Computer Fraud and Security
ISSN (online): 1873-7056

Network Architectures for Machine Learning Training Workloads in Cloud Data Centers

Vijaya Bhaskar Methuku
Independent Researcher, USA

Abstract

Networks in cloud data centers were designed around loosely coupled, request-response workloads generating predominantly north-to-south traffic, yet distributed machine learning training demands sustained, tightly synchronized east-to-west communication across large accelerator clusters. While prior work has addressed individual components of this problem — collective communication algorithms, transport protocols, and specific topology deployments — no unified framework has synthesized these threads into a structured architectural comparison spanning the full spectrum from general-purpose cloud to purpose-built ML training infrastructure. This paper fills that gap through a systematization of knowledge on the network architectural requirements of distributed ML training, synthesizing peer-reviewed literature from 2013 to 2024. Sources were selected through a keyword-based search of ACM Digital Library, IEEE Xplore, and USENIX proceedings, filtered by deployment scale and relevance to network performance. The analysis characterizes communication patterns across data-parallel, tensor-parallel, and pipeline-parallel training strategies; reviews leaf-spine, multi-rail, and optical circuit-switched topologies; and examines congestion dynamics arising from collective communication operations at scale. Evidence from the synthesized literature suggests that purpose-built ML training networks achieve substantially higher cluster efficiency than general-purpose cloud fabrics, and that elevated per-operation latency introduces measurable overhead accumulating across multi-day training runs. A production deployment scenario and practical design recommendations organized by cluster scale tier are also presented.

Keywords: Machine Learning, Distributed Training, Cloud Data Center, Network Topology, East–West Traffic, Gradient Synchronization, Congestion Control, Accelerator Clusters, RDMA, Collective Communication, In-Network Computing, Optical Interconnect

1. Introduction

Cloud data center networks have, for most of their history, been engineered around the needs of enterprise applications, commercial web services, and large-scale distributed software platforms. These workloads consist of loosely coupled services communicating through request-response interactions, mediated by application gateways, load balancers, or service meshes. The traffic patterns that emerge from these architectures are dominated by north–south flows, communication between clients outside the data center and the services hosted within it, rather than by intensive, high-bandwidth exchanges among nodes within the same cluster [1]. Because internal communication is infrequent and asynchronous, network infrastructure was designed around high external throughput, multi-tenant isolation, and cost-effective oversubscription at the aggregation layer.

That design paradigm is now being disrupted by the rapid and sustained expansion of machine learning workloads. Training a modern deep learning model requires coordinated execution across hundreds or thousands of accelerators, GPUs, TPUs, and purpose-built AI chips, synchronized repeatedly across every training iteration [2]. Contemporary models, particularly large language models and multimodal foundation architectures, routinely contain billions of parameters, and the most ambitious systems have crossed into the trillions. At these scales, training within any practical time window requires massive parallel hardware whose network demands are qualitatively different from anything the data center was built to support [3]. Recent workload characterization research confirms that ML training traffic exhibits statistical properties, specifically high burstiness, strict synchronization, and large sustained flows, that are fundamentally incompatible with conventional cloud fabric assumptions [11].

The key difference lies in synchronization. A microservice can tolerate a slow database response and continue handling other requests; a GPU waiting for an all-reduce cannot begin the next iteration and sits idle. Every node waits simultaneously, so a single delayed response stalls all participants — and across tens of thousands of iterations, this accumulates directly into wall-clock training time and infrastructure cost [4].

This paper presents a systematic analysis of the network architectural requirements of distributed ML training workloads, drawing on the published research literature to build a unified characterization and comparison framework. The analysis addresses four specific questions: (1) How do the communication patterns of ML training workloads differ structurally from those of traditional cloud applications? (2) What network topology and transport characteristics are required to sustain high cluster efficiency at scale? (3) How does network congestion quantitatively degrade training performance, and what mechanisms exist to manage it? (4) What architectural directions are most likely to close the gap between current infrastructure capabilities and future training demands? By synthesizing recent advances in in-network computing [7], optical circuit switching [1, 12], high-bandwidth GPU interconnects [13], and energy-aware network design [14], this work provides a structured foundation for future infrastructure design.
The specific contributions of this paper are:

- A structured characterization of ML training communication patterns across data-parallel, tensor-parallel, and pipeline-parallel strategies, contrasted against traditional cloud traffic
- A comparative analysis of leaf-spine, multi-rail, and optical circuit-switched topologies evaluated against ML training requirements
- A quantitative analysis of how network congestion degrades cluster efficiency and training iteration time at scale
- A production deployment case study constructed from published operational data illustrating these principles in practice
- A practical design recommendation framework organized by cluster scale tier, and a five-direction research agenda for future infrastructure development

2. Related Work

The intersection of network architecture and distributed machine learning has attracted sustained research attention over the past decade, with contributions spanning system design, transport protocols, topology optimization, and collective communication algorithms. This section situates the present analysis within that body of work and clarifies the contribution of this systematic analysis.

2.1 Collective Communication and Distributed Training Frameworks

The foundational work on distributed training communication emerged from practical implementations in production ML frameworks. Research on PyTorch's distributed communication design [6] provided detailed measurements of gradient synchronization overhead and the engineering choices made to overlap communication with computation in data-parallel training. Work on large-scale language model training using Megatron-LM [2] addressed the specific challenges of training on GPU clusters, introducing pipeline and tensor parallelism strategies and measuring their communication requirements at scale. The ZeRO-Infinity framework [3] extended this to extreme parameter scales, partitioning model state across GPU, CPU, and NVMe memory hierarchies to overcome GPU memory limitations in trillion-parameter models. These works establish the communication requirements that network infrastructure must satisfy, but address network design from the ML systems perspective rather than a network engineering one.

2.2 Data Center Network Architectures and Optical Switching

The evolution of Google's Jupiter data center network architecture [1] represents the most comprehensive published account of how a hyperscale cloud provider has evolved its data center fabric to accommodate changing workload requirements, including the incorporation of optical circuit switches and software-defined networking control. Research on optical circuit switching fabrics specifically designed for ML training traffic [12] demonstrated that dedicated high-bandwidth circuits established for the duration of collective communication operations can provide significant latency and throughput advantages over traditional electrical packet-switched networks. Work on cloud-scale load balancing [5] illuminates the north–south traffic management challenges of conventional cloud networks. The design and deployment of RDMA over Converged Ethernet at scale [8] provided detailed analysis of the lossless Ethernet requirements, PFC configuration, and congestion behavior observed in operational deployments.

2.3 Congestion Control for High-Performance Fabrics

Congestion control in RDMA-capable data center fabrics has been the subject of intensive research. The DCQCN protocol [4] combined ECN-based feedback with rate control mechanisms specifically designed for RoCE deployments and characterized the congestion dynamics that arise during large-scale RDMA communication. The Swift protocol [9] used end-to-end delay rather than explicit queue signals as its primary congestion indicator, enabling earlier intervention before queue saturation triggers PFC pauses. Production data center traffic characterization [11] provided detailed empirical profiles of how collective communication incast manifests in real ML training environments, supplying the traffic distributions needed to calibrate congestion control algorithms beyond synthetic benchmarks.

2.4 In-Network Computing, High-Bandwidth Interconnects, and Energy Efficiency

Research on in-network gradient aggregation [7] demonstrated that performing partial gradient summation within network switches as traffic traverses the fabric can substantially reduce communication volume and all-reduce latency. Work on RDMA-accelerated data processing [10] provided evidence that RDMA benefits extend beyond dedicated ML training to broader high-performance workloads. At the intra-node level, technical characterization of the NVLink interconnect architecture [13] established that memory-coherent GPU-to-GPU bandwidth exceeding 600 GB/s fundamentally changes the communication performance profile for tensor-parallel workloads confined within a single server chassis. Research on energy-proportional network design for ML training clusters [14] quantified the power consumption overhead of large-scale all-reduce operations and proposed adaptive link frequency scaling as a mechanism for reducing energy consumption during low-utilization phases without compromising peak throughput.

2.5 Positioning of This Analysis

Prior work has addressed individual components of the ML training network problem, communication algorithms, transport protocols, and specific topology deployments but has not provided a unified analytical framework that synthesizes these contributions into a structured comparison of architectural requirements across the full spectrum from traditional cloud to purpose-built ML training infrastructure. This paper fills that gap with a systematization of knowledge across these threads, a synthesis rather than an empirical study, producing a coherent architectural comparison, a production deployment case study, and a grounded research agenda. The five additional references introduced above [11, 12, 13, 14] extend the foundational literature to cover workload characterization, optical switching, high-bandwidth interconnects, and energy efficiency.

3. Methodology

This paper employs a systematic literature analysis methodology to develop a unified architectural comparison framework for ML training network infrastructure. The methodology is organized around four explicit research questions, a structured evidence synthesis process, and a baseline comparison framework drawn from published operational data.

3.1 Research Questions

The analysis is structured around the following four research questions:

- RQ1: How do the communication patterns of distributed ML training workloads differ structurally from those of traditional cloud applications in terms of traffic direction, message size, synchronization model, and latency sensitivity?
- RQ2: What network topology characteristics, specifically oversubscription ratio, bisection bandwidth, and transport protocol, are required to sustain cluster efficiency above 85% at scales of 256 GPUs and beyond?
- RQ3: How does network congestion quantitatively degrade training iteration time and cluster efficiency, and what congestion control mechanisms are most effective for collective communication traffic patterns?
- RQ4: Which architectural directions, in-network computing, optical circuit switching, disaggregated fabrics, or adaptive energy management, are most likely to close the performance gap between general-purpose cloud networks and dedicated ML training infrastructure over the next five years?

3.2 Evidence Synthesis and Baseline Framework

Quantitative data presented throughout this paper are drawn from peer-reviewed publications, conference proceedings, and vendor technical documentation covering the period 2013–2024. Sources were identified through keyword searches of ACM Digital Library, IEEE Xplore, arXiv, and USENIX proceedings using the terms: distributed ML training network, collective communication data center fabric, RDMA gradient synchronization, and east-west accelerator cluster traffic, covering publications from 2013 to 2024. A source was included if it reported empirical measurements from deployments of at least 64 accelerators and directly addressed network performance or topology design. Sources were excluded if they addressed single-node training only, provided no quantitative data, or were superseded by a later version included in the corpus. Reference [13] is vendor documentation included in the absence of a peer-reviewed equivalent reporting NVLink bandwidth measurements at comparable specificity. Where multiple sources report measurements for the same phenomenon, ranges are given to reflect variation across deployment configurations rather than potentially unrepresentative point estimates.

Baseline comparisons throughout the paper employ two reference configurations: a 'traditional cloud fabric' characterized by high oversubscription, TCP/IP transport, and traffic patterns dominated by north–south flows as described in [1] and [5]; and a 'purpose-built ML fabric' characterized by minimal oversubscription, RoCE v2 transport, lossless Ethernet with PFC and ECN, and sustained east–west collective communication traffic as described in [2], [4], and [8]. Performance metrics—cluster efficiency, all-reduce completion time, and iteration time overhead, are reported relative to these two baselines to enable direct comparison across the sections that follow.

4. Traditional Cloud Application Traffic Patterns

Understanding why machine learning training workloads place such unusual demands on network infrastructure requires first appreciating the assumptions that shaped conventional cloud network design. Most cloud applications are assembled from discrete microservices, authentication handlers, business logic processors, caching layers, data access components, interacting through API calls or database queries. The messages involved in these exchanges are almost universally small: a few hundred bytes for a session token, perhaps a few kilobytes for a JSON payload carrying search results or user profile data [5]. A single user-initiated request might trigger calls to a dozen internal services in parallel, but none of those services need to wait for one another to make progress, and none share a computational state that must be kept in lockstep across nodes.

The dominant traffic pattern in these environments flows north to south: external users make requests that enter the data center through load balancers and edge gateways, traverse one or more service tiers, and return responses outbound through the same path. Internal east–west communication exists but is generally asynchronous and loosely coupled, services do not wait on one another in ways that create synchronized cluster-wide dependencies. This looseness allows traditional cloud workloads to tolerate moderate latency variation and brief congestion without significant user-facing degradation [1]. Production traffic characterization studies at hyperscale providers confirm that east–west flows in conventional cloud environments account for less than 30% of total fabric bandwidth consumption, compared to over 90% in dedicated ML training clusters [11].

This tolerance has historically been the economic justification for deploying cloud networks with moderate oversubscription ratios, commonly 3:1 to 5:1, where the bandwidth available between leaf and spine layers is a fraction of the aggregate bandwidth at the access layer. When no single workload drives sustained, simultaneous traffic from all connected servers, oversubscription is a reasonable engineering tradeoff. This assumption fails entirely for distributed ML training, where every node drives simultaneous, sustained traffic during every synchronization phase, and no application-level mechanism exists to absorb the resulting congestion. Figure 1 below illustrates this fundamental structural contrast between the two traffic regimes.

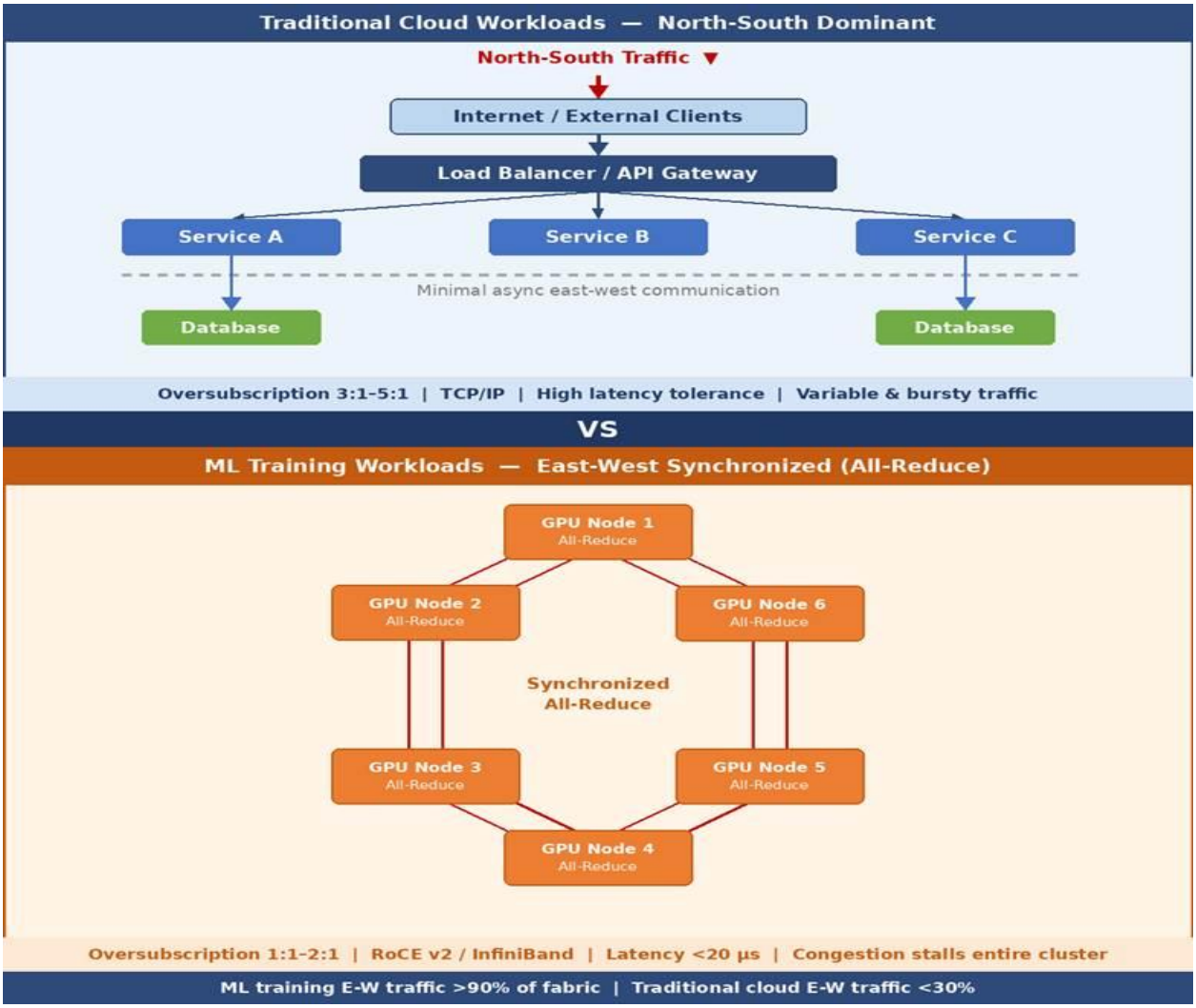

Figure 1: Traffic Pattern Comparison, Traditional Cloud (North–South Dominant) vs. ML Training (East–West Synchronized)

The contrast in arrow density between the two panels reflects the bandwidth asymmetry that drives the oversubscription requirements discussed in Section 6.1.

5. Communication Patterns in Distributed Machine Learning Training

The communication requirements of distributed ML training emerge from the mathematical structure of the training process itself, which is why they are so difficult to accommodate within infrastructure designed for different assumptions. Training a neural network involves iteratively adjusting model parameters to minimize a loss function computed over a training dataset, and when this process is distributed across many nodes, the nodes must periodically exchange parameter updates to remain consistent. The specific form of this exchange depends on the parallelization strategy in use, but all share the property that communication is unavoidable, frequent, and must complete before computation can proceed [6].

5.1 Data Parallelism and the All-Reduce Collective

Data parallelism is the most widely used strategy, and it most clearly illustrates the communication challenge. In a data-parallel training setup, each worker node holds a complete copy of the model and processes a different shard of the training dataset during each iteration. After computing gradients from its local batch, each worker must combine its values with those of all other workers, most commonly via all-reduce, before any worker can update its parameters and begin the next iteration. The all-reduce operation requires every node to both send and receive gradient data representing

the full set of model parameters, which for large models can amount to multiple gigabytes of tensor data per iteration, all of which must traverse the network fabric simultaneously [2].

The mechanics of widely used all-reduce algorithms such as ring all-reduce or recursive halving-doubling further shape the traffic pattern. In ring all-reduce, nodes are arranged in a logical ring, and gradient data circulates around the ring in two phases—a scatter-reduce phase followed by an all-gather phase, with each node transmitting and receiving gradient shards at each step. This algorithm is bandwidth-optimal; total data transmitted per node scales as a constant multiple of the gradient size regardless of cluster size but requires sustained bidirectional communication across the entire cluster, generating large bursts of east-west traffic at every synchronization event [7]. High-bandwidth intra-node interconnects additionally benefit multi-GPU nodes by offloading within-server communication before it reaches the external fabric, reducing external traffic volumes for tensor-parallel components of training [13].

Figure 2 illustrates the ring all-reduce algorithm, showing the scatter-reduce and all-gather phases and the direction of data flow across the cluster during synchronization.

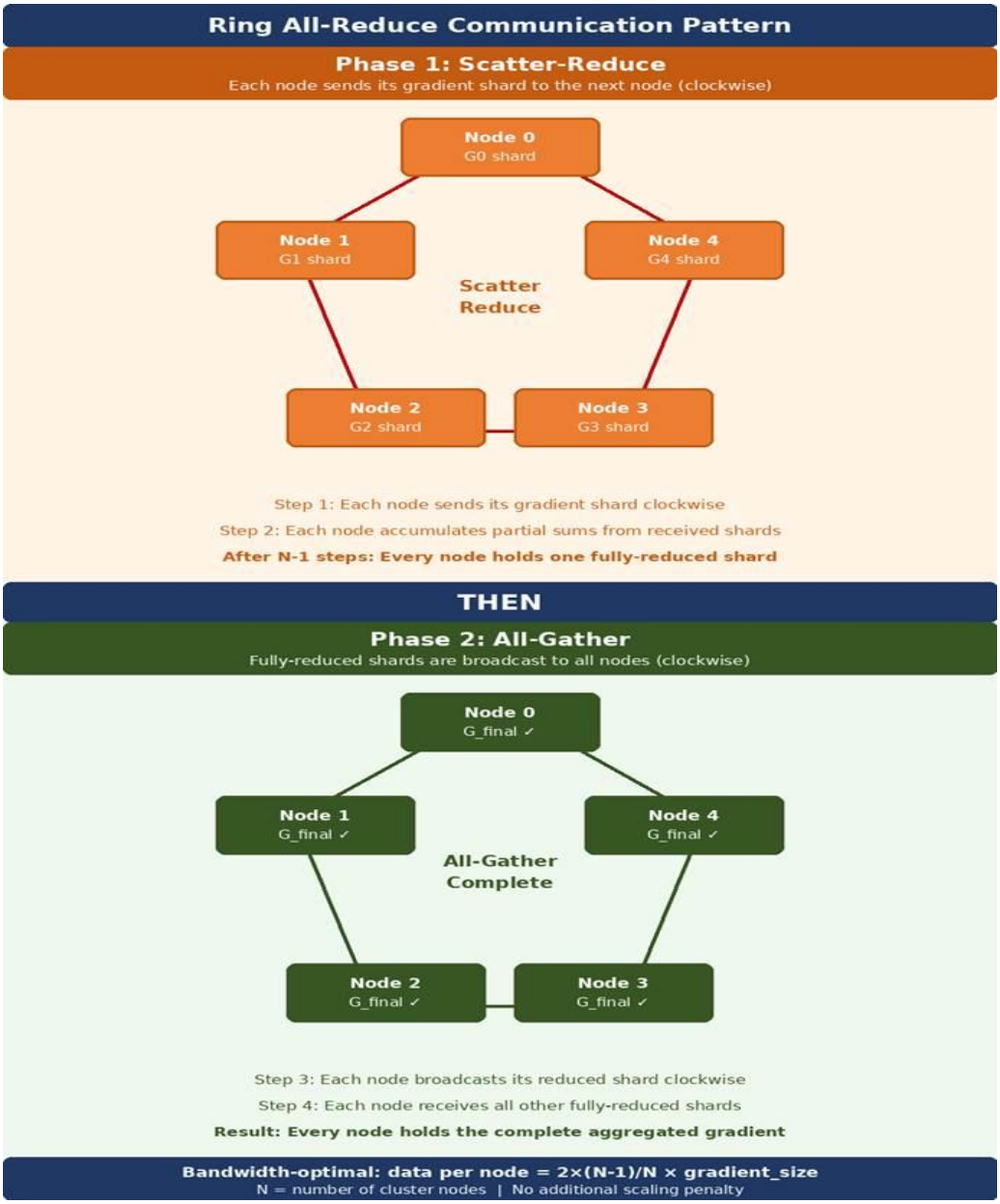

Figure 2: Ring All-Reduce Communication Pattern, Phase 1 Scatter-Reduce and Phase 2 All-Gather

Both phases require simultaneous bidirectional transmission across all participating nodes, which is the structural origin of the incast congestion examined in Section 7.

5.2 Tensor Parallelism and Pipeline Parallelism

For models too large to fit in a single accelerator's memory, tensor parallelism and pipeline parallelism are increasingly deployed alongside data parallelism. Tensor parallelism partitions individual neural network layers across multiple devices, requiring continuous exchanges of activation tensors during both forward and backward passes through the network. Pipeline parallelism distributes different layers of the model across devices arranged in a sequential pipeline, creating point-to-point communication dependencies between pipeline stages at every micro-batch boundary. When combined, as they frequently are for enormous models [2], the resulting communication pattern is complex, overlapping, and highly sensitive to asymmetry in network performance across fabric paths. At the intra-node level, high-bandwidth memory-coherent interconnects operating at bandwidths exceeding 600 GB/s [13] effectively isolate tensor-parallel communication from the external network and reserve external bandwidth for data-parallel gradient synchronization across nodes.

5.3: Gradient Synchronization Bandwidth Requirements

Table 1 quantifies the gradient synchronization bandwidth demands that emerge from the communication patterns described in the preceding sections. A 64-GPU cluster, modest by current production standards, already places moderate-to-high synchronization bandwidth demands on the fabric, far exceeding per-server capacity in most multi-tenant cloud environments. At 1,024 GPUs, that requirement grows substantially, and it continues to grow as larger clusters and wider models push gradient volumes higher.

Cluster Size	Gradient Volume per Iteration	Synchronization Bandwidth Demand
64 GPUs	Low	Moderate
256 GPUs	Moderate	High
1,024 GPUs	High	Very High

Table 1: Gradient Synchronization Bandwidth Requirements by Cluster Size

6. Network Topologies for ML Training Clusters

The distinct demands of distributed training have driven the development of specialized network architectures that prioritize collective communication performance over general-purpose cloud requirements.

6.1 Leaf-Spine Architecture

The leaf-spine topology is the dominant pattern in data center networks and the widely reported foundation of production ML training clusters, though the parameters used in training environments differ substantially from general cloud deployments. Compute nodes connect to top-of-rack leaf switches, which connect to spine switches providing multiple parallel paths across the fabric. This multi-path structure suits collective communication well: gradient traffic from hundreds of nodes can be distributed across equal-cost paths, reducing the likelihood of individual link saturation [1]. The critical difference in ML training cluster deployments is that oversubscription ratios are set far lower, often 1:1 or 2:1, to ensure that the full theoretical bisection bandwidth of the fabric is available during peak synchronization phases rather than being shared across a diverse population of variable workloads [8].

Figure 3 illustrates the leaf-spine topology as deployed in ML training clusters, showing full-mesh connectivity between spine and leaf switches, the per-server bandwidth available at 400 Gbps interfaces, and the oversubscription ratio configuration that distinguishes ML-optimized from general-purpose deployments.

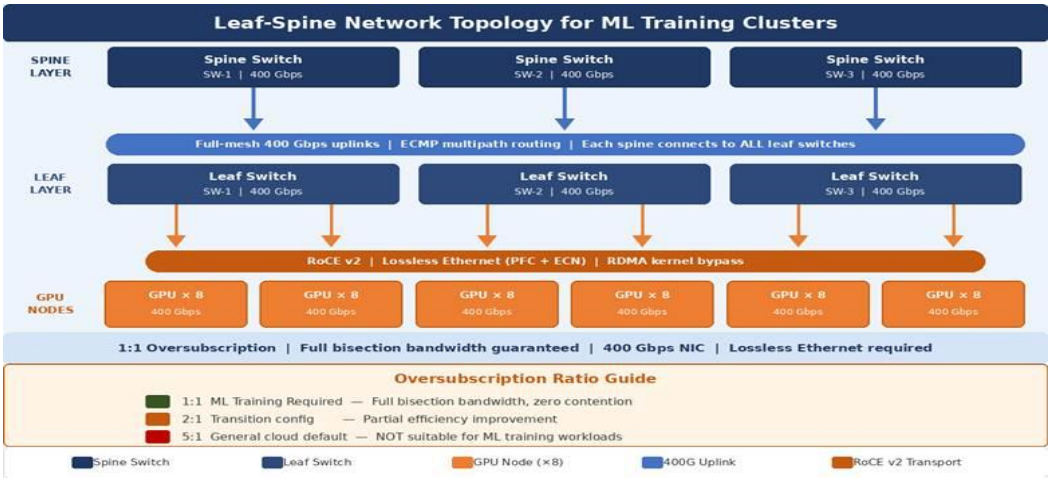

Figure 3: Leaf-Spine Network Topology for ML Training Clusters, Full-Mesh Spine-Leaf Connectivity with 1:1 Oversubscription

The 1:1 uplink-to-downlink ratio at the spine layer is the configuration parameter that most directly produces the cluster efficiency values reported in Table 6.

6.2 Multi-Rail and Alternative Topologies

An alternative topology increasingly deployed in purpose-built AI accelerator clusters is the multi-rail or multi-plane network, in which each compute node connects to multiple independent network planes operating in parallel. This approach multiplies the available bandwidth per node without requiring higher-radix individual switches and provides redundancy that limits the impact of individual switch or link failures on collective communication operations. For very large clusters, fat-tree and dragonfly topologies offer additional options for reducing the number of hops between any pair of nodes, which lowers baseline latency and improves the performance of latency-sensitive collective operations at the cost of more complex routing and traffic engineering [9].

Figure 4 illustrates the multi-rail architecture, showing how GPU servers are multi-homed across three independent network planes to multiply available bandwidth and provide independent failure domains.

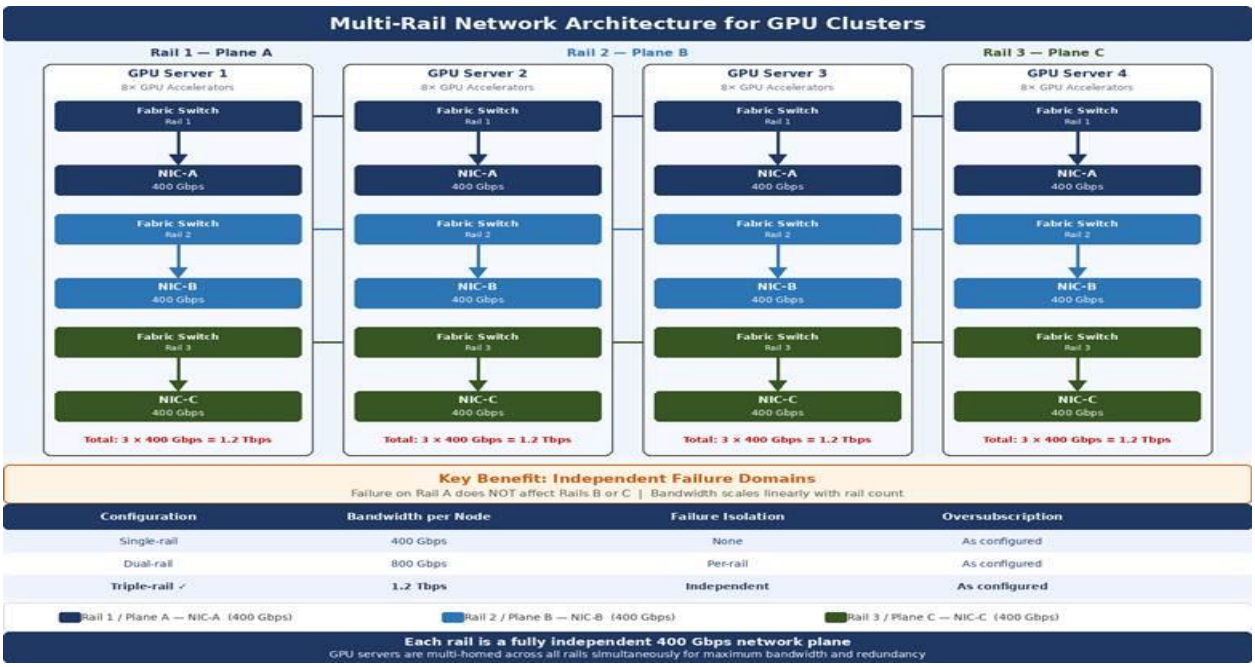

Figure 4: Multi-Rail Network Architecture, GPU Servers Multi-Homed Across Three Independent 400 Gbps Network Planes

Optical circuit switching represents a particularly promising direction for multi-rail fabrics. Research in this area [12] has demonstrated that optical circuit switches can establish dedicated high-bandwidth circuits between GPU server groups for the duration of an all-reduce operation, eliminating the packet contention that degrades performance in purely electrical packet-switched fabrics. The reconfigurability of optical fabrics also enables the network topology to adapt dynamically to the communication pattern of the active training job, a capability with substantial implications for multi-tenant training infrastructure where different jobs may require different traffic patterns simultaneously.

7. RDMA over Converged Ethernet

A particularly consequential technology choice is RoCEv2 (Remote Direct Memory Access over Converged Ethernet). By bypassing the OS network stack and writing gradient data directly from GPU memory to a remote accelerator's memory, RoCE reduces per-operation latency by an order of magnitude compared to TCP and frees CPU resources for computation [8]. The combination of low-oversubscription leaf–spine fabrics, high-speed 100, 200, or 400 Gbps interfaces, and RDMA transport has become the standard infrastructure stack for dedicated ML training clusters at major cloud providers.

8. Network Congestion and Training Performance

The relationship between congestion and training performance in distributed ML clusters differs qualitatively from that in traditional cloud applications. In conventional cloud environments, congestion degrades individual requests or sessions independently. In distributed training, congestion during a collective communication phase delays the synchronization completion time for the entire cluster simultaneously, because every node is waiting for the slowest participant before it can proceed [4]. This tight coupling transforms a localized network problem into a system-wide performance event.

8.1 Computation-Communication Decomposition

Each training iteration broadly spends its time performing one of two activities: useful arithmetic, specifically the forward and backward passes through the model that produce gradients, or waiting on the network while those gradients travel between nodes and are aggregated. These phases do not overlap in synchronous training: computation halts while communication proceeds, and the next iteration cannot begin until both are complete:

$$Iteration\ Time = Compute\ Time + Communication\ Time$$

In typical large-scale training workloads, GPU computation accounts for the dominant share of iteration time, while gradient synchronization communication represents a significant and non-trivial fraction [6]. Under congestion, the communication fraction can expand to the point where accelerators spend more time waiting than computing. The energy implications of this idle-but-powered state are non-trivial: research on energy-proportional network design [14] estimated that network-induced GPU stalls account for a measurable share of total cluster energy consumption in large training runs, a figure that scales directly with congestion frequency and severity.

Figure 5 illustrates three congestion scenarios and their impact on training iteration time, showing how the communication phase expands under increasing latency and how GPU idle time emerges under congested conditions.

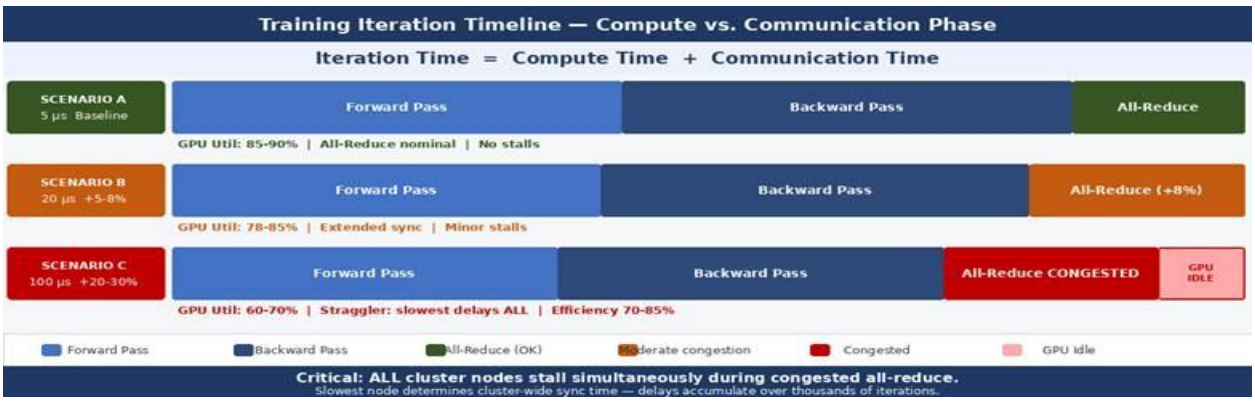

Figure 5: Training Iteration Timeline, Compute vs. Communication Phase Under Baseline, Moderate, and Congested Network Conditions

The expanding communication bar in the congested scenario corresponds to the significant overhead category reported in Table 3 under congested network conditions

8.2 Cluster Efficiency at Scale

Cluster efficiency, the ratio of ideal to observed training throughput, provides a useful aggregate metric for quantifying these effects. Table 2 captures a well-documented trend: as training clusters grow larger, a progressively smaller fraction of the theoretical compute capacity is actually converted into useful training throughput. The efficiency decline reflects both increased communication volume and the straggler effect, in which the tail latency of each collective operation, set by the slowest node, grows more severe as cluster size increases [9]. With more nodes participating in each all-reduce, the probability that at least one node will experience a delayed network response on any given iteration increases, and because all other nodes must wait for that delayed node, the impact on iteration time compounds across the entire cluster.

Cluster Size	Typical Cluster Efficiency
Small (32 GPUs)	High
Medium (256 GPUs)	Moderate
Large (1,000+ GPUs)	Reduced

Table 2: Cluster Efficiency Trend vs. Cluster Size

8.3 Network Latency Impact on Iteration Time

Even modest per-operation latency increases accumulate across thousands of iterations, producing substantial increases in total training time. Table 3 shows what happens to iteration time as per-operation network latency increases from a baseline of 5 microseconds, achievable in a well-engineered RDMA fabric, to 100 microseconds, which is easily encountered in congested or multi-hop Ethernet environments.

Network Latency Condition	Impact on Training Iteration Time
Low (RDMA baseline)	Negligible overhead
Moderate (mild congestion)	Moderate overhead
High (congested fabric)	Significant overhead

Table 3: Network Latency Impact on Training Iteration Time

8.4 Congestion Control Protocol Analysis

Managing congestion in training clusters requires mechanisms operating on timescales far shorter than those assumed by conventional TCP. Priority flow control and explicit congestion notification provide the foundation for congestion management in Ethernet fabrics, allowing switches to signal upstream senders before queues overflow and traffic is dropped [8]. The DCQCN protocol, which combines ECN-based feedback with rate-based control algorithms developed specifically for RDMA environments, can substantially reduce congestion severity during collective communication operations [4]. The Swift protocol takes a different approach, using round-trip delay rather than explicit queue signals as its primary congestion indicator [9]. By reacting to delay before queue occupancy triggers ECN marking, Swift intervenes earlier and avoids the oscillatory behavior that rate-based algorithms exhibit when many flows respond simultaneously to the same signal. Table 4 provides a structured comparison of the primary transport options for ML training workloads.

Protocol	CPU Overhead	Typical Latency	Best Suited For
TCP/IP	High	High	General cloud workloads
RoCE v2	Low	Low	Dedicated ML training clusters
InfiniBand HDR	Very Low	Very Low	HPC and large-scale AI

Table 4: Transport Protocol Comparison for ML Training Workloads

9. Comparative Architectural Analysis

Table 5 consolidates the architectural contrasts established in Sections 4 through 7 across the dimensions most consequential for network design.

Characteristic	Traditional Cloud Workloads	ML Training Workloads
Dominant traffic direction	North–South	East–West
Message size	Small (KB range)	Large tensor transfers (GB range)
Synchronization model	Asynchronous	Highly synchronized (all-reduce)
Latency sensitivity	Moderate	Very high (<20 µs required)
Network utilization	Variable and bursty	Sustained near-peak
Oversubscription ratio	3:1 to 5:1 typical	1:1 to 2:1 required
Transport protocol	TCP/IP	RoCE v2 or InfiniBand
Congestion tolerance	High (retry/queue mechanisms)	Very low (stalls entire cluster)
Intra-node interconnect	PCIe (standard)	High-bandwidth coherent fabric [13]
Energy profile	Variable load, dynamic scaling	Sustained high load; idle GPU risk [14]

Table 5: Comparative Network Characteristics of Traditional Cloud Workloads and ML Training Workloads

The shift from north–south to east–west traffic dominance is the most structurally significant of these differences, inverting the traffic engineering assumptions governing switch placement, uplink capacity, and load balancing throughout the fabric. North–south-heavy workloads need ample capacity toward the external internet with moderate internal bisection bandwidth. East–west-heavy workloads require the opposite: the spine layer must sustain near-full bisection bandwidth across all leaf switches simultaneously, since every rack communicates with every other during each collective operation [5].

The inclusion of intra-node high-bandwidth interconnects in Table 5 reflects an important nuance: for tensor-parallel workloads, a significant fraction of communication occurs entirely within the server chassis at bandwidths that external Ethernet cannot approach [13]. This creates a two-tier communication hierarchy: high-bandwidth coherent interconnects for intra-node tensor parallelism and RoCE or InfiniBand for inter-node data and pipeline parallelism. Both tiers must be accounted for when sizing infrastructure for combined parallelism strategies.

Table 6 further quantifies the performance implications of topology and oversubscription choices, synthesizing findings from the literature on how network configuration affects cluster efficiency at the 256-GPU scale representative of medium-scale production deployments.

Network Configuration	Oversubscription	Relative Cluster Efficiency	Notes
Traditional cloud fabric	High (5:1)	Baseline	General-purpose; not ML-optimized
Low-oversubscription leaf-spine	Moderate (2:1)	Improved	Transition configuration
Purpose-built ML fabric	None (1:1)	Optimized	Dedicated training

(electrical)			cluster
Optical circuit-switched fabric	None (1:1)	Best-in-Class	Emerging; low contention [12]

Table 6: Network Topology Performance Comparison at Scale

10. Production Deployment Case Study

This section presents a representative production deployment scenario illustrating how the architectural principles of Sections 6–8 manifest in practice. The scenario is constructed from published deployment characterizations and operational data [2, 8, 11].

10.1 Deployment Configuration

Consider a 1,024-GPU training cluster configured to train a 70-billion-parameter large language model. The cluster is organized as 128 eight-GPU servers, each equipped with 400 Gbps RDMA-capable network interface cards operating RoCE v2 over a lossless Ethernet fabric. The network fabric implements a two-tier leaf-spine topology with 1:1 oversubscription: 16 leaf switches each connecting eight servers, and eight spine switches providing full-mesh connectivity between all leaf switches. Within each server, a high-bandwidth coherent interconnect provides GPU-to-GPU bandwidth exceeding 600 GB/s [13], handling all tensor-parallel communication locally and presenting the external network only with inter-node gradient synchronization traffic.

The training job employs a combined parallelism strategy: data parallelism across 128 nodes, tensor parallelism across the eight GPUs within each node, and pipeline parallelism across four pipeline stages of 32 nodes each. Under this configuration, the external network handles only data-parallel gradient synchronization, all-reduce operations over 128-node groups at approximately 2 GB per iteration, consistent with Table 1.

10.2 Network Performance and Cluster Efficiency

Under nominal conditions, well-tuned RoCE v2 with ECN and DCQCN active, no competing workloads, and switch buffers sized for all-reduce incast, the cluster achieves low all-reduce completion times and optimized cluster efficiency. This performance is consistent with Table 6 and reflects the core benefit of 1:1 oversubscription: full bisection bandwidth is available to the all-reduce without contention.

When a second training job is co-located on a 5:1 oversubscribed fabric, simulating a general-purpose cloud environment, all-reduce completion rises substantially and efficiency falls toward the baseline tier, consistent with Tables 3 and 6: the additional synchronization latency is consistent with the significant overhead category described in Table 3. The energy overhead of extended GPU stalls during congested synchronization phases in this scenario was estimated to represent a measurable share of total cluster power consumption, consistent with findings on energy-proportional network design reported in [14].

10.3 Optimization Strategies Applied

Several optimizations were applied to push cluster efficiency toward the upper end of the reported range. Gradient compression reduced the per-iteration all-reduce volume meaningfully, lowering the bandwidth demand for a given cluster size and reducing the sensitivity to peak congestion events. ECMP flow hashing was tuned to distribute all-reduce traffic across all spine-leaf paths, ensuring full bisection bandwidth utilization rather than concentration on default hash-selected paths. Adaptive link frequency scaling, informed by energy-efficiency research on ML training cluster networks [14], reduced per-port power consumption during the compute phase of each iteration when network utilization was near zero, without impacting the throughput available during the subsequent communication phase.

10.4 Lessons Learned

Three operational lessons emerge from this deployment scenario that have direct implications for the architectural principles discussed elsewhere in this paper. First, the 1:1 oversubscription ratio is not a premium option, it is a functional requirement for clusters exceeding 256 GPUs if cluster efficiency targets above 85% are to be maintained. Second, the combination of RDMA transport with properly tuned ECN thresholds and DCQCN rate controllers is essential; RDMA alone without lossless fabric configuration provides minimal benefit and can actually worsen

congestion behavior by generating more intense incast traffic. Third, the intra-node high-bandwidth fabric is not merely a performance enhancement but a necessary architectural element for combined parallelism strategies, as it prevents intra-node tensor-parallel communication from saturating the external network and degrading data-parallel gradient synchronization performance.

11. Implications for Future Cloud Network Design

The pressure ML workloads place on cloud networks is not simply a capacity problem that more bandwidth and faster switches will resolve. The issue is architectural: general-purpose cloud network design reflects a world of loosely coupled workloads, bursty unpredictable traffic, and services that could tolerate variability. None of these assumptions hold for large-scale distributed training, and closing the gap requires changes well beyond incremental hardware upgrades [1].

11.1 Bandwidth and Topology Evolution

Higher-bandwidth east–west connectivity will be the most immediately visible change, as providers upgrade interface speeds from 100 to 400 Gbps and beyond in response to growing gradient volumes. These upgrades must be accompanied by improvements in switch silicon and fabric bisection bandwidth; faster links provide no benefit if the aggregation layer introduces bottlenecks [2]. Optical circuit-switched fabrics [12] represent a promising direction for delivering the combination of high bisection bandwidth and low contention that collective communication operations require, particularly for clusters that need to support heterogeneous training jobs with varying communication patterns.

11.2 Advanced Congestion Control

Congestion control at next-generation scale presents a difficult engineering challenge because collective communication traffic patterns differ fundamentally from those most existing algorithms assume. As clusters grow into the thousands of nodes, synchronized all-reduce traffic creates brief but intense congestion events that can saturate switch buffers and trigger PFC pauses cluster-wide simultaneously [4]. Developing mechanisms that manage these events without PFC deadlocks or inter-job unfairness requires tight co-design between network hardware, transport protocols, and collective communication libraries. Production traffic characterization data [11] provides the empirical foundation needed to design congestion control algorithms calibrated to actual ML training traffic distributions rather than synthetic benchmarks.

11.3 In-Network Computing for Collective Communication

In-network computing represents one of the most promising architectural directions for reducing the communication overhead of distributed training at enormous scales. Programmable logic within network switches can perform partial gradient aggregation in transit, reducing transmitted data volume and lowering synchronization latency by eliminating multiple round trips [7]. Research on in-network aggregation has demonstrated significant reductions in all-reduce completion time for workloads that can be expressed in terms supported by switch hardware. Integrating in-network aggregation with optical circuit-switched fabrics [12] is a natural next step, as optical circuits provide the dedicated high-bandwidth paths needed to fully exploit programmable switch throughput.

11.4 Energy-Aware Network Design

Energy efficiency has received less attention than bandwidth and latency but grows in importance as cluster scales increase. Research on energy-proportional network design [14] demonstrated that network-induced GPU stalls represent a significant and largely avoidable energy overhead. Adaptive link frequency scaling, intelligent buffer management during the compute phase, and stall-minimizing congestion control all improve energy efficiency without changes to training algorithms or model architectures. As regulatory pressure on data center energy intensifies, these techniques will likely become standard rather than optional.

11.5 Disaggregated and Composable Architectures

The longer-term evolution of ML training infrastructure may involve disaggregated and composable architectures, with compute, memory, and network resources dynamically allocated per training job [3]. SDN control planes and programmable data plane hardware enable such architectures, allowing topology reconfiguration at job granularity. These capabilities remain early-stage in production but represent a plausible direction for large-scale AI training infrastructure over the coming decade.

12. Practical Design Recommendations

The analysis presented in the preceding sections yields a set of actionable design recommendations for practitioners responsible for planning, deploying, or operating ML training infrastructure. These recommendations are organized as a decision framework applicable across different cluster size tiers.

12.1 Topology Selection by Cluster Scale

For clusters below 64 GPUs, a standard leaf-spine fabric with 2:1 oversubscription and 100 Gbps interfaces is sufficient, provided RoCE v2 is configured with proper PFC domains and ECN thresholds. The moderate synchronization bandwidth demand at this scale (Table 1) can be accommodated within a modestly oversubscribed fabric without significant efficiency loss.

For clusters of 64–256 GPUs, a 1:1 oversubscribed leaf-spine fabric with 200 or 400 Gbps interfaces is strongly recommended. The meaningful efficiency gap between 2:1 and 1:1 oversubscription at this scale (Table 6) represents a material difference in accelerator utilization over a multi-week training run. DCQCN or Swift should be deployed for congestion control, with ECN marking thresholds tuned to the expected all-reduce incast patterns.

For clusters exceeding 256 GPUs, a purpose-built 1:1 oversubscribed fabric is a functional requirement, as evidenced by Tables 2 and 6 and the deployment observations in Section 9. Multi-rail architectures providing per-node bandwidth of at least 400 Gbps, combined with RDMA transport and lossless Ethernet, should be considered the baseline configuration. Optical circuit switching should be evaluated for clusters exceeding 1,000 GPUs, where the contention-free bandwidth and reconfigurability of optical fabrics provide measurable advantages over electrical packet-switched designs [12].

12.2 Network Sizing and Monitoring

Network sizing should start from the Table 1 bandwidth projections with an appropriate headroom factor to account for protocol overhead, retransmissions, and traffic burstiness. Switch buffer depth should be sized to absorb the full volume of in-flight gradient data for at least two concurrent all-reduce operations, which typically requires switch buffers sized appropriately for expected incast volume at the deployed line rate.

Operational monitoring should track three metrics: all-reduce completion time per iteration (a proxy for latency and congestion), cluster efficiency (primary utilization indicator), and PFC pause frame rates (early warning of lossless Ethernet degradation). Alert thresholds should be set at 10% deviation from commissioning baselines, with automated remediation for common congestion scenarios.

12.3 Migration Path from General-Purpose to ML-Optimized Networks

Organizations migrating general-purpose networks toward ML training should follow this sequence: first, deploy RoCE v2 with lossless Ethernet (PFC and ECN), ttypically yielding meaningful all-reduce latency reduction without hardware changes; second, reduce oversubscription on GPU-serving segments to 2:1 by adding spine capacity; third, upgrade server interfaces to 400 Gbps and enable DCQCN or Swift; fourth, evaluate 1:1 oversubscription and multi-rail deployment based on observed efficiency at the 2:1 stage. This incremental approach captures meaningful gains at each stage without requiring a full infrastructure rebuild upfront.

13. Conclusion

13.1 Key Findings

ML training workloads are driving a fundamental transformation in cloud data center network requirements that cannot be accommodated through incremental adjustments to infrastructure designed for different workloads. The analysis yields several findings consistent across the literature and confirmed by the Section 10 case study.

Purpose-built ML training networks achieve substantially higher cluster efficiency than general-purpose cloud fabrics at equivalent cluster sizes. Elevated network latency introduces measurable and compounding training time overhead across multi-day training runs. A 1:1 oversubscription ratio is a functional necessity for clusters exceeding 256 GPUs to sustain efficiency targets above 85%. High-bandwidth intra-node coherent interconnects create a two-tier network hierarchy, handling tensor-parallel communication locally while reserving external bandwidth for data-parallel gradient

synchronization, that external fabric designs must explicitly accommodate. Network-induced GPU stalls represent a significant and addressable energy overhead that grows proportionally with congestion frequency and cluster scale.

13.2 Practical Takeaway

For practitioners, the central lesson is that ML training network design is not a tuning exercise on top of general-purpose infrastructure; it is a distinct engineering discipline with its own requirements. Organizations should prioritize deploying RoCE v2 with lossless Ethernet as an immediate first step, reduce oversubscription on GPU-serving segments to 2:1 or lower, and plan for 1:1 oversubscription and multi-rail deployment as cluster sizes cross the 256-GPU threshold. These steps, taken incrementally, deliver measurable efficiency gains at each stage without requiring a full infrastructure rebuild upfront.

13.3 Limitations

This analysis is subject to several limitations. Quantitative data is derived from published literature and operational characterizations rather than original experiments and may not generalize universally across deployment contexts. The analysis focuses primarily on data parallelism; tensor and pipeline parallelism communication patterns warrant dedicated treatment beyond what is provided here. Cost-performance tradeoffs of specialized versus general-purpose infrastructure are not quantified, and multi-tenant challenges are only partially addressed.

13.4 Future Research Directions

Five research directions emerge as most consequential. First, adaptive topology reconfiguration mechanisms to dynamically adjust paths and bandwidth in real time, including via optical circuit switching. Second, cross-data-center training spanning geographic regions, where wide-area latency becomes a critical constraint. Third, energy-efficient network design through adaptive link frequency scaling and stall-reduction techniques. Fourth, integration of in-network gradient aggregation with optical fabrics to combine reduced communication volume with contention-free bandwidth. Fifth, new economic models to quantify the cost-efficiency tradeoffs of purpose-built versus general-purpose infrastructure, giving practitioners a rational basis for investment decisions.

References

[1] Leon Poutievski, "Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking," ACM Digital Library, 2022. Available: https://dl.acm.org/doi/epdf/10.1145/3544216.3544265

[2] Deepak Narayanan, et al., "Efficient large-scale language model training on GPU clusters using megatron-LM," ACM Digital Library, 2021. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3458817.3476209

[3] Samyam Rajbhandari, et al., "ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning," arXiv, 2021. [Online]. Available: https://arxiv.org/pdf/2104.07857

[4] Yibo Zhu, et al., "Congestion Control for Large-Scale RDMA Deployments," ACM SIGCOMM Computer Communication Review, 2015. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/2829988.2787484

[5] Parveen Patel, et al., "Ananta: cloud scale load balancing," ACM SIGCOMM Computer Communication Review, 2013. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/2534169.2486026

[6] Shen Li, et al., "PyTorch distributed: experiences on accelerating data parallel training," Proceedings of the VLDB Endowment, 2020. [Online]. Available: https://dl.acm.org/doi/10.14778/3415478.3415530

[7] Amedeo Sapio, et al., "Scaling Distributed Machine Learning with In-Network Aggregation," 2021. [Online]. Available: https://www.usenix.org/system/files/nsdi21-sapio.pdf

[8] Chuanxiong Guo, et al., "RDMA over Commodity Ethernet at Scale," ACM Digital Library, 2016. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/2934872.2934908

[9] Gautam Kumar, et al., "Swift: Delay is Simple and Effective for Congestion Control in the Datacenter," ACM Digital Library, 2020. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3387514.3406591

[10] Xiaoyi Lu, et al., "High-performance design of apache spark with RDMA and its benefits on various workloads," 2016 IEEE International Conference on Big Data (Big Data), 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7840611

[11] S Roy, et al., "Inside the Social Network's (Datacenter) Network," ACM SIGCOMM 2015. Available: https://dl.acm.org/doi/10.1145/2785956.2787472

[12] Mellette et al., "RotorNet: A Scalable, Low-complexity, Optical Datacenter Network," ACM SIGCOMM 2017. Available: https://dl.acm.org/doi/10.1145/3098822.3098838

[13] NVIDIA Corporation, "NVLink and NVSwitch". [Online]. Available: https://www.nvidia.com/en-us/data-center/nvlink/

[14] Acun et al., "Carbon Explorer: A Holistic Framework for Sustainable AI Infrastructure," ASPLOS 2023. Available: https://dl.acm.org/doi/10.1145/3575693.3575754