NEST

Network- and Memory- Aware Device Placement for Distributed Deep Learning

Irene Wang, Vishnu Varma Venkata, Arvind Krishnamurthy, Divya Mahajan

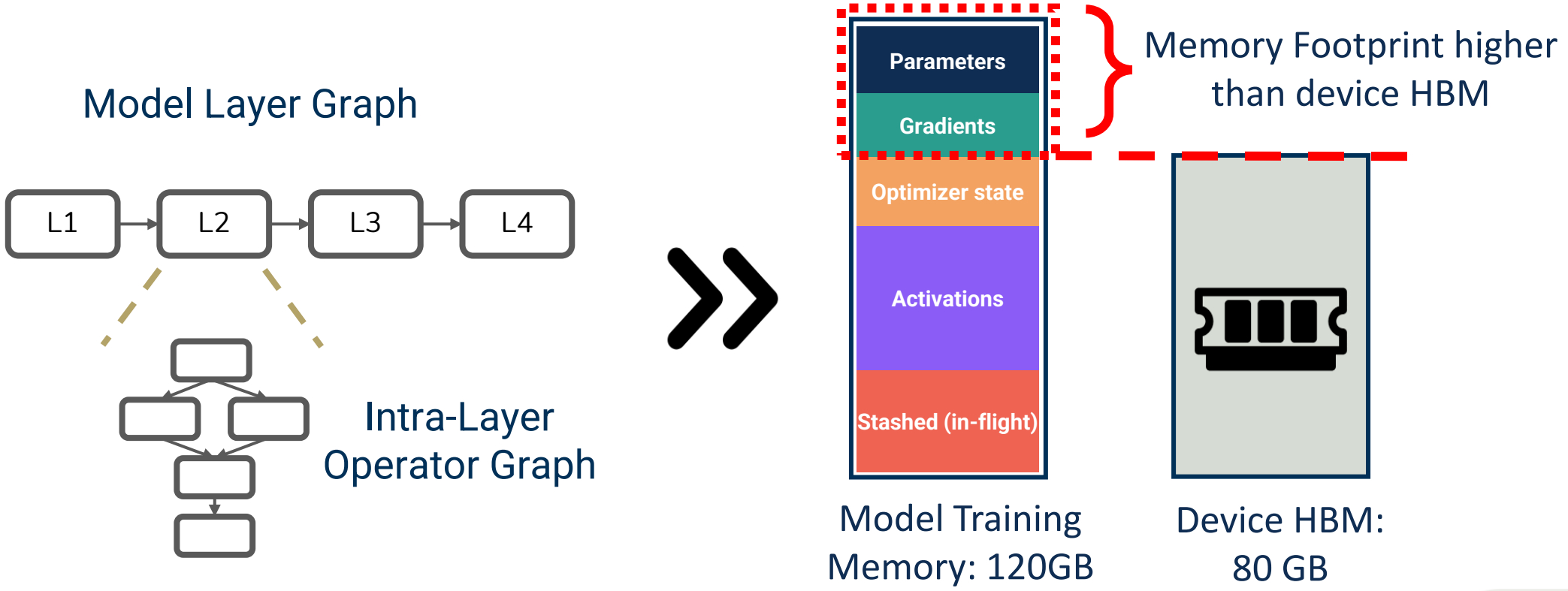
SCAI
SYSTEMS FOR SCALABLE INTELLIGENCE

Georgia Tech

UNIVERSITY of WASHINGTON

Training a Deep Learning Model

Training each deep learning model has ever **increasing memory requirement**.

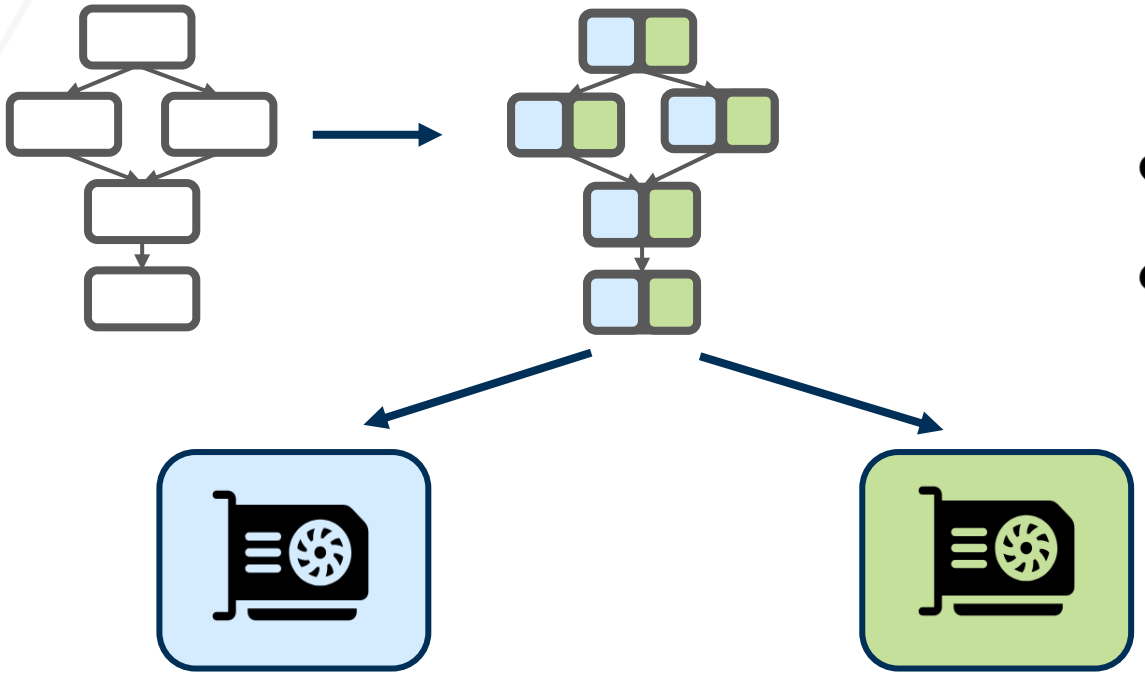


Training a Deep Learning Model

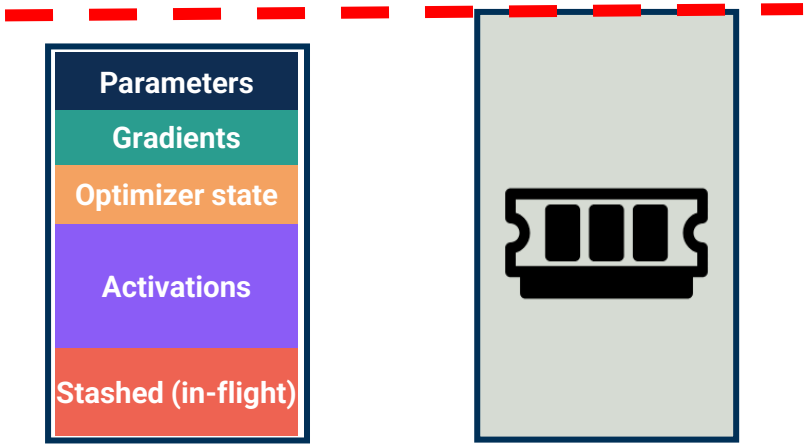
Partition models across multiple machines

Operator Graph

Tensor Model Parallelism



Per Device Memory requirement **lower than** device capacity



Per Device Model Training Memory: 74 GB

Device HBM: 80 GB

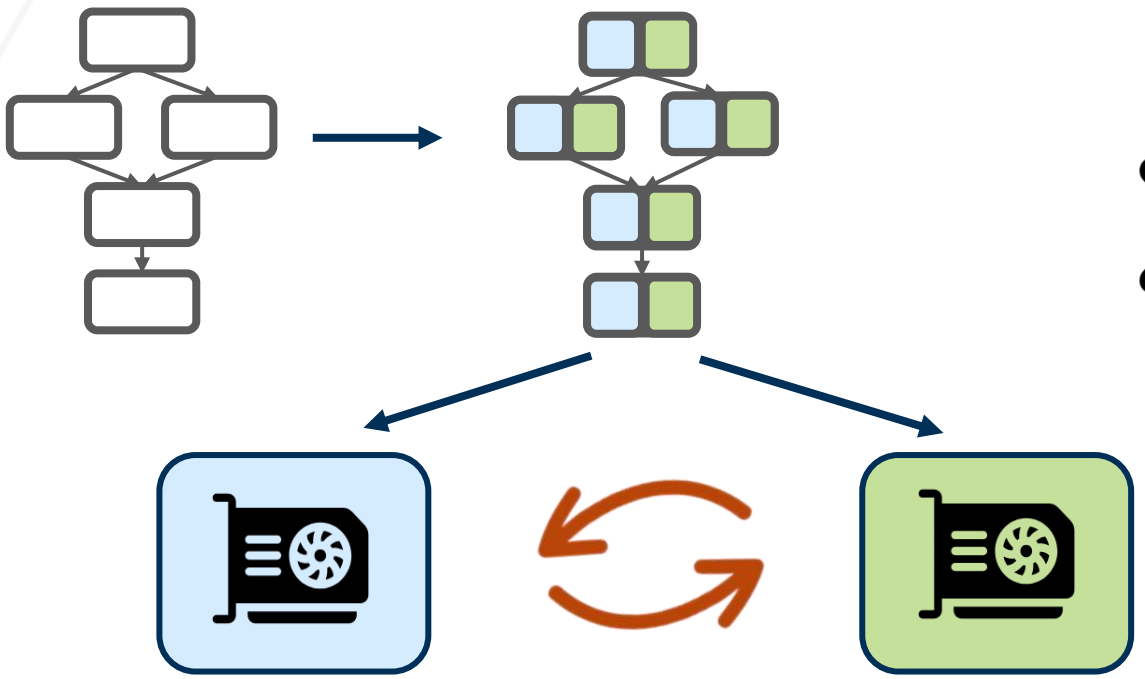
Training a Deep Learning Model

Partition models across multiple machines

Parallelization strategies reduce per-device memory or compute requirements,
but incur additional runtime overheads

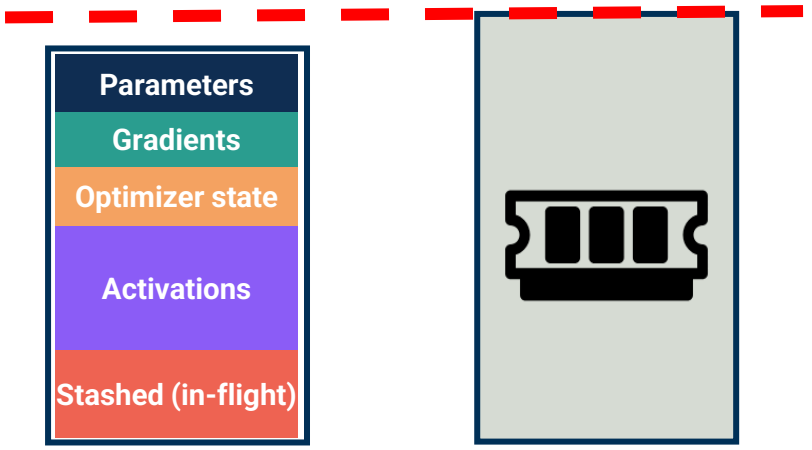
Operator Graph

Tensor Model Parallelism



AllReduce Communication

Per Device Memory requirement **lower than** device capacity

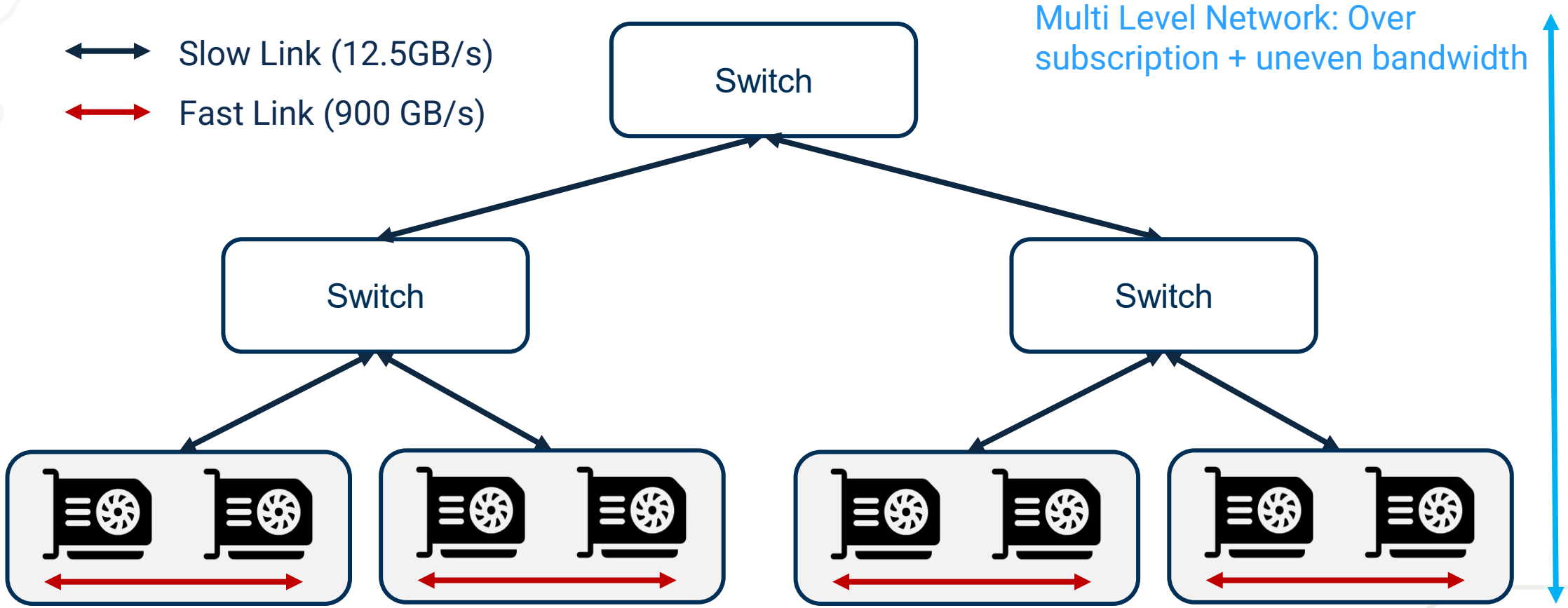


Per Device Model Training Memory: 74 GB

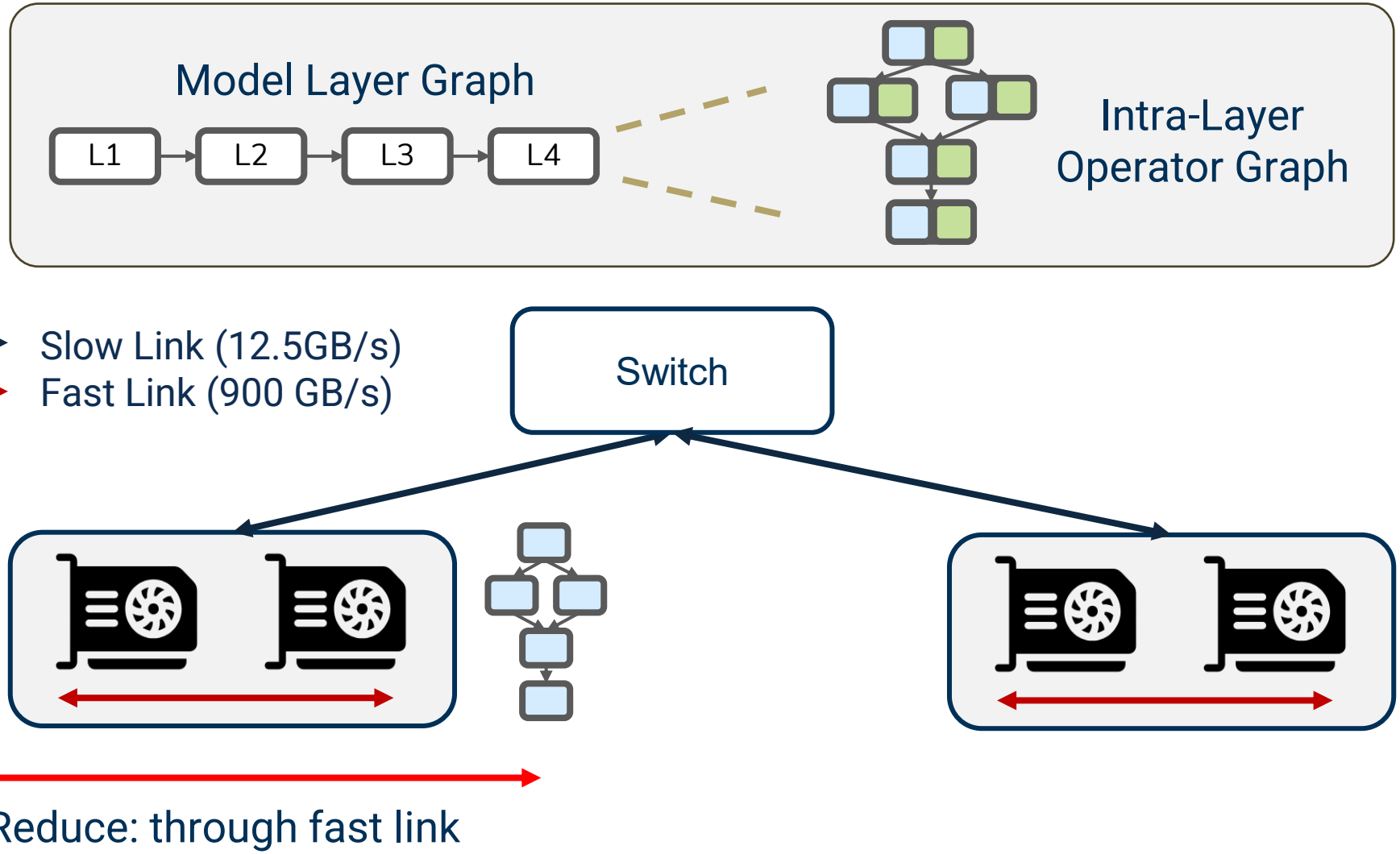
Device HBM: 80 GB

Training a Deep Learning Model

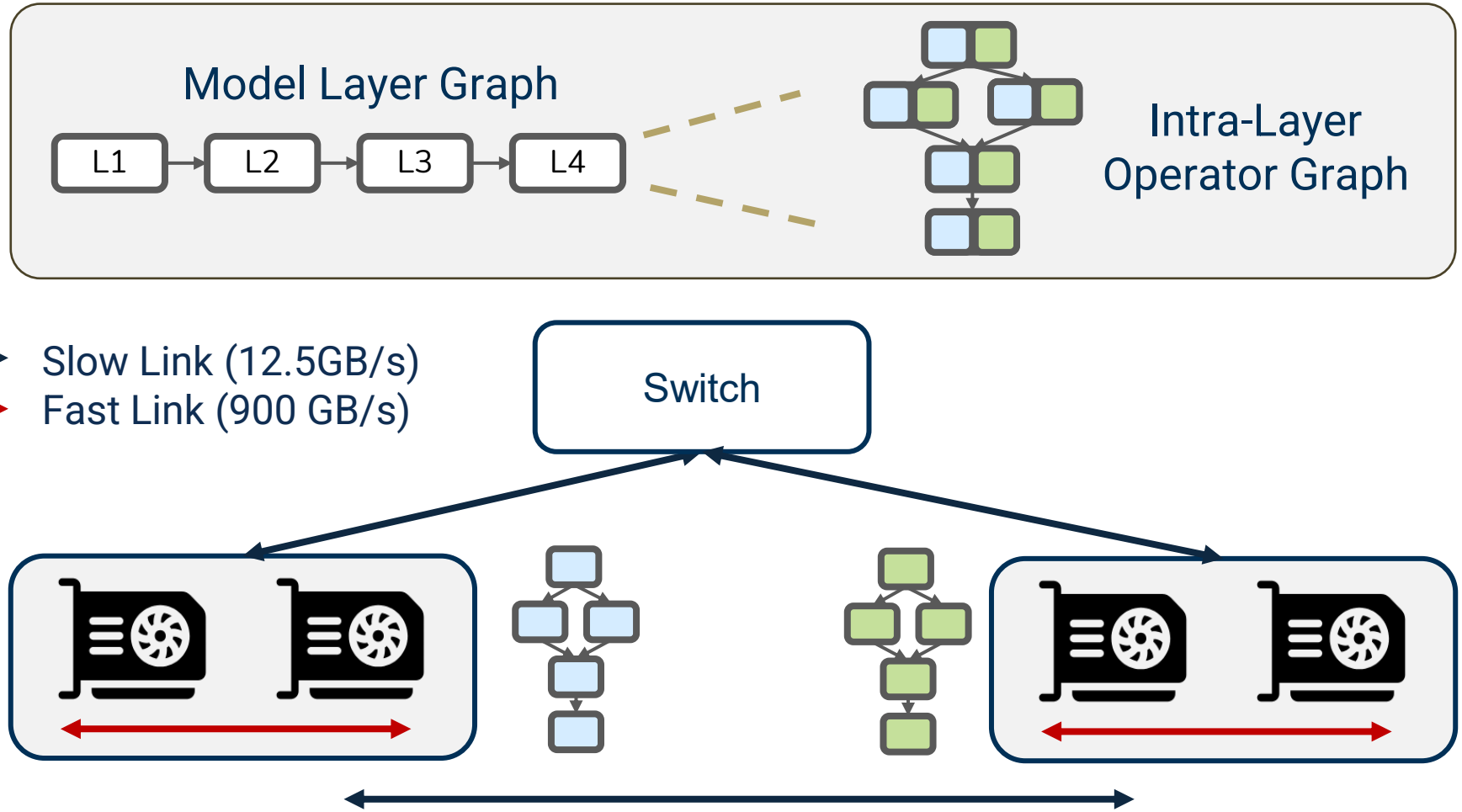
Place the partitions across machines in the network



Training a Deep Learning Model



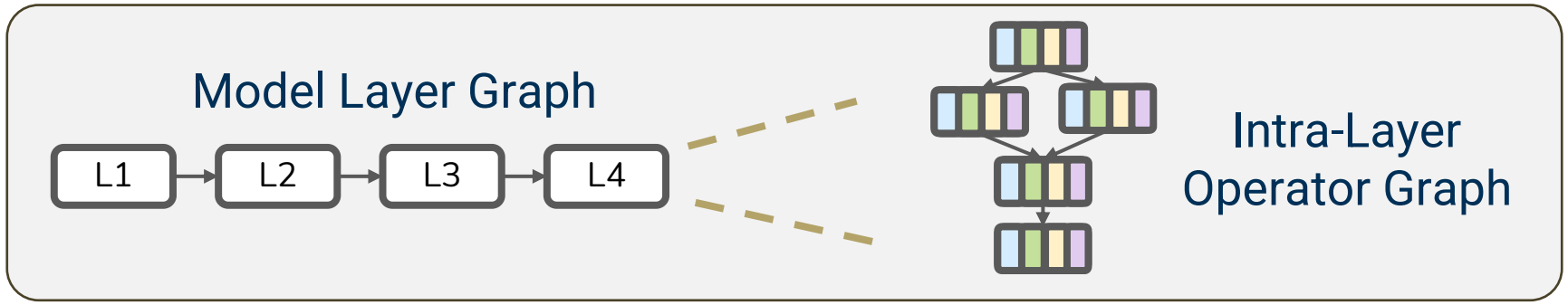
Training a Deep Learning Model

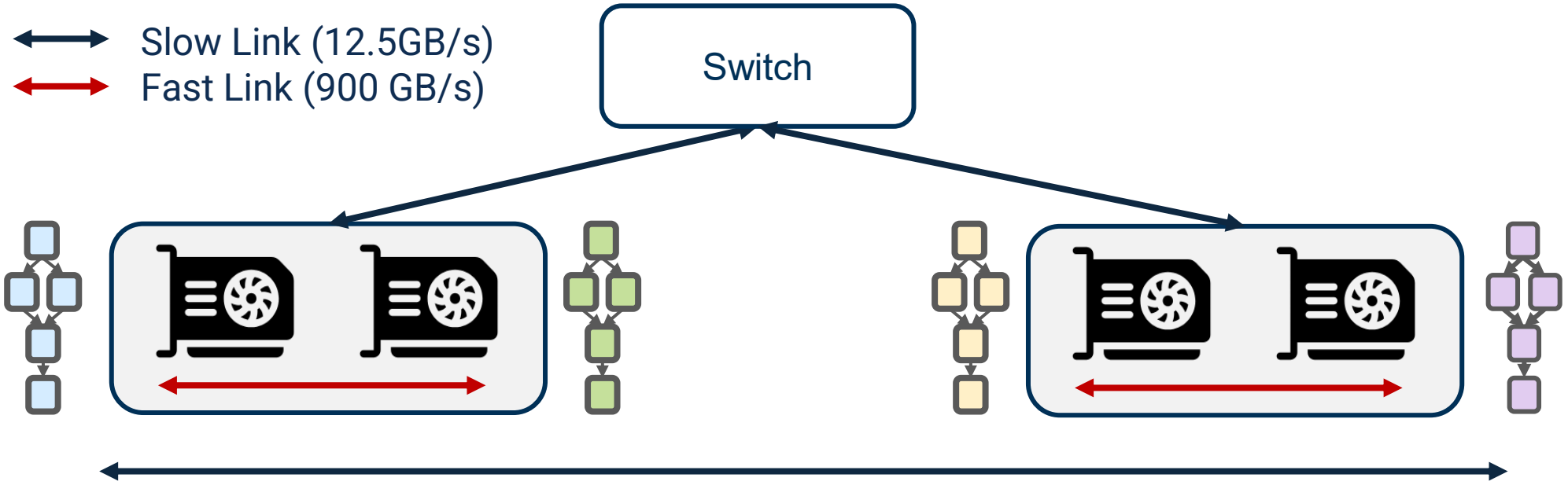


AllReduce: Through slower links

Communication time > 9x difference !

Training a Deep Learning Model



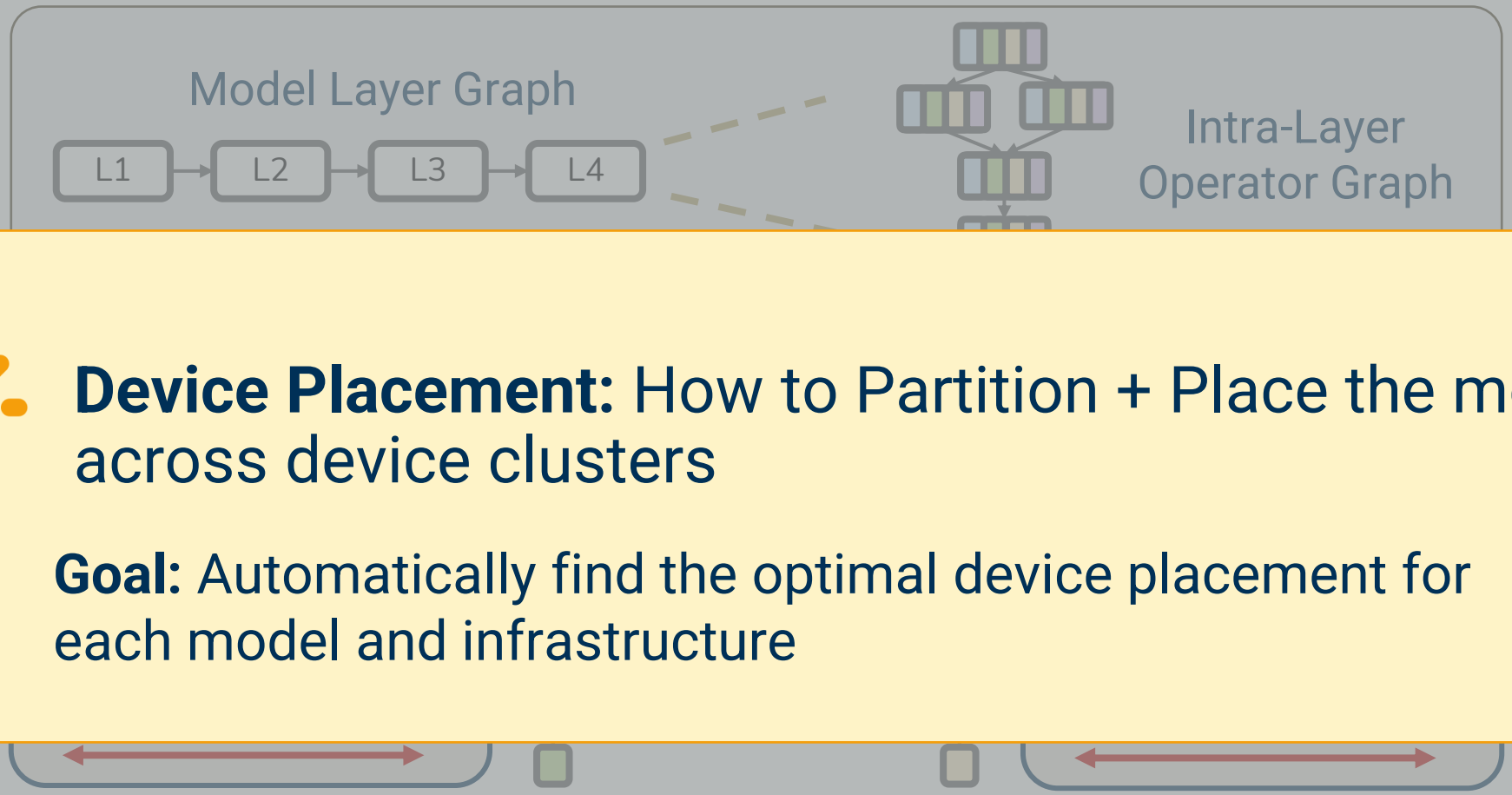


AllReduce: Through slower links

Tradeoff

Further reduces Per-device Memory footprint and less per-device "Compute" Time

Training a Deep Learning Model



Device Placement: How to Partition + Place the model across device clusters

Goal: Automatically find the optimal device placement for each model and infrastructure

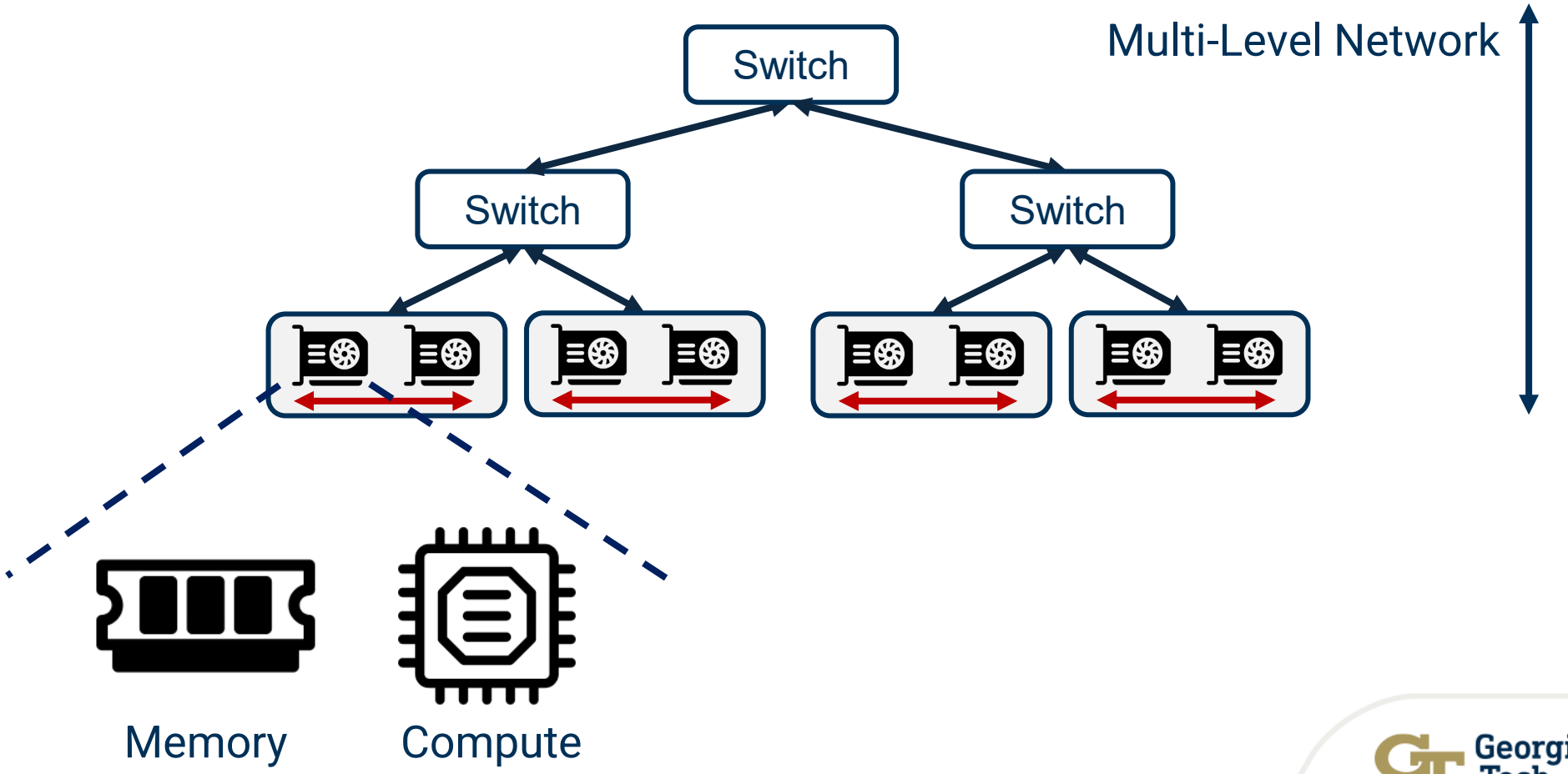
All Reduce: Through slower links

Tradeoff

Further reduces Per-device Memory footprint and less per-device “Compute” Time

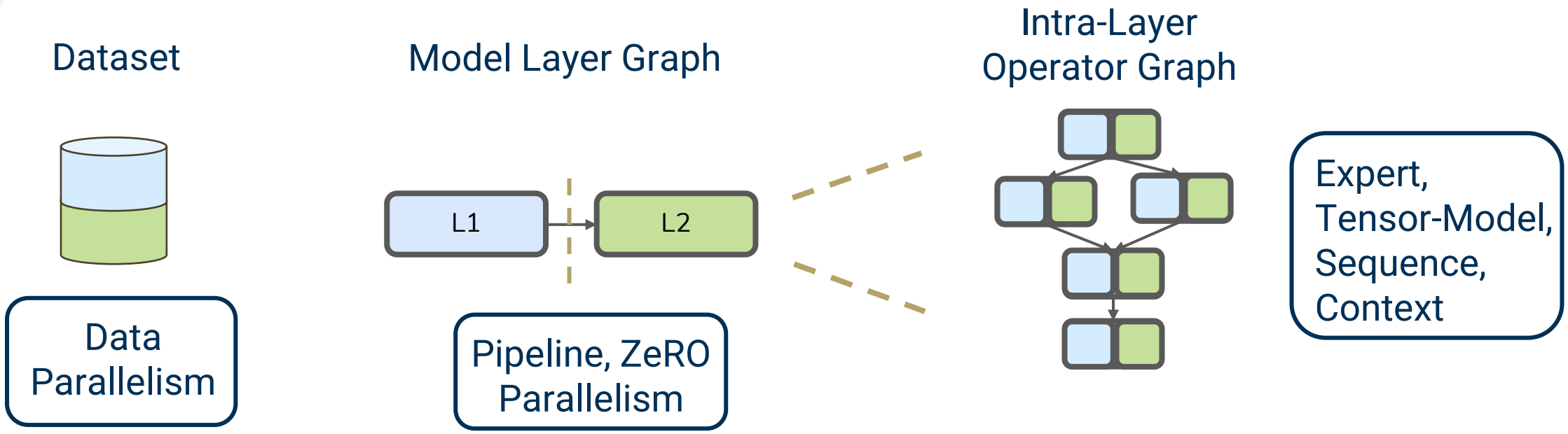
What Make Device Placement Challenging?

Optimal Strategy depends and **tightly couples Network, Memory, and Compute**, forming a large and complex search space



What Make Device Placement Challenging?

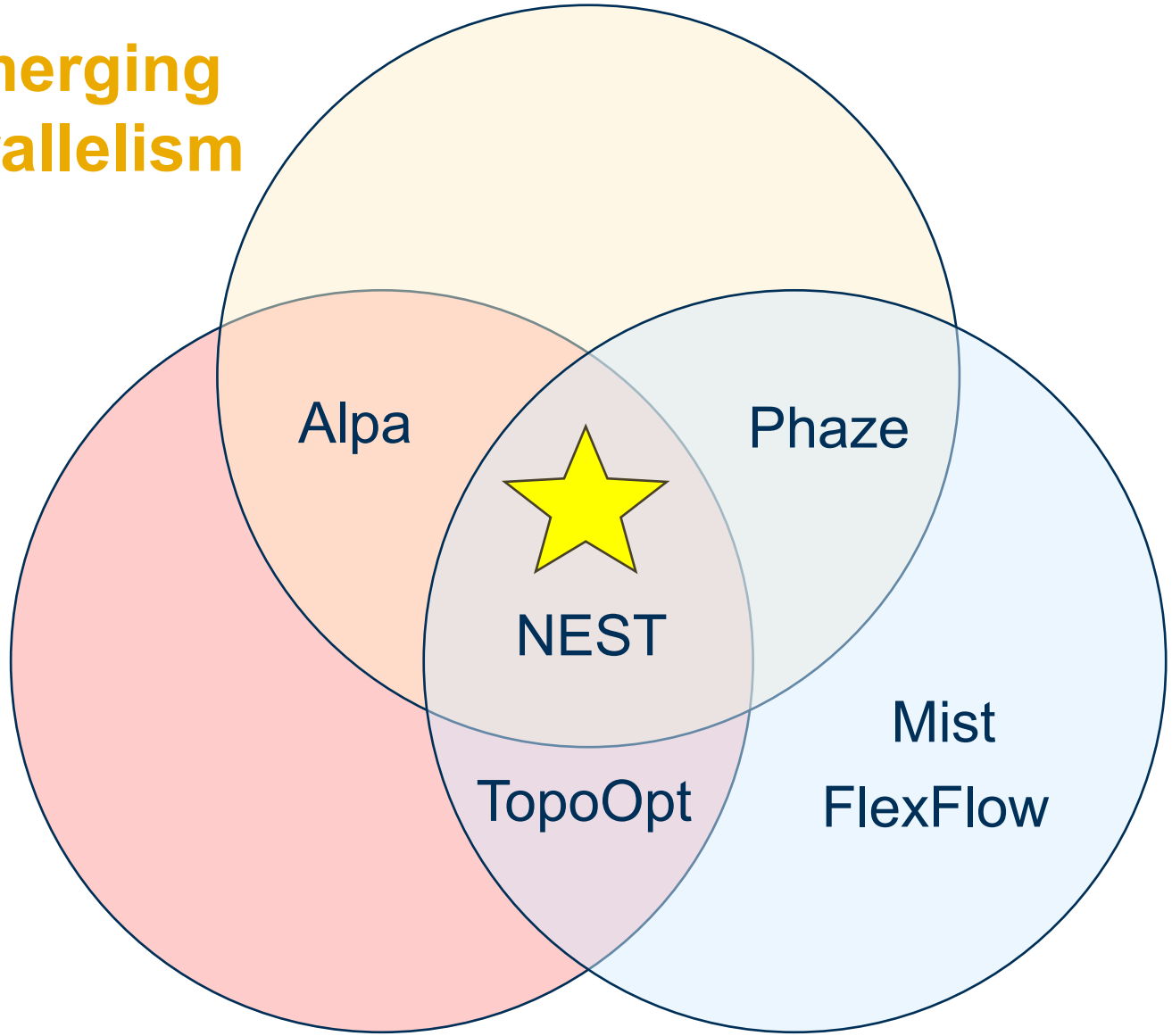
Device Placement often employs **a hybrid of multiple parallelization strategies**



Each method affects memory usage differently and incurs unique runtime costs!

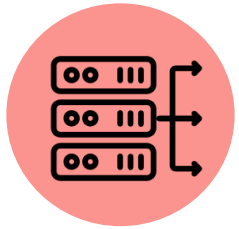
Prior Works: Reduced Search Space

Prior Works are not Network-, Memory-, and Compute-Aware — **All at Once**

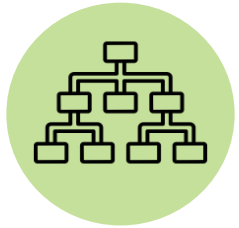


NEST

We introduce NEST, an **Efficient and Scalable** Dynamic Programming-based placement framework, while guaranteeing optimality:



Supports growing number of **parallelism strategies**

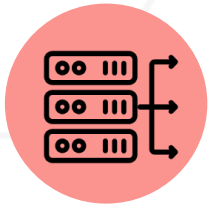


Models impact of placement on **diverse network**

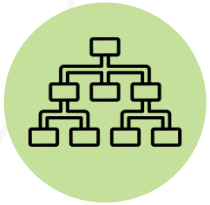
Integrates **memory feasibility** tracking and optimization

Georgia Tech

Our Approach



Strategy Explosion



Heterogeneous Networks

Memory Footprint

Categorize parallelism along orthogonal dimensions.

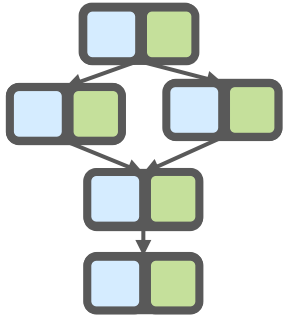
SUB-GRAPH (per-layer) is pre-profiled offline; GRAPH-GLOBAL is the DP search axis.

Insight 1: Categorize the Parallelism Strategies

Based on the **scale** they operate at:

Strategy 1: Sub Graph

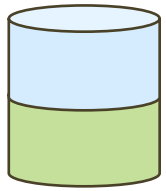
- Single Layer Decisions: Partitioning **operator dimensions** or **graph shapes**



Intra-Layer Operator Graph

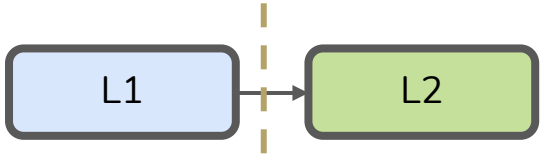
Strategy 2: Graph-Global

- Global Designs: Partition model **layers, parameters and data** across devices



Data

Layers, Parameters, Gradients, Optimizers

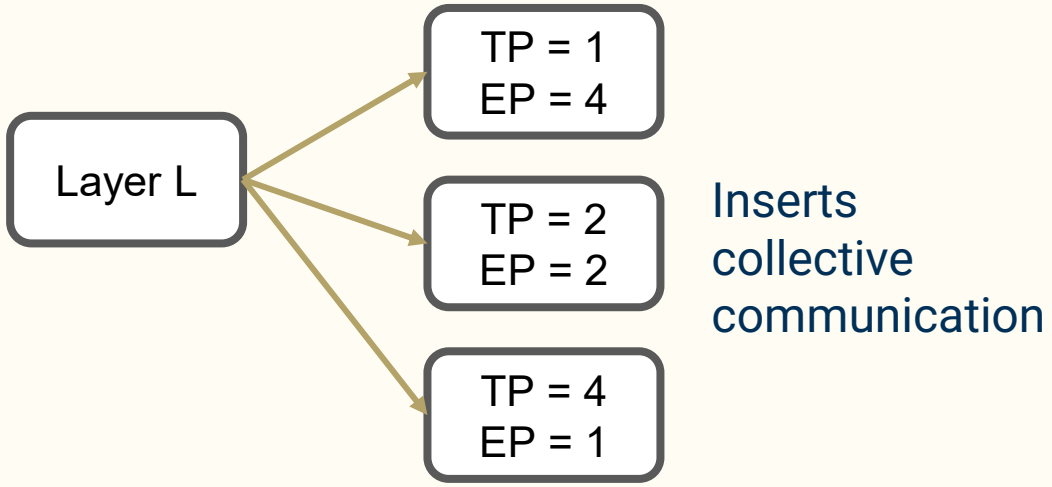


Insight 1: Two Types of Parallelism = Two Roles in the Framework

Sub-Graph Parallelisms

Local: Modifies a single Layer

TP | EP | SP | CP

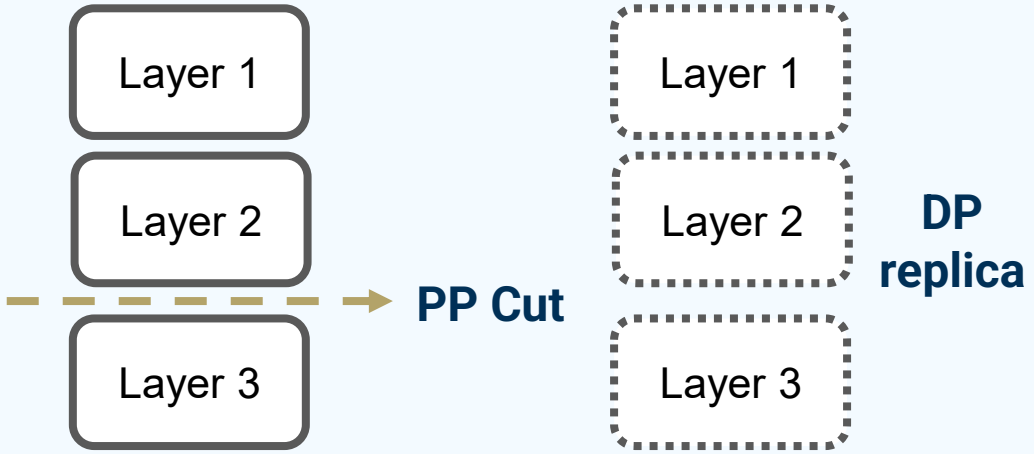


Generates transformed **operator graphs**, and profile offline

Graph-Global Parallelisms

Global decisions: in Dynamic Program

PP | DP | ZeRO



The **Dynamic Program** decides over the optimal stage boundaries

Insight 1: Parallelism Categorization - Scalability

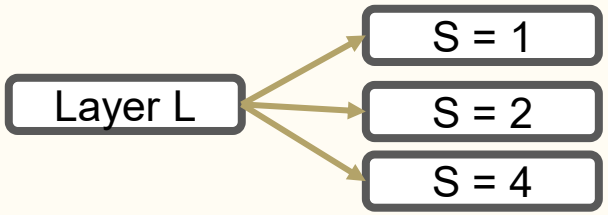
This approach allows NEST to easily extend new parallelism strategies

1. New Strategy Arrives

`Strategy S`

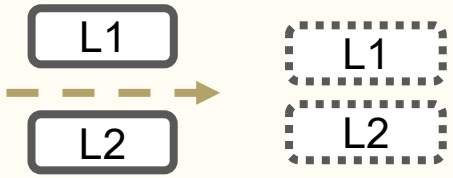
Novel way to split operators across devices

2. Provide Graph + Cost



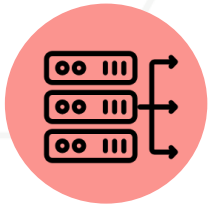
Transform Operator graph + Comm/Compute cost annotation

3. Dynamic Program Doesn't Change



Dynamic Program iterates.
No change to the algorithm

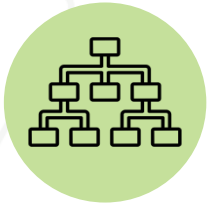
Our Approach



Strategy Explosion

Categorize parallelism along orthogonal dimensions.

SUB-GRAPH (per-layer) is pre-profiled offline; GRAPH-GLOBAL is the DP search axis.



Heterogeneous Networks

Replace device pairs with a few network levels.

Only recur over 3-5 network levels while maintaining optimality .

Memory Footprint

Building a Network Aware Dynamic Program

dp [D] [k] [s]: Each entry tracks the minimum possible latency to execute the partition of:

D : Downset index
Set of layers

k : Devices
Number of devices

s : Stages
Number of pipeline stages

Find the Optimal Pipeline split of :

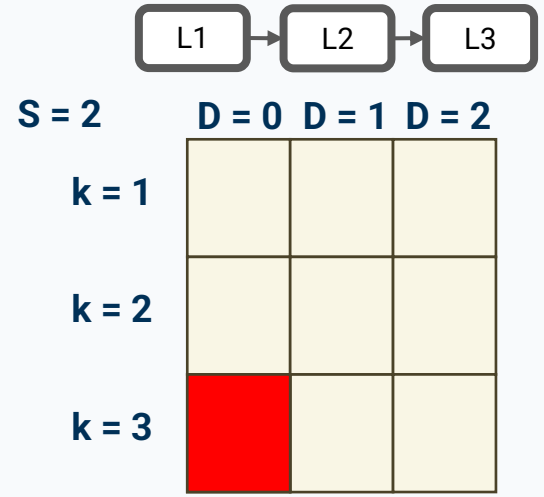
Device 1

Device 2

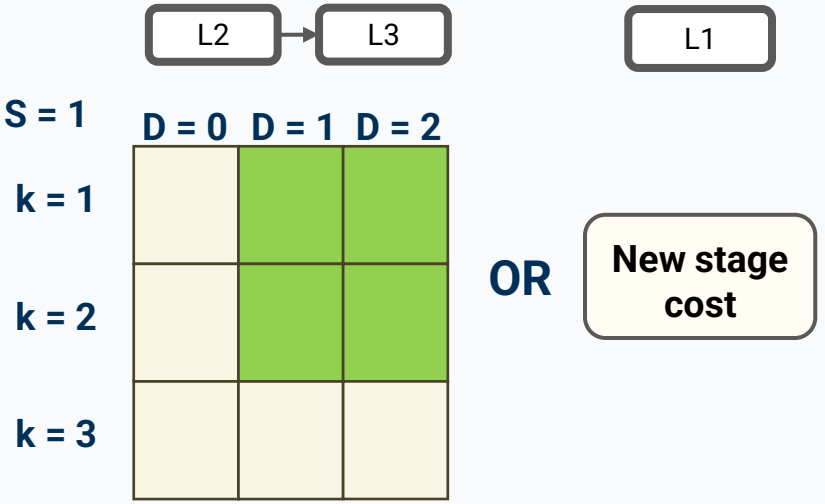
Device 3

Maximum Num Stages = 2

Current Cell: dp [D=0] [k=2] [s=2]:



Depends on the previous partitions :



Challenge: The Unknow Producer Problem

When network is considered, we not only need to decide the **pipeline stage partition**, but also **the placement of each stage**.

Load already placed
Backward pass
latency fixed

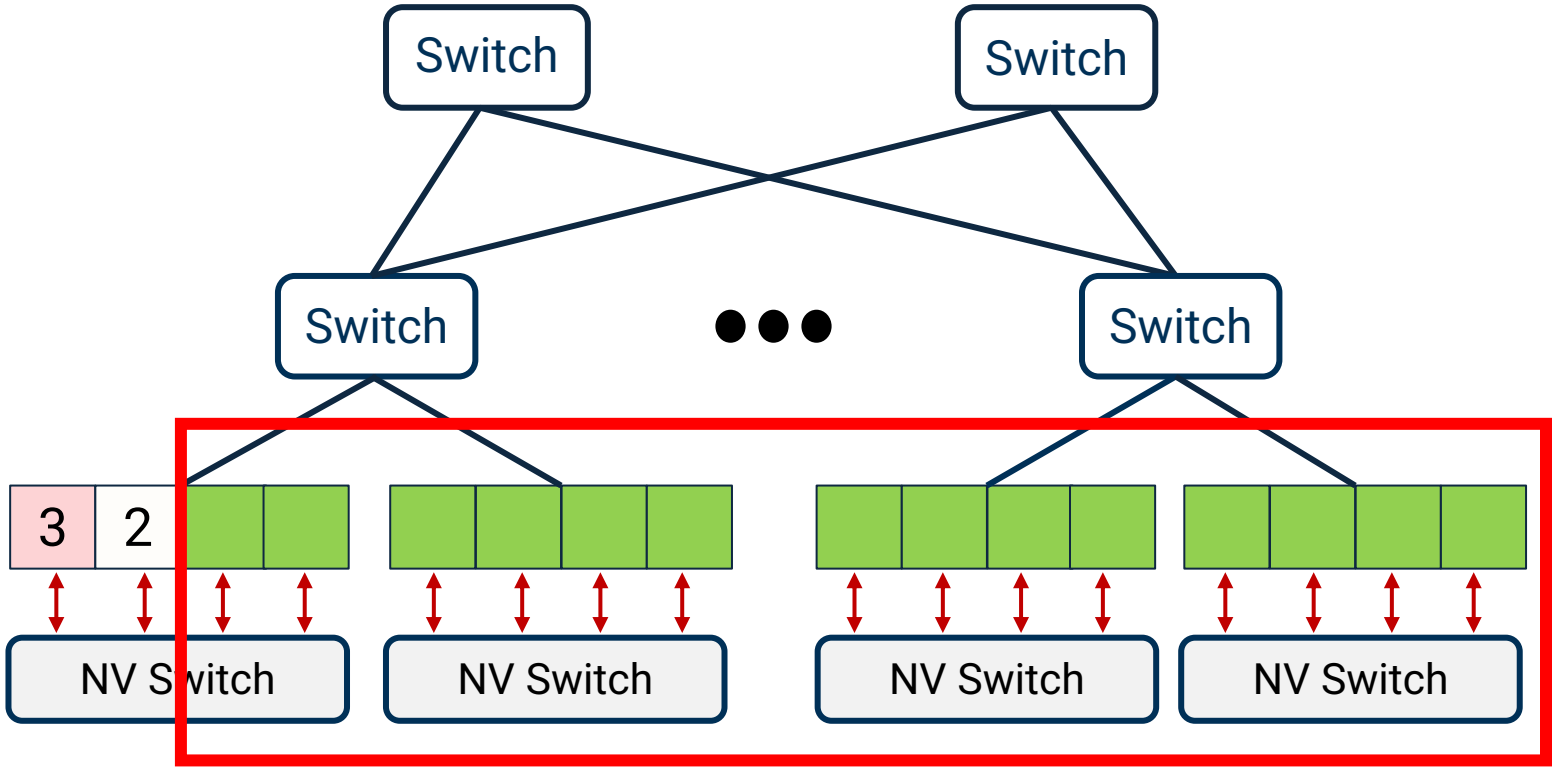
3

We want to
place this layer

2

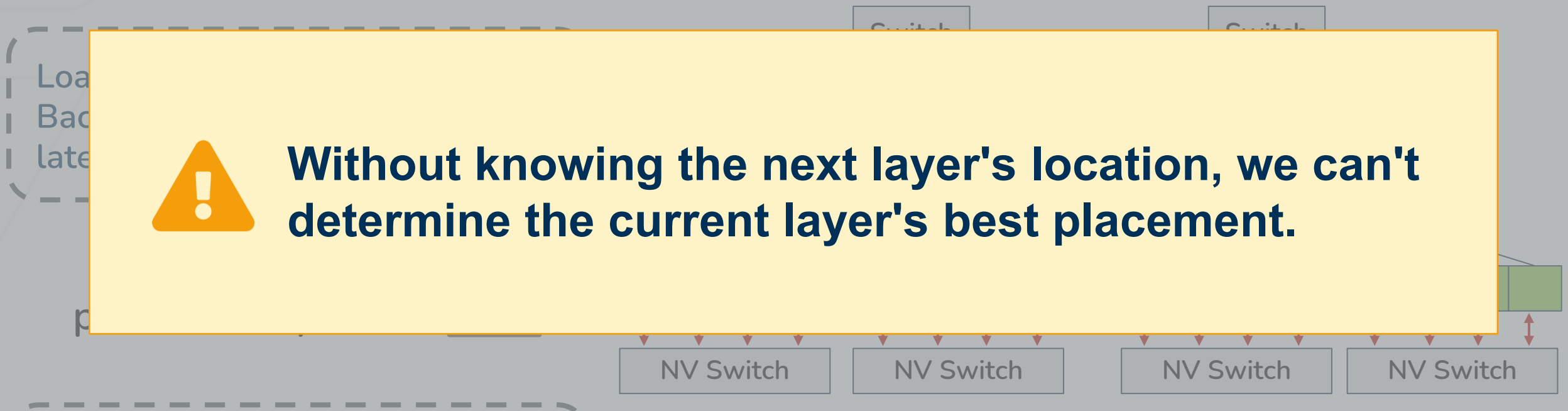
Potential placement options
Forward pass
latency unknown

1



The Unknow Producer Problem

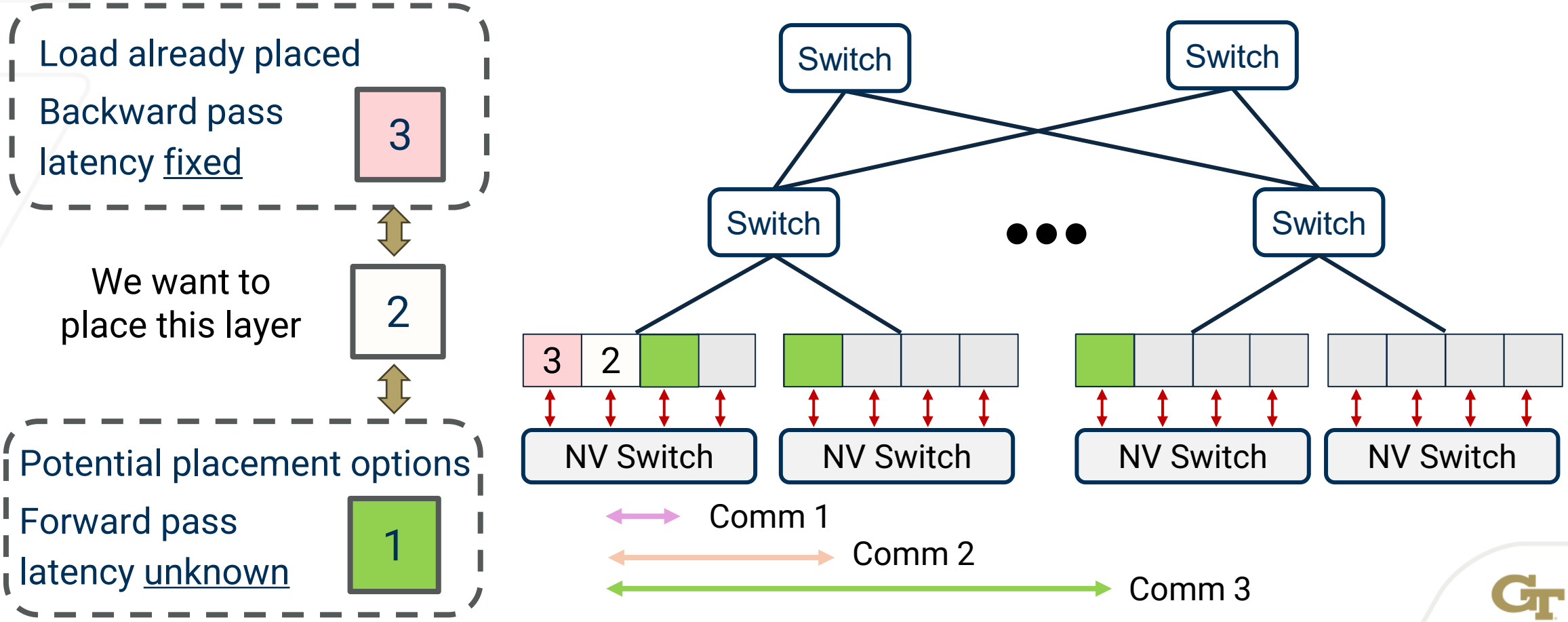
When network is considered, we not only need to decide the **pipeline stage partition**, but also **the placement of each stage**.



Without knowing the next layer's location, we can't determine the current layer's best placement.

Insight 2: Network Layer Abstraction

Unknown placements are abstracted as a discrete communication levels



Insight 2: Network Layer Abstraction

Group devices by communication locality

Instead of N^2 device pairs, the DP tracks just a handful of network levels.

Level 0

Intra-node

NVLink / NVSwitch

≈ 900 GB/s (H100)

Level 1

Intra-rack

Leaf switch

≈ 100 GB/s

Level 2

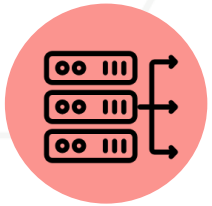
Inter-rack

Spine / oversub.

≈ 12.5 GB/s

dp [ℓ] [D] [k] [s] : Each entry in the DP table tracks cost per communication level

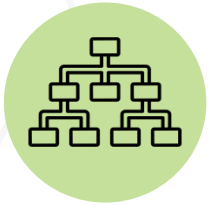
Our Approach



Strategy Explosion

Categorize parallelism along orthogonal dimensions.

SUB-GRAPH (per-layer) is pre-profiled offline; GRAPH-GLOBAL is the DP search axis.



Heterogeneous Networks

Replace device pairs with a few network levels.

Only recur over 3-5 network levels while maintaining optimality .

Memory Footprint

Track memory inside the DP, prune infeasible states.

ZeRO and recompute become incremental knobs the optimizer can turn on as needed.

Insight 3: Integrated Memory Optimizations

dp [ℓ] [D] [k] [s]: Each entry tracks the minimum possible latency to execute the partition of:

ℓ : Level	**D : Downset index**	**k : Devices**	**s : Stages**
Network Level	Set of layers	Number of devices	Number of pipeline stages

Eliminate Invalid Subgraph Variants

Operator Graph Memory Profiling

EP = 1
CP = 1

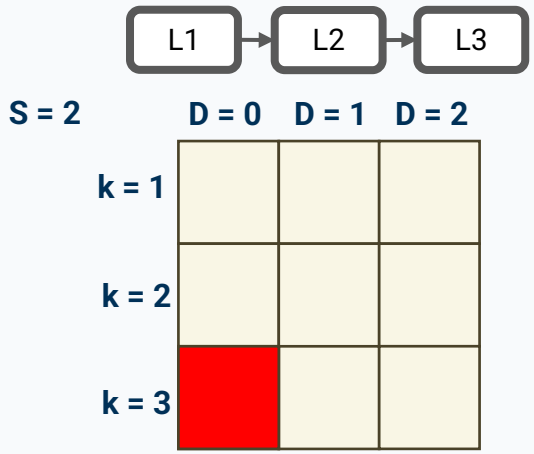
doesn't fit ✗

Apply
Zero / Recomputation / Batch size tuning

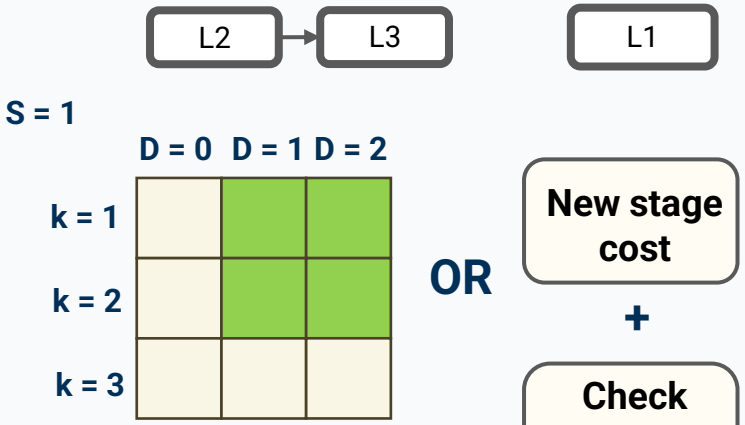
EP = 1
CP = 1

Eliminate if still memory usage still too high

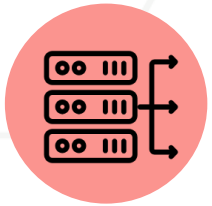
Enter DP Algorithm



Depends on the previous partitions :



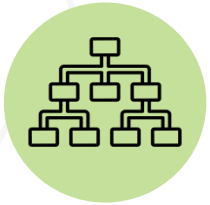
Our Approach



Strategy Explosion

Categorize parallelism along orthogonal dimensions.

SUB-GRAPH (per-layer) is pre-profiled offline; GRAPH-GLOBAL is the DP search axis.



Heterogeneous Networks

Replace device pairs with a few network levels.

Only recur over 3-5 network levels while maintaining optimality .

Memory Footprint

Track memory inside the DP, prune infeasible states.

ZeRO and recompute become incremental knobs the optimizer can turn on as needed.

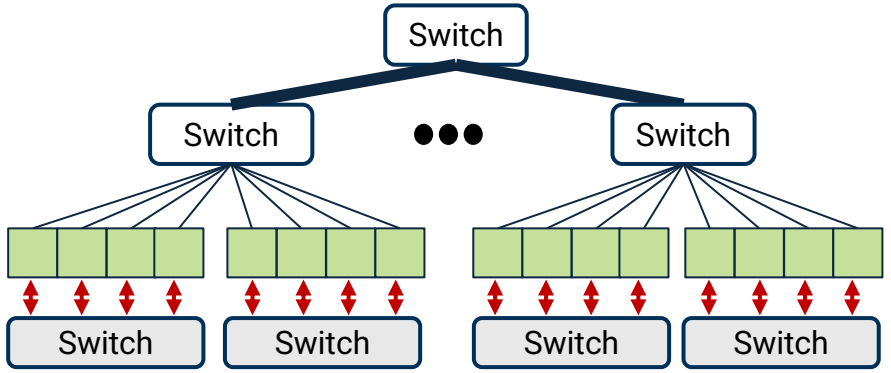
Combined inside a single Dynamic Programming based framework → **NEST**

Evaluations

Evaluation Scenarios

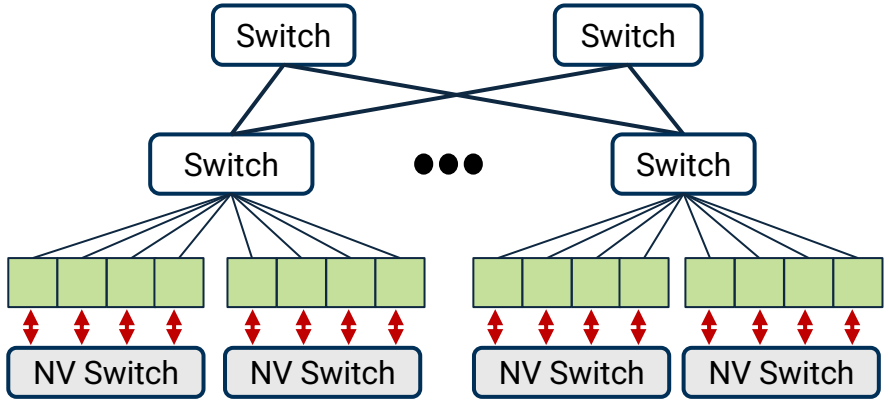
TPUv4 across Fat-Tree:

- Bandwidth: Intra-node: 900GB/s, Inter-node: 100 GB/s

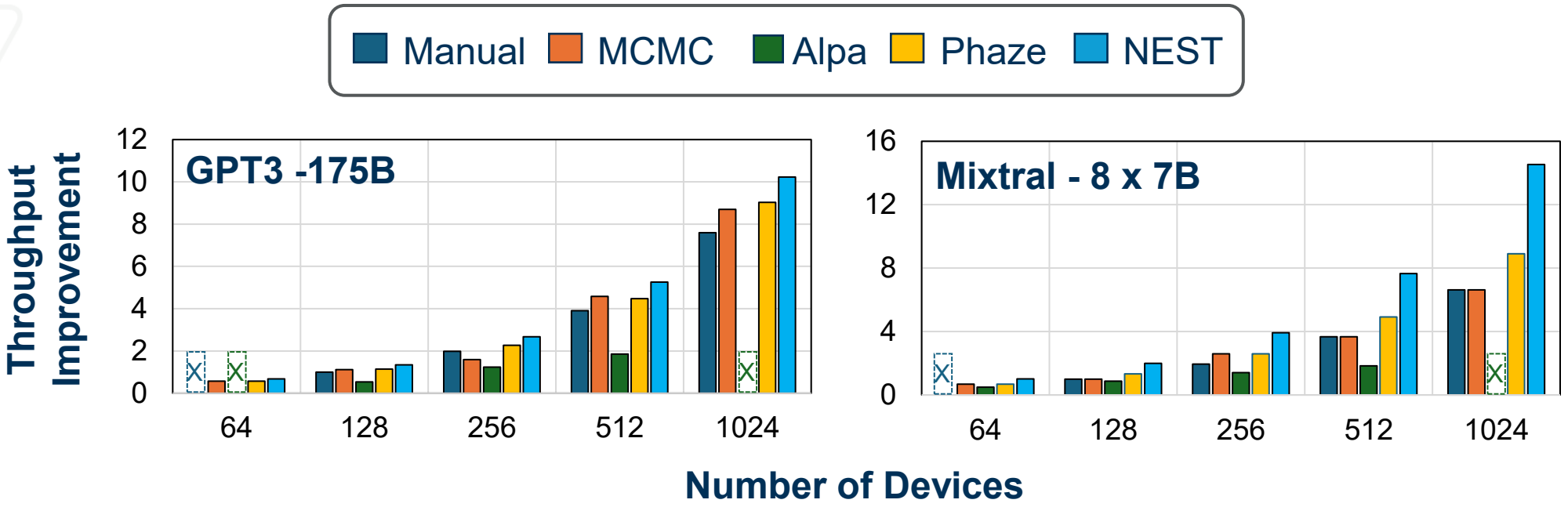


H100 across oversubscribed Spine-Leaf

- Bandwidth: Intra-node: 900GB/s, Inter-node: 12.5 GB/s



Results – Case 1: TPUv4 Fat Tree Topology



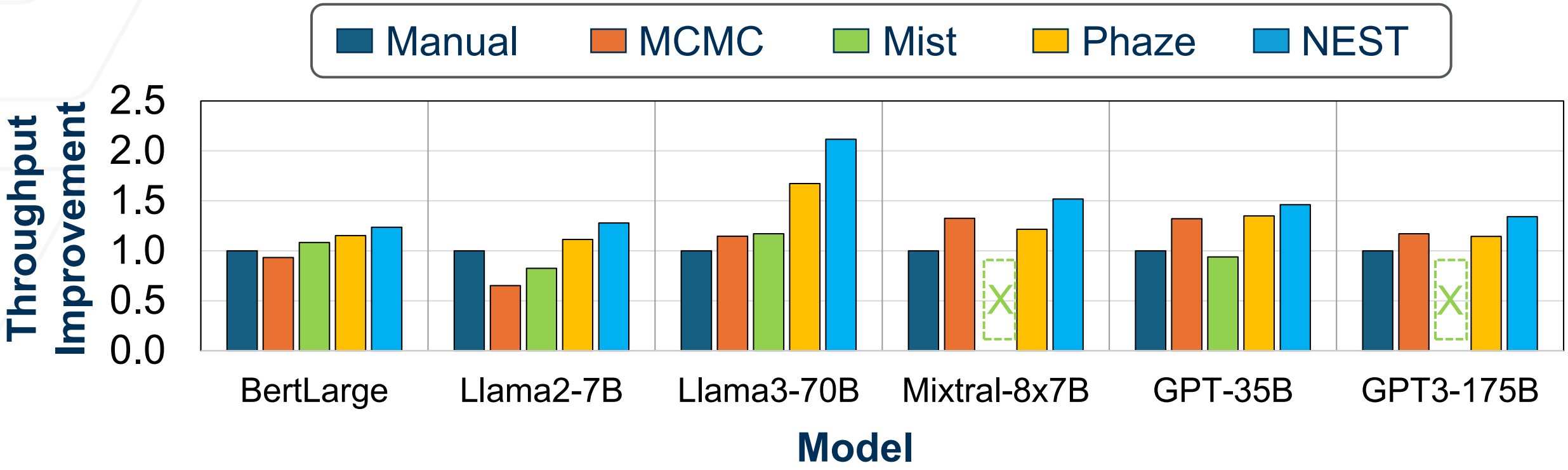
Memory Unaware (Alpa): infeasible on small clusters, over-sharded on large clusters.

Network Unaware (Phaze): Uses similar strategies, but worse comm/compute balance than NEST.

Memory- and Network- aware (NEST): Near-linear scaling by aligning pipeline cuts with bandwidth tiers.

Placement awareness + integrated memory optimization matters !

Results – Case 2: H100 Spine-Leaf Topology



NEST generalizes across hardware and even to constrained networks demonstrating up to **1.5x throughput benefits**

Conclusion

NEST: a network- , compute- and memory aware device placement framework

Unifies model parallelism and placement via **structured dynamic programming**.

NEST enables **efficient, scalable training** across diverse hardware and network. Also provides easy integration to new strategies without exploding the search time

Evaluations demonstrate **consistent gains** in throughput, memory efficiency, and scalability over state-of-the-art baselines.

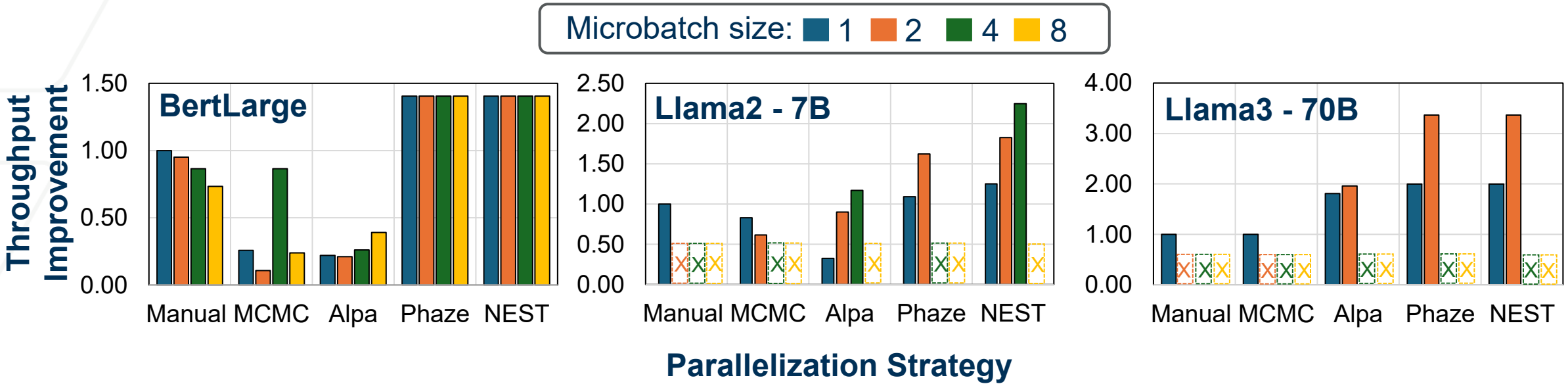
This work provides a foundation for co-designing parallelization strategies and AI datacenter infrastructure

Thank you!

Paper

Code

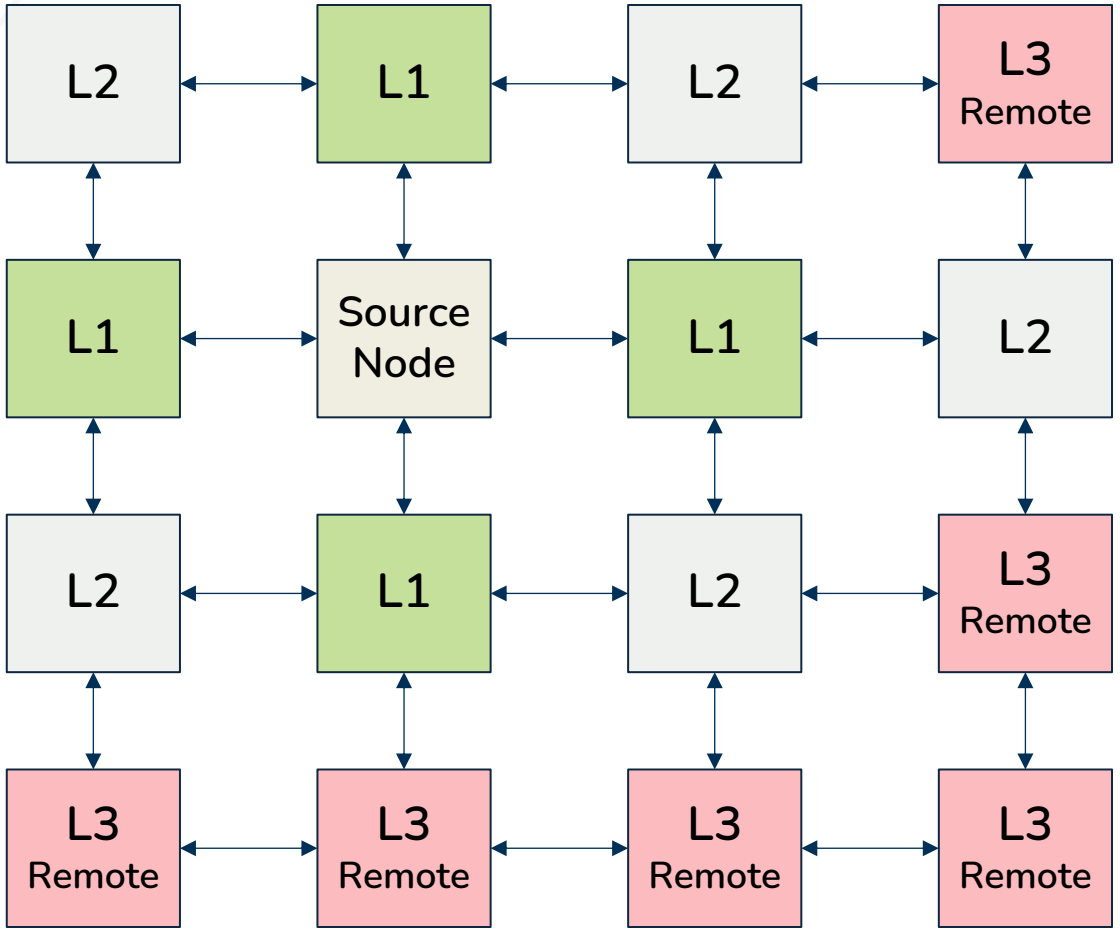
Results – Microbatch scaling



The optimal micro-batch size varies across models and strategies. Changing the micro-batch size shifts compute intensity and memory footprint, altering the ideal parallelism configuration. NEST's efficiency enables joint optimization at scale.

Generalizing to Mesh Topologies

NEST maps physical hop distances to discrete communication levels (l) relative to a reference source node.



- **Level 1(L1):** Immediate neighbors (1-hop) or high-bandwidth local tiles with the lowest communication latency
- **Level 2 (L2):** Nodes at a two-hop distance or connected through intermediate-bandwidth links.
- **Level 3 (L3 / Remote):** Nodes beyond a predefined distance threshold or those connected through lower-bandwidth inter-mesh links.

Georgia Tech