Network Support for Machine Learning

A Foundational Component of ML Infrastructure

Topic Outline (~3 Lectures)

- **Background**
- **Topology**: TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs, NSDI 2023
- **Transports**: Towards Domain-Specific Network Transport for Distributed DNN Training, NSDI 2024
- **Deployment**: MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs, NSDI 2024 (Experience Paper)

Topic Outline

- This outline is similar to DCN topic
  - LLM is ALSO a workload running in DCN, so similar problems arise
    - Topology
    - Transport
    - …
  - LLM is different because it is **TOO IMPORTANT** nowadays so that it is worthwhile to **co-design DCN with LLM training/inference** from scratch

Background

- Modern ML (especially LLMs) requires **distributed training/inference**
- **Scaling = more GPUs + more communication**

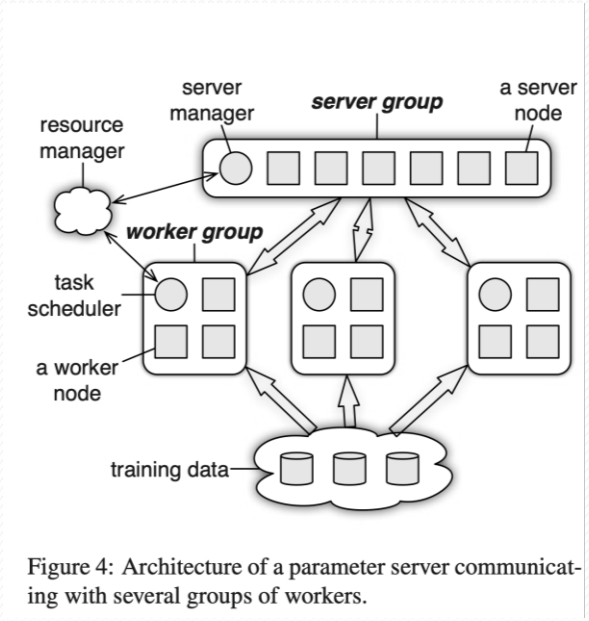
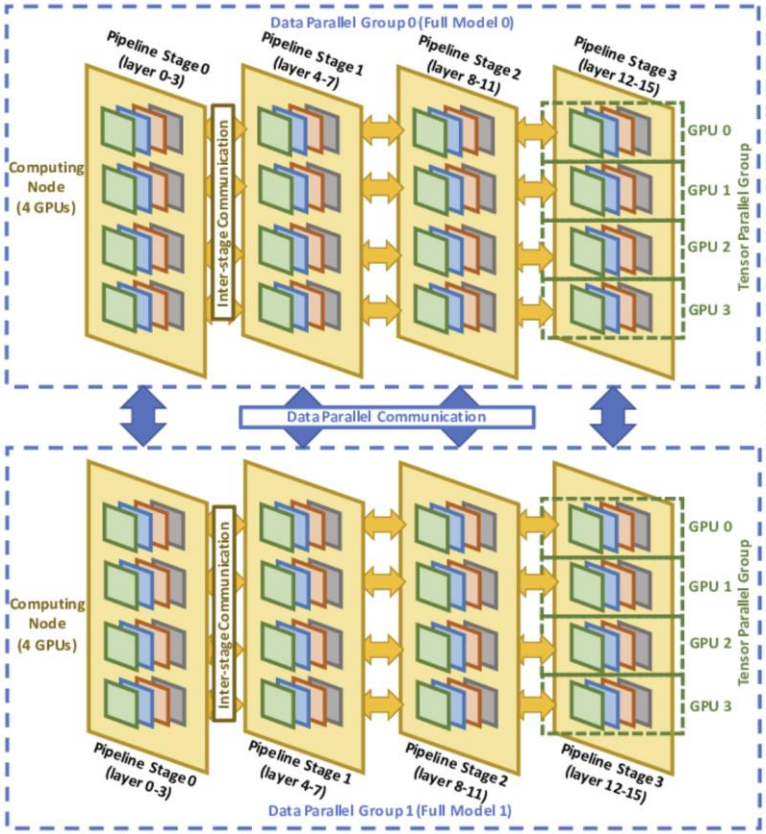
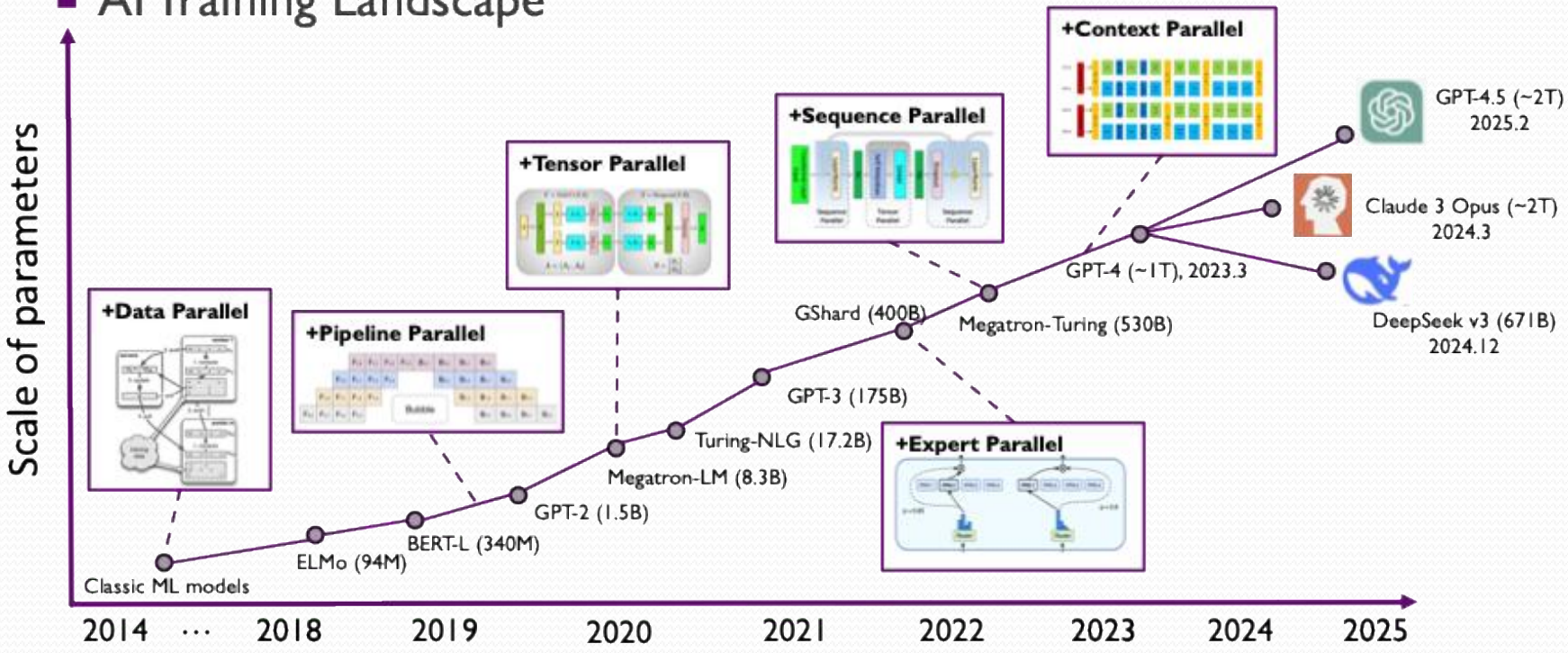


Figure 4: Architecture of a parameter server communicating with several groups of workers.



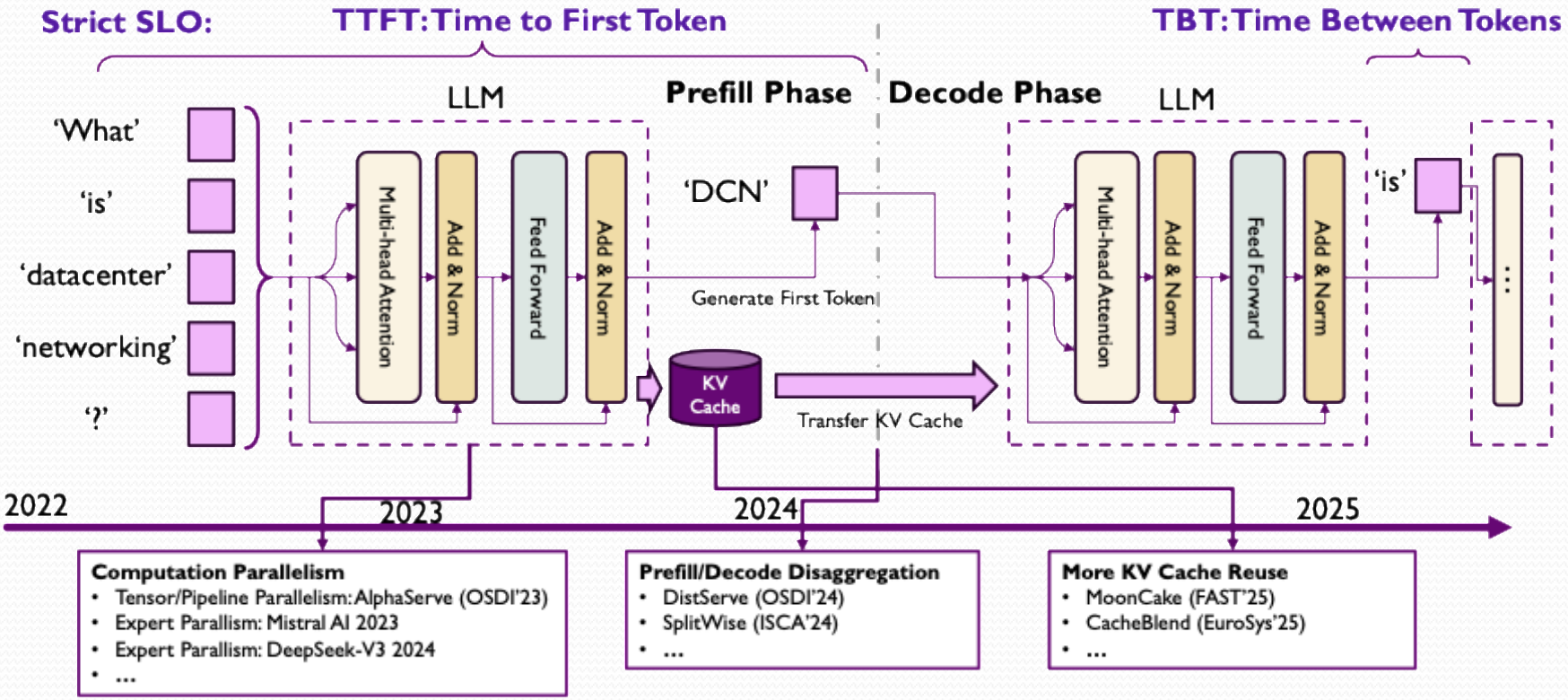
Background

- AI Training Landscape



Background

- Emerging AI inference/serving workflow



Why Network Matters?

- **Training**
  - Communication-intensive operations:
  - AllReduce
  - Pipeline/model parallelism
  - Checkpoint synchronization
- **At scale**
  - Network overhead can dominate training time

Why Network Matters?

- **LLM inference is increasingly distributed**
  - Tensor parallel serving
  - MoE expert routing
  - KV-cache disaggregation
- **Requirements:**
  - Ultra-low latency
  - High throughput
  - Predictable tail performance

Traditional Assumption is Breaking

Traditional DC Networks	ML Workloads
Short, bursty flows	Large, structured flows
Unpredictable traffic	Highly predictable
General-purpose design	Domain-specific needs

Co-Design ML Systems with Networking

- Instead of:
  - Fix network → optimize ML
- We should:
  - **Co-design Compute + Network + ML System**
  - This leads to many new key research questions, examples:
    - **Topology** — What network structure fits ML?
    - **Transport** — How should data be moved?
    - **Deployment** — How does it work at scale?

More Reading Materials

- LLM Training
  - Alibaba HPC: A Data Center Network for Large Language Model Training, **SIGCOMM 2024**
  - ByteScale: Communication-Efficient Scaling of LLM Training with a 2048K Context Length on 16384 GPUs, **SIGCOMM 2025**
- LLM Inference
  - Efficient Memory Management for Large Language Model Serving with PagedAttention, **SOSP 2023**
  - Distserve: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving, **OSDI 2024**

TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs

Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, Anthony Kewitsch

NSDI 2023

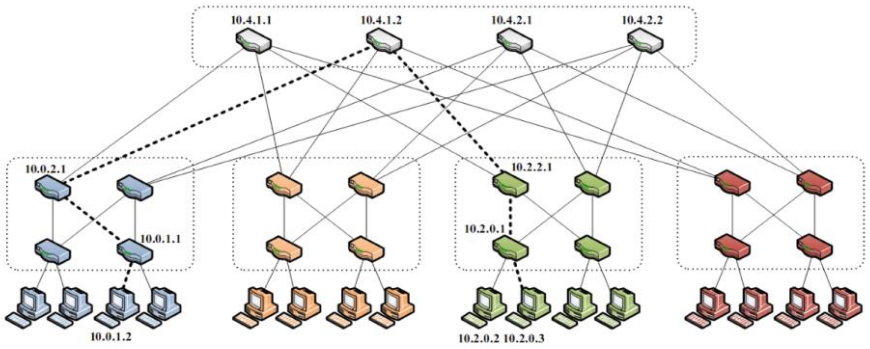
Outline

- **Background and Motivation**
- TopoOpt Design
- Implementation and Evaluation
- Review

Background

- **Fat-Tree network topology**
  - Fat-Trees provide uniform bandwidth and latency between server pairs
  - Ideal when the workload is **unpredictable** and consists mostly of short transfers

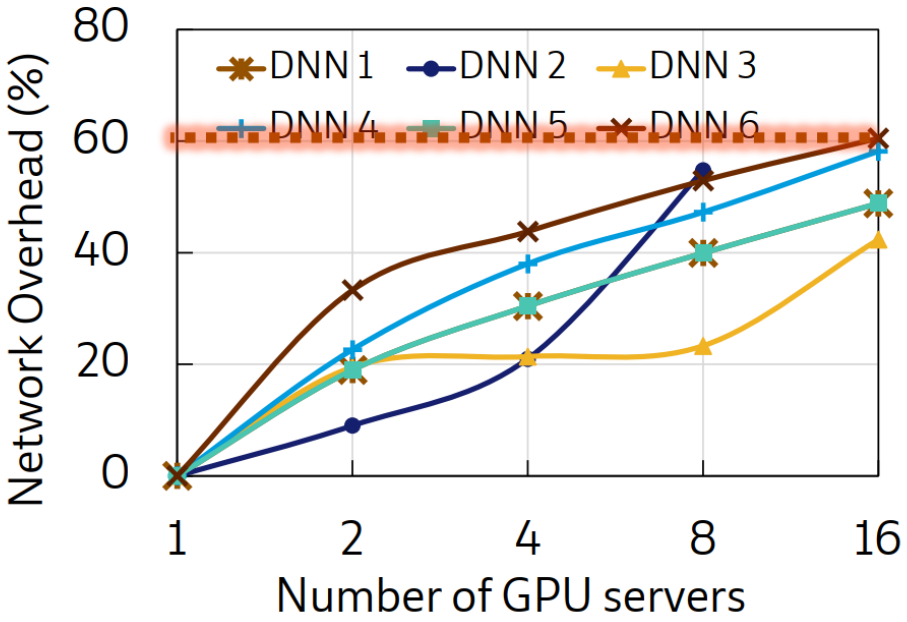
Fat-Tree networks are not the best network topology for DNN training !



A Scalable, Commodity Data Center Network Architecture
Mohammad Al-Fares et al., SIGCOMM '08

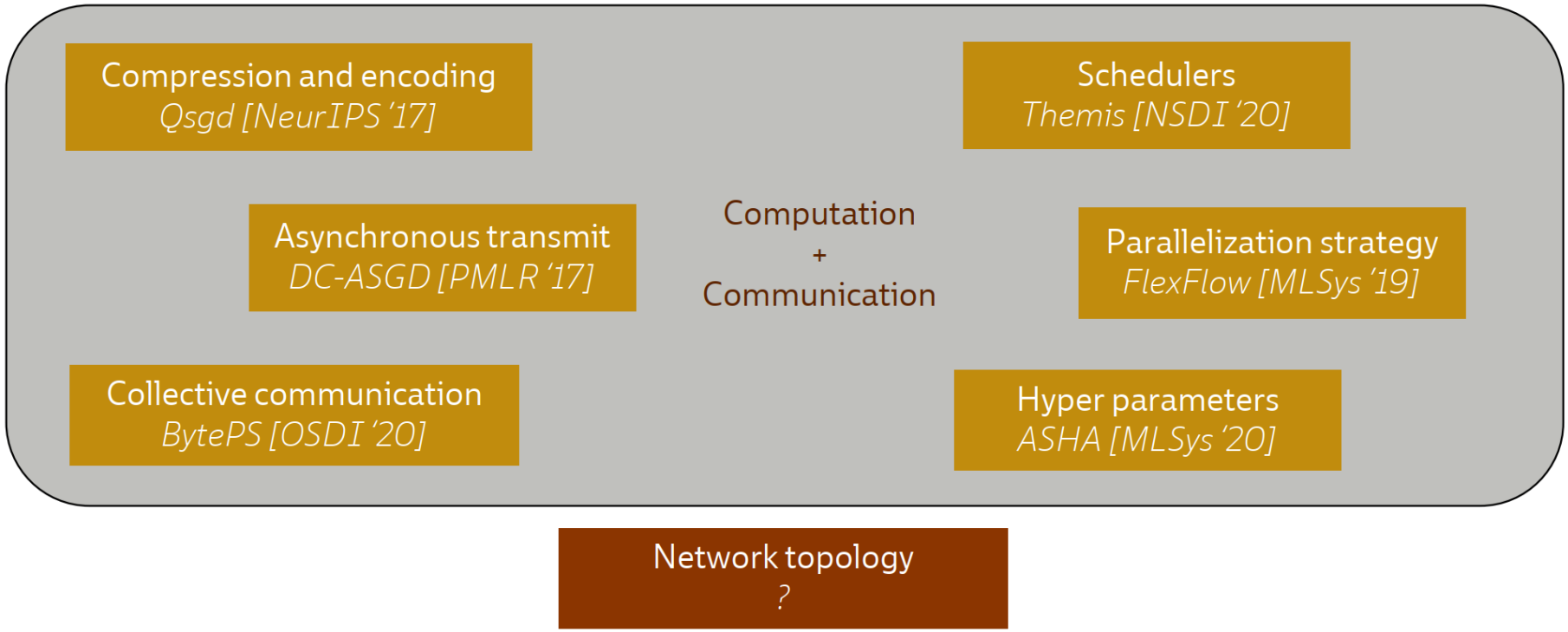
Motivation

- Fat-Tree based DNN training infrastructures are facing a **network bottleneck**
  - Network Bottleneck: the amount of time spent on communication only



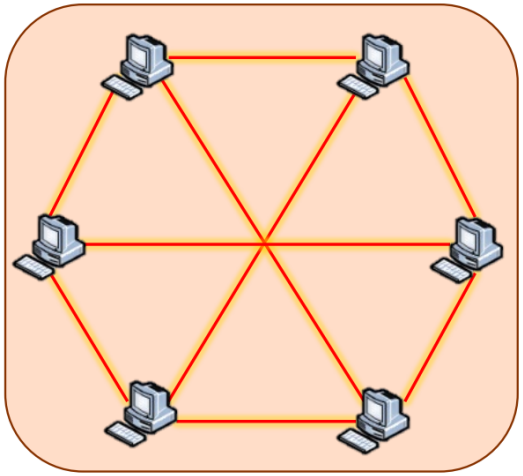
Motivation

- Previous work on distributed DNN training optimization **does NOT consider physical topology**

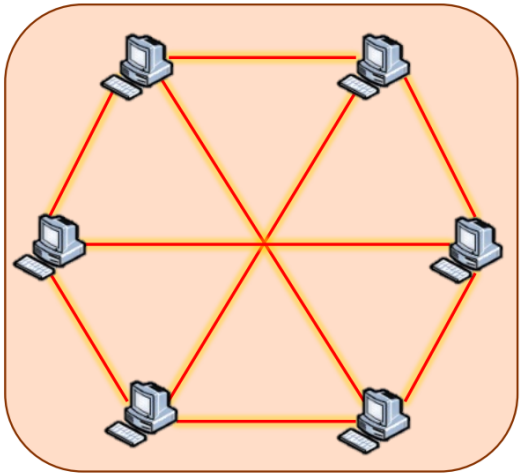


Motivation

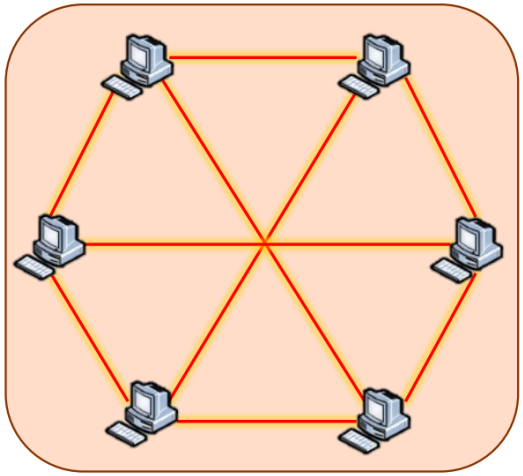
- Can we achieve topology reconfiguration?



Topology A



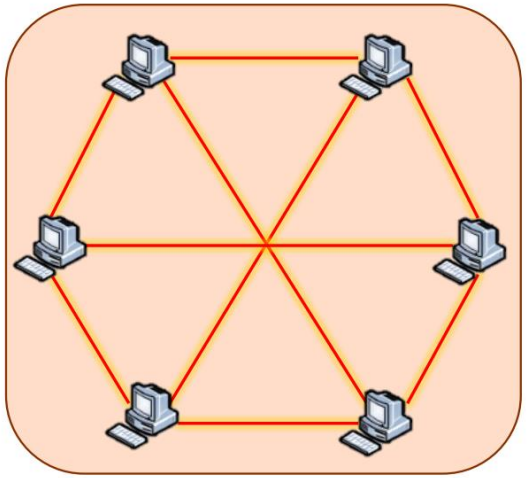
Topology A



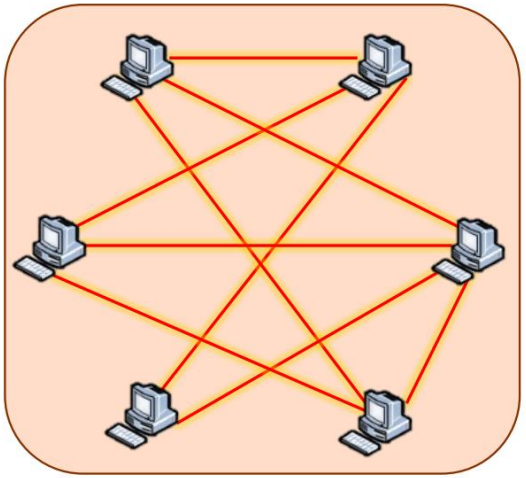
Topology A

Motivation

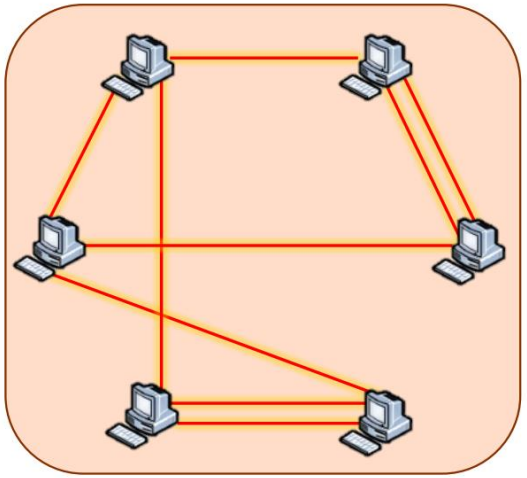
- What if **different topologies** for different training workloads?



Topology A



Topology B



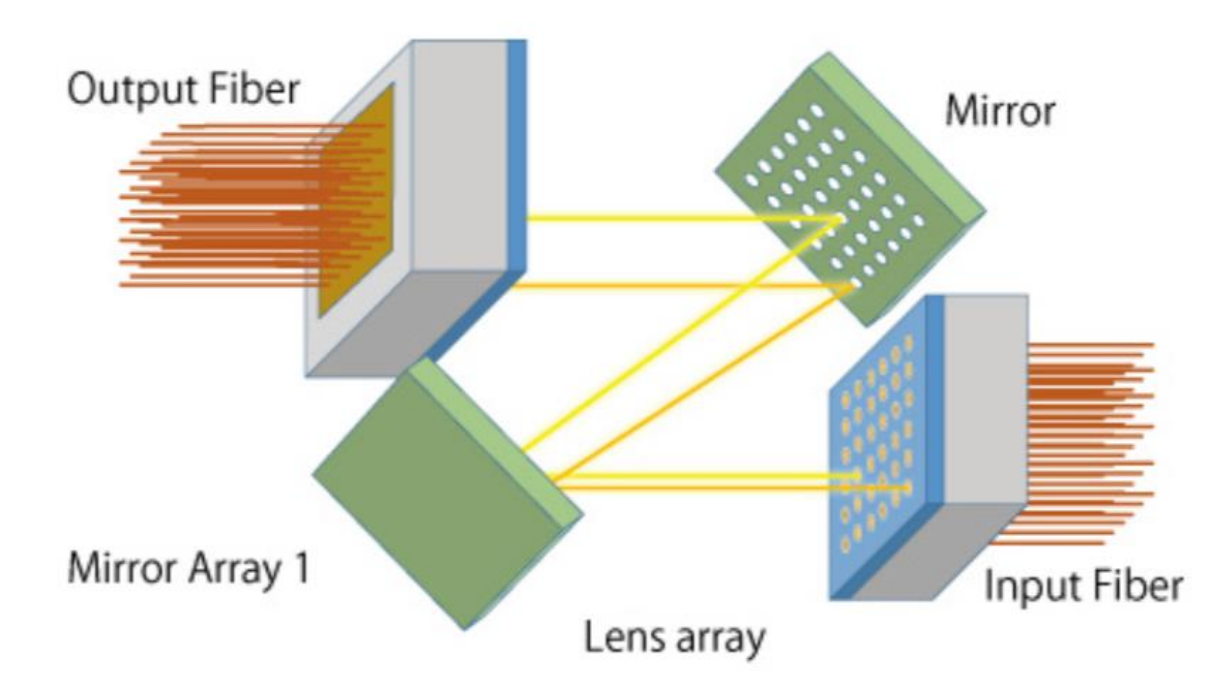
Topology C

Motivation

- How can we achieve **topology reconfiguration**?
- Answer: **use Optical Switching (OCS)**

Background

- Optical Switch (OCS)



Background

- **Optical Switch (OCS)**
  - Dynamic high-bandwidth optical connections
  - Reconfigurable topology
  - Direct GPU-to-GPU connectivity
- However, **reconfiguration has COST**

Technology	Reconfiguration Latency
Patch panels	minutes
3D MEMS OCS (Current practice)	**~10 ms**
Emerging photonic switches	µs–ns

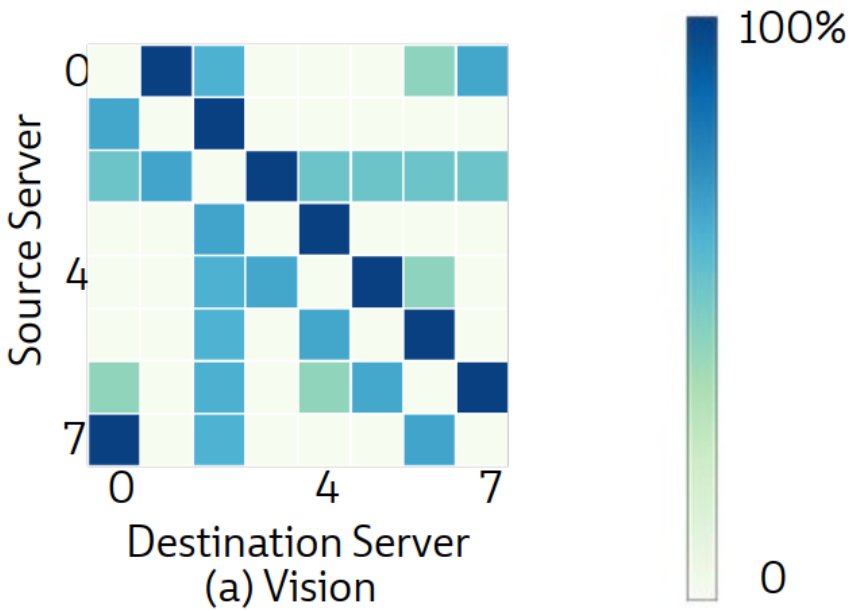
Background

Electrical Packet Switching	Optical Circuit Switching
Flexible packet forwarding	Dedicated optical paths
Multi-hop communication	Near single-hop communication
Static topology	Dynamically **reconfigurable** (with overhead)
More transceivers (high cost/low reliability)	Fewer transceivers

If traffic pattern is **highly predictable**, OCS is a good choice!

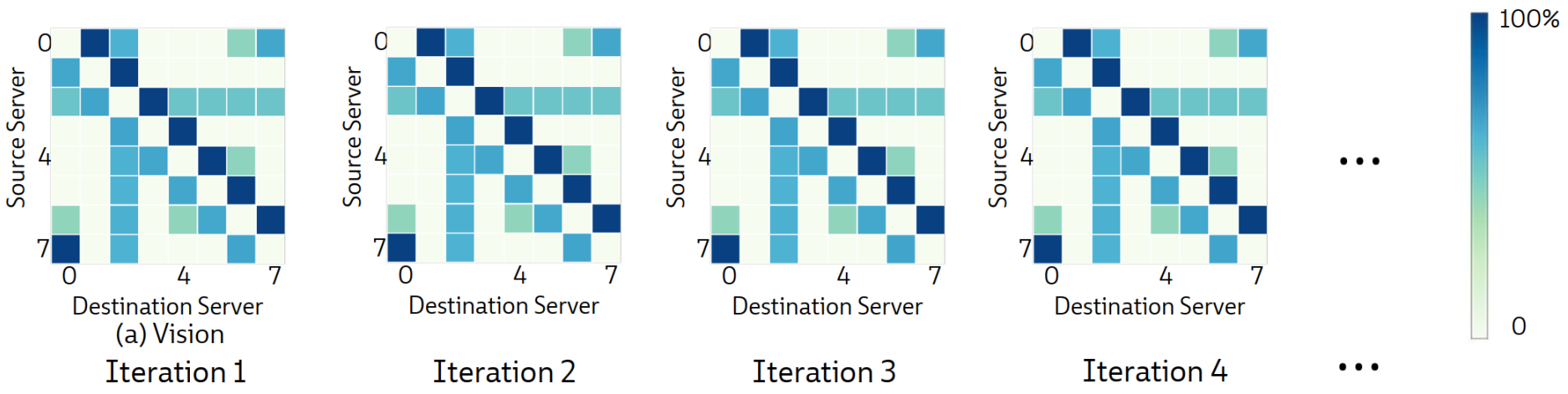
Motivation

- How to deal with **reconfiguration overhead**?



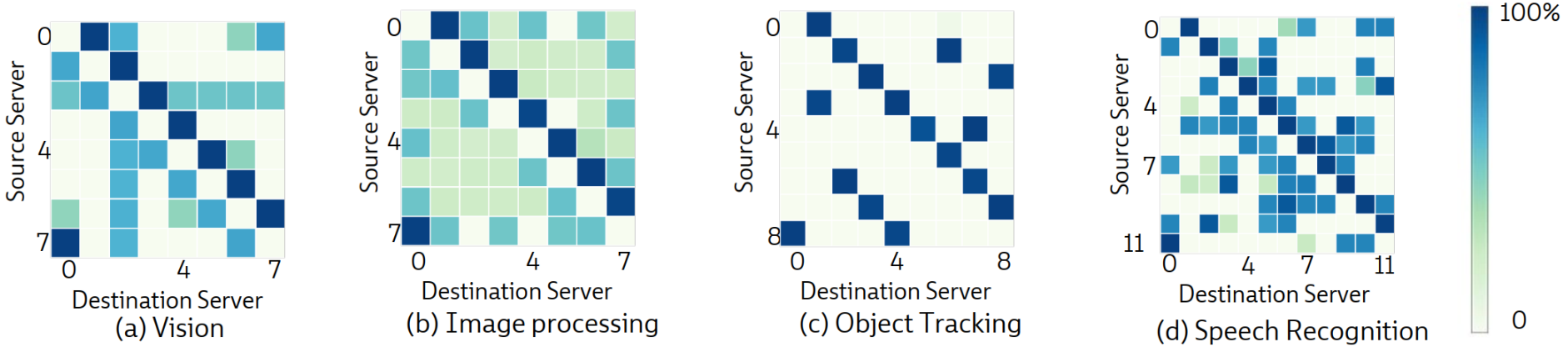
Motivation

- **Observation1:** Traffic patterns are **predictable**, and **do not change across training iterations**



Motivation

- **Observation2:** Traffic patterns are **model-dependent**
  - Depend on the **parallelization strategy** and **device placement** of a training job



Outline

- Background and Motivation
- **TopoOpt Design**
- Implementation and Evaluation
- Review

TopoOpt

- The first system to leverage reconfigurable network, to **co-optimize network topology and parallelization strategy** for distributed training
- TopoOpt achieves 3.4x faster training time for DNN training

TopoOpt

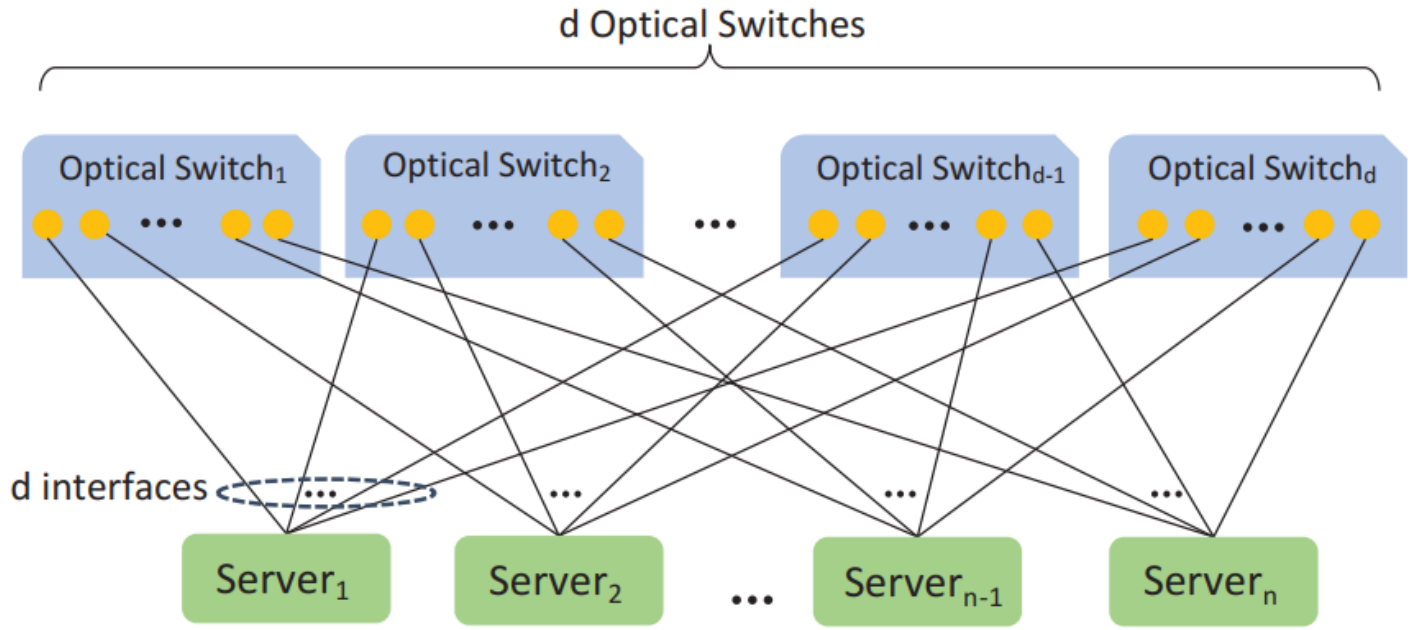


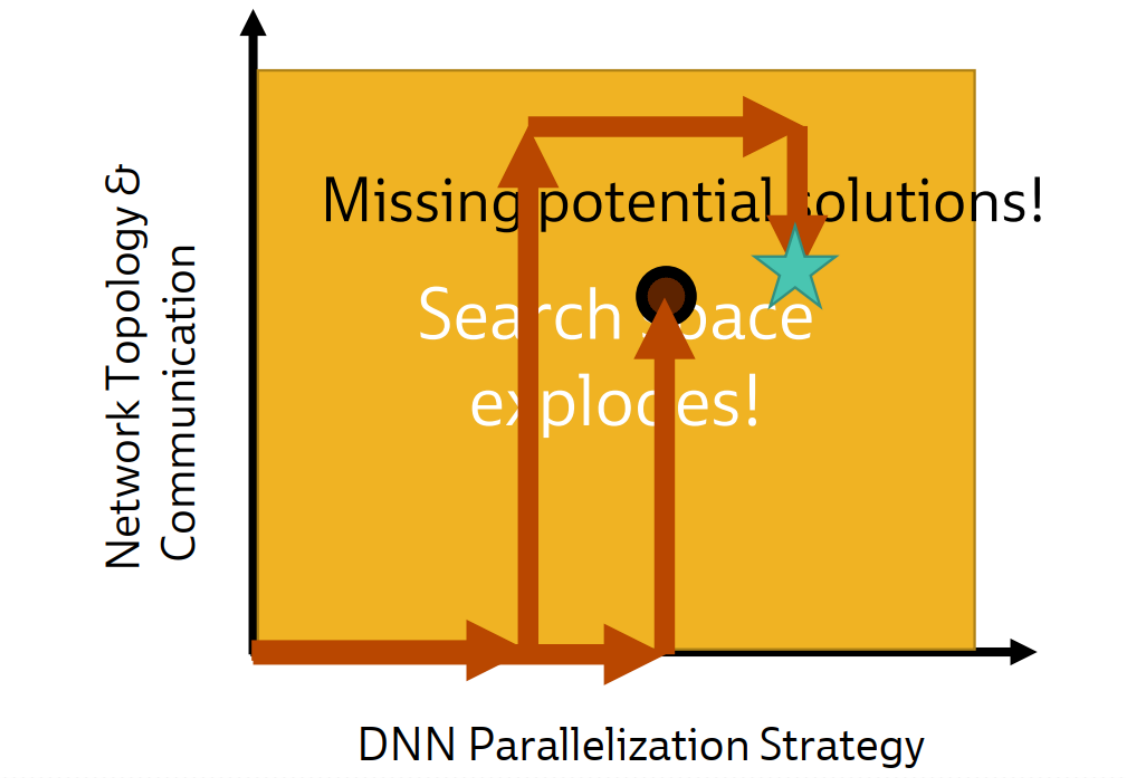
Figure 5: Illustration of TOPOOPT's interconnect.

TopoOpt

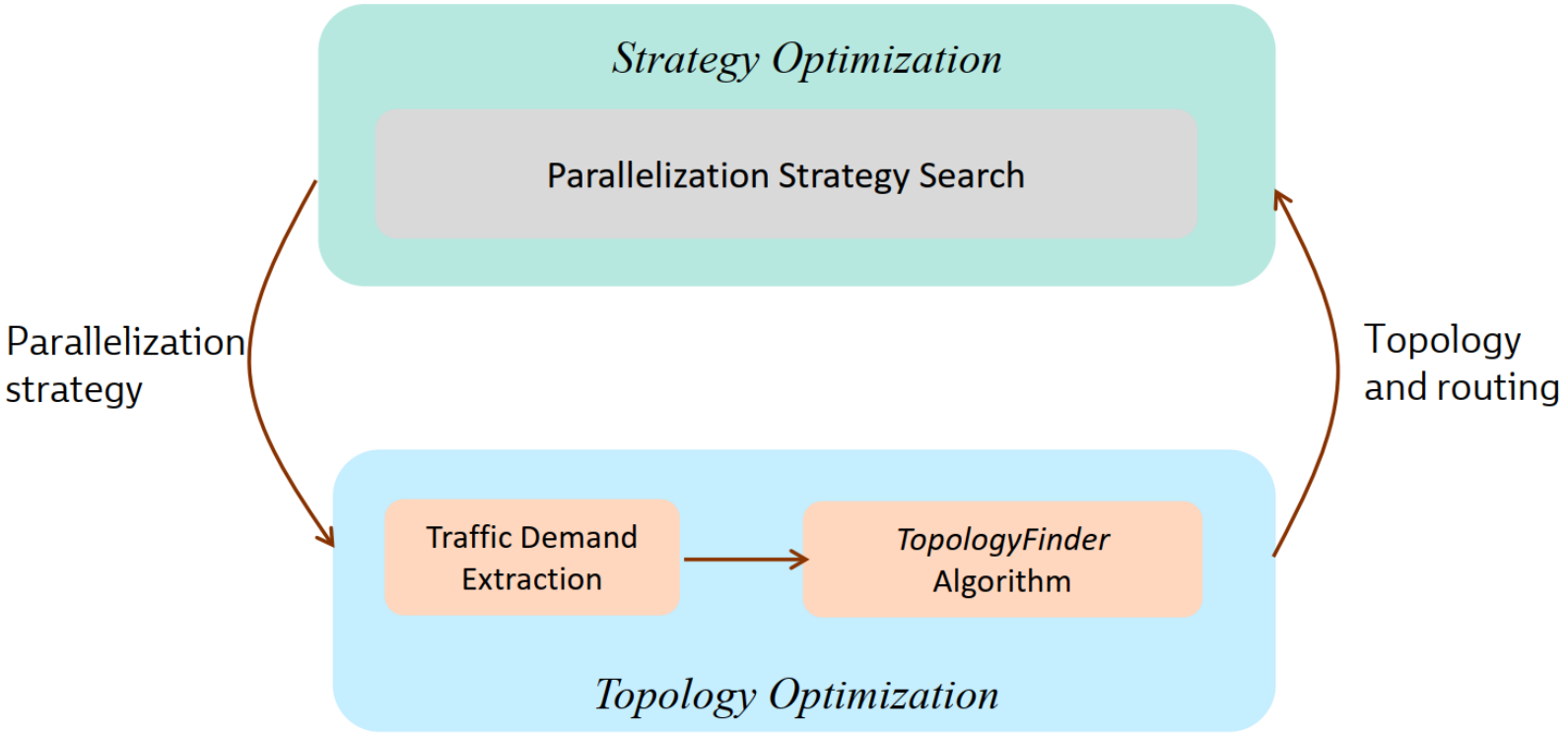
- **Interconnection:** A TopoOpt cluster is a *shardable* direct-connect fabric where each server has ***d* interfaces** connected to a core layer of ***d* optical switches**
- **Host-based forwarding:** To ensure traffic is not blocked when there is no direct link between two servers, we use a technique called *host-based forwarding*, where hosts act as switches and forward incoming traffic toward the destination
- **One-shot reconfiguration:** TOPOOPT uses a one-shot reconfiguration technique based on an offline co-optimization framework that jointly optimizes the parallelization strategy and topology

Co-optimization Challenge

- **Huge search space for optimal DNN training**

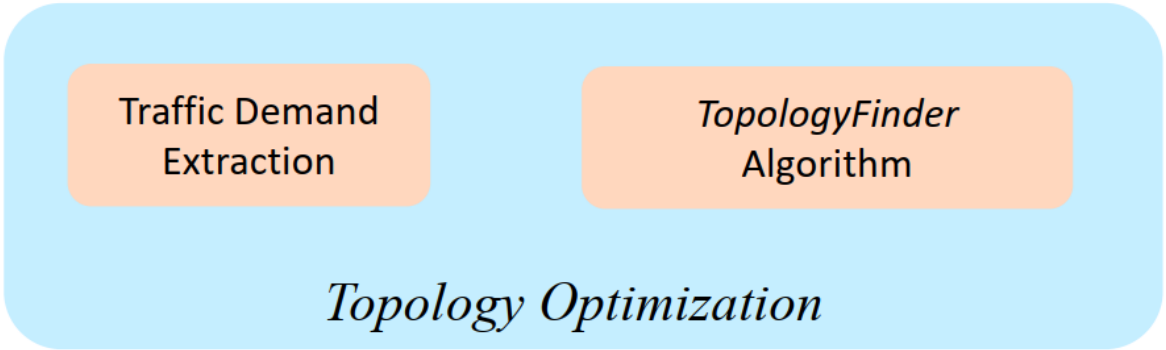


Co-optimization Challenge



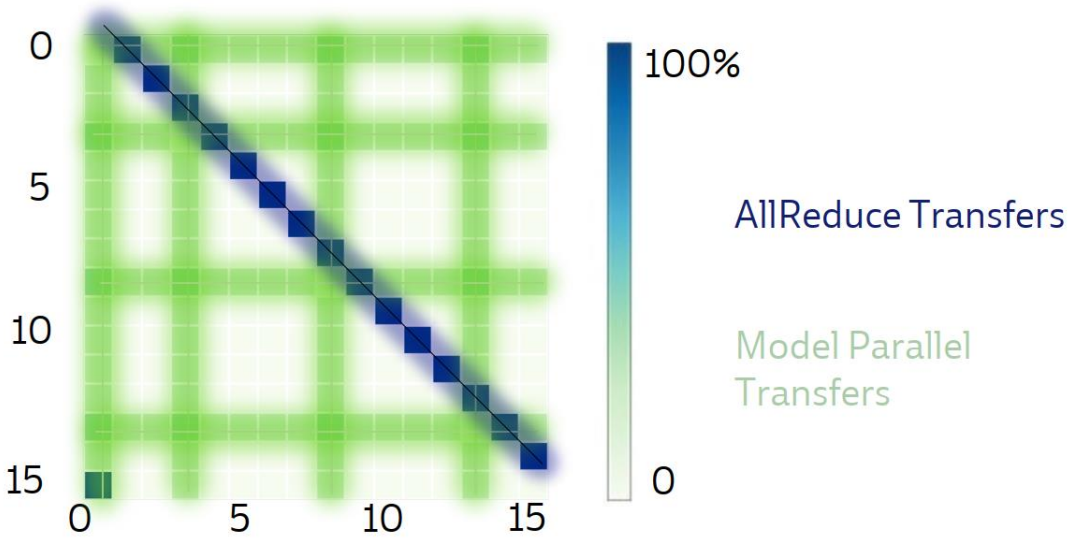
Co-optimization Challenge

- What algorithm should we use to find the topology in this framework?



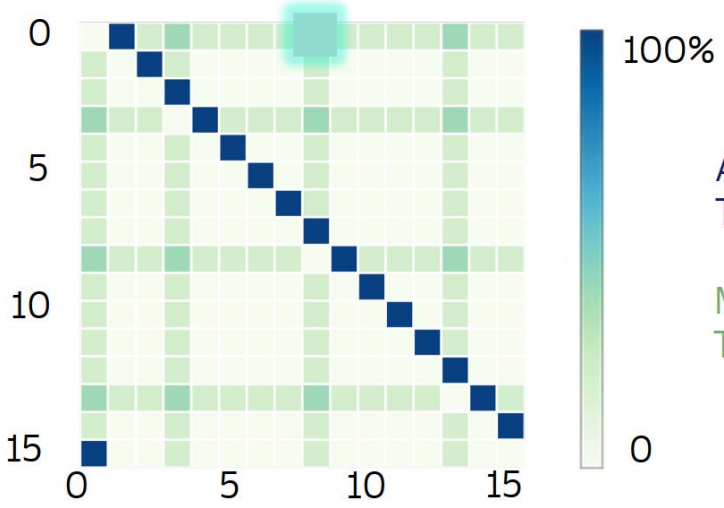
Co-optimization Challenge

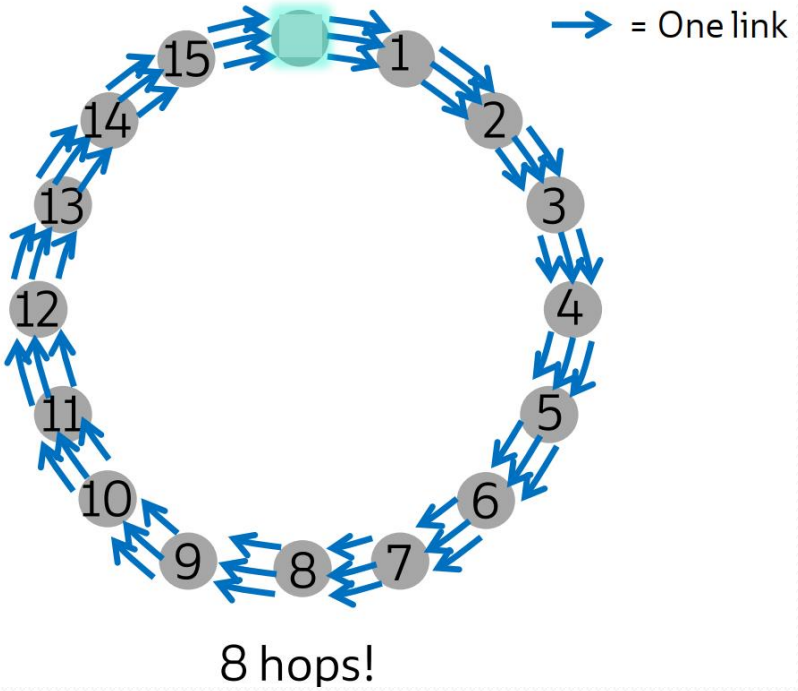
- Characteristics of DNN training traffic



Finding a Good Network Topology for Both AllReduce and Model-Parallel Transfers

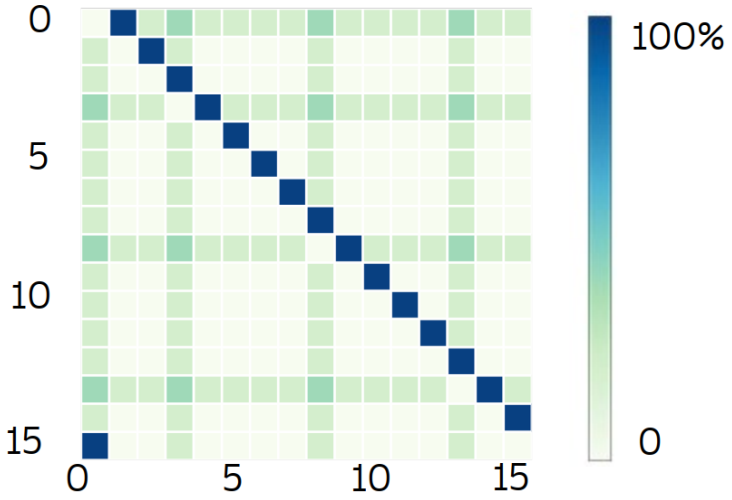
- Degree (d) = 3, unidirectional





Finding a Good Network Topology for Both AllReduce and Model-Parallel Transfers

- Degree (d) = 3, unidirectional

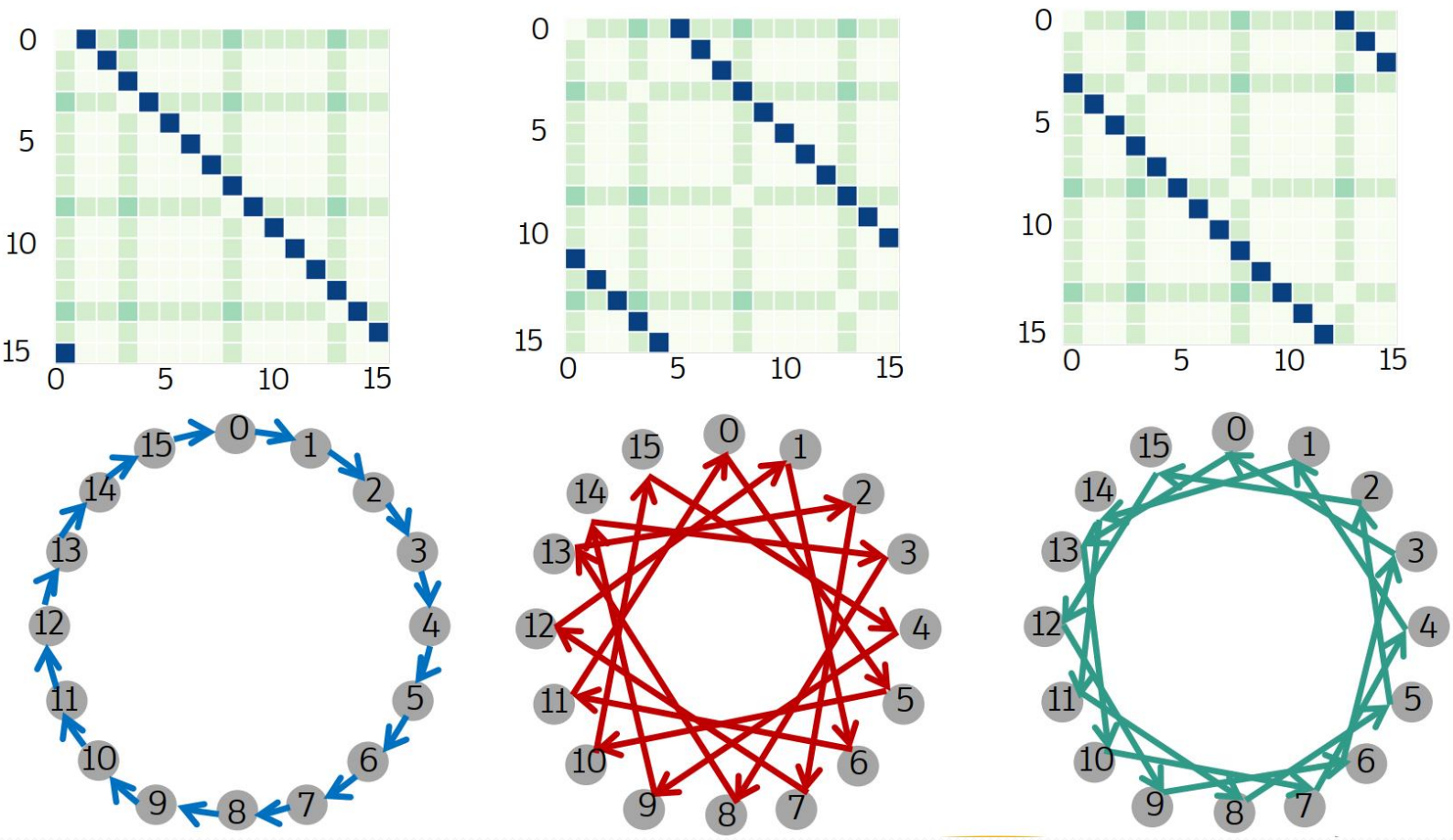


Transfer Type	Characteristics	Network Requirement
AllReduce Transfers	Large, Sparse	Ample Bandwidth
Model Parallel Transfers	Small, Dense	Low hop-count

Key Idea: Mutate the Traffic Matrix

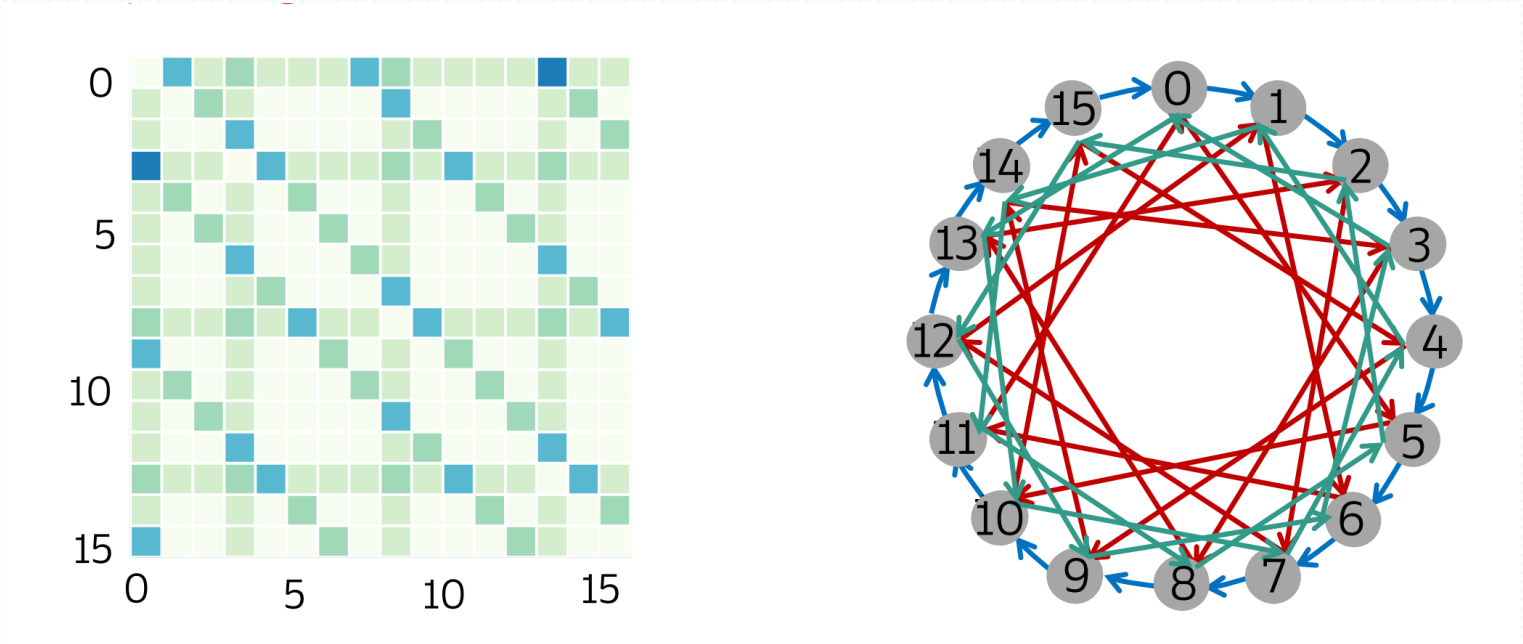
- AllReduce transfers are **mutable**.
- Model transfers are **NOT mutable**.

Key Idea: Mutate the Traffic Matrix



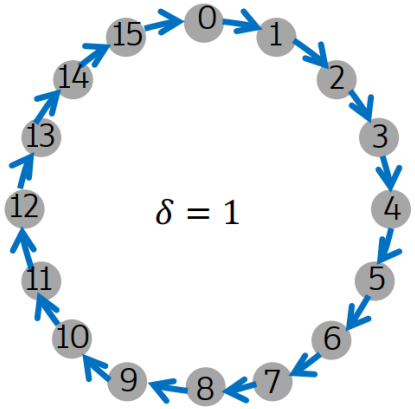
Splitting AllReduce Traffic

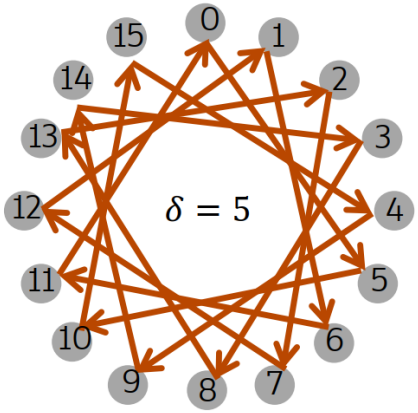
- Leverage the mutability of AllReduce transfers to achieve **high bandwidth** for AllReduce & **low hop-count** for Model-Parallel!



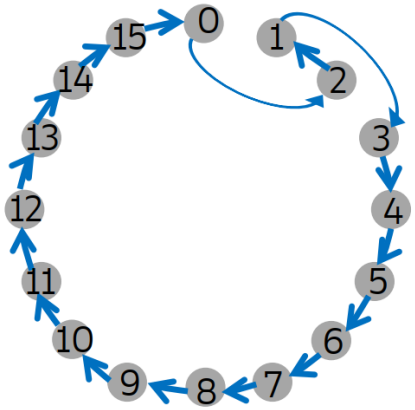
Regular Permutations

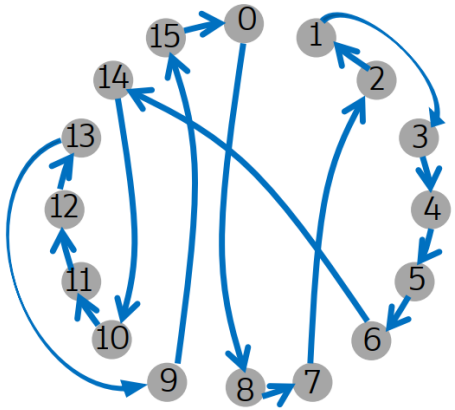
- n total accelerator, each with degree d





Regular permutations – every server connects to another one with a fixed distance δ





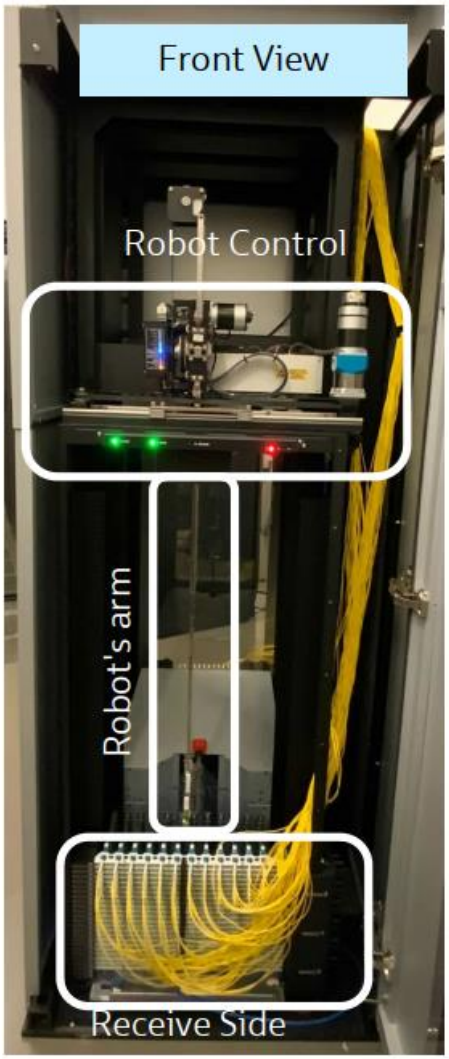
Irregular permutations

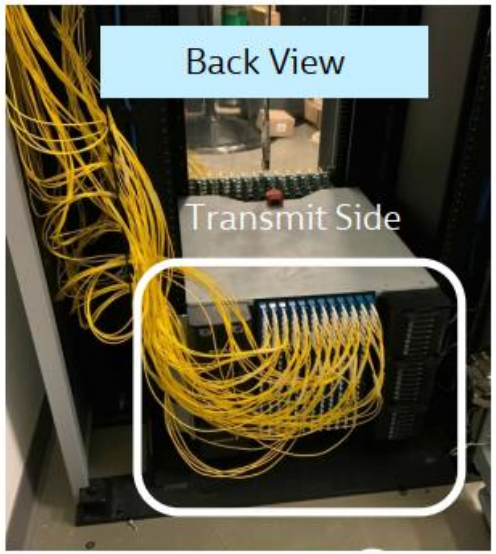
TopoOpt bounds the cluster diameter to $O(d \cdot \sqrt[d]{n})$

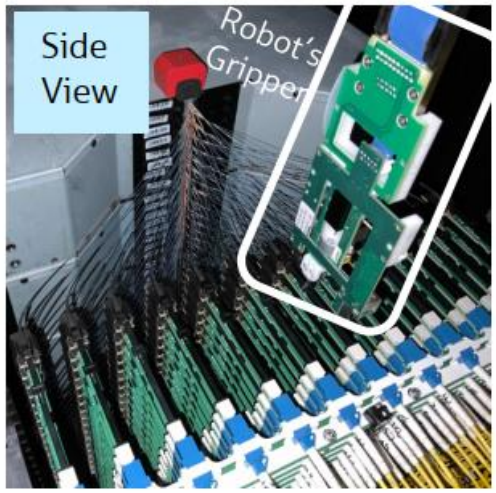
Outline

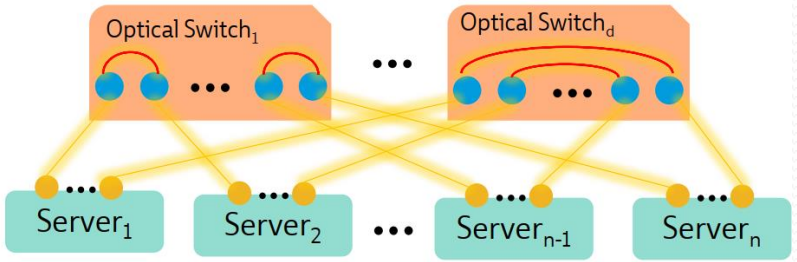
- Background and Motivation
- TopoOpt Design
- **Implementation and Evaluation**
- Review

Implementation





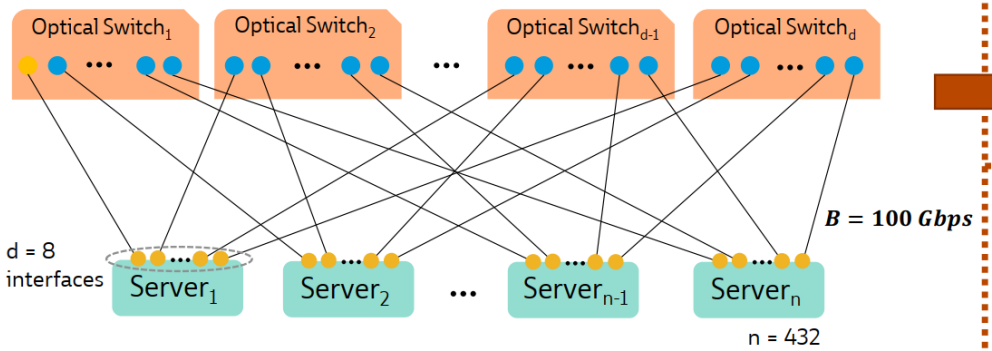


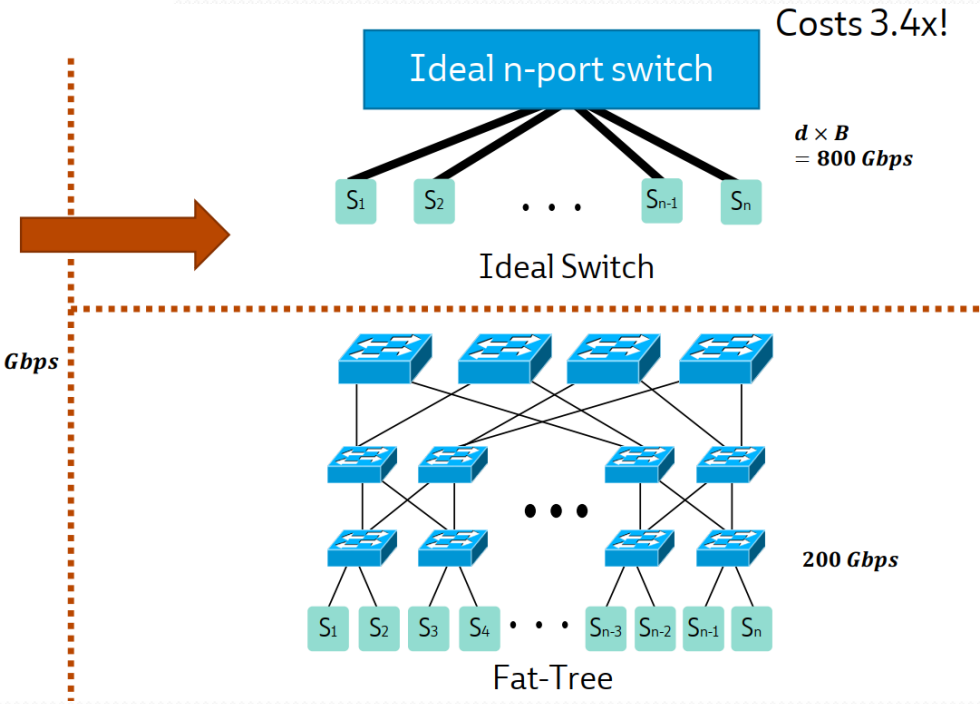


Fully functional 12-node, degree 4 testbed integrated with NCCL

Evaluation

- Evaluate TopoOpt with large scale simulation and a small-scale prototype
- Artifact code can be found at http://TopoOpt.csail.mit.edu





Evaluation

- Cost Analysis
- Performance Comparison on Dedicated Clusters
- Impact of All-to-all Traffic
- Impact of Host-based Forwarding
- Performance on Shared Clusters
- Impact of Reconfiguration Latency

Cost Analysis

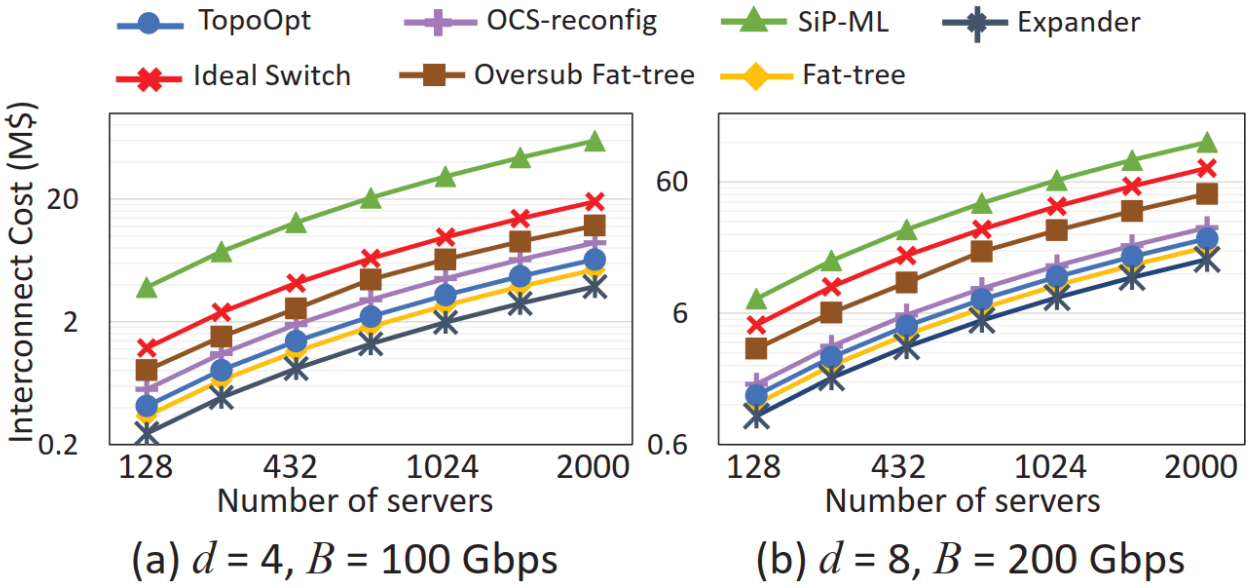


Figure 10: Interconnect cost comparison.

- Ideal Switch: a full-bisection Fat-tree of the same bandwidth
- Using OCSs for TOPOOPT is more expensive (1.33 ×, on average) than patch panels
- The cost of TOPOOPT overlaps with that of the Fat-tree
- Third, the ratio of Ideal Switch's cost to TOPOOPT's cost is 3.2 × on average.

Performance Comparison on Dedicated Clusters

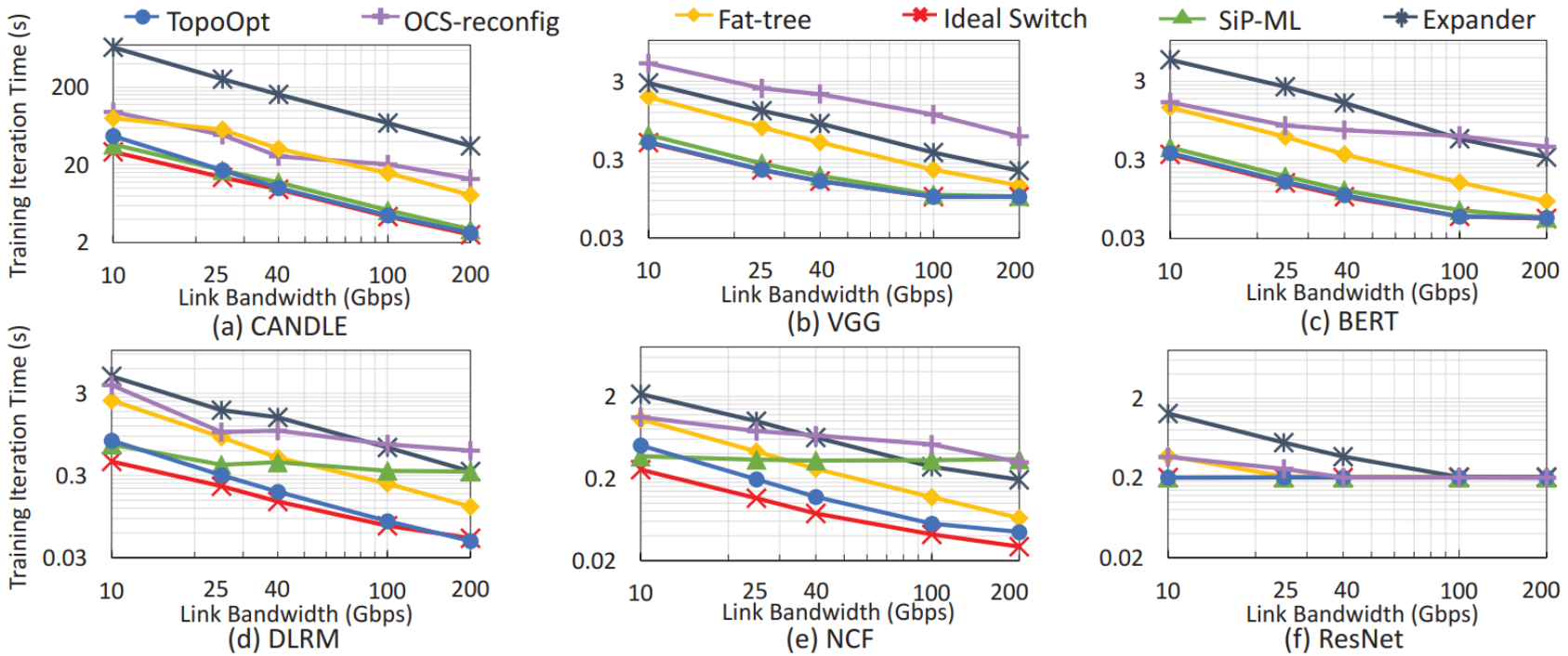
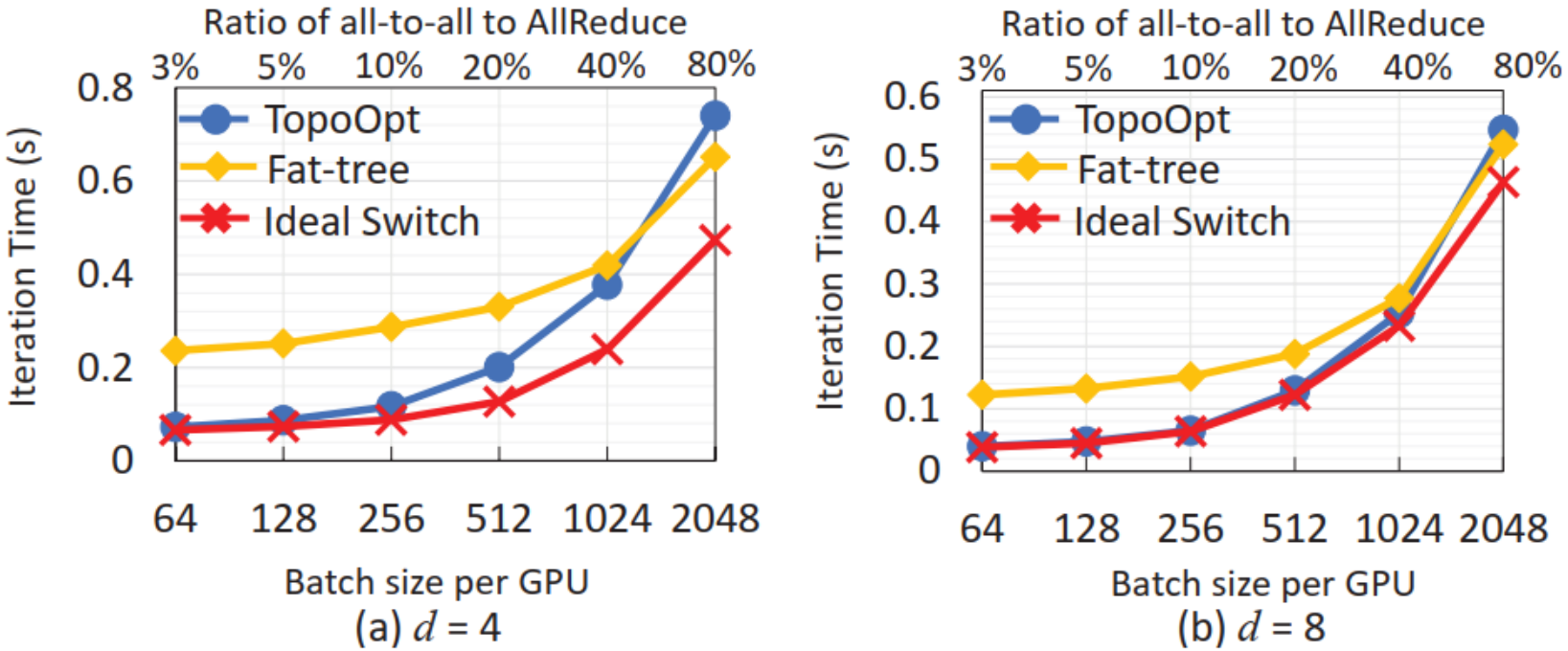


Figure 11: Dedicated cluster of 128 servers ($d = 4$).

Impact of All-to-all Traffic



- As shown in Figure 12a, when the batch size is 128 and d = 4, TopoOpt's performance matches that of Ideal Switch, while Fat-tree is a factor of 2.7 slower
- Specifically, when the batch size is 2048 and all-toall traffic is 80% of AllReduce traffic, TOPOOPT performs poorly, and the iteration time is a factor of 1.1 higher than that of the Fat-tree architecture. **Increasing the server degree d mitigates the problem**, as shown in Figure 12b

Impact of Host-based Forwarding

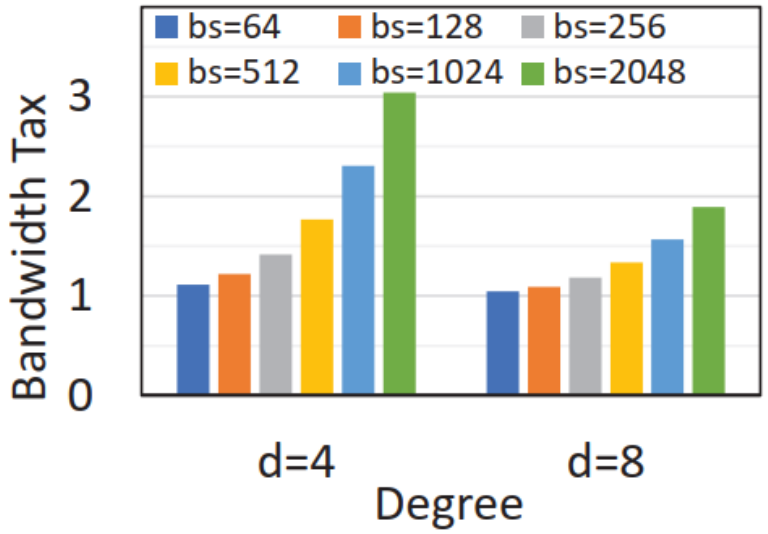


Figure 13: Bandwidth tax.

- At batch size 64 with $d = 4$, TopoOpt experiences a bandwidth tax of 1.11, indicating that host-based forwarding creates 11% extra traffic in the network.
- Increasing the degree to $d = 8$ further improves this number to 1.05

Performance on Shared Clusters

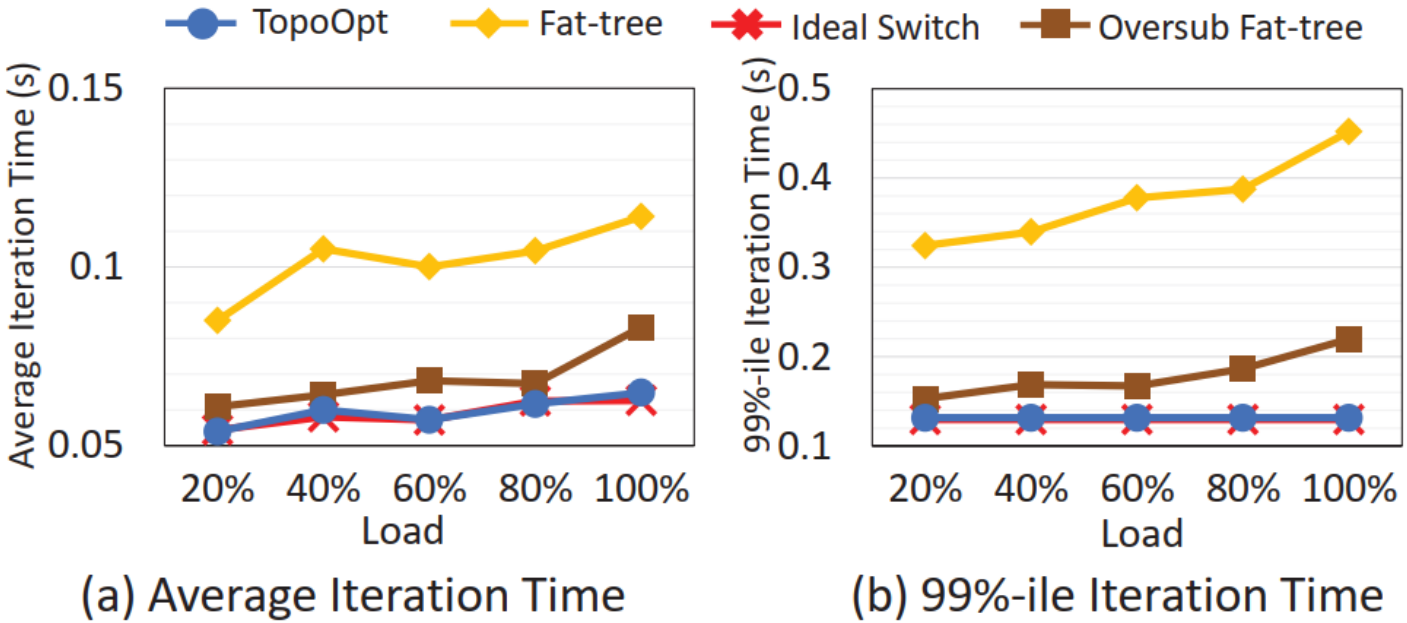


Figure 16: Shared cluster of 432 servers ($d = 8, B = 100$ Gbps).

- Compare the performance of different network **architectures when the cluster is shared across multiple DNN jobs**.
- Figure 16a shows that TOPOOPT improves the average iteration time by 1.7× and 1.15×, compared to the Fat-tree and Oversub. Fat-tree architectures, respectively. A similar trend for the tail iteration completion times, depicted in Figure 16b

Impact of Reconfiguration Latency

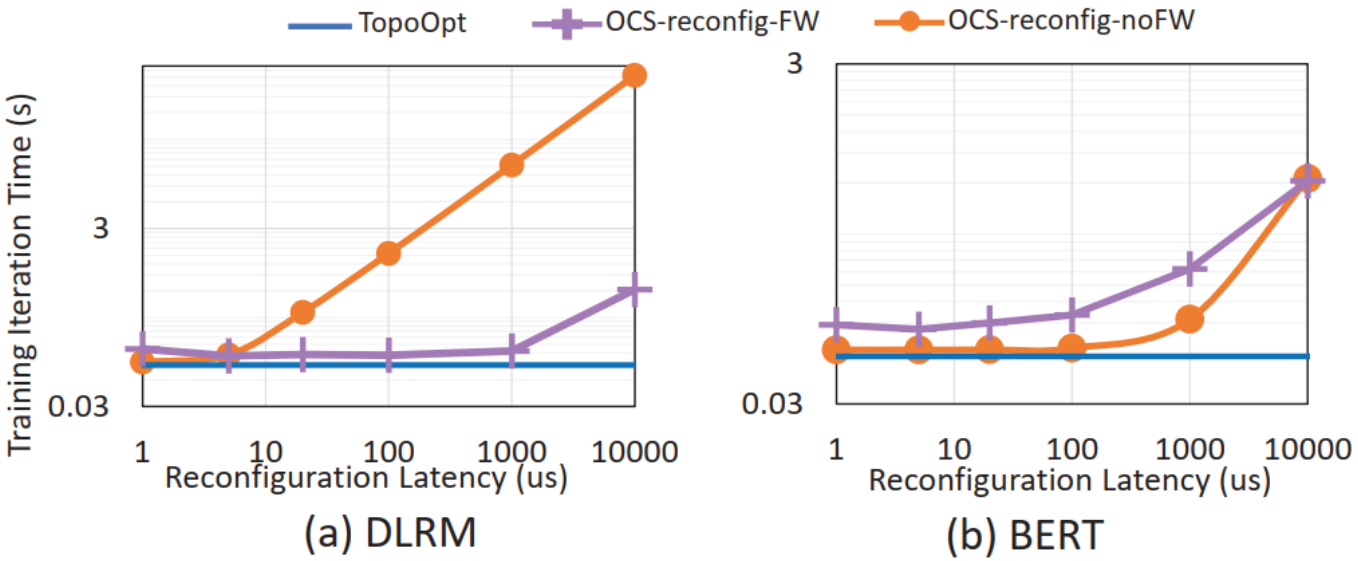


Figure 17: Impact of reconfiguration latency (d=8, B=100 Gbps).

Outline

- Background and Motivation
- TopoOpt Design
- Implementation and Evaluation
- **Review**

Discussion

- **Target workload:** The most suitable workload for a TOPOOPT cluster is a set of large DNN training jobs with hybrid data and model parallelism (or simply data parallelism).
- **Storage and control plane traffic:** Training clusters consist of custom-designed servers, each with eight GPUs, eight dedicated NICs for training traffic (GPU NICs), and four additional NICs for storage and other traffic (CPU NICs). **TOPOOPT only considers GPU NICs as server degree and partitions the network dedicated for training traffic**.

Discussion

- **Supporting dynamic scheduling and elasticity:** TOPOOPT is designed for the vast number of long-lasting training jobs that do not change dynamically. In cases where elasticity is required, instead of using patch panels, we use OCSs (or other fast reconfigurable optical switches) to change the servers participating in a job quickly.
- **Handling failures:** When a fiber fails, TOPOOPT can temporarily use a link dedicated to MP traffic to recover an AllReduce ring.

Discussion

- **Supporting multi-tenancy:** TOPOOPT can leverage NVIDIA's MIG to treat one physical server as multiple logical servers in its topology.
- **TOPOOPT's limitations:** TOPOOPT's approach assumes the traffic pattern does not change between iterations. However, this assumption may not hold for Graphic Neural Network models or **Mixture-of-Expert (MoE) models**
  - *MixNet: A Runtime Reconfigurable Optical-Electrical Fabric for Distributed Mixture-of-Experts Training*