

Reconfiguration granularity. Fine-grained reconfiguration in the optical rail network is required in hybrid parallelisms. Though the parallelism dimension of a GPU's communication undergoes sequential orders, allowing us to define the time window for reconfiguration, traffic from different parallelisms may occur simultaneously in two groups of GPUs within one job. In the example of Fig. 3(b), stage 2 ranks' DP AllGather and other stages' Send/Recv can be concurrent. Therefore, the circuit configurations should be conducted on the granularity of communication groups. Coarse-grained reconfiguration (directing all east-west links to north-south) would introduce conflicts with communication schedules of the ML frameworks. This demands flexible hardware switching for dynamic sets of ports (achievable by [60]) and per-rank control logic implementation. The flexibility is also important for multi-tenant environments.

Fault tolerance. Failure properties and fault handling techniques in the optical-rail topology are similar to those used in the rail-optimized topology and rail-only topology [78]. Using optical circuit switches does not introduce additional failure scenarios. Instead, the optical rails significantly reduce the failure points since it removes transceivers, lasers, and other electrical devices in the switching layer. Using passive optical switches such as AWGR and WaS link technologies could further reduce the failure probability [3, 4]. Having additional rails in the scale-out could offer redundancy when there are server uplink, transceiver (laser), NIC, or switch failures. When there are failures within the scale-up domain (*e.g.,* GPU or server failures), network operators and users can choose among (1) application layer workload resharding [19, 30], (2) network-protocol-layer reconfiguration [36](with scale-out topology reconfiguration if a reconfigured job consists of GPUs from a different scale-up), and (3) job migrations, depending on the severity and resource allocation status. Traditional fault-tolerance mechanisms used in SDN controller [5, 29] could be applied to Opus controllers to mitigate the single point of failure.

Limitations. Opus requires tight coordination across networking layers. The specifications of the reconfigurable switches (such as the reconfiguration speed, overhead per reconfiguration, radix, *etc.*) and the control plane scheduling decisions, should meet the application-level demands, specifically the deployment scale, the window sizes and the frequency of parallelism transitions in the scale-out. If an application's scaling techniques cause volatile switchings between parallelisms, or long-lasting unpredictable traffic patterns, solely relying on Opus's dynamic network reconfiguration strategy is unsuitable. Such applications could instead benefit from having a larger all-to-all scale-up network.

Opportunities. Our reconfigurable rail design does not require modifications to the existing applications, as the control layer handles circuit allocation. However, the control layer has no leverage on the window sizes and collective orderings for hiding the reconfiguration delay. Back-to-back or partially-overlapped traffic from two parallelism axes need to be spaced, resulting in bubbles and GPU idling. We believe there are opportunities for further reducing the switching overhead by presenting the network resources (*i.e.,* circuit connectivity) as callable abstractions to the application, similar to the abstractions for compute acceleration (*e.g.,* `cuda.amp` for tensor cores in PyTorch [62]). Utilizing the abstraction, the application, *e.g.,* PyTorch, could schedule computation kernels, communication collectives and the circuit allocation together, exploiting the overlapping possibilities between computation and network reconfiguration.

6 Conclusion

We propose a control layer to support ML workloads with hybrid parallelisms on reconfigurable optical rail-optimized networks. We show that performing circuit switching driven by parallelism shifts within a job can have 70.5% cost saving and 95.84% power reduction without significant impact on application performance, comparing to the fully-connected baselines. This is achieved through the tight integration between the application collective communication layer and the network controller, enabling fine-grained, parallelism-aware circuit provisioning and switching.

Acknowledgements

We thank our shepherd, Sujata Banerjee, and anonymous HotNets reviewers for their feedback. This work was supported in part by ACE, one of the seven centers sponsored by the Semiconductor Research Corporation (SRC) and DARPA under the Joint University Microelectronics Program 2.0 (JUMP 2.0). This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award NERSC DDR-ERCAP0032726.

References

[1] Saksham Agarwal, Qizhe Cai, Rachit Agarwal, David Shmoys, and Amin Vahdat. 2024. Harmony: A congestion-free datacenter architecture. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 329–343.

[2] Daniel Amir, Nitika Saran, Tegan Wilson, Robert Kleinberg, Vishal Shrivastav, and Hakim Weatherspoon. 2024. Shale: A practical, scalable oblivious reconfigurable network. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 449–464.

[3] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, et al. 2020. Sirius: A flat datacenter network with nanosecond optical switching. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 782–797.

[4] Kaoutar Benyahya, Ariel Gomez Diaz, Junyi Liu, Vassily Lyutsarev, Marianna Pantouvaki, Kai Shi, Shawn Yohanes Siew, Hitesh Ballani, Thomas Burridge, Daniel Cletheroe, et al. 2025. Mosaic: Breaking the Optics versus Copper Trade-off with a Wide-and-Slow Architecture and MicroLEDs. In *Proceedings of the ACM SIGCOMM 2025 Conference*. 234–247.

[5] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. 2014. ONOS: towards an open, distributed SDN OS. In *Proceedings of the third workshop on Hot topics in software defined networking*. 1–6.

[6] Maciej Besta and Torsten Hoefler. 2014. Slim Fly: A Cost Effective Low-Diameter Network Topology. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 348–359. doi:10.1109/SC.2014.34

[7] Broadcom Inc. 2025. BCM78909 51.2-Tb/s Multilayer Co-Packaged Optics Switch. Online; accessed July 5, 2025. https://www.broadcom.com/products/fiber-optic-modules-components/co-packaged-optics/switches/bcm78909 A high-radix, high-bandwidth CPO switch supporting up to 64×800GbE or 128×400GbE..

[8] Broadcom Inc. 2025. Co-Packaged Optics (CPO). https://www.broadcom.com/info/optics/cpo. Accessed: 2025-07-03.

[9] Broadcom Inc. Optical Systems Division. 2021. *SiPh Chiplets In Package (SCIP)*. Technical Report. Broadcom Inc., Irvine, CA, USA. https://docs.broadcom.com/doc/siph-chiplets-in-package-scip OSD CPO SCIP_20211106 V5.

[10] CALIENT Technologies, Inc. 2022. Calient's Optical Circuit Switch (S-Series) Datasheet. https://www.calient.net/wp-content/uploads/2022/06/Datasheet_Calients-Optical-Circuit-Switches.pdf. Accessed: 2025-07-03.

[11] Li Chen, Kai Chen, Zhonghua Zhu, Minlan Yu, George Porter, Chunming Qiao, and Shan Zhong. 2017. Enabling {Wide-Spread} Communications on Optical Fabric with {MegaSwitch}. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 577–593.

[12] Weiwei Chu, Xinfeng Xie, Jiecao Yu, Jie Wang, Amar Phanishayee, Chunqiang Tang, Yuchen Hao, Jianyu Huang, Mustafa Ozdal, Jun Wang, et al. 2025. Scaling Llama 3 Training with Efficient Parallelism Strategies. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*. 1703–1716.

[13] Coherent Corp. 2025. Optical Circuit Switch (OCS). https://www.coherent.com/networking/optical-circuit-switch. Accessed: 2025-07-10; Based on press release published March 25,2024; Coherent's liquid-crystal-based OCS architecture supports up to 300×300 ports and is optimized for AI/ML data center fabrics.

[14] EpiPhotonics Corp. 2025. Products. http://epiphotonics.com/products.html. Accessed: 2025-07-03.

[15] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*. 339–350.

[16] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research* 23, 120 (2022), 1–39.

[17] FS.COM. n.d.. Cisco Compatible 400GBASE-XDR4 QSFP-DD PAM4 1310nm 2km Module. https://www.fs.com/products/110530.html?attribute=94270&id=4477813. Accessed: 2025-07-02.

[18] FS.COM. n.d.. N9510-64D 64-Port Ethernet L3 Data Center Switch (Broadcom Tomahawk-4, 64×400GbE). https://www.fs.com/products/149853.html. Accessed: 2025-07-02.

[19] Swapnil Gandhi, Mark Zhao, Athinagoras Skiadopoulos, and Christos Kozyrakis. 2024. Recycle: Resilient training of large dnns using pipeline adaptation. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*. 211–228.

[20] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, et al. 2024. Rdma over ethernet for distributed training at meta scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 57–70.

[21] Alexandru M Gherghescu, Vlad-Andrei Bădoiu, Alexandru Agache, Mihai-Valentin Dumitru, Iuliu Vasilescu, Radu Mantu, and Costin Raiciu. 2024. I've Got 99 Problems But FLOPS Ain't One. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 195–204.

[22] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. 2016. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 216–229. doi:10.1145/2934872.2934911

[23] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[24] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A Scalable and Flexible Data Center Network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication* (Barcelona, Spain) *(SIGCOMM '09)*. Association for Computing Machinery, New York, NY, USA, 51–62. doi:10.1145/1592568.1592576

[25] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. 2009. BCube: a high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 63–74.

[26] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R. Das, Jon P. Longtin, Himanshu Shah, and Ashish Tanwer. 2014. FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-Space Optics. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 319–330. doi:10.1145/2619239.2626328

[27] Vipul Harsh, Sangeetha Abdu Jyothi, and P Brighten Godfrey. 2020. Spineless data centers. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 67–73.

[28] Hewlett Packard Enterprise. 2021. HPE Cray EX Supercomputer Overview. https://www.hpe.com/psnow/doc/a50002546enw. Accessed: 2025-07-09.

[29] Patrick Hunt, Mahadev Konar, Flavio P Junqueira, and Benjamin Reed. 2010. {ZooKeeper}: Wait-free coordination for internet-scale systems. In *2010 USENIX Annual Technical Conference (USENIX ATC 10)*.

[30] Insu Jang, Zhenning Yang, Zhen Zhang, Xin Jin, and Mosharaf Chowdhury. 2023. Oobleck: Resilient distributed training of large models using pipeline templates. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 382–395.

[31] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. 2023. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th annual international symposium on computer architecture*. 1–14.

[32] Mehrdad Khani, Manya Ghobadi, Mohammad Alizadeh, Ziyi Zhu, Madeleine Glick, Keren Bergman, Amin Vahdat, Benjamin Klenk, and Eiman Ebrahimi. [n. d.]. SiP-ML: High-Bandwidth Optical Network Interconnects for Machine Learning Training. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*.

[33] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems* 5 (2023), 341–353.

[34] Abhishek Vijaya Kumar, Arjun Devraj, Darius Bunandar, and Rachee Singh. 2024. A case for server-scale photonic connectivity. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks* (Irvine, CA, USA) *(HotNets '24)*. Association for Computing Machinery, New York, NY, USA, 290–299. doi:10.1145/3696348.3696856

[35] Abhishek Vijaya Kumar, Eric Ding, Arjun Devraj, Darius Bunandar, and Rachee Singh. 2025. LUMION: Fast Fault Recovery for ML Jobs Using Programmable Optical Fabrics. arXiv:2505.23105 [cs.LG] https://arxiv.org/abs/2505.23105

[36] ChonLam Lao, Minlan Yu, Aditya Akella, Jiamin Cao, Yu Guan, Pengcheng Zhang, Zhilong Zheng, Yichi Xu, Ennan Zhai, Dennis Cai, et al. 2024. TrainMover: Efficient ML Training Live Migration with No Memory Overhead. *arXiv e-prints* (2024), arXiv–2412.

[37] Cong Liang, Xiangli Song, Jing Cheng, Mowei Wang, Yashe Liu, Zhenhua Liu, Shizhen Zhao, and Yong Cui. 2024. NegotiaToR: Towards A Simple Yet Effective On-demand Reconfigurable Datacenter Network. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 415–432.

[38] Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, et al. 2024. TorchTitan: One-stop PyTorch native solution for production ready LLM pre-training. *arXiv preprint arXiv:2410.06511* (2024).

[39] Xudong Liao, Yijun Sun, Han Tian, Xinchen Wan, Yilun Jin, Zilong Wang, Zhenghang Ren, Xinyang Huang, Wenxue Li, Kin Fai Tse, et al. 2025. mFabric: An Efficient and Scalable Fabric for Mixture-of-Experts Training. *arXiv preprint arXiv:2501.03905* (2025).

[40] Lightmatter, Inc. 2025. Passage Technology. https://lightmatter.co/products/passage/. Accessed: 2025-07-03.

[41] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[42] Hong Liu, Ryohei Urata, Kevin Yasumura, Xiang Zhou, Roy Bannon, Jill Berger, Pedram Dashti, Norm Jouppi, Cedric Lam, Sheng Li, et al. 2023. Lightwave fabrics: at-scale optical circuit switching for datacenter and machine learning systems. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 499–515.

[43] Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889* (2023).

[44] Lumentum Holdings Inc. 2025. Lumentum Optical Circuit Switch to Improve Next-Generation AI Data Center Scalability. https://www.lumentum.com/en/media-room/news-releases/lumentum-optical-circuit-switch-improve-next-generation-ai-data-center. Accessed June 20, 2025.

[45] William M Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C Snoeren, and George Porter. 2017. Rotornet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 267–280.

[46] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740* (2017).

[47] National Energy Research Scientific Computing Center (NERSC). 2025. Perlmutter Architecture — NERSC Documentation. https://docs.nersc.gov/systems/perlmutter/architecture/. Accessed: 2025-07-04.

[48] NVIDIA. 2025. Llama-3.1-405B DGXC Benchmarking Recipe. https://catalog.ngc.nvidia.com/orgs/nvidia/teams/dgxc-benchmarking/resources/llama31-405b-dgxc-benchmarking-a. Version 24.11.1, modified January 29, 2025.

[49] NVIDIA Corporation. 2020. *NVIDIA Collective Communication Library (NCCL): Creating a Communicator*. NVIDIA. https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/communicators.html Accessed July 6, 2025.

[50] NVIDIA Corporation. 2022. Doubling all-to-all Performance with NCCL 2.12: Introducing PXN (PCI X NVLink). NVIDIA Developer Blog. https://developer.nvidia.com/blog/doubling-all2all-performance-with-nvidia-collective-communication-library-2-12/ Describes PXN, which enables GPU-to-NIC communication via NVLink to optimize rail-aligned collective performance.

[51] NVIDIA Corporation. 2024. ConnectX-7 400G Adapters Datasheet. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/connectx-7-datasheet. Accessed: 2025-07-02.

[52] NVIDIA Corporation. 2025. Co-Packaged Silicon Photonics Networking Switches. Online; accessed July 5,2025. https://www.nvidia.com/en-us/networking/products/silicon-photonics/ Describes NVIDIA's co-packaged optics (CPO) switches with integrated silicon photonics.

[53] NVIDIA Corporation. 2025. *NVIDIA Announces Spectrum-X Photonics, Co-Packaged Optics Networking Switches to Scale AI Factories to Millions of GPUs*. Press Release. NVIDIA Corporation, Santa Clara, CA, USA. https://nvidianews.nvidia.com/news/nvidia-spectrum-x-co-packaged-optics-networking-switches-ai-factories Unveiled at GTC 2025.

[54] NVIDIA Corporation. 2025. *NVIDIA Collective Communications Library (NCCL)*. NVIDIA Developer. https://developer.nvidia.com/nccl Version 2.x; MPI-compatible multi-GPU / multi-node collective communication library.

[55] NVIDIA Corporation. 2025. *NVIDIA DGX H200 Datasheet*. Datasheet. NVIDIA Corporation, Santa Clara, CA. https://resources.nvidia.com/en-us-dgx-systems/dgx-h200-datasheet Includes specifications of the DGX H200 system, featuring 8× H200 GPUs, dual Xeon Platinum 8480C CPUs, 2 TB system memory, 30 TB NVMe SSD, and full NVIDIA AI Enterprise software stack.

[56] NVIDIA Corporation. 2025. *NVIDIA DGX SuperPOD*. NVIDIA. https://www.nvidia.com/en-us/data-center/dgx-superpod/ Full-stack data center platform scaling to tens of thousands of GPUs; includes compute, networking, storage, and software.

[57] NVIDIA Corporation. 2025. *NVIDIA HGX Platform*. NVIDIA. https://www.nvidia.com/en-us/data-center/hgx/ Reference architecture combining GPUs, NVLink/NVSwitch, networking, and AI/HPC software stack.

[58] NVIDIA Corporation. 2025. *Rail Optimized Topology Validation*. NVIDIA Networking, Santa Clara, CA. https://docs.nvidia.com/networking/display/ibdiagnetusermanualv221/Rail+Optimized+Topology+Validation Part of the ibdiagnet InfiniBand Fabric Diagnostic Tool User Manual; describes cabling validation and compute-fabric alignment in DGX SuperPOD rail-optimized fabrics.

[59] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. Pmlr, 1310–1318.

[60] Polatis (a HUBER+SUHNER company). n.d.. Series 7000 — 384×384-port Software-Defined Optical Circuit Switch. https://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuit-switch-sdn-enabled.asp. Accessed: 2025-07-01.

[61] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. 2022. Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. 66–85.

[62] PyTorch Team. 2025. Automatic Mixed Precision package (`torch.amp`). https://pytorch.org/docs/stable/amp.html. Accessed: 2025-07-09.

[63] Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. 2023. Zero bubble pipeline parallelism. *arXiv preprint arXiv:2401.10241* (2023).

[64] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3505–3506.

[65] Peter Sanders, Jochen Speck, and Jesper Larsson Träff. 2009. Two-tree algorithms for full bandwidth broadcast, reduction and scan. *Parallel Comput.* 35, 12 (2009), 581–594.

[66] Ken-ichi Sato. 2023. Optical switching will innovate intra data center networks [Invited Tutorial]. *Journal of Optical Communications and Networking* 16, 1 (2023), A1–A23.

[67] Aashaka Shah, Vijay Chidambaram, Meghan Cowan, Saeed Maleki, Madan Musuvathi, Todd Mytkowicz, Jacob Nelson, Olli Saarikivi, and Rachee Singh. 2023. TACCL: Guiding Collective Algorithm Synthesis using Communication Sketches. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 593–612. https://www.usenix.org/conference/nsdi23/presentation/shah

[68] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[69] Vishal Shrivastav, Asaf Valadarsky, Hitesh Ballani, Paolo Costa, Ki Suh Lee, Han Wang, Rachit Agarwal, and Hakim Weatherspoon. 2019. Shoal: A Network Architecture for Disaggregated Racks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. USENIX Association, Boston, MA, 255–270. https://www.usenix.org/conference/nsdi19/presentation/shrivastav

[70] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. *SIGCOMM Comput. Commun. Rev.* 45, 4 (aug 2015), 183–197. doi:10.1145/2829988.2787508

[71] Ankit Singla, P Brighten Godfrey, and Alexandra Kolla. 2014. High throughput data center topology design. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 29–41.

[72] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. 2012. Jellyfish: Networking Data Centers Randomly. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX Association, San Jose, CA, 225–238. https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/singla

[73] Sithara P Sreenilayam, Dermot Brabazon, and Yuri P Panarin. 2019. Fast ferroelectric liquid crystal based optical switch: simulation and experiments. *Crystals* 9, 8 (2019), 388.

[74] Nouamane Tazi, Ferdinand Mom, Haojun Zhao, Phuc Nguyen, Mohamed Mekkouri, Leandro Werra, and Thomas Wolf. 2025. The Ultra-Scale Playbook: Training LLMs on GPU Clusters. https://huggingface.co/spaces/nanotron/ultrascale-playbook. Accessed: 2025-05-16.

[75] Telescent Inc. n.d.. Products | Telescent. https://www.telescent.com/products. Accessed: 2025-07-01.

[76] Rajeev Thakur and William D Gropp. 2003. Improving the performance of collective operations in MPICH. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*. Springer, 257–267.

[77] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. 2010. C-Through: Part-Time Optics in Data Centers. In *Proceedings of the ACM SIGCOMM 2010 Conference* (New Delhi, India) *(SIGCOMM '10)*. Association for Computing Machinery, New York, NY, USA, 327–338. doi:10.1145/1851182.1851222

[78] Weiyang Wang, Manya Ghobadi, Kayvon Shakeri, Ying Zhang, and Naader Hasani. 2024. Rail-only: A low-cost high-performance network for training LLMs with trillion parameters. In *2024 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, 1–10.

[79] 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)*. 739–767.

[80] Zhenguo Wu, Liang Yuan Dai, Ziyi Zhu, Asher Novick, Madeleine Glick, and Keren Bergman. 2023. SiP Architecture For Accelerating Collective Communication in Distributed Deep Learning. In *2023 Optical Fiber Communications Conference and Exhibition (OFC)*. 1–3. doi:10.1364/OFC.2023.W1G.1

[81] Zhenguo Wu, Liang Yuan Dai, Yuyang Wang, Songli Wang, and Keren Bergman. 2024. Flexible silicon photonic architecture for accelerating distributed deep learning. *Journal of Optical Communications and Networking* 16, 2 (2024), A157–A168.

[82] Eric P Xing, Qirong Ho, Wei Dai, Jin-Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. 2015. Petuum: A new platform for distributed machine learning on big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1335–1344.

[83] Sharada Yeluri. 2023. *Optimizing Power Consumption in High-End Routers*.

[84] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277* (2023).

[85] Yazhou Zu, Alireza Ghaffarkhah, Hoang-Vu Dang, Brian Towles, Steven Hand, Safeen Huda, Adekunle Bello, Alexander Kolbasov, Arash

Rezaei, Dayou Du, Steve Lacy, Hang Wang, Aaron Wisner, Chris Lewis, and Henri Bahini. 2024. Resiliency at Scale: Managing Google's TPUv4 Machine Learning Supercomputer. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 761–774.