mFabric: An Efficient and Scalable Fabric for Mixture-of-Experts Training

Xudong Liao[1] Yijun Sun[1] Han Tian[1] Xinchen Wan[1] Yilun Jin[1] Zilong Wang[1]
Zhenghang Ren[1] Xinyang Huang[1] Wenxue Li[1] Kin Fai Tse[1] Zhizhen Zhong[2*]
Guyue Liu[3] Ying Zhang[4] Xiaofeng Ye[5] Yiming Zhang[6] Kai Chen[1*]

[1]Hong Kong University of Science and Technology [2]Massachusetts Institute of Technology
[3]Peking University [4]Meta [5]EmbedWay [6]Xiamen University

ABSTRACT

Mixture-of-Expert (MoE) models outperform conventional models by selectively activating different subnets, named *experts*, on a per-token basis. This gated computation generates dynamic communications that cannot be determined beforehand, challenging the existing GPU interconnects that remain *static* during the distributed training process. In this paper, we advocate for a first-of-its-kind system, called mFabric, that unlocks topology reconfiguration *during* distributed MoE training. Towards this vision, we first perform a production measurement study and show that the MoE dynamic communication pattern has *strong locality*, alleviating the requirement of global reconfiguration. Based on this, we design and implement a *regionally reconfigurable high-bandwidth domain* on top of existing electrical interconnects using optical circuit switching (OCS), achieving scalability while maintaining rapid adaptability. We have built a fully functional mFabric prototype with commodity hardware and a customized collective communication runtime that trains state-of-the-art MoE models with *in-training* topology reconfiguration across 32 A100 GPUs. Large-scale packet-level simulations show that mFabric delivers comparable performance as the non-blocking fat-tree fabric while boosting the training cost efficiency (e.g., performance per dollar) of four representative MoE models by 1.2×–1.5× and 1.9×–2.3× at 100 Gbps and 400 Gbps link bandwidths, respectively.

1 INTRODUCTION

Mixture-of-Experts (MoE) models [6, 10, 16, 31, 33, 41, 44, 73, 83] have gained significant traction in the machine learning community to improve the performance of large language models (LLMs) [10, 16]. Unlike traditional methods that scale LLMs by stacking dense layers, which leads to a linear increase in computational costs as model sizes expand, MoE models utilize multiple parallel expert layers and activate only a subset of them based on the input token for each training iteration (e.g., xAI discloses that 25% weights are active in Grok-1 [10]). This dynamic approach enables models to grow to large sizes without a proportional cost increase in computation.

However, such dynamic expert activations require *all-to-all* communications in and out of expert layers in each training iteration. Among parallelization strategies, *expert parallelism (EP)*, which assigns expert layers to different GPUs, requires a high volume of traffic that is comparable to that of *tensor parallelism (TP)*, and much larger than other parallelisms. Moreover, the token-specific activation of experts in EP results in *temporally non-deterministic* and *spatially non-uniform* communication patterns that vary across training iterations, challenging existing GPU interconnects.

Today's GPU interconnects contain intra-server *scale-up networks* (e.g., NVSwitch [27] or NVLink [26]) and inter-server *scale-out networks* (e.g., Ethernet or Infiniband). Both of them are currently dimensioned with uniform and static network topologies (e.g., fully-connected crossbar topology for scale-up networks [27], and Clos-style fat-tree for scale-out networks [34, 76]). To accommodate the temporal and spatial variations of MoE communication patterns, these fabrics contain over-provisioned full bisection bandwidth that is mostly under-utilized. Some recent proposals on the use of optical circuit switching (OCS) to enable topology reconfiguration for *spatially* non-uniform traffic distributions come with the prerequisite of stable *temporal* patterns such that no reconfigurations happen during the entire training process [55, 80]. As a result, these interconnect architectures face bottlenecks, leading to inefficient resource usage and slowdowns in distributed MoE training with both spatial and temporal fluctuations.

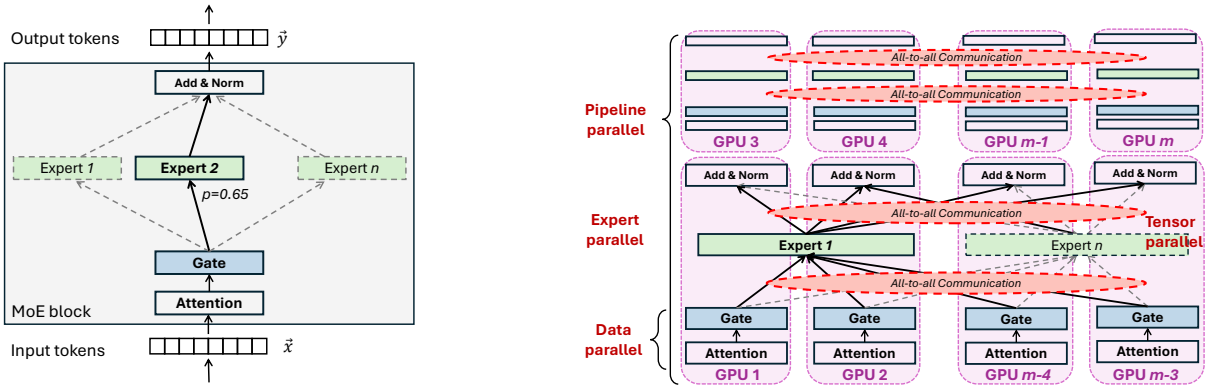
Therefore, to fully unlock the computational advantages of MoE models, we need to design a novel GPU interconnect fabric that is adaptable to dynamic all-to-all communication patterns at runtime. This means that the topology reconfiguration needs to happen *during* the distributed MoE training process. This is very challenging because today's OCS technologies face fundamental trade-offs between *low reconfiguration latency* (to enable reconfiguration during training) and *high scalability* (to interconnect tens of thousands of GPUs) (more details in Table 2).

To understand the problem space, we first perform a comprehensive measurement study in a production cluster to investigate the real-world communication patterns of distributed MoE training. Our measurements reveal that although EP generates strong variability during training, its dynamic range is strictly within an MoE block, creating *strong locality* for all-to-all traffic on a global scale (§3).

Based on this insight, we introduce mFabric, a novel system designed to overcome these challenges by enabling efficient topology reconfiguration during distributed MoE training. At the core of mFabric is a *regionally reconfigurable high-bandwidth domain* architecture that takes advantage of millisecond-scale OCS for inter-host networks on top of existing electrical interconnects. This hybrid design allows for rapid localized reconfiguration, improving both the scalability and adaptability of the interconnect (§4).

mFabric contains the following key components: 1) mFabric leverages the *semi-predictability* of all-to-all transfers in the MoE

*Corresponding authors: zhizhenz@mit.edu (Z. Zhong), kaichen@cse.ust.hk (K. Chen).



(a) An MoE block with multiple experts. In this example, the *gate* only activates *Expert 2*.

(b) Example of a hybrid parallelism for distributed MoE training that combines DP (gate), EP (parallel expert layers), PP (MoE blocks) and TP (a single expert layer).

Figure 1: Illustration of the MoE model architecture and its distributed training strategies.

layer to implement a traffic monitor to track regional network demands (§5.1); 2) Based on the obtained demands, mFabric uses a greedy algorithm to generate the OCS topology, balancing algorithmic complexity with topology efficiency (§5.2); 3) Finally, mFabric implements a custom collective communication manager for inter-host DP and EP to route traffics via both electrical interconnects and OCS (§5.3).

To evaluate mFabric, we build a fully functional prototype with 32 NVIDIA A100 GPUs, 16 Mellanox 100G NICs [22], a Polatis millisecond-scale OCS [30], and a 100G Ethernet switch. In addition, we develop a custom collective communication runtime based on NCCL [19] to support in-training topology reconfigurations. We successfully train three state-of-the-art MoE models using this prototype (§6).

To study the performance of mFabric at scale, we conduct packet-level simulations using four representative real-world MoE models. Our simulation results reveal that mFabric outperforms the state-of-the-art benchmarks, exhibiting comparable training speed to the best electrical (rail-optimized [7] and Fat-tree [34]) fabrics, while improving cost-efficiency by up to 1.5× and 2.3× for 100Gbps and 400Gbps links, respectively. (§7). We also observe that mFabric outperforms TopoOpt [80] by up to 2.5× and exhibits scalability with the cluster size increasing to 30K+ GPUs.

As the first system to enable in-training topology reconfiguration for distributed MoE training, our code will be available at the time of publication to facilitate further discussions.

2 BACKGROUND

In this section, we first describe MoE's model architecture and training parallelization strategies (§2.1). Then, we discuss several existing technologies for GPU interconnects (§2.2).

2.1 Distributed MoE Training

MoE model architecture. MoE models contain several sequential MoE blocks (layers) [44, 73]. As shown in Figure 1a, each MoE block has one attention layer, a gate unit, and several parallel Feed-Forward Networks (FFNs) called *experts*. The input token x is first fed into the attention layer. After that, the gate unit to select the most relevant experts based on the output of the attention layer. For example, *expert 2* is activated while other experts are omitted. This is called computation-based routing and is the key to enabling MoE's sparse architecture that scales to trillions of parameters without a linear increase in computation cost. The output token, y, is the weighted sum of the outputs of all selected experts.

Models Size	Mixtral 8×7B	LLaMA-MoE 6.7B	Qwen-MoE 14.3B
# of MoE blocks	32	32	24
# experts	8	16	64
EP degree	8	16	16
TP degree	4	1	1
PP degree	4	4	4
Seq. len.	4096	4096	4096
Micro-batch size	8	8	8

Table 1: State-of-the-art MoE training configurations.

Data Parallelism (DP). In DP [60], the model parameters are replicated across multiple GPUs, and each GPU hosts a different subset of the training data. Because only gradients are transferred across GPUs, the traffic volume is relatively small compared to other parallelisms. In MoE models, DP is usually applied to smaller components, such as gate units, and add & norm layers (Figure 1b). DP also applies as we create replicas of the whole model onto several different clusters.

Pipeline Parallelism (PP). In PP [67, 74], multiple sequential stages of the model are distributed to different GPUs (Figure 1b). Therefore, only hidden activation states are transferred through point-to-point all-reduce collective communication primitives, hence generating the least amount of communication with deterministic volume.

Tensor Parallelism (TP). TP [74] is a technique to partition an individual layer between multiple GPUs. In the context of MoE training (Figure 1b), the expert layers are so large that they need to be partitioned across different GPUs, with intermediate hidden

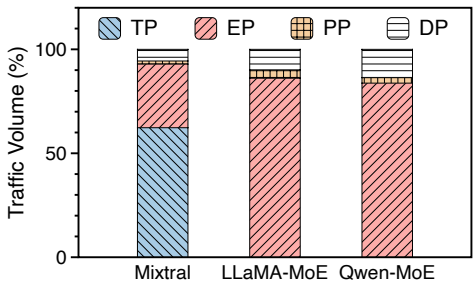


Figure 2: Traffic volume distribution of different parallelism of three state-of-the-art MoE models.

Commodity OCS	Port count	Reconfig. delay
Robotic (Telescent) [80]	1008×1008	several minutes
Piezo (Polatis) [30]	576×576	10-25 *ms*
3D MEMS (Calient) [5, 71]	320×320	10-15 *ms*
2D MEMS (Google Palomar) [61]	136×136	Not reported
RotorNet (InFocus) [65, 66]	128×128	10 μs
Silicon Photonics (Lightmatter) [13]	32×32	7 μs
PLZT (EpiPhotonics) [18]	16×16	10 *ns*

Table 2: Tradeoff between port count and reconfiguration delay in commodity OCS technologies.

states being transferred through broadcast, all-gather and reduce-scatter collective communication primitives. Therefore, TP is the most communication-intensive operation, and its communication scale is generally limited to fewer than eight GPUs in practice [74].

Expert Parallelism (EP). In MoE models, different experts in an MoE block are allocated to different GPUs [44, 58, 62] (Figure 1b). Since each GPU needs to send its local states to other experts and receive remote states from other GPUs, the dispatching of intermediate hidden states and the collection of expert outputs are performed via two all-to-all communications. EP's all-to-all communication is non-uniform and non-deterministic across different training iterations (§3).

Traffic volume of different parallelisms. To understand the traffic volumes of different parallelization strategies, we profile three state-of-the-art MoE models, Mixtral 8×7B MoE [16], Llama-MoE [83] and Qwen-MoE [31], with Megatron-LM [74] and measure the total amount of data transfer. The detailed model configurations are shown in Table 1. We plot the distribution of traffic volume in one MoE training iteration in Figure 2. For Mixtral 8×7B, we observe that TP generates the highest traffic volume, which accounts for above 60% of the total traffic volume. The EP generates the second-highest traffic volume (30% of the total traffic volume). PP and DP generate the least traffic volume (less than 6%). For LLaMA-MoE and Qwen-MoE, we find that EP becomes the most communication intensive, generating more than 80% of the total traffic volume. This is due to the absence of TP, as the size of the largest layer (e.g., expert layers) is smaller than that of a single GPU memory.

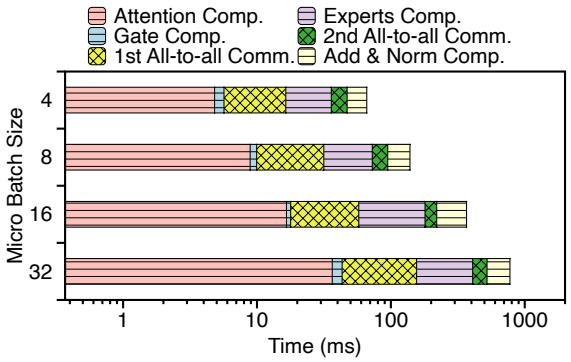


Figure 3: [Mixtral 8×7B in production] EP's all-to-all communications occupy 10 ms to 114 ms (21% to 33%) of the total training iteration time at different batch sizes under 400 Gbps Infiniband network (Note that x-axis is log-scaled).

2.2 GPU Interconnects

Scale-up interconnects: NVLink and NVSwitch. NVLink and NVSwitch are proprietary technologies provided by NVIDIA to support GPU communications within a host server [21, 26]. They offer a higher bandwidth (1.8 TB/s) than PCIe (128 GB/s). In particular, NVSwitch is a non-blocking crossbar architecture that works similarly to a circuit switch [27].

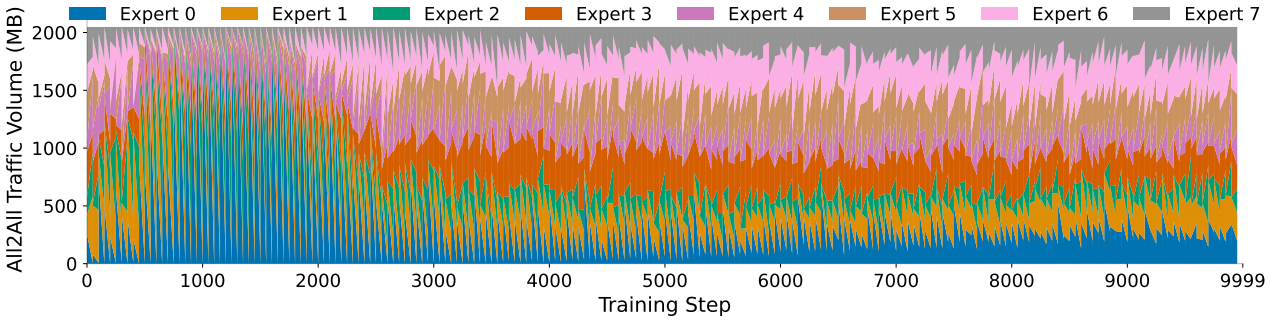
Scale-out interconnects: electrical packet switching (EPS). Ethernet- and Infiniband-based EPS has been widely adopted in data center networks with clos-style topologies [34, 47–49, 52, 76]. In such networks, data are encapsulated into packets and switched at layer 2 (MAC) or above. EPS has the advantage of massive scalability to hundreds of thousands of host servers in modern data centers [34]. However, EPS networks are fixed in topology that cannot be easily reconfigured.

Scale-out interconnects: optical circuit switching (OCS). OCS is an layer-1 (PHY) switching technology that creates dedicated reconfigurable optical circuits between hosts. As depicted in Table 2, today's commodity OCSes have a fundamental tradeoff between the *scalability* (in terms of port counts) and *agility* (in terms of reconfiguration delay). Technologies like the robot optical patch panel [80] scale up to thousands of ports at the cost of several minutes of reconfiguration delay. While at the other end of the spectrum, waveguide-based OCS like silicon photonics [13] and PLZT [18] scores microseconds or nanoseconds latency with limited port counts.

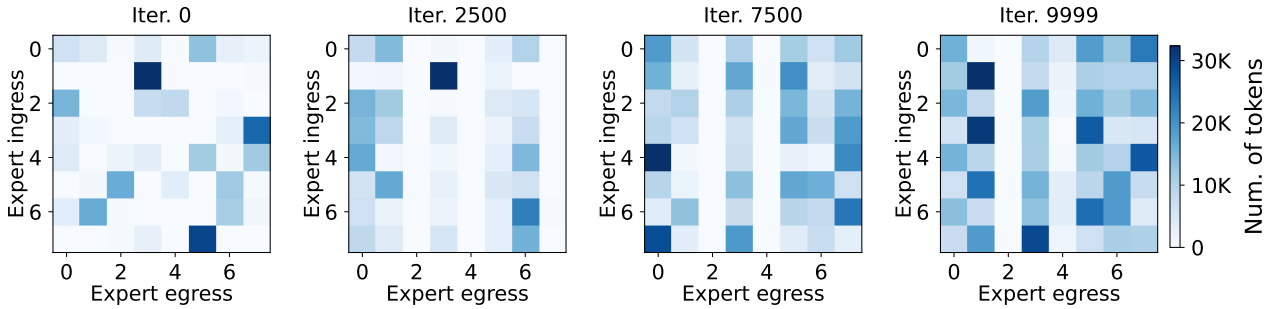
3 MOE PRODUCTION MEASUREMENTS

Unlike conventional parallelism (e.g., TP, PP and DP) whose communication patterns are deterministic, EP's communications are determined by the gating unit at runtime due to the semantic heterogeneity of the input tokens. In this section, we measure the communication patterns of Mixtral 8×7B [16] in a production data center. We use a hybrid parallelism that combines an EP degree of 8, TP degree of 4, PP degree of 4 at a sequence length of 4096 and micro-batch size of 8 [42].

The production compute fabric. We perform a comprehensive measurement study using a Certified NVIDIA DGX SuperPOD



(a) In temporal dimension, the all-to-all traffic volume of each node varies across different training iterations.



(b) In spatial dimension, the all-to-all traffic volume is non-uniform across different experts.

Figure 4: [Mixtral 8×7B in production] All-to-all traffic dynamics during MoE training.

platform [20] with 128 H-series (Hopper architecture) GPUs and 128 ConnectX-7 400Gbps NICs in a production data center. The compute fabric is connected in a rail-optimized topology [7] managed by NVIDIA. The NVIDIA Collective Communications Library (NCCL) [19] is used to optimize communications on the DGX platform.

Reconfiguration opportunity for all-to-all communications. We first measure the time it takes for each step in Mixtral 8×7B's forward pass computation[1] and show the results in Figure 3. We observe that the expert computation takes 30ms and is proportional to the used micro-batch size. For the typical micro-batch size used in production (e.g., 8), the expert computation takes more than 100ms, which is much larger than the reconfiguration latency of existing optical switches (for example, the MEMS OCS in Table 2). Therefore, it provides an opportunity to reconfigure the OCS for the second all-to-all in the expert computation phase. For backward propagation, it allows us to hide the reconfiguration latency in the attention computation of its later layer (for the second all-to-all) and expert computation period (for the first all-to-all) as the backward computation often takes more time than the forward.

MoE's all-to-all communications are temporally and spatially dynamic. We measure the all-to-all communication dynamics across the 10000 iterations of Mixtral 8×7B in Figure 4. In particular, Figure 4a plots the total communication volume that each expert receives in all-to-all communication in each MoE layer, which represents the intensity of activation of each expert and therefore determines its total ingress/egress communication volumes. We find that the activation intensities of each expert vary significantly across different iterations, which indicates the non-deterministic nature of the EP traffic. Figure 4b further plots the detailed all-to-all communication volume matrix of layer 0 through different iterations. We observe that each traffic matrix of all-to-all communication is non-uniform, with heavy communication only between several GPU pairs. We also find that as training progresses, the variability of the overall communication volume among experts decreases. The decreasing variability can be attributed to the use of the load balancing loss, which requires that all experts should have similar loads of tokens. However, even as the overall communication volumes of experts appear to converge, the sparsity of all-to-all traffic matrices persists, as illustrated in Figure 4b. Additionally, recent advancements in the ML community have introduced MoE training techniques where certain specialized experts are intentionally to left underutilized to achieve improved inference performance [40, 63]. This further implies the communication sparsity in MoE training.

All-to-all communications have strong locality. Figure 5 shows the all-to-all communications among all the 128 GPUs during the training of Mixtral 8×7B. We observe that the EP traffic exhibits *strong locality*. This is because only the expert layers within the same MoE block need all-to-all communications, while expert layers across different MoE blocks at different PP stages do not communicate directly.

[1]The backward pass is a reverse process of the forward pass.

	Traffic Patterns			Ideal Interconnects			Technology
	Volume	Temporal Variability	Spatial Variability	Bandwidth	Reconfigurability	Scalability	
TP	Highest	No	Local & All-Reduce	High	Slow & One-Shot	Small	Crossbar (NVSwitch)
EP	High	Yes	Regional & All-to-All	High	Fast & In-Training	Medium	Circuit Switching (Optical)
DP	Low	No	Large & All-Reduce	Low	Slow &One-Shot	Large	Electrical Packet Switching
PP	Low	No	Large & Point-to-Point	Low	Slow &One-Shot	Large	Electrical Packet Switching

Table 3: The quest for a best fit between interconnect fabric and the MoE parallelization strategies.

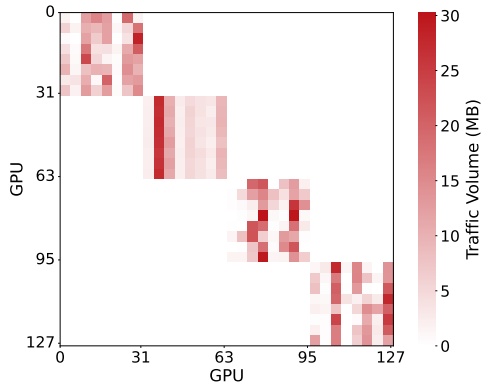


Figure 5: [Mixtral 8×7B in production] Traffic matrix of all GPUs showing strong locality.

The above observations are stemmed from the inherent sparse activation characteristic of MoE layer and the gradual refinement of the gating unit during training. We note that other papers in ML community [6, 59] have also observed similar behaviors, suggesting that these characteristics are common across different MoE models.

4 MFABRIC ARCHITECTURE DESIGN

So far, we have shown that MoE introduces unique traffic patterns that non-deterministic and non-uniform. Now, an important question to address is *How to design a network architecture that best serves the requirements of distributed MoE training?* In this section, we first study an ideal yet practical fabric for distributed MoE training (§4.1). Then, we describe our key proposal in MFABRIC, the regionally reconfigurable high-bandwidth domain with OCS (§4.2).

4.1 Towards A Ideal yet Practical Fabric

We start with a thought experiment from first principles on designing an ideal yet practical fabric for distributed MoE training.

The ideal fabric. Designing an ideal fabric for distributed MoE training requires a best fit between the traffic patterns presented by parallelization strategies and the properties of the interconnect. In Table 3, we first summarize the traffic requirements in terms of volume, predictability, and locality for different parallelisms (discussed in §2.1). Then, we list the ideal fabric considering its bandwidth, reconfigurability and scalability. For conventional parallelisms like TP, DP, and PP, they only require one-shot reconfiguration because their communication patterns are fully deterministic. While TP requires high bandwidth within a small radix because sharding a single layer into too many GPUs is not efficient, DP and PP are on the opposite side which could possibly span a large number of GPUs with relatively low communication bandwidth requirements. Instead, EP exhibits significant differences in traffic patterns, hence the corresponding interconnect. In particular, the non-deterministic traffic patterns require in-training topology reconfiguration, and the regional all-to-all communications among experts require a medium fabric radix. Therefore, the ideal fabric for MoE training should be a globally reconfigurable network capable of adjusting its topology as soon as traffic patterns vary within a training iteration (less than a second in practice). Recall the fact in Figure 3 that the topology reconfiguration must be completed before each all-to-all communication phase, the time window left for topology reconfiguration is on the order of tens of milliseconds.

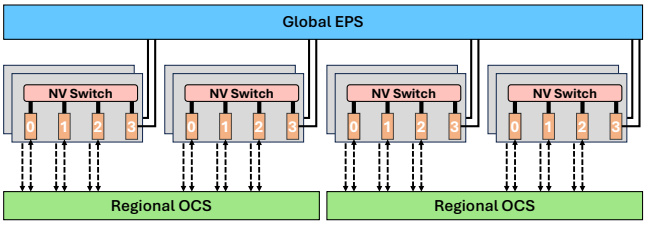


Figure 6: MFABRIC network architecture. Orange boxes represent GPUs. For clarity, we depict each server with 4 GPUs.

Challenges of the ideal fabric. Implementing this ideal fabric presents significant challenges. Recall that in Table 2 we review a trade-off in commodity OCS technologies[2] between reconfiguration delay and port count. Achieving millisecond scale reconfiguration times typically limits the number of ports an OCS can support (typically less than a few hundred ports), making it difficult to scale to all nodes in a large MoE interconnect (hundreds of thousands). In contrast, increasing the port count to support a global network results in slower reconfiguration times, failing to meet the rapid adaptation required within training iterations.

Landing the ideal fabric in practice. To reconcile these challenges, we leverage the key observation that despite the non-deterministic and non-uniform characteristics of MoE all-to-all traffic, there is a strong communication locality for traffic variations. Therefore, instead of building a globally reconfigurable OCS fabric, we propose designing several *regionally reconfigurable OCS* networks (described next in §4.2). Figure 6 depicts the network architecture of MFABRIC.

[2]There exist small-scale laboratory prototypes that may break this tradeoff by leveraging advanced device innovations [36, 56], and they are out of the scope of this paper. We focus only on commodity solutions that are readily deployable at scale.

By partitioning the network into several domains where the communication locality is strong (for EP), each regional OCS rapidly adjusts to local traffic demands without the complexity of global reconfiguration. This approach mitigates the trade-off between reconfiguration speed and port count by limiting the scope of each OCS, allowing for rapid reconfiguration within regions. By implementing regionally reconfigurable OCS networks, we can leverage this locality to achieve fast reconfiguration within smaller, manageable regions. This approach balances the need for adaptability and reconfigurability with practical hardware limitations, effectively supporting the dynamic communication patterns of MoE training.

4.2 Regionally Reconfigurable OCS

mFabric, as the first fabric to support the in-training topology reconfiguration, highlights the core idea of building regionally reconfigurable OCS to offload dynamic EP traffic in the existing electrical fabric. By leveraging the strong locality inherent in MoE's all-to-all traffic, we partition the network into regions where communication demands are non-deterministic among expert layers. This regional approach allows for rapid reconfiguration within each partition, effectively overcoming the fundamental trade-off between reconfiguration speed and port count in OCS technology. By focusing on regional reconfigurability, mFabric achieves scalability while maintaining rapid adaptability to dynamic communication patterns of MoE training, alleviating the complexity of global network reconfiguration.

Where to deploy regionally reconfigurable OCS? To best serve EP traffic's locality, OCS is connected to a cluster of servers where each server splits its NICs between OCS and EPS. Today's fast OCSes with reconfiguration delays of less than tens of milliseconds support up to 500 ports (Table 2). Therefore, in a typical server configuration with eight NICs, the reconfigurable OCS region interconnects around 80 to 250 servers (with each server assigning two to six NICs to OCS).

How to reconfigure the OCS topology? In mFabric, the OCS fabric is arranged into multiple isolated slices for each reconfigurable high-bandwidth region. Therefore, the regional OCS topology is controlled by its localized topology controller that frequently collects traffic demands from the host servers. Within each training iteration, there are four all-to-all communications with the same traffic pattern. However, the traffic patterns are non-deterministic across training iterations. Hence, we need to develop a mechanism to reconfigure the topology within each training iteration. The localized topology reconfiguration mechanism implies that mFabric does not require a centralized topology controller, which avoids scalability concerns on the control plane.

When to perform topology reconfiguration? Different from EPS networks that are connectionless, OCS networks are connection-oriented. During topology reconfiguration, OCS networks are not available to carry packets[3]. Therefore, to reconfigure the topology without blocking the training process, e.g., for the second all-to-all communication during FP, the best case is that we can hide the reconfiguration latency in the expert computation period.

[3]Most commodity optical switches require tens of nanoseconds to several milliseconds or (Table 2) to reconfigure their topology.

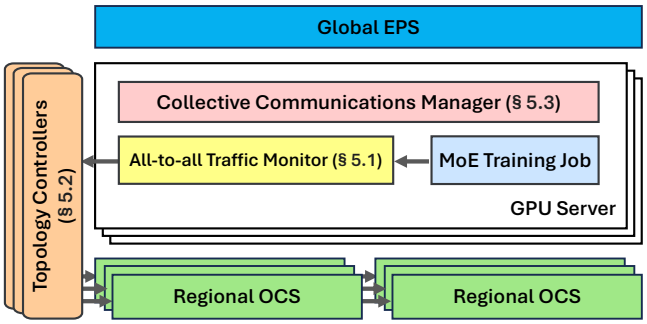


Figure 7: mFabric control plane implementation.

Towards a mixed optical-electrical fabric. mFabric seeks the best fit between the traffic patterns of the parallelization strategies and the corresponding switching technologies. Therefore, mFabric uses server-scale NVSwitch for tensor parallelism, regionally reconfigurable OCS for expert parallelism, and large-scale EPS for data parallelism and pipeline parallelism. To distribute data movement tasks on different fabrics, a new collective communication library that supports topology reconfiguration is needed.

5 MFABRIC SYSTEM IMPLEMENTATION

To enable architecture design in §4, we design a system implementation of the control plane for mFabric. As shown in Figure 7, mFabric's control plane contains a traffic monitor that not only keeps track of the traffic demands in EP but also makes predictions for subsequent all-to-all communications (§5.1). Based on these monitored traffic demands, a centralized topology controller generates topology reconfiguration and dispatch the results to regional OCS (§5.2). Finally, a collective communication manager is responsible to steer the traffic in the mFabric fabric (§5.3).

5.1 All-to-All Traffic Monitor

As shown in Figure 1b, each MoE layer experiences a total of four all-to-all communication phases during each training iteration—two during the forward pass and two during the backward pass. The first all-to-all communication in each layer occurs after the gating unit computation. The output from the gating unit[4] across the expert parallel (EP) groups determines the traffic matrix for this communication phase. These four all-to-all communications share the same communication matrix due to the symmetry of the token dispatching and collection. Therefore, we observe *semi-predictability* on all-to-all transfers, as the second all-to-all in FP and two all-to-all transfers in BP can be determined after the gating process, while FP's first all-to-all cannot be fully predicted in advance.

mFabric's objective is to reconfigure the OCS to dynamically allocate bandwidth for expert parallelism communication on demand. For the latter three times of all-to-all communication on a MoE layer, mFabric is able to predetermine their demands and therefore reconfigure the OCS to provision efficient optical circuits in the

[4]Referred to here as the expert load of tokens. The actual output of the gating unit is the dispatching probability distribution for each token. The expert load is derived directly from the probability distribution using the top-k parameter.

Algorithm 1 ReconfigureOCS

1: **procedure** ReconfigureOCS(E, α, N, V)
▷ **input** E: all-to-all communication demands of experts
▷ **input** α: current optical degree
▷ **input** N: number of servers
▷ **input** V: server node set
▷ **output** S: NIC level mapping in OCS
▷ Initialize
2: C ← zero matrix of size $N \times N$
3: $avail\_ocs[v] \leftarrow \alpha$ for $v \in V$
4: Initialize finish time $T = \infty$ if $D[i][j] \neq 0$, otherwise $T = 0$
▷ D is translated into an upper triangular matrix as we allocate TX and RX together.
5: D ← calculate_server_demand(E)
6: **while** True **do**
7: (i, j) ← findBottleneckLink(T, C, V)
▷ Allocate OCS link
8: **if** $avail\_ocs[i] > 0$ and $avail\_ocs[j] > 0$ **then**
▷ Create a link between i and j
9: $C[i][j] \leftarrow C[i][j] + 1$ and $C[j][i] \leftarrow C[j][i] + 1$
10: **for** $v \in \{i, j\}$ **do**
11: $avail\_ocs[v] \leftarrow avail\_ocs[v] - 1$
12: **else**
13: Break
▷ Update the time matrix
14: $T[i][j] \leftarrow \frac{D[i][j]}{C[i][j]}$, $T[j][i] \leftarrow \frac{D[j][i]}{C[j][i]}$
▷ Calcuate OCS reconfigure schedule
15: $S \leftarrow$ GetNICMapping(C)
▷ Permute the schedule for intra-host topology optimization
16: $S \leftarrow$ permuteLinks(S)
▷ Reconfigure the OCS
17: reconfigureOCS(S)
18: **return** S

regionally HB domain. The reconfiguration time can be hidden into MoE computation.

For communication in the first all-to-all in FP, given that the mFabric network architecture is provisioned with millisecond-scale reconfigurable OCS, there are two options. mFabric reconfigures the OCS but blocks the training process, as this reconfiguration time cannot be hidden in the computation. On the other hand, mFabric can choose to utilize a random topology/reuse topology from previous MoE layers, which, however, cannot benefit from the efficient circuits schedule.

We also observe the *partial predicabilty* of the FP's first all-to-all, which offers an opportunity to proactively reconfigure the OCS for it in advance. We offload the details of the prediction algorithm for all-to-all traffic demand in §B.

Note that mFabric does not introduce extra demand collection overhead as the state-of-the-art MoE training framework has already implemented a mechanism to collect this information [4] to perform on-demand all-to-all transmission.

5.2 Topology Controller

mFabric reconfigures the regional OCS topology in different EP groups based on the aforementioned predicted all-to-all communication demands. As finding an optimal topology and deriving an optical schedule is an NP-hard and time-consuming problem [45], we address this challenge by proposing a lightweight greedy algorithm. The key insight is that all-to-all communication time is bottlenecked by the largest transfers, which implies that the corresponding GPU pair should be allocated with more circuits. Thus, in each iteration, we identify the communication pairs with the longest transmission time and greedily assign them with direct optical links in the OCS topology. As presented in Algorithm 1, the detailed OCS reconfiguration algorithm works as follows:

Step 1: Obtain the inter-server demand (line 5). Given the predicted all-to-all communication demands, the algorithm first maps the traffic matrix to an actual inter-server communication demand, with respect to the number of experts per GPU and the number of GPUs per server. Note that we provision the TX and RX bandwidth of each OCS link together, thus making the inter-server demand matrix upper triangular via adding the TX and RX demands together.

Step 2: Find communication bottleneck (line 7). The algorithm then iteratively finds the bottleneck of the current allocated links. The bottleneck link is defined as the link that has the longest completion time given the demand matrix D and allocated link matrix C. We greedily calculate the bottleneck link by calculating the completion time of each link and return the server pairs with the longest completion time.

Step 3: Allocate OCS circuit (line 9-11). The algorithm first allocates the OCS link for the found bottleneck server pairs. If the OCS NICs of two servers are not fully allocated, the algorithm will assign the link for them accordingly.

Step 4: Generate OCS topology (line 15-16). The algorithm then generates the OCS topology by mapping the TX and RX of NICs based on the allocated link matrix C. Note that, if there are multiple links between two servers, the algorithm first permutes the connection to achieve a NUMA optimized topology to avoid intra-host congestion. For example, if there are two links between server A and server B, the algorithm will permute the connection to make sure the corresponding TX and RX NICs are in two different NUMA nodes for further intra-host traffic forwarding (§5.3).

Step 5: Reconfigure OCS (line 17). With the permuted OCS topology, the final step is to reconfigure the OCS cross-link connections accordingly. The topology manager leverages the TX/RX pair mappings to establish the optical path for the aforementioned pairs.

5.3 Collective Communications Manager

With the generated topology, the next step is to allocate network traffic from different parallelisms to mFabric and generate routing schedules. In the following, we illustrate how mFabric's collective communications manager routes network traffic from various parallelisms in the data path.

TP and PP. TP traffic is limited to intra-host high bandwidth domain, utilizing the ultra-high-bandwidth NVSwitch. PP traffic

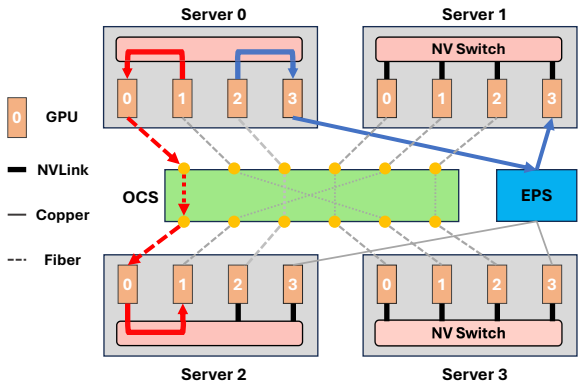


Figure 8: Routing of all-to-all communication in MFABRIC. For simplicity, the TX/RX links are merged into a single link in the OCS. For example, there are two direct paths from server 0 to server 2 within the OCS fabric, while communication from server 0 to server 1 must utilize the EPS fabric.

occurs across different PP stages and relies on the high-fanout EPS fabric in the MFABRIC architecture. As a result, no special arrangements are needed for these two types of parallelism.

Hierarchical DP. DP traffic typically spans the entire training cluster. Therefore, MFABRIC routes it through the EPS fabric. To further improve the communication efficiency of DP transfers, we leverage the hierarchical all-reduce algorithm [53, 77] to reduce the outbound traffic volume from each server. Specifically, DP parameter synchronization occurs in three stages across the cluster. First, the GPUs within each server perform an intra-host reduction to aggregate the parameters to a gateway DP GPU connected to the EPS NIC. Next, all servers engage in a global ring all-reduce among the gateway DP GPUs to synchronize the model parameters. Finally, each server broadcasts the synchronized parameters from the gateway DP GPU to all other GPUs. The first and third stages of communication use the high-speed NVSwitch, while the second stage relies on the relatively lower-bandwidth EPS fabric. If multiple EPS NICs are available in the fabric, MFABRIC utilizes a multi-ring all-reduce method to fully exploit the bandwidth and reduce communication time.

Topology-aware EP. EP traffic is expected to use the regionally reconfigurable high-bandwidth domain. After reconfiguration, MFABRIC essentially offers a direct-connect topology for EP transfers. According to this topology, MFABRIC utilizes the following steps to route the EP traffic. For ease of understanding, we illustrate this process in Figure 8, which depicts a reconfigured topology among four servers with a total of 16 GPUs (EP degree equals to 16).

(1) Each GPU looks up the topology to identify its intra-server communication delegation GPU for all communication pairs. MFABRIC prioritizes directly connected optical circuits over EPS. For example, the delegation GPUs from server 0 to server 2 are GPU 0 and 2, as they are allocated with optical connections. Similarly, server 0's GPUs need to relay corresponding traffic to GPU 1 if they want to communicate with server 3. However, to perform communications between server 0 and server 1, they have to use GPUs that are connected in the EPS.
(2) With the gateway information, each server performs an intra-host `gather`, gathering outbound data to the corresponding delegation GPUs via NVSwitch. Note that in §5.2, we balanced the number of NICs across each NUMA node to mitigate intra-host congestion when multiple links are provisioned between a server pair. MFABRIC aims to distribute the traffic load across delegation NICs as evenly as possible.
(3) Each server initiates the inter-host all-to-all communication across all delegation GPUs using NICs in both the EPS and OCS fabrics.
(4) Each server performs an intra-host all-to-all communication among local experts via NVSwitch.
(5) The delegation GPUs in each server `scatter` the received all-to-all data to its final destination.

As the dataflows in phases 3 and 4 do not interfere with each other, MFABRIC overlaps the communication in these two phases to reduce overall completion time.

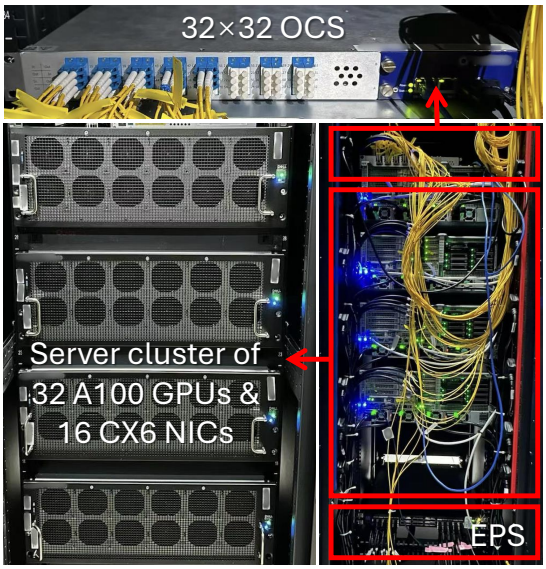


Figure 9: MFABRIC prototype using commodity hardware.

6 MFABRIC PROTOTYPE

To evaluate MFABRIC, we build a fully functional prototype using commodity hardware[5] capable of training state-of-the-art MoE models.

Prototype hardware setup. Figure 9 is a picture of our prototype, which contains four GPU servers, each equipped with eight NVIDIA A100 GPUs and four Mellanox ConnectX-6 100G NICs. Three NICs of each server are connected to a Polatis MEMS OCS [30], while the remaining one NIC is connected to a NVIDIA SN3700 Ethernet switch [28]. We use 100 Gbps QSFP28 optical transceivers and duplex LC fibers [12]. All NICs operate in RoCEv2 mode. Each server has a total of four NVLinks that connect two adjacent GPUs.

[5]Due to NVIDIA's warranty restrictions, we are not allowed to modify the topology of the DGX SuperPod used in the measurement study §3.

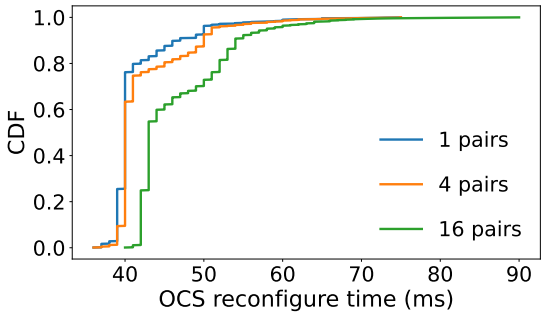


Figure 10: [Testbed] Reconfiguration delay across different number of pairs.

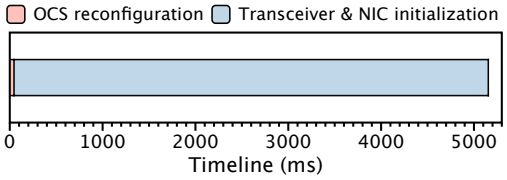


Figure 11: [Testbed] Overall timeline of one OCS control.

Prototype software runtime. We implement mFabric using approximately 6K lines of code in C++, including the topology generator, the OCS controller, and the custom collective communication library supporting in-training topology reconfigurations. For DP and PP communication over the static EPS fabric, we utilize NCCL [19] to provide high-speed intra-host communication and inter-host all-reduce/point-to-point operations. For EP's all-to-all communications that involve both EPS and OCS, we implement a custom collective communication library that leverages RDMA for high-speed data transfer using the raw `ibverbs` library. We port the mFabric runtime to Python to integrate with Megatron-LM [74] for training real-world MoE models.

Prototype profiling. We profiled the overall reconfiguration turnaround time of our OCS, and the results are shown in Figure 10. The OCS is controlled by issuing TL1 commands over Ethernet. We observed that as the number of pairs increases, the reconfiguration time slightly rises. The average reconfiguration time is approximately 41.44 ms for 1 pair, 42.44 ms for 4 pairs, and 46.75 ms for 16 pairs. The 99th percentile reconfiguration times are around 60 ms for 1 pair, 62 ms for 4 pairs, and 68 ms for 16 pairs. Notably, 99% of the reconfiguration times are under 70 ms, which is acceptable for MoE training, given the relatively long expert computation times typically used in practice (e.g., 122 ms for a batch size of 16).

Figure 11 illustrates the overall timeline from issuing an OCS reconfiguration control command to the successful completion of an RDMA send, providing a detailed view of the control process. The process consists of two main stages: (1) the control server sends a reconfiguration command to the OCS; (2) the transceiver and NIC initialize the physical link and set up the network device. Our observations indicate that the overall turnaround time of one reconfiguration is predominantly influenced by the physical link initialization and NIC device initialization.

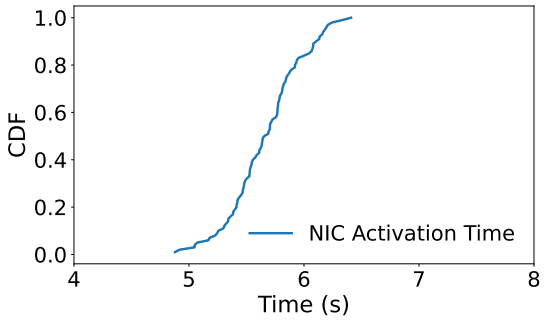


Figure 12: [Testbed] CDF of Time Elapsed from OCS Reconfiguration Completion to NIC Becoming Active.

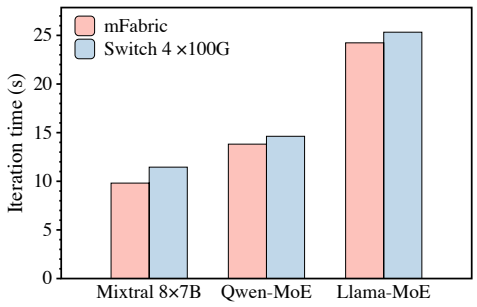
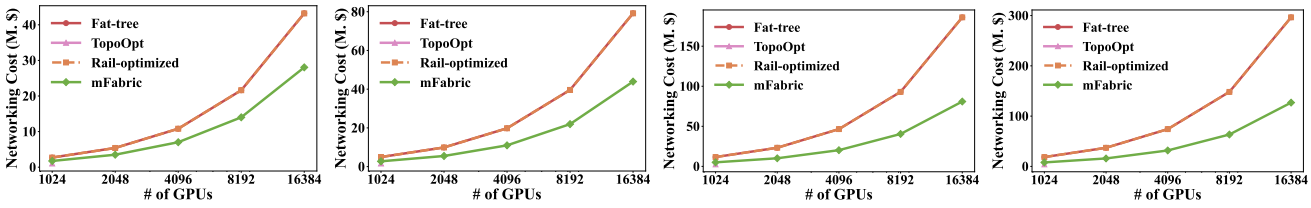


Figure 13: [Testbed] End-to-end training iteration time.

We further plot the CDF of the time elapsed from the OCS reconfiguration completion to the NIC becoming active. The results are shown in Figure 12. The average NIC activation time is approximately 5.67 s and the 99 percentile is around 6.33 s. These findings align with previous observations from [65], highlighting that some commodity transceivers and NICs are not well optimized for fast reconfiguration. Therefore, we exclude this NIC activation time to calculate the actual training time in mFabric testbed experiments. Due to the limited number of GPUs, we cannot train the full MoE models as shown in Table 1. We only run 7 layers of Mixtral 8×7B, 16 layers of LLaMA-MoE, and 12 layers of Qwen-MoE.

Training state-of-the-art MoE models. Figure 13 shows that mFabric achieves comparable performance to the 4 × 100G switch-based baseline. mFabric utilizes one NIC in the EPS fabric and configures the remaining three NICs in an optical circuit fabric (a total of 12 optical ports and 4 electronic ports). In contrast, the 4 × 100G baseline uses the four 100 Gbps ConnectX-6 NICs in a non-blocking EPS fabric with 16 electronic ports. mFabric's performance stems from its ability to efficiently provision high-bandwidth optical circuits for communication-intensive pairs in sparsely non-uniform all-to-all traffic, without compromising the transfer speed of DP and TP traffic. It is important to note that mFabric *does not alter the parallelization strategies* used in MoE training, but only accelerates network transmissions through its architectural design and efficient circuit-switching algorithm. As a result, mFabric does not affect the training accuracy of MoE models.



(a) 100Gbps (b) 200Gbps (c) 400Gbps (d) 800Gbps

Figure 14: [Simulation] Networking cost analysis.

7 LARGE-SCALE SIMULATIONS

This section evaluates the performance of mFabric over large-scale production clusters through large-scale simulation. First, we describe the simulation setup and methodology in §7.1. Next, we compare the cost of mFabric against other interconnects in §7.2. We then evaluate the end-to-end training speed of mFabric in §7.3. Finally, we provide a deep dive into the impact of optical degree and perform scalability analysis of mFabric in §7.5.

7.1 Setup

Packet-level simulation methodology. The simulation process is divided into two phases. In the first phase, we utilize a simulator built on top of FlexFlow [8]. We extend FlexFlow to support pipeline parallelism and rectify its profiler to ensure that the profiled computation time aligns with the actual runtime on the testbed. The simulator is fed with the micro-batch size, an MoE model, and a specified parallelization strategy, and generates a task DAG that describes the computation and communication tasks for the cluster. Using this DAG, we then utilize an event-driven packet-level simulator based on `htsim`[6] [11], which simulates packet-based communication between GPUs. The link propagation delay is set to 1 μs. We set the number of NICs and GPUs per server to d, with each NIC having a bandwidth of B. In our setup, each server is equipped with eight GPUs, interconnected via a high-speed NVSwitch (900 GB/s), and eight NICs, reflecting typical configurations used in production environments. The training process for the MoE model is simulated across multiple iterations.

Simulated MoE models and parallelization strategies. We simulate the training process of four MoE models: Mixtral 8×22B [14], Mixtral 8×7B [16], Qwen-MoE [31] and LLaMA-MoE [83]. For Mixtral 8×22B, we use a hybrid parallelism that combines an EP degree of 8, TP degree of 8, PP degree of 8 at a sequence length of 4096 and micro-batch size of 16. For other models, we reuse the same configurations in Table 1.

Simulated GPU interconnect fabrics. We compare the performance of mFabric with the following interconnects:

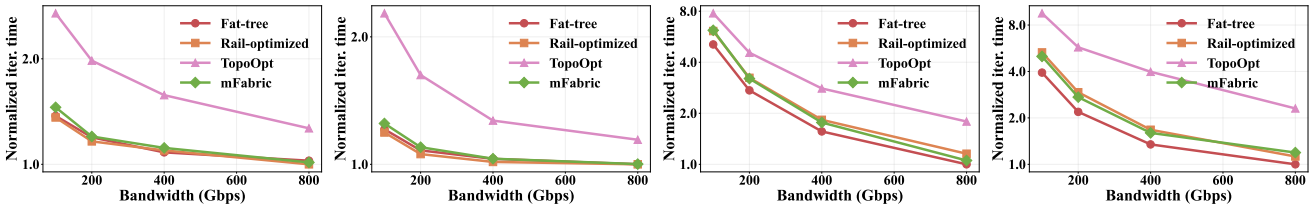
- mFabric **(this work)** In mFabric, each server connects two NICs to the EPS fabric using a fat-tree topology and connects the remaining six NICs to the OCS fabric by default. Following the architecture of the regionally reconfigurable high-bandwidth domain in mFabric, the optical circuit switch only needs to connect the GPUs within a single EP group, which is a maximum of 64 GPUs in our configuration. This can be easily supported by commodity OCS technologies (Table 2). mFabric blocks the network for 25 ms during the reconfiguration of the OCS for the first all-to-all communication in the forward pass and hides the reconfiguration time during computation for subsequent all-to-all communications, as discussed in §5.1.
- **Fat-tree.** We simulate a 1:1 non-blocking *Fat-tree* interconnect to investigate the performance gap between an ideal fabric and mFabric.
- **Rail-optimized [7].** The rail-optimized topology has been the state-of-the-art electronic GPU interconnect used by NVIDIA. It differs from the fat-tree by connecting GPUs of the same rank to the same ToR switch, providing low latency for GPUs within the same rail.
- **TopoOpt [80].** The state-of-the-art optical interconnect that co-optimizes both model parallelization and network topology to minimize communication overhead. For TopoOpt, all NICs are *optimistically* connected via a large and flat optical patch panel with an unlimited number of ports.

[6] `htsim` has been used by many datacenter proposals, e.g., , NDP [51], Opera [64] and TopoOpt [80].

7.2 Networking Cost Analysis

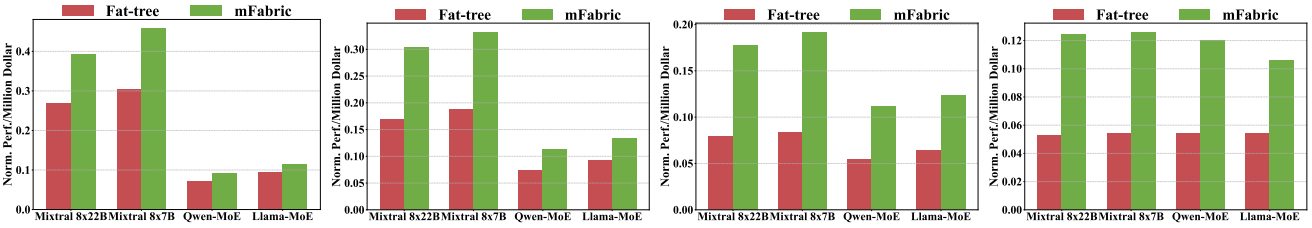
We present the cost analysis of mFabric in Figure 14, using a common production setup where each server contains 8 GPUs, following the same methodology as [80]. The networking cost is analyzed with link bandwidths from 100 Gbps to 800 Gbps across different cluster sizes. It is important to note that we only account for the number of *actually used* switch ports in calculating the cost, as the cluster may not fit perfectly within a fat-tree/rail-optimized topology with a reasonable K. Further details regarding the cost of each networking component can be found in Appendix C.

We make the following observations. First, compared to the non-blocking fat-tree and rail-optimized topologies, mFabric reduces networking costs by an average of 1.9×, as it organizes its high-bandwidth domain using OCS interconnects, which is significantly cheaper than EPS fabrics at high link bandwidth. Specifically, as shown in Figure 14c, mFabric's OCS fabric incurs 2.3× lower cost on average than fat-tree topology at 400 Gbps. Second, we acknowledge that mFabric incurs slightly higher expenses than the *optimistic* TopoOpt at the cluster size of 128 servers for two reasons: (1) mFabric requires EPS fabric to maintain global network-wide connectivity, and (2) mFabric's high-bandwidth domains assume millisecond-level reconfigurable OCS to adapt to runtime MoE traffic, which is more expensive than TopoOpt's slowly reconfigurable patch panel. However, TopoOpt requires a multi-tier patch panel fabric to form a network capable of interconnecting more than 1K



(a) Mixtral 8×22B (b) Mixtral 8×7B (c) Qwen-MoE (d) LLaMA-MoE

Figure 15: [Simulation] Training speed ups in a cluster of 128 servers with 1024 GPUs.



(a) 100G (b) 200G (c) 400G (d) 800G

Figure 16: [Simulation] Performance-cost comparison of different interconnects.

GPUs. Achieving this necessitates extensive patch panel ports and expensive long-reach transceivers to compensate for the insertion loss of optical signals across multiple switching layers. As a result, it remains unclear whether TopoOpt is able to interconnect such large clusters and maintain its cost-efficiency.

Additionally, we note that the mFabric fabric incurs lower energy consumption compared to rail-optimized and fat-tree topologies, as its high-bandwidth domain utilizes passive optical switches.

7.3 Performance: Training Speed Ups

This section compares the end-to-end training iteration time of mFabric against other interconnects across four MoE models on the cluster with 128 servers and 1024 GPUs.

Figure 15a compares the training iteration time of various interconnects for the Mixtral 8×22B model. We observe that mFabric achieves performance very close to the ideal Fat-tree and Rail-optimized topologies. mFabric's high performance is attributed to its efficient bandwidth allocation. The Mixtral 8×22B model employs a TP degree of 8, where all GPUs within each server are assigned to a single expert. In this scenario, mFabric can nearly always schedule direct optical circuits for all server pairs with high traffic demands during all-to-all communication (24 optical circuits over 8 EP participants), providing high bandwidth, low latency, and avoiding congestion. Compared to TopoOpt, mFabric reduces iteration time by an average of 1.5×, as TopoOpt's static topology cannot adapt to varying all-to-all traffic demands in real-time, leading to longer iteration times. Figure 15b compares the performance for Mixtral 8×7B model, exhibiting a similar trend to Mixtral 8×22B. Compared to TopoOpt, mFabric reduces the iteration time by 1.4× on average. Besides, we observe that these two Mixtral models show similar trends as link bandwidth increases. This can be explained by two factors. First, with a micro-batch size of 8, both models are predominantly computation-bound, as shown in Figure 3. As a result, increasing link bandwidth yields diminishing returns on overall training speed. Second, at higher link bandwidths, communication overhead becomes a smaller fraction of the total iteration time, naturally reducing the performance gap between mFabric and other interconnects.

Figure 15c and Figure 15d compare mFabric with other interconnects for Qwen-MoE and LLaMA-MoE, which differ from Mixtral models by employing larger EP degrees across more experts. We observe that mFabric is comparable to Rail-optimized and Fat-tree topologies and outperforms TopoOpt. For example, mFabric reduces the training iteration time by 2.1× on average compared to TopoOpt in LLaMA-MoE model. Compared to Mixtral models, it is interesting that mFabric exhibits larger performance gaps compared to ideal Fat-tree topology. The reason is that, as the number of experts increases (16 in LLaMA-MoE compared to 8 in Mixtral 8×22B), more bandwidth-intensive GPU pairs require more direct optical circuits, which slightly exceeds current mFabric's fanout (6 optical circuits over 16 EP participants). Unlike Mixtral models, Qwen-MoE and LLaMA-MoE are communication-bound. This is because these models not only have smaller model sizes (lower computation load), but also employ a larger number of experts and higher EP degrees (higher communication demand), leading to more intensive communication patterns. Consequently, these models show more performance improvements when link bandwidth increases.

7.4 Cost Efficiency: Performance per Dollar

To investigate the cost-efficiency of mFabric, we present the performance per dollar comparison of different interconnects in Figure 16. We observe that mFabric significantly outperforms Fat-tree

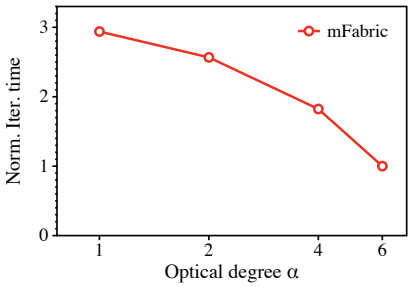


Figure 17: [Simulation] Impact of optical degree α in mFabric

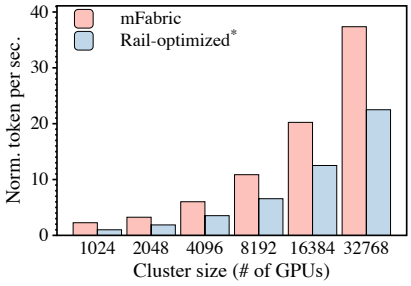


Figure 18: [Simulation] Performance of mFabric with different cluster sizes.

across all four evaluated models. At 100 Gbps link bandwidth, mFabric achieves 1.2× to 1.5× higher cost-efficiency compared to Fat-tree, with Mixtral 8×7B showing the highest improvement. For 200 Gbps link bandwidth, mFabric achieves 1.4× to 1.8× higher cost-efficiency compared to Fat-tree. The improvement becomes more pronounced in commonly-deployed 400 Gbps networks, where mFabric demonstrates even higher cost-efficiency gains: 2.3× for Mixtral 8×7B, 2.2× for Mixtral 8×22B, and 2.0× for Qwen-MoE.

Notably, the cost-efficiency advantage is consistent across different link bandwidths. Even at forward-looking 800 Gbps networks, mFabric maintains advantages from 2.0× to 2.4×. This superior cost-efficiency can be attributed to two key factors. First, mFabric's architecture efficiently provisions network resources by directly interconnecting regional GPU pairs with high communication demands through optical circuits, eliminating the need for excessive electrical switches and transceivers required in Fat-tree. Second, while Fat-tree fabric maintains full-bisection bandwidth with uniform connectivity, much of this bandwidth remains underutilized in MoE training due to its sparse and non-uniform communication patterns. In contrast, mFabric allocates bandwidth resources based on actual communication demands, significantly reducing hardware costs while maintaining high training performance.

7.5 Design Space Exploration

Impact of optical degree. We show the impact of the optical degree on mFabric's performance in Figure 17. We evaluate the Mixtral 8×22B model on a cluster of 128 servers with 100 Gbps link bandwidth. The optical degree α in mFabric is varied to adjust its connectivity in the OCS. We reduce the bandwidth of each

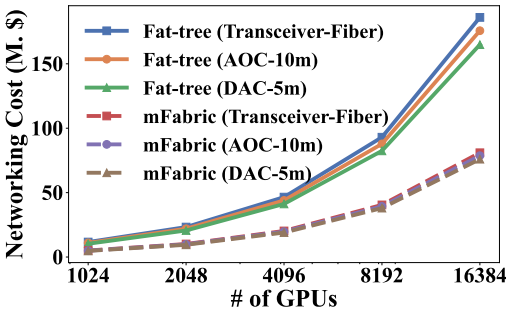


Figure 19: [Simulation] Cost comparison of different EPS links at 400 Gbps bandwidth.

electronic port when increasing their number, to ensure a cost-equivalent comparison. Our findings show that, as the optical degree increases, mFabric further reduces iteration time, as more communication-intensive GPU pairs can be provisioned with dedicated high-bandwidth optical circuits.

Scalability. We demonstrate the scalability of mFabric in Figure 18. The Mixtral 8×7B model is evaluated at 100 Gbps bandwidth, with the cluster size varying from 128 servers to 4,096 servers, covering up to 32768 GPUs. mFabric demonstrates scalability by fundamentally relaxing the stability limits of OCS through the introduction of several separate regionally reconfigurable domains, allowing mFabric to scale similarly to a fat-tree topology. We observe that mFabric scales effectively with the number of servers and outperforms the cost-equivalent rail-optimized topology by an average of 1.8× in terms of training tokens per second.

Different EPS link options. The OCS portion of mFabric requires optical transceivers with pluggable fibers to allow optical switching. For the EPS part of mFabric, especially short-reach rack-scale links between the servers and ToR switches, Direct Attach Copper (DAC) cables or Active Optical Cables (AOC) are more cost-effective alternatives to optical transceivers plus fibers (typically used for long-reach links). We analyze the cost implications of these link options in Figure 19. The results show that replacing the EPS links with DAC or AOC slightly reduce the costs for both fat-tree interconnect and mFabric. Most importantly, the cost effectiveness of mFabric is orthogonal to the choices of EPS links, and maintains significant cost advantages over fat-tree topology. For example, with 400 Gbps DAC cables option in a 4096-GPU cluster, mFabric achieves 2.2× lower total cost compared to fat-tree topology.

8 RELATED WORK

Networks for distributed training. MegaScale [54] uses a Clos-based topology for training LLMs on more than 10,000 GPUs. Alibaba HPN [70] introduced a dual-plane network to enhance resilience to failure. NVIDIA developed a rail-optimized network [7] to fully leverage the heterogeneous networking capabilities of different fabrics, which has been widely adopted in its computing clusters. [79] further proposed a rail-only design that removes the core switching layer for inter-rail GPUs, albeit at the cost of degrading cross-rail traffic performance. Meta [46] shared its insights

on the tuning of routing strategies, optimizing collective operations, and strengthening network resilience to design large-scale RoCE networks for AI training. TopoOpt [80] recently proposed optimizing the network topology for machine learning jobs using reconfigurable networks. To the best of our knowledge, mFabric is the first to propose an optically reconfigurable network for large-scale MoE training, based on a first-principles analysis of the traffic demands of MoE training.

Reconfigurable data center networks. There has been a decades-long research agenda focused on designing reconfigurable networks for data centers [35, 36, 38, 39, 43, 50, 64, 66, 68, 78, 80, 81]. These proposals target generic data center networks, which are not optimized to provide cost-efficient solutions for large-scale MoE training. In particular, traffic-oblivious solutions, such as RotorNet [66] and Opera [64], result in suboptimal performance for MoE training, as they cannot deliver timely transfers for bandwidth-intensive all-to-all traffic. Meanwhile, the hardware innovations like RotorNet [65, 66] and Sirius [36] features faster optical switching latency, hence allowing mFabric to achieve much faster topology reconfiguration delay. Shoal [75] proposes the use of rapidly reconfigurable electronic circuit switches, but its design focuses on rack-scale networks, making it unsuitable for large-scale MoE training.

OCS deployments in data centers. Google has pioneered the deployments of OCS technology in production data centers. In particular, they have advanced its data center networking technologies, transitioning from electronic packet switching (EPS) to hybrid and fully optical solutions [34, 52, 61, 69, 76]. Early designs like Jupiter [76] utilized Clos topologies with EPS to achieve scalability and high bandwidth but faced limitations in power efficiency and adaptability to dynamic workloads. Subsequent innovations in Jupiter Evolving [69] introduced optical circuit switches (OCS) to complement EPS, creating a hybrid architecture that improved cost and power efficiency. Most recently, Lightwave Fabrics [61] incorporated reconfigurable MEMS-based OCS to enable millisecond-scale topology reconfiguration in data centers and TPU supercomputers, achieving performance gains in machine learning (ML) workloads through dynamic bandwidth allocation. While TPUv4's reconfigurable fabric focuses on cube-level optimization, mFabric specifically explores server-scale optical interconnects for MoE training, delivering fine-grained bandwidth management for sparse and non-uniform communication patterns.

Emerging OCS hardware devices and systems. There are recent proposals on designing novel OCS hardware at server scale for chip-to-chip interconnects [13, 37, 72, 82]. However, these emerging devices requires system-level design to be practical. Similar as other system-level work [57], mFabric's regional OCS and its algorithmic designs are compatible with this vibrant line of exploration on novel OCS hardware.

9 CONCLUSION

This paper presented mFabric, a novel cost-effective reconfigurable fabric for large-scale MoE training. At the core of mFabric is the design and implementation of regionally reconfigurable high-bandwidth domain. Through small-scale prototype and large-scale packet simulations, evaluation results show mFabric outperforms state-of-the-art electrical and optical interconnects.

REFERENCES

[1] [n. d.]. 100GBASE-SR4 850nm 100m DOM MPO-12/UPC MMF Optical Transceiver Module. https://www.fs.com/products/48354.html. ([n. d.]).
[2] [n. d.]. 200GBASE-SR4 850nm 100m DOM MPO-12/UPC MMF Optical Transceiver Module. https://www.fs.com/products/139696.html. ([n. d.]).
[3] [n. d.]. 400GBASE-SR4 PAM4 850nm 100m DOM MPO-12/APC MMF Optical Transceiver Module. https://www.fs.com/products/226577.html. ([n. d.]).
[4] [n. d.]. All-to-all traffic demand collection in Megatron-LM. https://github.com/NVIDIA/Megatron-LM/blob/461b06cd6d1fb4a625cebdbca499dac9484087fc/megatron/core/transformer/moe/token_dispatcher.py#L432. ([n. d.]).
[5] [n. d.]. Calient Optical Circuit Switch. www.calient.net. ([n. d.]).
[6] [n. d.]. DeepSeek-V3. https://github.com/deepseek-ai/DeepSeek-V3/. ([n. d.]).
[7] [n. d.]. Doubling all2all Performance with NVIDIA Collective Communication Library 2.12. https://developer.nvidia.com/blog/doubling-all2all-performance-with-nvidia-collective-communication-library-2-12/. ([n. d.]).
[8] [n. d.]. FlexFlow. https://github.com/flexflow/FlexFlow. ([n. d.]).
[9] [n. d.]. Generic Compatible 800GBASE-SR8 QSFP-DD PAM4 850nm 50m DOM MPO-16/APC MMF Optical Transceiver Module. https://www.fs.com/products/200921.html?attribute=93760&id=3569240. ([n. d.]).
[10] [n. d.]. Grok. https://x.ai/blog/grok-1.5v. ([n. d.]).
[11] [n. d.]. htsim simulator. https://github.com/nets-cs-pub-ro/NDP/wiki/NDP-Simulator. ([n. d.]).
[12] [n. d.]. LC UPC to LC UPC, Duplex, 2 Fibers. https://www.fs.com/products/40191.html. ([n. d.]).
[13] [n. d.]. LightMatter. https://lightmatter.co/products/passage/. ([n. d.]).
[14] [n. d.]. Mixtral 8x22B. https://huggingface.co/mistralai/Mixtral-8x22B-Instruct-v0.1. ([n. d.]).
[15] [n. d.]. Mixtral-8x7B-Instruct-v0.1. https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1. ([n. d.]).
[16] [n. d.]. Mixtral of experts. https://mistral.ai/news/mixtral-of-experts/. ([n. d.]).
[17] [n. d.]. MSN4700-WS2FC, NVIDIA® Mellanox Spectrum-3 Based 32-Port Ethernet L3 Data Center Switch, 32 x 400Gb QSFP-DD. https://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=4135. ([n. d.]).
[18] [n. d.]. Nano-Second Speed PLZT Photonics. http://epiphotonics.com/products.html. ([n. d.]).
[19] [n. d.]. NVIDIA Collective Communications Library (NCCL). https://developer.nvidia.com/nccl. ([n. d.]).
[20] [n. d.]. NVIDIA DGX SuperPOD. https://www.nvidia.com/en-us/data-center/dgx-superpod/. ([n. d.]).
[21] [n. d.]. NVIDIA GB200 NVL72 Delivers Trillion-Parameter LLM Training and Real-Time Inference. https://developer.nvidia.com/blog/nvidia-gb200-nvl72-delivers-trillion-parameter-llm-training-and-real-time-inference/. ([n. d.]).
[22] [n. d.]. NVIDIA Mellanox MCX515A-CCAT ConnectX®-5 EN Network Interface Card, 100GbE Single-Port QSFP28. https://www.fs.com/products/119648.html. ([n. d.]).
[23] [n. d.]. NVIDIA Mellanox MCX653105A-HDAT ConnectX®-6 InfiniBand/VPI Adapter Card 200GbE/HDR, Single-Port QSFP56. https://www.fs.com/products/168437.html. ([n. d.]).
[24] [n. d.]. NVIDIA Mellanox MCX75310AAS-NEAT ConnectX®-7 InfiniBand/VPI Adapter Card 400GbE/NDR, Single-Port OSFP. https://www.fs.com/products/212161.html. ([n. d.]).
[25] [n. d.]. NVIDIA Mellanox Spectrum-4 SN5600 800G 64-Port 51.2Tb/s 2U Data Center Switch. https://fireowls.com/products/920-9n42f-00ri-7c0-nvidia-spectrum-sn5600-ethernet-switch?srsltid=AfmBOoqC5-OQefz1MpAi_2QkzW2tnaCTXA_xGtgoj9b3NphWUXw8dBrl. ([n. d.]).
[26] [n. d.]. NVIDIA NVLink and NVLink Switch. https://www.nvidia.com/en-us/data-center/nvlink/. ([n. d.]).
[27] [n. d.]. NVIDIA NVSwitch Technical Overview. https://images.nvidia.com/content/pdf/nvswitch-technical-overview.pdf. ([n. d.]).
[28] [n. d.]. NVIDIA Spectrum SN3700. https://marketplace.nvidia.com/en-us/enterprise/networking/sn3700/. ([n. d.]).
[29] [n. d.]. Polatis Optical Circuit Switch. https://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuitswitch-sdn-enabled.asp. ([n. d.]).
[30] [n. d.]. Polatis Optical Switches. http://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuit-switch-sdn-enabled.asp. ([n. d.]).
[31] [n. d.]. Qwen1.5-MoE-A2.7B. https://huggingface.co/Qwen/Qwen1.5-MoE-A2.7B. ([n. d.]).
[32] [n. d.]. Telescent G4 Network Topology Manager. https://www.telescent.com/products. ([n. d.]).
[33] [n. d.]. XVERSE-MoE-A36B MoE base model. https://huggingface.co/xverse/XVERSE-MoE-A36B. ([n. d.]).
[34] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM New York, NY, USA, 63–74.
[35] Daniel Amir, Nitika Saran, Tegan Wilson, Robert Kleinberg, Vishal Shrivastav, and Hakim Weatherspoon. 2024. Shale: A Practical, Scalable Oblivious Reconfigurable

Network. In *Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24)*. https://doi.org/10.1145/3651890.3672248

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

[37] Darius Bunandar, Shashank Gupta, Jessie Rosenberg, Clifford Chao, Kuang Liu, and Nicholas C Harris. 2024. Optical communication substrate using glass interposer. (Oct. 24 2024). US Patent App. 18/638,820.

[38] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. 2012. OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX Association.

[39] Li Chen, Kai Chen, Zhonghua Zhu, Minlan Yu, George Porter, Chunming Qiao, and Shan Zhong. 2017. Enabling Wide-Spread Communications on Optical Fabric with MegaSwitch. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 577–593. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/chen

[40] Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277* (2022).

[41] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. 2024. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. (2024). arXiv:cs.CL/2405.04434 https://arxiv.org/abs/2405.04434

[42] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[43] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference (SIGCOMM '10)*. Association for Computing Machinery. https://doi.org/10.1145/1851182.1851223

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

[45] Klaus-Tycho Foerster, Manya Ghobadi, and Stefan Schmid. 2018. Characterizing the algorithmic complexity of reconfigurable data center architectures. In *Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems*. 89–96.

[46] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, et al. 2024. RDMA over Ethernet for Distributed Training at Meta Scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 57–70.

[47] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, Sushant Sengupta, Jennifer Rexford, et al. 2009. VL2: A scalable and flexible data center network. In *ACM SIGCOMM Computer Communication Review*, Vol. 39. ACM New York, NY, USA, 51–62.

[48] Chuanxiong Guo, Hui Wu, Kai Tan, Lei Shiy, Yongguang Zhang, and Songnian Lu. 2009. Bcube: A high performance, server-centric network architecture for modular data centers. In *ACM SIGCOMM Computer Communication Review*, Vol. 39. ACM New York, NY, USA, 63–74.

[49] Chuanxiong Guo, Hui Wu, Kai Tan, Lei Shiy, Yongguang Zhang, and Songnian Lu. 2010. Dcell: A scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM Computer Communication Review*, Vol. 40. ACM New York, NY, USA, 63–74.

[50] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R Das, Jon P Longtin, Himanshu Shah, and Ashish Tanwer. 2014. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 319–330.

[51] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-architecting datacenter networks and stacks for low latency and high performance. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. https://doi.org/10.1145/3098822.3098825

[52] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. 3–14. https://doi.org/10.1145/2486001.2486019

[53] Yimin Jiang, Yibo Zhu, Chang Lan, Bairen Yi, Yong Cui, and Chuanxiong Guo. 2020. A unified architecture for accelerating distributed {DNN} training in heterogeneous {GPU/CPU} clusters. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 463–479.

[54] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, et al. 2024. {MegaScale}: Scaling large language model training to more than 10,000 {GPUs}. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 745–760.

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

[56] Mehrdad Khani, Manya Ghobadi, Mohammad Alizadeh, Ziyi Zhu, Madeleine Glick, Keren Bergman, Amin Vahdat, Benjamin Klenk, and Eiman Ebrahimi. 2021. SiP-ML: high-bandwidth optical network interconnects for machine learning training. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 657–675.

[57] Abhishek Vijaya Kumar, Arjun Devraj, Darius Bunandar, and Rachee Singh. 2024. A case for server-scale photonic connectivity. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 290–299.

[58] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668* (2020).

[59] Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. 2023. Merge, then compress: Demystify efficient SMoe with hints from its routing policy. *arXiv preprint arXiv:2310.01334* (2023).

[60] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. 2020. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704* (2020).

[61] Hong Liu, Ryohei Urata, Kevin Yasumura, Xiang Zhou, Roy Bannon, Jill Berger, Pedram Dashti, Norm Jouppi, Cedric Lam, Sheng Li, Erji Mao, Daniel Nelson, George Papen, Mukarram Tariq, and Amin Vahdat. 2023. Lightwave Fabrics: At-Scale Optical Circuit Switching for Datacenter and Machine Learning Systems. In *Proceedings of the ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23)*. https://doi.org/10.1145/3603269.3604836

[62] Juncai Liu, Jessie Hui Wang, and Yimin Jiang. 2023. Janus: A unified distributed training framework for sparse mixture-of-experts models. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 486–498.

[63] Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not All Experts are Equal: Efficient Expert Pruning and Skipping for Mixture-of-Experts Large Language Models. *arXiv preprint arXiv:2402.14800* (2024).

[64] William M Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C Snoeren, and George Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 1–18.

[65] William M Mellette, Alex Forencich, Rukshani Athapathu, Alex C Snoeren, George Papen, and George Porter. 2024. Realizing RotorNet: Toward Practical Microsecond Scale Optical Networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 392–414.

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

[67] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. 2019.

PipeDream: Generalized pipeline parallelism for DNN training. In *Proceedings of the 27th ACM symposium on operating systems principles*. 1–15.
[68] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. Integrating microsecond circuit switching into the data center. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 447–458.
[69] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. 2022. Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. https://doi.org/10.1145/3544216.3544265
[70] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24)*.
[71] R Ryf, J Kim, JP Hickey, A Gnauck, D Carr, F Pardo, C Bolle, R Frahm, N Basavanhally, C Yoh, et al. 2001. 1296-port MEMS transparent optical crossconnect with 2.07 petabit/s switch capacity. In *OFC 2001. Optical Fiber Communication Conference and Exhibit. Technical Digest Postconference Edition (IEEE Cat. 01CH37171)*, Vol. 4. IEEE, PD28–PD28.
[72] Tae Joon Seok, Niels Quack, Sangyoon Han, Richard S Muller, and Ming C Wu. 2016. Large-scale broadband digital silicon photonic switches with vertical adiabatic couplers. *Optica* 3, 1 (2016), 64–70.
[73] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).
[74] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. (2020). arXiv:cs.CL/1909.08053 https://arxiv.org/abs/1909.08053
[75] Vishal Shrivastav, Asaf Valadarsky, Hitesh Ballani, Paolo Costa, Ki Suh Lee, Han Wang, Rachit Agarwal, and Hakim Weatherspoon. 2019. Shoal: A network architecture for disaggregated racks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 255–270.
[76] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. https://doi.org/10.1145/2785956.2787508
[77] Xinchen Wan, Hong Zhang, Hao Wang, Shuihai Hu, Junxue Zhang, and Kai Chen. 2020. Rat-resilient allreduce tree for distributed machine learning. In *Proceedings of the 4th Asia-Pacific Workshop on Networking*. 52–57.
[78] Guohui Wang, David G Andersen, Michael Kaminsky, Konstantina Papagiannaki, TS Eugene Ng, Michael Kozuch, and Michael Ryan. 2010. c-Through: Part-time optics in data centers. In *Proceedings of the ACM SIGCOMM 2010 Conference*. 327–338.
[79] Weiyang Wang, Manya Ghobadi, Kayvon Shakeri, Ying Zhang, and Naader Hasani. 2024. Rail-only: A Low-Cost High-Performance Network for Training LLMs with Trillion Parameters. (2024). arXiv:cs.NI/2307.12169 https://arxiv.org/abs/2307.12169
[80] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. 2023. TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 739–767. https://www.usenix.org/conference/nsdi23/presentation/wang-weiyang
[81] Yiting Xia, Mike Schlansker, TS Eugene Ng, and Jean Tourrilhes. 2015. Enabling Topological Flexibility for Data Centers Using {OmniSwitch}. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*.
[82] Xiaosheng Zhang, Ming Chiang A Wu, Andrew S Michaels, and Johannes Henriksson. 2022. Beam-steering system based on a MEMS-actuated vertical-coupler array. (Sept. 13 2022). US Patent 11,441,353.
[83] Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. LLaMA-MoE: Building Mixture-of-Experts from LLaMA with Continual Pre-training. *arXiv preprint arXiv:2406.16554* (2024). https://arxiv.org/abs/2406.16554

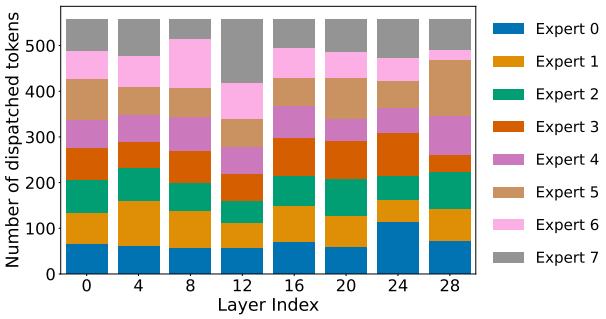


Figure 20: [Mixtral 8×7B in production] Non-uniform token distribution across MoE blocks.

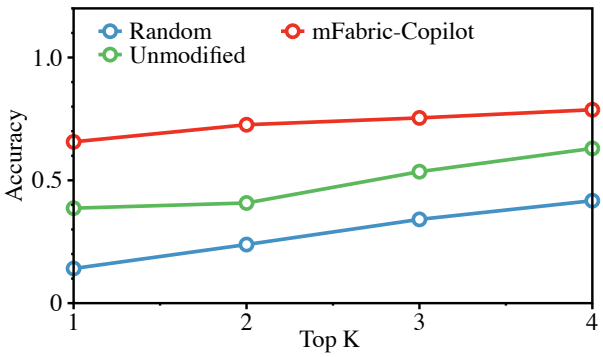


Figure 21: Average prediction accuracy of mFabric-Copilot.

Appendix

A NON-UNIFORM TOKEN DISTRIBUTION IN TRAINED MOE MODEL

We measured the all-to-all token distribution in the forward pass of pre-trained Mixtral 8×7B [15], as shown in Figure 20. We observe that the number of tokens dispatched to each expert is non-uniform and varies across different MoE blocks, which argues for a necessary mechanism to adapt to the dynamic traffic in EP even when the model has largely converged.

B TRAFFIC DEMAND PREDICTION

mFabric aims to handle the first all-to-all communication in the forward pass with a predictive approach. By default, the OCS topology for this initial communication is either randomly generated (e.g., for the first all-to-all in the first layer) or remains unchanged from the previously used topology (e.g., the first all-to-all in the second layer). The traffic demand prediction algorithm predicts the *conditional probability* of the traffic matrix, denoting the conditional probability of a token gated to expert j given that it is gated the expert i in the last layer. With the conditional probability matrix and the empirical token distribution in the previous layer, we can predict the traffic distribution in the current layer.

Matrix Estimation: For each layer, mFabric estimates the conditional probability matrix with the traffic demand records in recent

Link Bandwidth	Transceiver ($)	NIC ($)	Elec. switch port ($)	OCS port ($)	Patch panel port ($)
100 Gbps	99 [1]	659 [22]	187 [80]	520 [29]	100 [32]
200 Gbps	239 [2]	1079 [23]	374 [80]	520 [29]	100 [32]
400 Gbps	659 [3]	1499 [24]	1090 [17]	520 [29]	100 [32]
800 Gbps	1399 [9]	2248[1][24]	1400 [25]	520 [29]	100 [32]

[1] Conservatively estimated as 1.5 times the price of 400G NIC, as 800G products are not yet commercially available.

Table 4: Cost of network components.

iterations. Focusing on the recent expert load distributions, we employ a weighted average within a fixed window of traffic records in time series. For each layer, the optimization objective is to minimize the square error between the predicted load distribution and the ground truth (for simplicity, we omit the layer index here):

$$\min_{P} \quad \sum_{i=1}^{k} w_i \cdot \Sigma_i \left((Y_i - PX_i)^2 \right), \tag{1}$$

where k is the window size. The transition matrix P is of size $N \times N$, representing the conditional probability of the current layer's expert load distribution, given the expert load distribution of the previous layer. X_i, Y_i are normalized expert load distribution vectors of two neighboring layers. Each element in P is constrained to be in the range $[0, 1]$, and the sum of each column is constrained to be 1 to ensure that P is a valid probability matrix.

We employ the Sequential Least Squares Programming (SLAP) method for optimization, as it is suitable for nonlinear problems with linear constraints. The algorithm is implemented via the `scipy.optimize` library in Python. During the inference, with the load distribution in layer i given, we can predict the expert load distribution for the next layer in advance for its first all-to-all communication.

We name this method mFabric-Copilot. Figure 21 compares the prediction accuracy of mFabric-Copilot against the aforementioned methods, i.e., the randomly assigned token distribution (uniform bandwidth allocation), and the unmodified token distribution from the previous layers (unchanged topology) on collected traces from measurements. Top K accuracy measures whether mFabric is able to find the top-k activation-intensive experts. We find that mFabric-Copilot exhibits significantly higher accuracy than other counterparts, which implies that mFabric-Copilot can find the most intensive pairs in of the all-to-all communication with high probability. Therefore, mFabric-Copilot offers an opportunity to proactively reconfigure the topology for the FP's first all-to-all in advance.

C COST OF NETWORK COMPONENTS

Table 4 lists the costs of network components used in §7.2. We reuse the prices for electronic switches at 100G, 200G as well as for NICs, OCS ports, and patch panel ports from TopoOpt [80], and we add the prices of transceivers, NICs, and electronic switch ports for 400 Gbps and 800 Gbps link accordingly. We also follow the same methodology as in TopoOpt when calculating the fiber costs.