Optimized Network Architectures for Training Large Language Models With Billions of Parameters

Weiyang Wang* Manya Ghobadi* Kayvon Shakeri† Ying Zhang† Naader Hasani†

*MIT †Meta

ABSTRACT

This paper challenges the well-established paradigm for building any-to-any networks for training Large Language Models (LLMs). We show that LLMs exhibit a unique communication pattern where only small groups of GPUs require high-bandwidth any-to-any communication within them, to achieve near-optimal training performance. Across these groups of GPUs, the communication is insignificant, sparse, and homogeneous. We propose a new network architecture that closely resembles the communication requirement of LLMs. Our architecture partitions the cluster into sets of GPUs interconnected with non-blocking any-to-any high-bandwidth interconnects that we call HB domains. Across the HB domains, the network only connects GPUs with communication demands. We call this network a "rail-only" connection, and show that our proposed architecture reduces the network cost by up to 75% compared to the state-of-the-art any-to-any Clos networks without compromising the performance of LLM training.

1 Introduction

The evolving field of Large Language Models (LLMs) holds great promise in revolutionizing our understanding of human language processing, driving technological advancement of artificial intelligence (AI). OpenAI's ChatGPT served over 100 million active users within three months of its release, making it the fastest-growing application ever [1]. Beyond chatbots, LLMs are progressively infiltrating our digital lives. Key service providers integrate these powerful models into co-pilot programs and search engines [2, 3, 4], transforming how humans interact with the digital sphere.

LLMs are among the largest and most computationally-intensive Deep Neural Networks (DNNs). The latest GPT4 model is estimated to have trillions of parameters and take months to train [5, 6]. Historically, researchers seek to enhance the performance of distributed DNN training and inference through optimizing parallelization strategies [7, 8, 9, 10], sophisticated scheduling [11, 12, 13], advanced compression [14], and even the reconfiguration of the network topology itself [15, 16, 17]. Despite these efforts, LLMs still require significant raw computing power. The GPT3 model from 2020 already requires 355 GPU-years on Nvidia's V100 GPUs [18, 19]. As Moore's law slows down, the growth rate of LLM size and computation requirement exceeds the advancement of accelerators, making hyper-scale GPU clusters inevitable. Our conversations with lead machine learning architects in the industry indicate that the next-generation LLMs likely require over 30,000 GPUs of computing power to finish training within a reasonable time.

A GPU-centric cluster typically employs two types of connection [20]. For the first one, a few GPUs (e.g., eight for a DGX H100 server) reside within a high-bandwidth domain, called HB domain, through a short-range communication protocol like NVLink [21]. The second connection forms a network capable of any-to-any GPU communication using RDMA-capable NICs, connected in a variant of the Clos network. The cluster uses the RDMA protocol on this network to benefit from bypassing CPU and OS entirely through GPU-Direct [22, 23]. However, scaling up RDMA networks to tens of thousands of GPUs is challenging. Previous work demonstrated that large-scale RDMA network are prone to deadlocking and PFC storms [24, 25, 26, 27, 28], degrading the performance. Furthermore, as the scale goes up, Clos architectures become prohibitively costly [16]. Datacenter providers often resort to over-subscription to tame the cost of the cluster, worsening the deadlocking problems.

Prior work proposed several techniques to enable large-scale RDMA networks and reduce their cost [29, 25, 30, 31]. These approaches fundamentally depend on the assumption that the network is capable of any-to-any communication. This assumption forms the bedrock upon which datacenters have been conceptualized and developed for several decades.

In this paper, we challenge this assumption and show that LLM training traffic *does not* require any-to-any connectivity across all GPUs in the network. We argue that with the optimal parallelization strategy, an LLM training workload requires high-bandwidth any-to-any connectivity only within small subsets of GPUs, and each subset fits within an HB domain. Across the HB domains, communication only happens for a few GPU pairs, and the traffic volume is insignificant. As a result, the conventional any-to-any approach for building datacenter interconnect adds unnecessary complexity and cost for distributed LLM training.

We propose a network architecture that accurately reflects LLM communication requirements. In this architecture, a

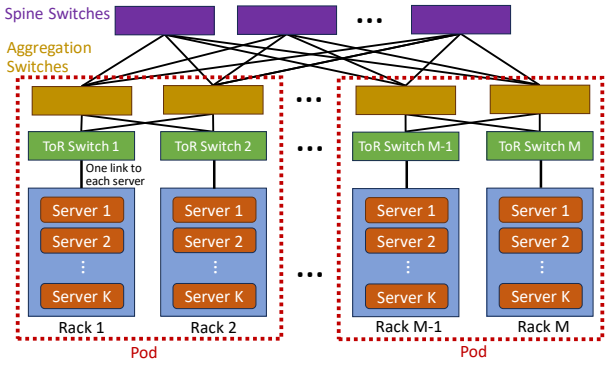


Figure 1: CPU cluster with a Clos network [32, 33].

cluster is first partitioned into multiple HB domains, and each interconnected with a full-bisection bandwidth any-to-any interconnect. Across the HB domains, instead of forming a Clos to support any-to-any communication, the network only connects sets of GPUs with network traffic. We demonstrate that our LLM-centric network architecture archives *the same performance* as a full-bisection bandwidth any-to-any Clos cluster while reducing the cost by 37% to 75%.

2 Motivation

In this section, we first introduce the architecture of a conventional GPU-centric cluster. Then we perform a thorough analysis of LLM traffic patterns to motivate a network architecture contrasting the conventional design.

2.1 State-of-the-Art GPU cluster Design

Conventional networked clusters are designed to serve CPU-heavy workloads using a multi-layer Clos network, illustrated in Figure 1 [32, 33]. This architecture, known as a Fat-Tree network, is deeply studied in the system and networking communities. In a typical Fat-Tree-based cluster, each server is equipped with one NIC (40 Gbps to 400 Gbps), and K servers are arranged into racks connecting to Top-of-the-Rack (ToR) switches. The ToR switches are then connected to the aggregation switches to provide connectivity across racks, forming a pod. Finally, the pods are interconnected with spine switches, allowing any-to-any communication across servers in a CPU cluster.

In contrast, the rise of network-heavy ML workloads led to the dominance of GPU-centric clusters, where *individual* GPUs have dedicated NICs [34]. Figure 2 illustrates the network architecture of a typical GPU cluster. Each GPU has two different communication interfaces: (i) An NVLink interface to support high-bandwidth but short-range interconnection and (ii) a conventional RDMA-enabled NIC. The NVLinks connect K GPUs to provide terabits of non-blocking any-to-any bandwidth in/out per GPU (7.68 Tbps for fourth-gen NVLink, for instance). This group of GPUs with fast interconnect forms a *high-bandwidth domain (HB domain)*. Traditionally, HB domains were restricted to a single server (e.g., DGX servers with $K = 8$ or 16 GPUs). However,

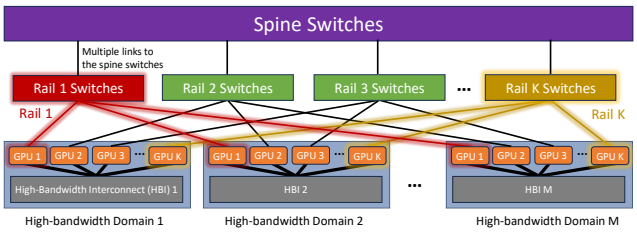


Figure 2: State-of-the-art GPU clusters are based on rail-optimized, any-to-any networks [20].

recently, Nvidia announced the GH200 supercomputer interconnecting $K = 256$ Grace Hopper Superchips to form one HB domain across multiple racks [35].

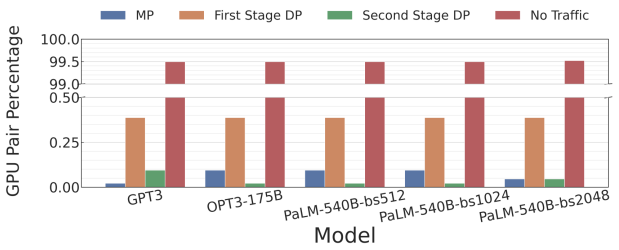
However, some LLMs take too long for a single HB domain to train, even with 256 GPUs. For instance, the PaLM-540B model would take 117 days to finish on a GH200 supercomputer, assuming perfect GPU utilization. These models require parallelization across multiple HB domains.

To enable training an LLM across multiple HB domains, GPU cluster operators use RDMA-capable NICs to interconnect multiple HB domains together. The conventional network architecture to interconnect HB domains is called a *rail-optimized* network [20]. In a rail-optimized architecture, GPUs within an HB domain are labeled from 1 to K. A *rail* is the set of GPUs with the same index (or rank) on different HB domains, interconnected with a rail switch. For instance, Figure 2 illustrates Rail 1 and Rail K in red and yellow color, respectively. These rail switches are subsequently connected to spine switches to form a full-bisection any-to-any Clos network topology. This network ensures any pair of GPUs in different HB domains can communicate at the network line rate (400 Gbps Infiniband network for GH200). For instance, traffic between `GPU 1, Domain 1` and `GPU 1, Domain 2` traverses through Rail Switch 1 only, while traffic between `GPU 1, Domain 1` and `GPU 2, Domain 2` goes through the respective rails and the spine switches.

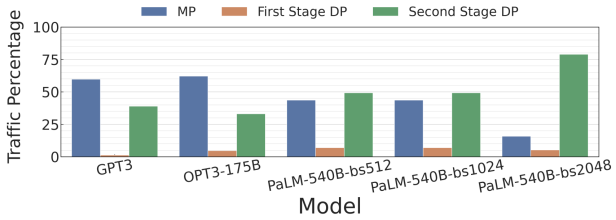
2.2 Analyzing Network Traffic of LLMs

We now analyze the traffic pattern and iteration time of OpenAI's GPT3 [18], Meta's OPT3-175B [36], and Google's PaLM-540B [37] distributed in a cluster composed of hundreds of Nvidia GH200 supercomputers [35]. These LLMs represent cutting-edge language models with publicly available parameters. Each GH200 supercomputer comprises a two-tier NVSwitch architecture, facilitating 2 Pbps of full-bisection bandwidth (7.68 Tbps per GPU) across 256 H100 GPUs. Additionally, each GPU has a Connect-X7 HCA Infiniband network interface [35], which provides 400 Gbps network bandwidth in/out of each GPU. In this setup, each GH200 supercomputer forms an HB domain.

Our analysis uses the benchmarking parallelization strategy from Nvidia [38, 39] to ensure optimal GPU utilization. We use hybrid data parallelism (DP) and intra-operator model parallelism (MP), but our conclusions remain similar for pipeline parallelism. The DP synchronization utilizes a

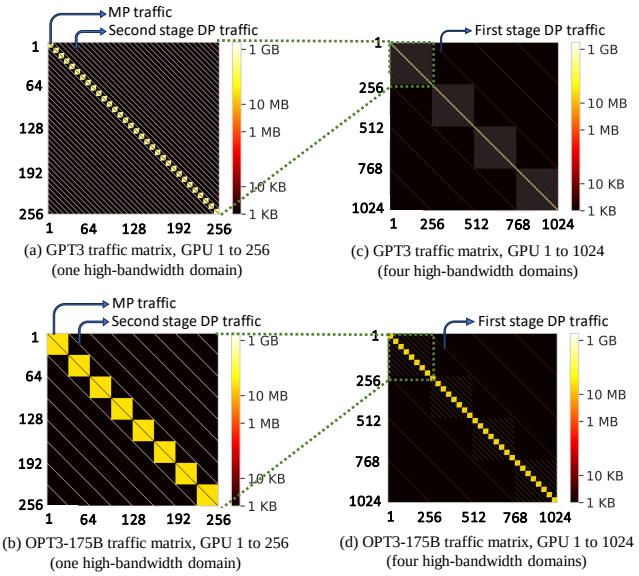


(a) Traffic type distribution for all pairs of GPUs



(b) Traffic size distribution for all pairs of GPUs

Figure 3: (a) The communication type across all GPU pairs. (b) The magnitude of traffic for all GPU pairs with non-zero communication.



(a) GPT3 traffic matrix, GPU 1 to 256 (one high-bandwidth domain)

(c) GPT3 traffic matrix, GPU 1 to 1024 (four high-bandwidth domains)

(b) OPT3-175B traffic matrix, GPU 1 to 256 (one high-bandwidth domain)

(d) OPT3-175B traffic matrix, GPU 1 to 1024 (four high-bandwidth domains)

Figure 4: Traffic heatmaps for GPT3 and OPT3-175B.

hierarchical AllReduce algorithm to limit traffic across HB domains. We employ the best parallelization strategy for each LLM model and compute the resulting traffic pattern. Note that for the PaLM model, the batch size varies throughout the training.

Figure 3a illustrates the traffic type distribution of one training iteration across server pairs for a cluster of 128 GH200 supercomputers, and Figure 3b shows the percentage of volume for each type of traffic. There are two primary types of communication: intra-operator model parallelism (MP) traffic and AllReduce traffic generated by data parallelism. The MP traffic happens within GPUs that participate in a *model parallel group*, which always fits in an HB domain. For data parallelism, the hierarchical AllReduce algorithm further partitions the communication into two stages, where the first stage synchronizes parameters across HB domains while the second stage is within them. The algorithm ensures the second stage carries more traffic to utilize the available bandwidth better. While these types of traffic do not overlap between different pairs of GPUs, Figure 3a indicates that over 99% of GPU pairs carry *no traffic* and less than 0.25% of GPU pairs carry MP and second stage DP traffic between them. Simultaneously, Figure 3b suggests these traffic types account for over 90% of the total transmitted data. Recall these two types of traffic stay within HB domains, suggesting efficient usage of HB domain bandwidth and low demand on the network fabric interconnecting HB domains. This pattern is consistent across all models, indicating that building a cluster with any-to-any connectivity on top of HB domains for LLM models is excessive.

Within HB domains, the interconnect needs to support heavy any-to-any communication for training a diverse set of LLMs. To illustrate this, Figure 4 shows heatmaps of the traffic matrices during a training iteration for GPT3 and OPT3-175B. In this plot, every consecutive set of 256 GPUs resides within the same HB domain. Figures 4a and 4b demonstrate the communication pattern within an HB domain. Note that the traffic volume is significant (up to 1 GB across server pairs), and the pattern varies due to different parallelization strategies across these models. The squares on the diagonal represent the MP traffic, while the rest of the off-diagonal lines represent the second stage DP traffic. An any-to-any interconnect within an HB domain to provides the maximum flexibility to accommodate different LLMs.

However, the high-bandwidth any-to-any connectivity required within HB domains is not needed across them. Figures 4c and 4d zoom out to the first four HB domains, where the first stage DP traffic across HB domains occurs. Compared to the traffic within an HB domain, the cross-HB traffic is much smaller ($\approx$1 MB per entry) and more sparse. This is because cross-HB domain communication only occurs between GPUs with the same rank (i.e., GPUs on the same rail). Furthermore, the pattern is homogeneous, as the cross-HB domain communication pattern remains the same across LLMs (the off-diagonal lines in Figures 4c and d).

These observations suggest that it is possible to remove links that do not carry any network traffic without hurting the training performance of LLMs. Our analysis shows that 33% of the links in an any-to-any 400 Gbps Clos network are removable. Figure 5 illustrates the training iteration times for this alternative network, labeled as *Any-to-Any Trimmed 400 Gbps*, compared to the state-of-the-art. Given that the disconnected links do not support any traffic, these two topologies deliver identical performances.

To put this into perspective, we also consider an ideal performance by assuming all GPUs in the cluster are interconnected with NVSwitch, forming a monolithic 7.68 Tbps full-bisection bandwidth any-to-any network, which represents a 19.2$\times$ increase in the full-bisection bandwidth compared to

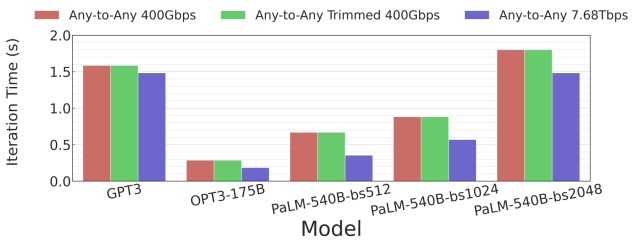


Figure 5: Computed iteration time.

the state-of-the-art. Such a network is impossible to build in practice. The performance improvement, however, is not proportional to the added bandwidth, ranging from $1.06\times$ to $1.88\times$ for all models, as shown in Figure 5. Such a result provides compelling arguments for LLM cluster operators to reassess the conventional any-to-any network design.

3 Our Proposed LLM Cluster

In this section, we propose a new network design specifically for LLM clusters. We parameterize our design with a mathematical model and put forward a comprehensive set of guidelines to determine parameters for such clusters.

3.1 Rail-only Network Design

We propose a network architecture that diverts from the any-to-any paradigm across all GPUs. Figure 6 illustrates our network architecture, which we name *rail-only*. Compared to a conventional rail-optimized GPU cluster, shown in Figure 2, our network keeps the HB domains and provides connectivity *only across the same rail*.

A straightforward way to realize our proposed architecture is to remove the spine switches from Figure 2 and repurpose all the uplinks connecting rail switches to the spine as down-links to GPUs. Hence, each rail is connected by a dedicated but separate Clos network. In the rest of this paper, we base our analysis on this realization of the rail-only network, though other technologies are also suitable for other rail interconnections (§5).

Our rail-only network architecture removes network connectivity across GPUs with different ranks in different rails. However, such communication is still possible by forwarding the data through HB domains. For instance, a message from `GPU 1, Domain 1` to `GPU 2, Domain 2` can be first routed through the first HB domain to `GPU 2, Domain 1` and then be transmitted to the final destination through the network. Although our analysis shows that LLM traffic does not require such forwarding, this connectivity might be needed for control messages, measurement, or training other DNN models in this cluster. We provide more discussions on handling other DNN models in Section 5.

3.2 Rail-only Network Analysis

Table 1 describes the parameters used in our analysis. We consider a network with N GPUs and an HB domain of size K. The bandwidth for each HB domain is B_F and the network bandwidth is B_S. We focus on the analysis of using hybrid two-stage data and model parallelism as the parallelization strategy and derive a mathematical model for the training iteration time and optimal model parallel group size.

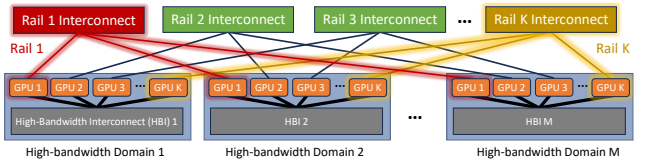


Figure 6: Our proposal: replace the any-to-any connectivity with a *rail-only* connection.

Name	Description
N	Total number of GPUs in the cluster, $N = N_M \cdot N_{Df} \cdot N_{Ds}$
N_M	Number of GPUs participating in a model parallel group
N_{Df}	Number of GPUs participating in the first level data parallel group
N_{Ds}	Number of GPUs participating in the second level data parallel group
K	HB domain size, $K = N_M \cdot N_{Ds}$
d_{model}	LLM Embedding dimension
M_{flop}	Amount of flops required for an iteration of training
M_{cc}	Rounds of collective communication needed for MP. This number is 8 for GPT and OPT (one multi-head attention layer per transformer block) and 12 for PaLM (two multi-head attention layers per transformer block)
l	Number of transformer block layers
B_F	HB domain bandwidth
B_S	GPU Network bandwidth
C	GPU Compute Speed (flops)
S_T	Size of a transformer block
b	Batch size
T_c	Compute time of a transformer block
T_{Df}	Communication time for first level DP
T_{Ds}	Communication time for second level DP
T_M	Communication time for MP

Table 1: Variables used in our analysis.

The aggregate time required for a single iteration is the sum of computation and communication times, formulated as $T_{iter} = T_c + T_{Df} + T_{Ds} + T_M$. We model the computational duration of the model as the ratio of the required amount of floating point operations to the aggregate FLOPs of the cluster. As for the collective communication durations, we calculate the bandwidth time only since the latency term is constant for our choice of collective communication algorithm and small relative to the bandwidth term. For all three types of communication, we base our calculations on bandwidth and latency optimal collective communication algorithms implemented with all-to-all communication within connected sets of GPUs. The total iteration time is:

$$T_{iter} = \frac{M_{flop}}{NC} + l\left(\frac{2S_T(N_{Df}-1)}{NB_S} + \frac{1}{B_F}\left(\frac{2S_T(N_{D_s}-1)}{K} + \frac{M_{cc}bd_{model}(N_M-1)}{N}\right)\right) \quad (1)$$

Leveraging this formula, we can calculate the optimal N_M:

$$N_M^* = \frac{\sqrt{NS_T}}{2\sqrt{bM_{cc}d_{model}}} \quad (2)$$

Intriguingly, N_M^* is independent of K, B_F, and B_S, indicating no dependency on the size of the HB domain or the exact bandwidth of any interconnect.

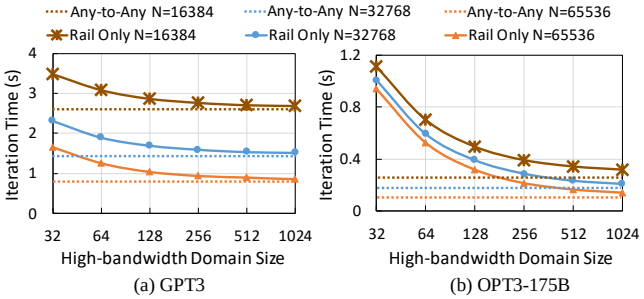


Figure 7: Iteration time as HB domain size changes.

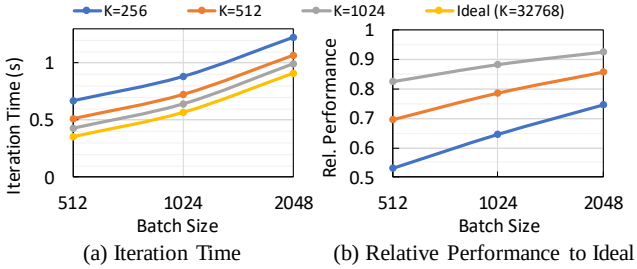


Figure 8: Iteration time and relative performance to the ideal case, as batch size changes, for PaLM-540B.

3.3 What Is the Ideal Size of an HB Domain?

The question that naturally follows the design process is, *what should be the ideal size of the HB domain?* In Figure 7, we vary the HB domain size (K) and plot the training iteration time for GPT3 and OPT3-175B for GPU clusters of sizes 16384, 32768, and 65536 GPUs. For each cluster size, we use the respective optimal N_M^* calculated from the bandwidth and computational ability parameters of GH200. We also compute the training iteration time of "Any-to-Any 7.68 Tbps" as the ideal-case performance. Recall that this design point represents the idealized scenario where *every GPU* is connected with a full-bisection NVLink fabric, or equivalently the case where $K = N$, and is unattainable in practice. The ideal case uses a non-hierarchical AllReduce algorithm to harness the uniform high-bandwidth interconnect, and its iteration time is:

$$T_{iter}^{ideal} = \frac{M_{flop}}{NC} + \frac{l}{B_F N}(2S_T(\frac{N}{N_M^*} - 1) + M_{cc} bd_{model}(N_M^* - 1)) \quad (3)$$

As depicted in Figure 7, the performance gain decreases as the HB domain size goes up. A transition of HB domain size from 32 to 64 accounts for a 12% and 37% reduction in training iteration times for GPT3 and OPT3-175B, respectively, whereas an increment from 512 to 1024 merely realizes a gain of 0.95% and 7.5%. This reduction in communication time gain can be attributed to Amdahl's law, as the computation time of the DNN remains constant across all instances. We argue that the current GH200 supercomputer, with an HB domain of size 256, is well-suited to the demands of LLM training today, provided an appropriate batch size is chosen. We defer the analysis of batch size in Section 3.4. At the same time, prospective technological advancements augmenting this size will further benefit the training performance, reducing the training iteration time closer to that of the ideal case without requiring any-to-any connectivity in the network across HB domains.

3.4 Impact of Batch Size on Network Design

Equation 2 indicates that with an increase in batch size b, the optimal model parallel group size shrinks. In this case, more data parallel communications can fit within the HB domain and thus benefit the overall performance. A comparative analysis between Figure 7a and 7b reveals this benefit since the GPT3 and OPT3-175B models have practically identical model structures, and the discrepancies in training iteration time solely come from the choice of batch size ($32M$ tokens for GPT3 versus $2M$ tokens for OPT3-175B).

To further understand the impact of batch size on training time, we analyze the performance of a PaLM-540B model on a 32768 GPU cluster while changing the HB domain size from $K = 256$ to 1024. During training, the PaLM model automatically changes its batch size from 512 to 2048 sequences ($1M$ to $4M$ tokens). Figure 8a plots the change in iteration time as the batch size varies. The iteration time exhibits the same trajectory for all HB domain sizes; however, due to Amdahl's law, the *relative performance* (the ratio to the iteration time for an HB domain size to that of the ideal case) improves as the batch size increases. Figure 8b represents this trend. When $K = 256$, the relative performance increases from 53% to 74% as the batch size goes up from 512 to 2048 sequences.

Prior studies have shown that LLM training benefits from a larger batch size [40, 18], making it a perfect fit for our rail-only design. Additionally, while the batch size parameter is typically an ML-centric metric for optimization, our analysis indicates that the impact of batch size on the comprehensive training performance goes beyond the total number of iterations required for convergence.

4 Network Cost Analysis

Our rail-only network architecture judiciously reduces the network resources for LLM training by eliminating unused network connections. This section compares the network cost of our proposed approach with the state-of-the-art rail-optimized GPU clusters. We calculate the number of switches (#SW) and transceivers (#TR) required for each network design and derive the total network equipment cost based on numbers reported in prior work [16][1]. This section focuses on using only electrical packet switches to construct the network; however, using optical direct connect technology can provide further cost reductions [41]. We defer the discussion about direct-connect networks to Section 5.

We enumerate the number of switches and transceivers required to build both the state-of-the-art network architecture

[1] $374 per transceiver, $748 per switch port for 400 Gbps.

# of GPUs (N)	Switch Radix	SOTA #SW	Rail-only #SW	SOTA #TR	Rail-only #TR	Cost Reduction
32768	32	7168	3072	262144	131072	54%
	64	2560	1536	196608	131072	37%
	128	1280	256	196608	65536	75%
	256	384	128	131072	65536	60%
65536	64	5120	3072	393216	262144	37%
	128	2560	1536	393216	262144	37%
	256	1280	256	393216	131072	75%

Table 2: Number of switches for different clusters.

and our proposed architecture in Table 2, accounting for variable cluster sizes and network switch radix. Note that for the state-of-the-art architecture, to use the least amount of network resources, each rail of GPUs is not physically separated in some cases. Thus, the datacenter operator must manually configure the switch to achieve the desired isolation across rails to achieve the rail-optimized design.

The last column of Table 2 illustrates our design's cost savings over that of the state-of-the-art for the total cost of switches and transceivers. Our rail-only design notably reduces the network cost by 37% to 75% compared to the state-of-the-art design while achieving equivalent performance. This reduction stems from eliminating core layer switches and decreasing the number of switch tiers within each rail. Surprisingly, switches with a radix of 64 provide the worst-case cost reduction in both cluster sizes. In this case, the state-of-the-art design requires a three-tier Clos network, while the rail-only design requires two tiers for each rail. Still, our design only requires three-quarters of the total number of switches while simultaneously achieving the same performance as the state-of-the-art design.

5 Discussion

LLM trend. The current growth rate of LLM computational requirement outpaces the advancements in AI accelerators and network speed as Moore's law slows down, necessitating hyper-scale clusters and more efficient interconnects [42, 43]. Our position to remove any-to-any network connectivity is the first step towards accommodating the network requirement for LLM training and sustaining the LLM growth trend. We also acknowledge ongoing efforts to reduce language models' size and resource requirements without compromising performance [44]. These works complement ours as our design reduces network resources and maintains performance even for smaller language models and clusters.

LLM Inference. This paper explores the training workload of LLMs, yet inference represents another significant part of the LLM product cycle. Inference demands fewer computational resources than training as it involves moving fewer data through the LLM and only computes the forward pass [45]. As such, each HB domain naturally becomes an inference-serving domain, and the rail-only connections help load-balance multiple inference domains. We leave a detailed investigation of LLM inference to future work.

Direct-connect network topology. As mentioned in Section 4, datacenter operators may leverage direct-connect network topologies for interconnection across the rails [41, 16, 17]. To maximize the effectiveness of such designs, we propose increasing the number of network interfaces connecting to each GPU through NIC interface splitting [16]. Additionally, we suggest using reconfigurable optical switches to provide greater flexibility for interconnections across HB domains. Such a design also allows reconfiguring some connections across rails for forthcoming workloads that behave differently from LLMs. We believe that combining our proposed design and optical reconfigurable network switches opens a new line of research in AI-ML clusters.

Other ML workloads and limitations. Although our proposed rail-only architecture focuses on network design specifically for LLMs, our design is efficient for many other DNN workloads when combined with other efforts. Recent works attempt to make the parallelization strategy and collective communication algorithms bandwidth-aware for any DNN model [8, 46], which already produce traffic patterns resembling that of LLMs. For parallelization strategies requiring a small amount of traffic for GPUs across the rails, the cluster can use the forwarding described in Section 3. Our design's primary challenge is the *all-to-all* communication across all GPUs, which commonly arises in recommendation models with large embedding tables [47, 34]. The forwarding scheme induces congestion and degrades the performance of all-to-all traffic. We acknowledge that all-to-all traffic is one of the most challenging traffic patterns that arise in ML workloads. Some potential solutions include reintroducing small any-to-any capacity through an over-subscribed network, utilizing a fast-reconfigurable network fabric, and decreasing the amount of all-to-all traffic initially generated by tweaking the ML model itself.

Fault tolerance. At first glance, the rail-only design might appear less fault tolerant than a standard Clos network. However, suppose a rail switch fails in either network. All the GPUs connected to the failed switch will become unavailable, rendering the two topologies identical regarding fault tolerance on rail switches. Conversely, our design requires fewer switches, which naturally reduces the points of failure. Datacenter operators can add redundant capacity by including extra rail switches, and our design remains more cost-effective compared to the state-of-the-art any-to-any network design. Fault tolerance can also be increased with a direct-connect network, as even if the control plane fails, an optical switch is likely to remain functional.

6 Conclusion

This paper challenges the conventional any-to-any network architecture for GPU clusters dedicated to training large language models. We propose a new architecture, called rail-only, that aligns with the distinct characteristics and demands of LLMs, leading to up to 75% cost reductions while maintaining identical performance to the current state-of-the-art Clos networks.

7 References

[1] Ubs: Chatgpt may be the fastest growing app of all time, 2023. URL https://aibusiness.com/nlp/ubs-chatgpt-is-the-fastest-growing-app-of-all-time.

[2] What's ahead for bard: More global, more visual, more integrated, 2023. URL https://blog.google/technology/ai/google-bard-updates-io-2023.

[3] Your ai pair programmer: Github copilot uses the openai codex to suggest code and entire functions in real-time, right from your editor., 2023. URL https://github.com/features/copilot.

[4] Confirmed: the new bing runs on openai's gpt-4, 2023. URL https://blogs.bing.com/search/march_2023/Confirmed-the-new-Bing-runs-on-OpenAI%E2%80%99s-GPT-4.

[5] OpenAI. Gpt-4 technical report, 2023.

[6] Gpt-4 has a trillion parameters - report, 2023. URL https://the-decoder.com/gpt-4-has-a-trillion-parameters/.

[7] Z. Jia, M. Zaharia, and A. Aiken. Beyond data and model parallelism for deep neural networks. *SysML*, 2019.

[8] L. Zheng, Z. Li, H. Zhang, Y. Zhuang, Z. Chen, Y. Huang, Y. Wang, Y. Xu, D. Zhuo, E. P. Xing, J. E. Gonzalez, and I. Stoica. Alpa: Automating inter- and Intra-Operator parallelism for distributed deep learning. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 559–578, Carlsbad, CA, July 2022. USENIX Association.

[9] C. Unger, Z. Jia, W. Wu, S. Lin, M. Baines, C. E. Q. Narvaez, V. Ramakrishnaiah, N. Prajapati, P. McCormick, J. Mohd-Yusof, X. Luo, D. Mudigere, J. Park, M. Smelyanskiy, and A. Aiken. Unity: Accelerating DNN training through joint optimization of algebraic transformations and parallelization. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 267–284, Carlsbad, CA, July 2022. USENIX Association.

[10] M. Wang, C.-c. Huang, and J. Li. Supporting very large models using automatic dataflow graph partitioning. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Y. Zhao, Y. Liu, Y. Peng, Y. Zhu, X. Liu, and X. Jin. Multi-resource interleaving for deep learning training. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 428–440, New York, NY, USA, 2022. Association for Computing Machinery.

[12] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, and L. Zhou. Gandiva: Introspective cluster scheduling for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 595–610, Carlsbad, CA, Oct. 2018. USENIX Association.

[13] J. Gu, M. Chowdhury, K. G. Shin, Y. Zhu, M. Jeon, J. Qian, H. Liu, and C. Guo. Tiresias: A GPU cluster manager for distributed deep learning. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 485–500, Boston, MA, Feb. 2019. USENIX Association.

[14] Y. Bai, C. Li, Q. Zhou, J. Yi, P. Gong, F. Yan, R. Chen, and Y. Xu. Gradient compression supercharged high-performance data parallel dnn training. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, SOSP '21, page 359–375, New York, NY, USA, 2021. Association for Computing Machinery.

[15] M. Khani, M. Ghobadi, M. Alizadeh, Z. Zhu, M. Glick, K. Bergman, A. Vahdat, B. Klenk, and E. Ebrahimi. Sip-ml: High-bandwidth optical network interconnects for machine learning training. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 657–675, New York, NY, USA, 2021. Association for Computing Machinery.

[16] W. Wang, M. Khazraee, Z. Zhong, M. Ghobadi, Z. Jia, D. Mudigere, Y. Zhang, and A. Kewitsch. TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 739–767, Boston, MA, Apr. 2023. USENIX Association.

[17] L. Zhao, S. Pal, T. Chugh, W. Wang, P. Basu, J. Khoury, and A. Krishnamurthy. Optimal direct-connect topologies for collective communications, 2022.

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[19] Openai's gpt-3 language model: A technical overview, 2020. URL https://lambdalabs.com/blog/demystifying-gpt-3.

[20] Nvidia dgx superpod: Next generation scalable infrastructure for ai leadership, reference architecture, 2023. URL https://docs.nvidia.com/dgx-superpod-reference-architecture-with-dgx-h100-systems.pdf.

[21] Nvlink and nvswitch: The building blocks of advanced multi-gpu communication—within and between servers., 2023. URL https://www.nvidia.com/en-us/data-center/nvlink/.

[22] G. Shainer, P. Lui, and T. Liu. The development of mellanox/nvidia gpudirect over infiniband: A new model for gpu to gpu communications. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, TG '11, New York, NY, USA, 2011. Association for Computing Machinery.

[23] Nvidia gpudirect: Enhancing data movement and access for gpus, 2023. URL https://developer.nvidia.com/gpudirect.

[24] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn. Rdma over commodity ethernet at scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 202–215, New York, NY, USA, 2016. Association for Computing Machinery.

[25] W. Bai, S. S. Abdeen, A. Agrawal, K. K. Attre, P. Bahl, A. Bhagat, G. Bhaskara, T. Brokhman, L. Cao, A. Cheema, R. Chow, J. Cohen, M. Elhaddad, V. Ette, I. Figlin, D. Firestone, M. George, I. German, L. Ghai, E. Green, A. Greenberg, M. Gupta, R. Haagens, M. Hendel, R. Howlader, N. John, J. Johnstone, T. Jolly, G. Kramer, D. Kruse, A. Kumar, E. Lan, I. Lee, A. Levy, M. Lipshteyn, X. Liu, C. Liu, G. Lu, Y. Lu, X. Lu, V. Makhervaks, U. Malashanka, D. A. Maltz, I. Marinos, R. Mehta, S. Murthi, A. Namdhari, A. Ogus, J. Padhye, M. Pandya, D. Phillips, A. Power, S. Puri, S. Raindel, J. Rhee, A. Russo, M. Sah, A. Sheriff, C. Sparacino, A. Srivastava, W. Sun, N. Swanson, F. Tian, L. Tomczyk, V. Vadlamuri, A. Wolman, Y. Xie, J. Yom, L. Yuan, Y. Zhang, and B. Zill. Empowering azure storage with RDMA. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 49–67, Boston, MA, Apr. 2023. USENIX Association.

[26] T. Schneider, O. Bibartiu, and T. Hoefler. Ensuring deadlock-freedom in low-diameter infiniband networks. In *2016 IEEE 24th Annual Symposium on High-Performance Interconnects (HOTI)*, pages 1–8, 2016.

[27] P. Goyal, P. Shah, K. Zhao, G. Nikolaidis, M. Alizadeh, and T. E. Anderson. Backpressure flow control. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 779–805, Renton, WA, Apr. 2022. USENIX Association.

[28] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen. Deadlocks in datacenter networks: Why do they form, and how to avoid them. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, page 92–98, New York, NY, USA, 2016. Association for Computing Machinery.

[29] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang. Congestion control for large-scale rdma deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 523–536, New York, NY, USA, 2015. Association for Computing Machinery.

[30] Z. Wang, L. Luo, Q. Ning, C. Zeng, W. Li, X. Wan, P. Xie, T. Feng, K. Cheng, X. Geng, T. Wang, W. Ling, K. Huo, P. An, K. Ji, S. Zhang, B. Xu, R. Feng, T. Ding, K. Chen, and C. Guo. SRNIC: A scalable architecture for RDMA NICs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1–14, Boston, MA, Apr. 2023. USENIX Association.

[31] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker. Revisiting network support for rdma. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 313–326, New York, NY, USA, 2018. Association for Computing Machinery.

[32] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity

data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, page 63–74, New York, NY, USA, 2008. Association for Computing Machinery.

[33] Introducing data center fabric, the next-generation facebook data center network, 2014. URL https://engineering.fb.com/2014/11/14/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/.

[34] D. Mudigere, Y. Hao, J. Huang, Z. Jia, A. Tulloch, S. Sridharan, X. Liu, M. Ozdal, J. Nie, J. Park, L. Luo, J. A. Yang, L. Gao, D. Ivchenko, A. Basant, Y. Hu, J. Yang, E. K. Ardestani, X. Wang, R. Komuravelli, C.-H. Chu, S. Yilmaz, H. Li, J. Qian, Z. Feng, Y. Ma, J. Yang, E. Wen, H. Li, L. Yang, C. Sun, W. Zhao, D. Melts, K. Dhulipala, K. Kishore, T. Graf, A. Eisenman, K. K. Matam, A. Gangidi, G. J. Chen, M. Krishnan, A. Nayak, K. Nair, B. Muthiah, M. khorashadi, P. Bhattacharya, P. Lapukhov, M. Naumov, A. Mathews, L. Qiao, M. Smelyanskiy, B. Jia, and V. Rao. Software-hardware co-design for fast and scalable training of deep learning recommendation models. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 993–1011, New York, NY, USA, 2022. Association for Computing Machinery.

[35] Nvidia dgx gh200, 2023. URL https://www.nvidia.com/en-us/data-center/dgx-gh200/.

[36] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[37] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: Scaling language modeling with pathways, 2022.

[38] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[39] V. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro. Reducing activation recomputation in large transformer models, 2022.

[40] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models, 2020.

[41] L. Poutievski, O. Mashayekhi, J. Ong, A. Singh, M. Tariq, R. Wang, J. Zhang, V. Beauregard, P. Conner, S. Gribble, R. Kapoor, S. Kratzer, N. Li, H. Liu, K. Nagaraj, J. Ornstein, S. Sawhney, R. Urata, L. Vicisano, K. Yasumura, S. Zhang, J. Zhou, and A. Vahdat. 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, page 66–85, New York, NY, USA, 2022. Association for Computing Machinery.

[42] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, and H. Williams. 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*, SIGCOMM '20, page 782–797, New York, NY, USA, 2020. Association for Computing Machinery.

[43] Openai: Ai and compute, 2023. URL https://openai.com/research/ai-and-compute.

[44] Hello dolly: Democratizing the magic of chatgpt with open models, 2023. URL https://www.databricks.com/blog/2023/03/24/hello-dolly-democratizing-magic-chatgpt-open-models.html.

[45] Z. Li, L. Zheng, Y. Zhong, V. Liu, Y. Sheng, X. Jin, Y. Huang, Z. Chen, H. Zhang, J. E. Gonzalez, and I. Stoica. AlpaServe: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, Boston, MA, July 2023. USENIX Association.

[46] L. Zhao and A. Krishnamurthy. Bandwidth optimal pipeline schedule for collective communication, 2023.

[47] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong, and M. Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems, 2019.