ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms

Saeed Rashidi[*], Srinivas Sridharan[†], Sudarshan Srinivasan[‡] and Tushar Krishna[*]

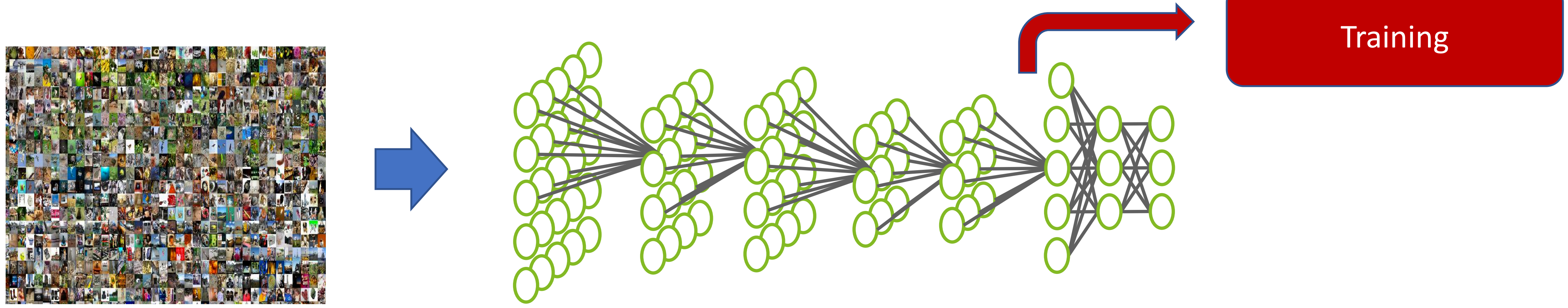
[*]Georgia Institute of Technology

[†]Facebook

[‡]Intel

Distributed Training

- Training a neural network involves using a training dataset to update the model weights to create a good mapping of inputs to outputs.
- Training time is increasing:
  - DNN Networks are becoming bigger (e.g. GNMT, BERT).
  - Training samples are becoming larger (e.g. DLRM).
  - Moore's law is dead!
- Solution?
  - **Distributed training:** scale the training across more compute nodes.

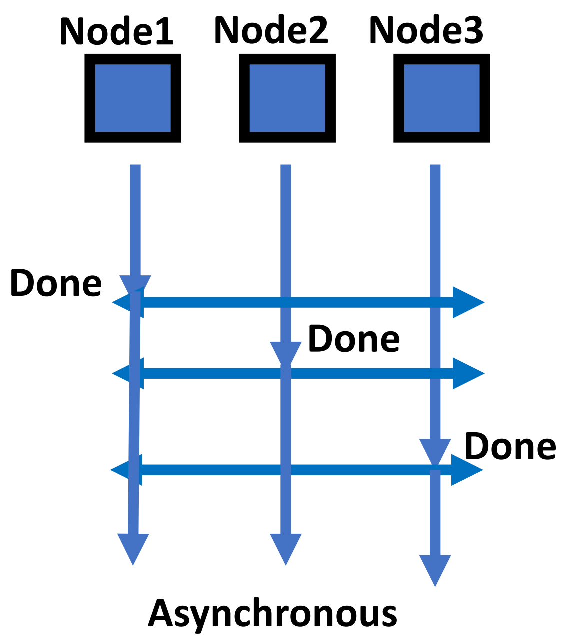


Challenges with Distributed Training

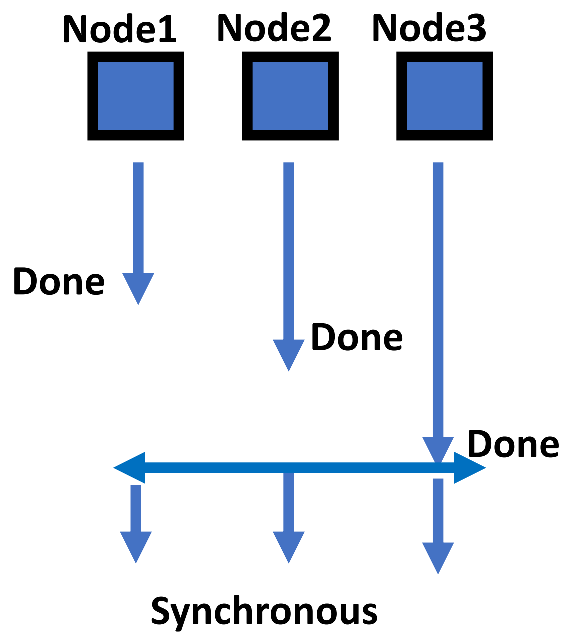
- Communication!
  - Inevitable in any distributed algorithm.
- What does communication depend on?
  - **synchronization scheme:** synchronous vs. asynchronous.
  - **parallelism approach:** data-parallel, model-parallel, hybrid-parallel.
- Is it a problem?
  - Depends … can we hide it behind compute?

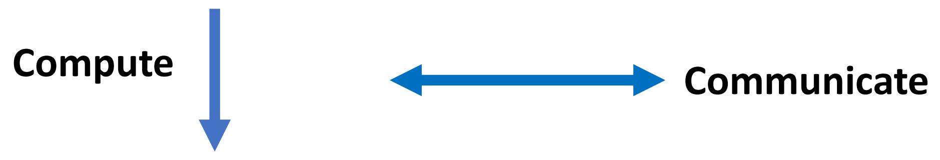
Background: Sync. vs. Async. Training

- Defines when nodes should exchange data.
  - Affects convergence time.



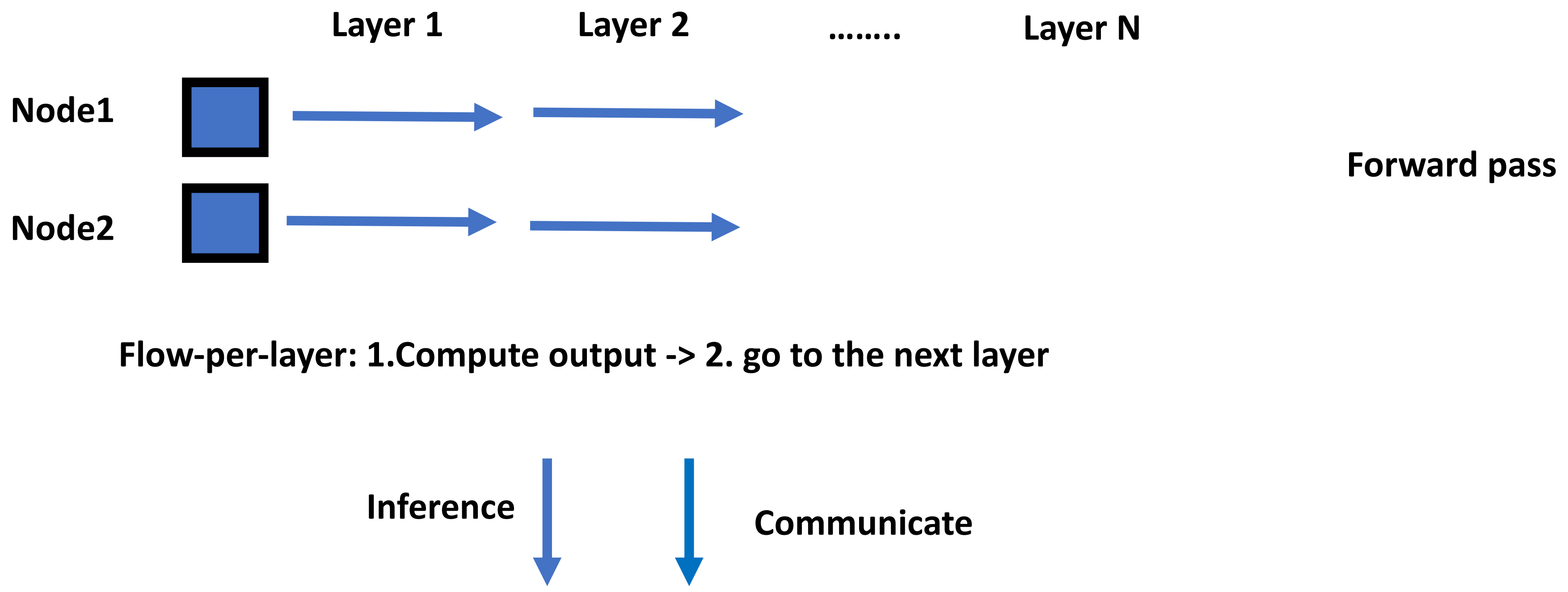
We focus on synchronous training





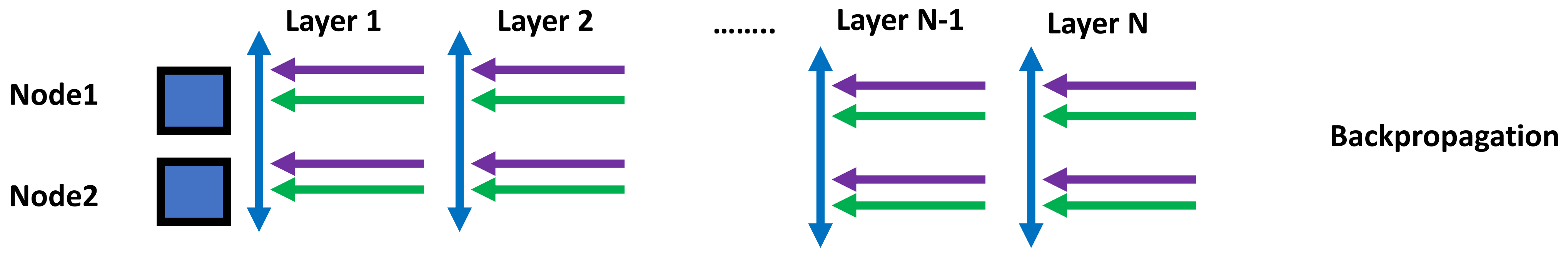
Background: Data-Parallel Training

- Distribute Data across multiple nodes and replicate model (network) along all nodes.
- **No communication** during the forward pass.

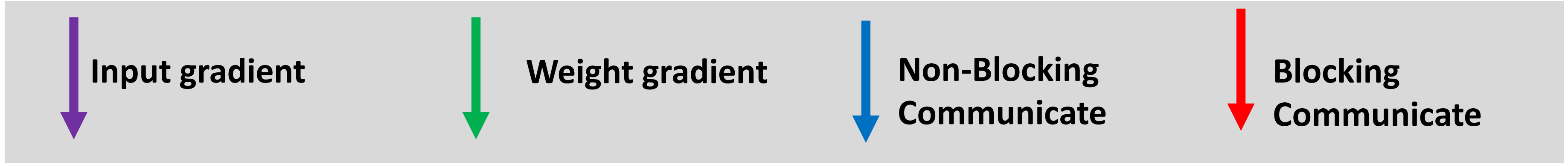


Background: Data-Parallel Training

- Distribute Data across multiple nodes and replicate model (network) along all nodes.
- **Communicate weight gradients** during the backpropagation pass.
  - Blocking wait during forward pass for collective of previous backpropagation for that layer.

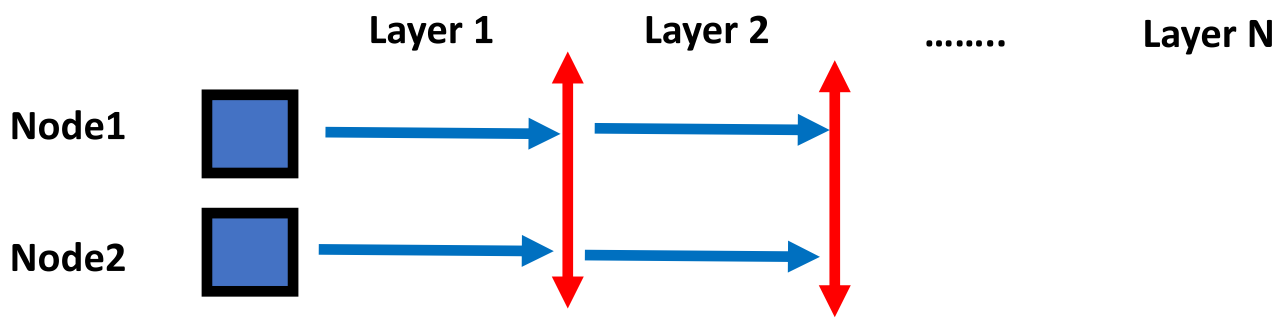


Flow-per-layer: 1.Compute weight gradient-> 2.issue weight gradient comm -> 3.compute input gradient -> 4. go to previous layer



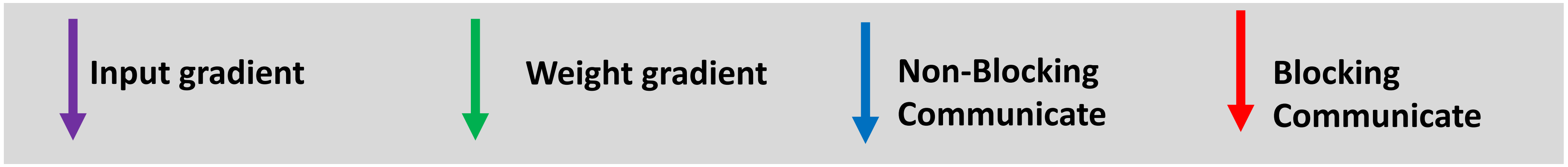
Background: Model-Parallel Training

- Distribute Model across all nodes and replicate data along all nodes.
- **Communicate outputs** during the forward pass.



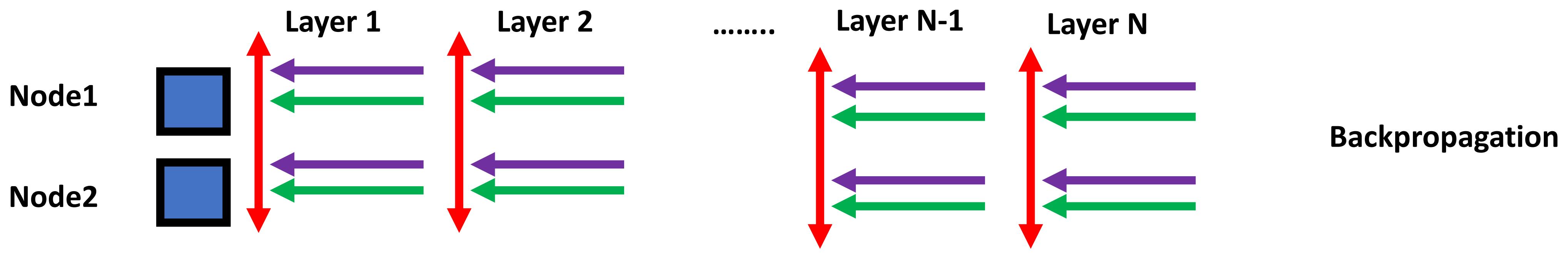
Forward pass

Flow-per-layer: 1.Compute output -> 2. issue output gradient comm -> 3.wait for gradient to be finished -> 4. go to the next layer

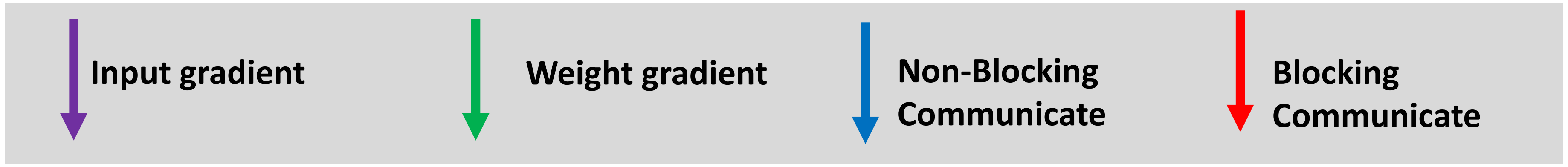


Background: Model-Parallel Training

- Distribute Model across all nodes and replicate data along all nodes.
- **Communicate input gradients** during the backpropagation pass.

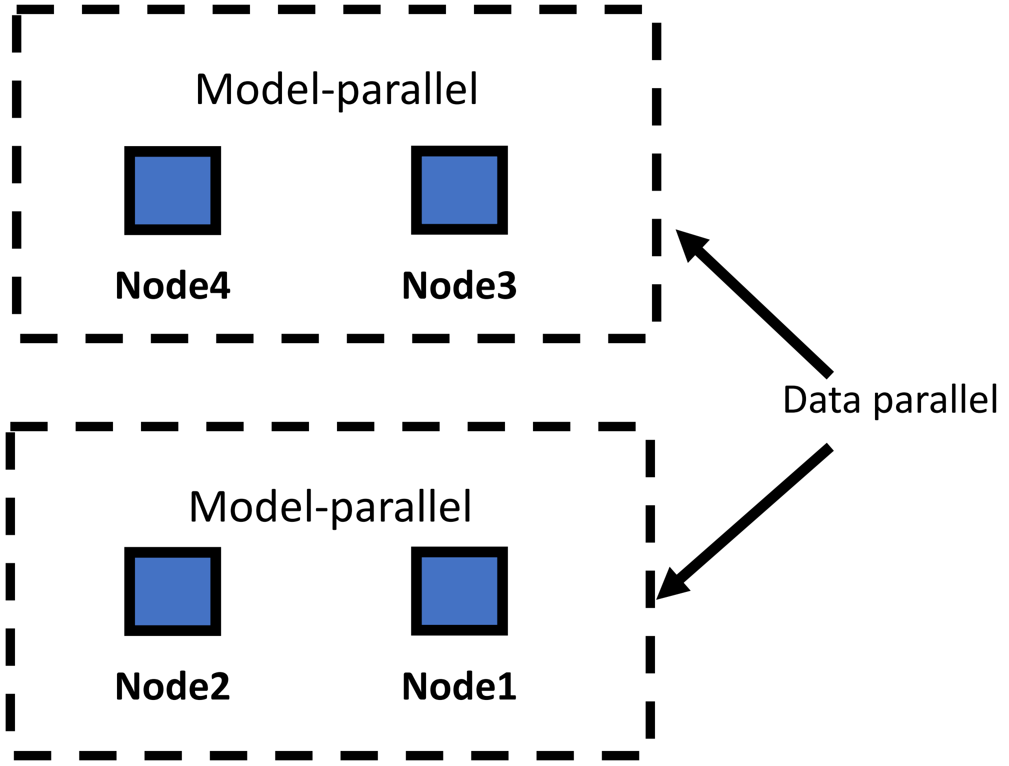


Flow-per-layer: 1.Compute input gradient-> 2.issue input gradient comm -> 3.compute weight gradient -> 4. wait for input gradient -> 5. go to previous layer



Background: Hybrid parallel

- Partition nodes into groups. Parallelism within a group is model-parallel, across the groups is data-parallel, or vice versa.

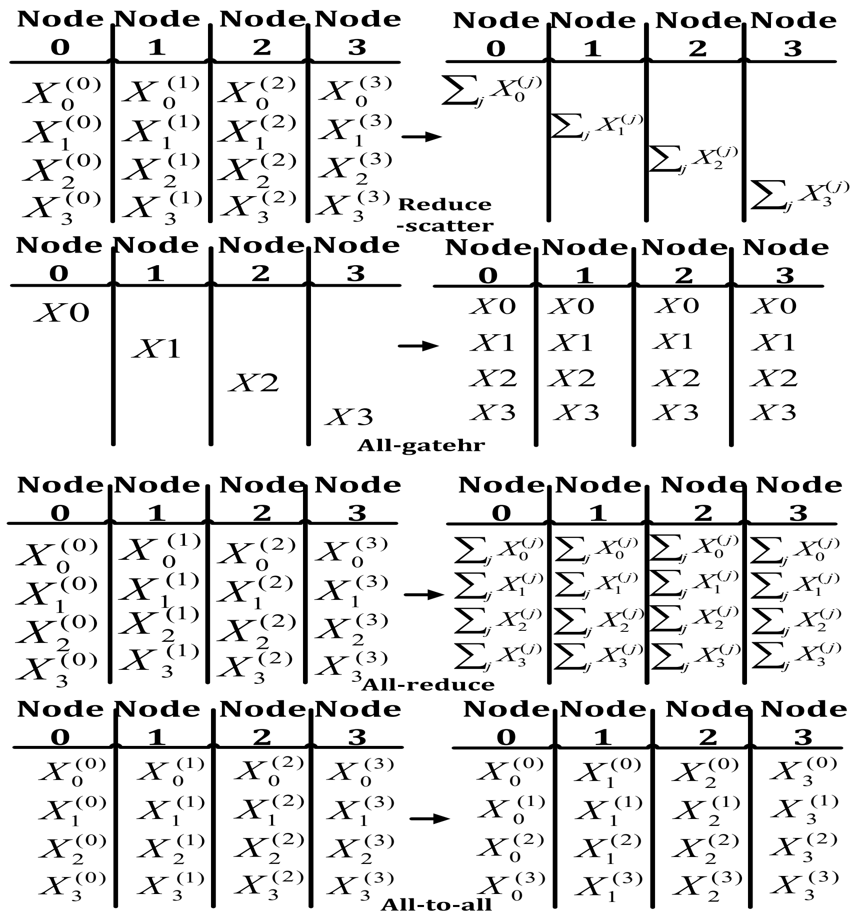


Parallelism	Activations during the forward pass	Weight gradients	Input gradients
Data		✓	
Model	✓		✓
Hybrid	partially	partially	partially

Communication during Distributed Training

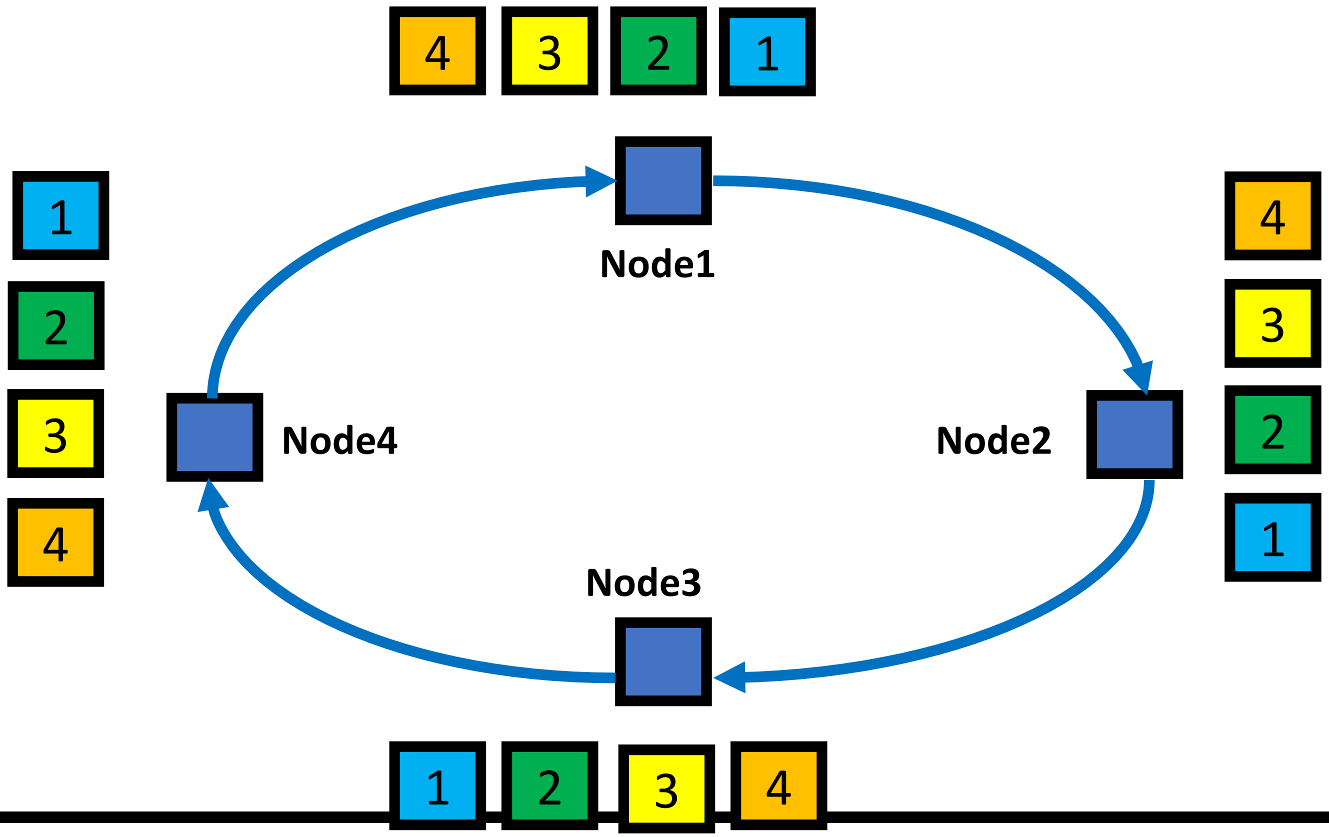
- Distributed Training introduces "**Collective Communication**"
  - All-Reduce
    - Reduce-Scatter + All-Gather
  - All-to-All
  - *One of these or a combination of these can occur depending on the DNN Model and Parallelization Strategy (Model/Data/Hybrid)*

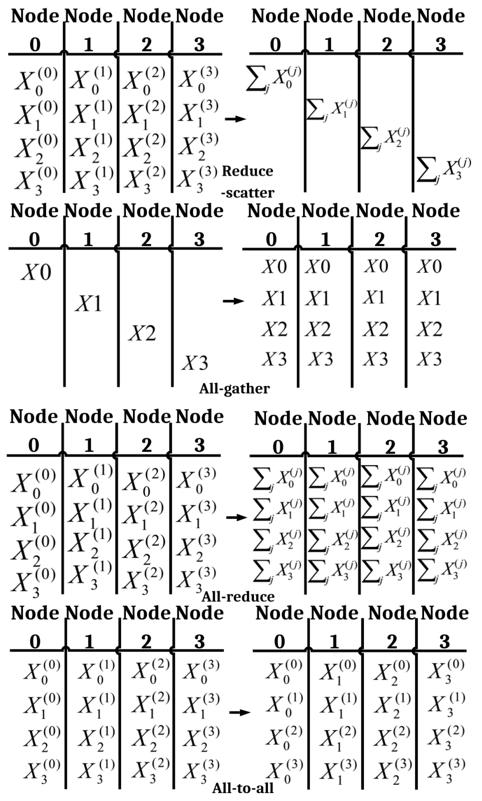
- **Research Questions**
  - What determines the runtime for a collective?
  - What is the compute-communication ratio during Distributed Training?



Example: Ring Based All-Reduce

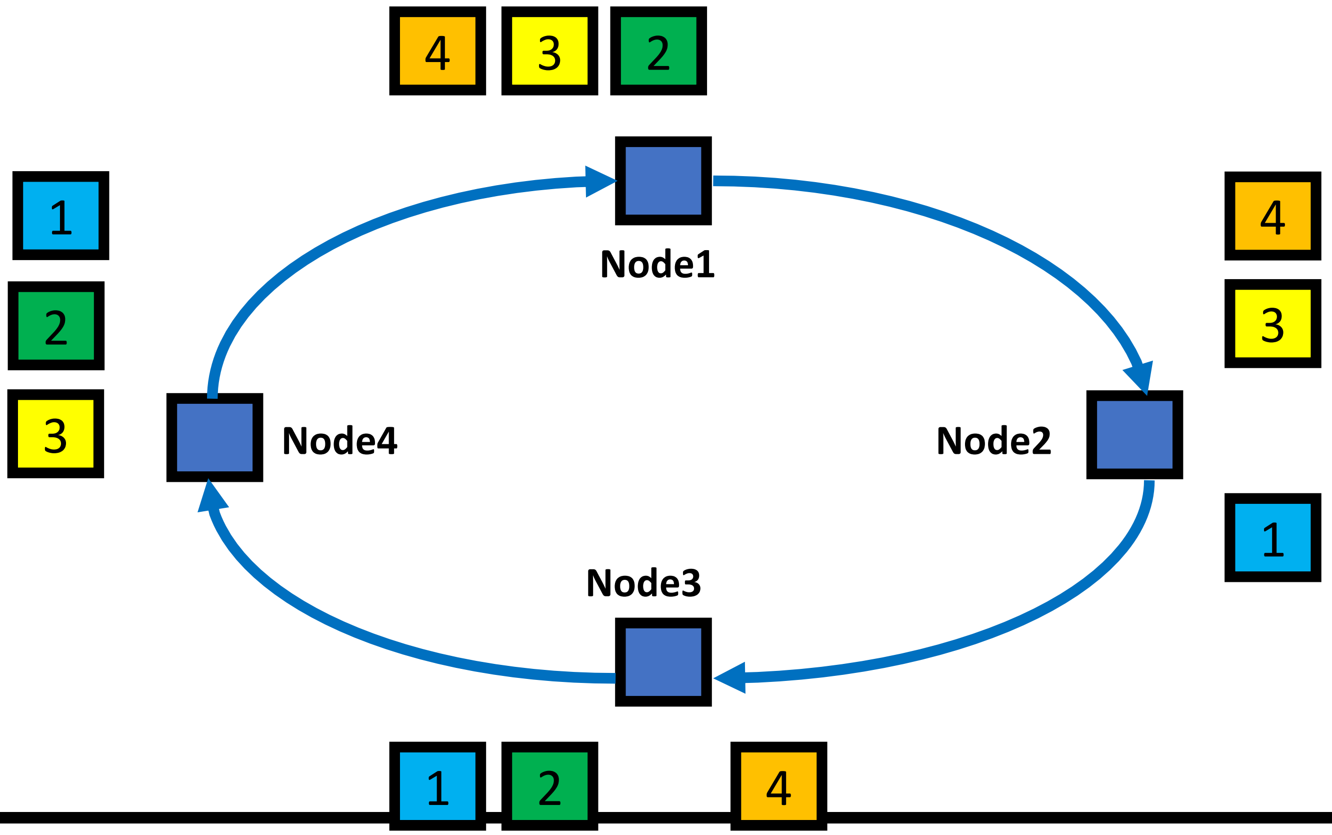
- A ring with N nodes partitions data to N messages
- Collective Communication Flow:

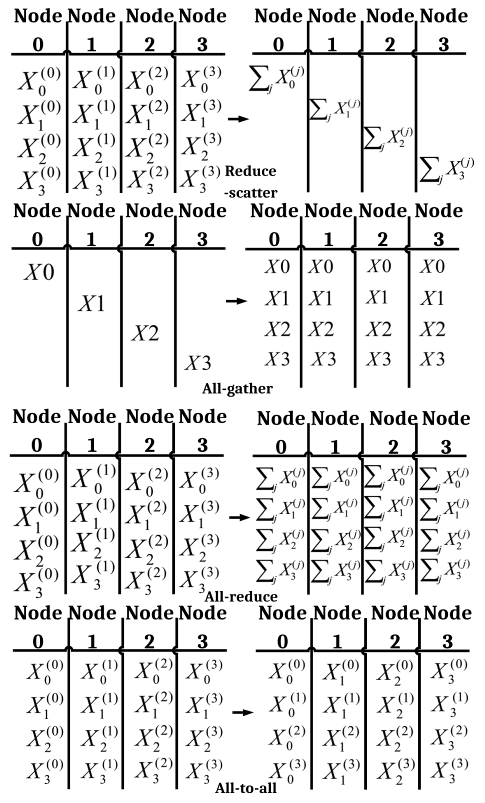




Example: Ring Based All-Reduce

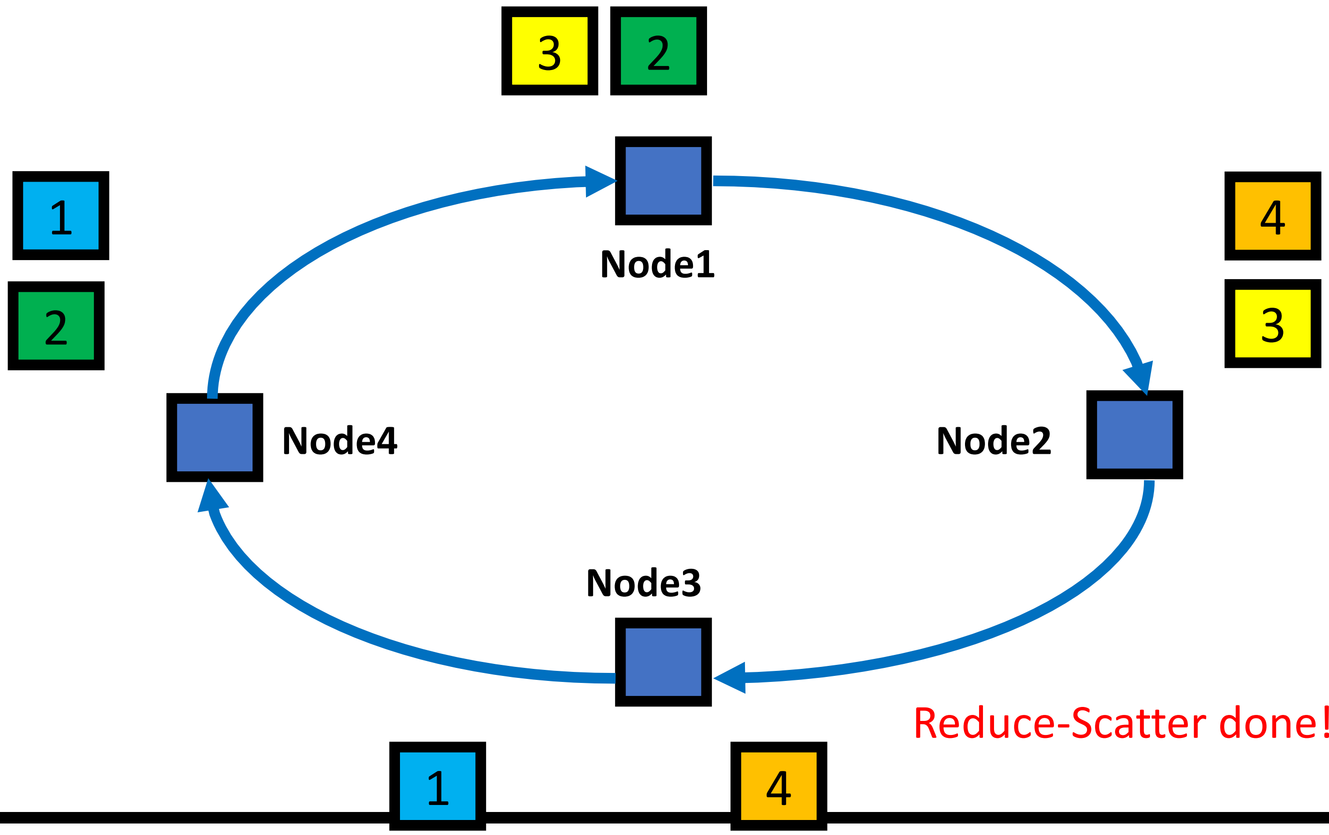
- A ring with N nodes partitions data to N messages
- Collective Communication Flow:



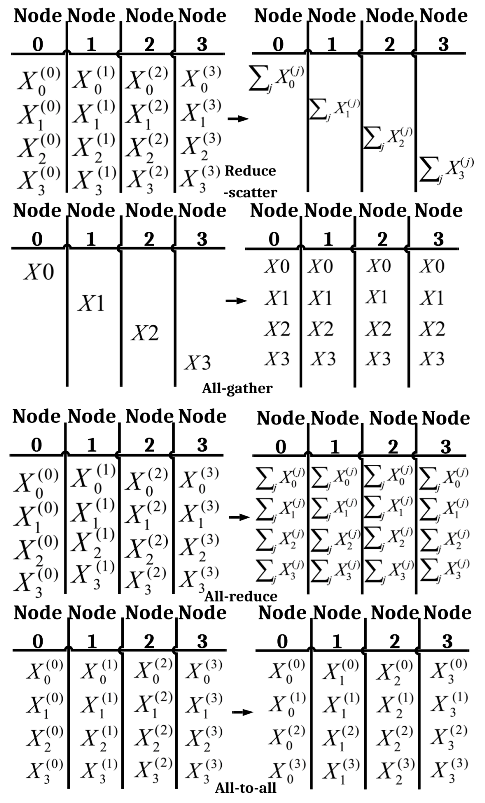


Example: Ring Based All-Reduce

- A ring with N nodes partitions data to N messages
- Collective Communication Flow:

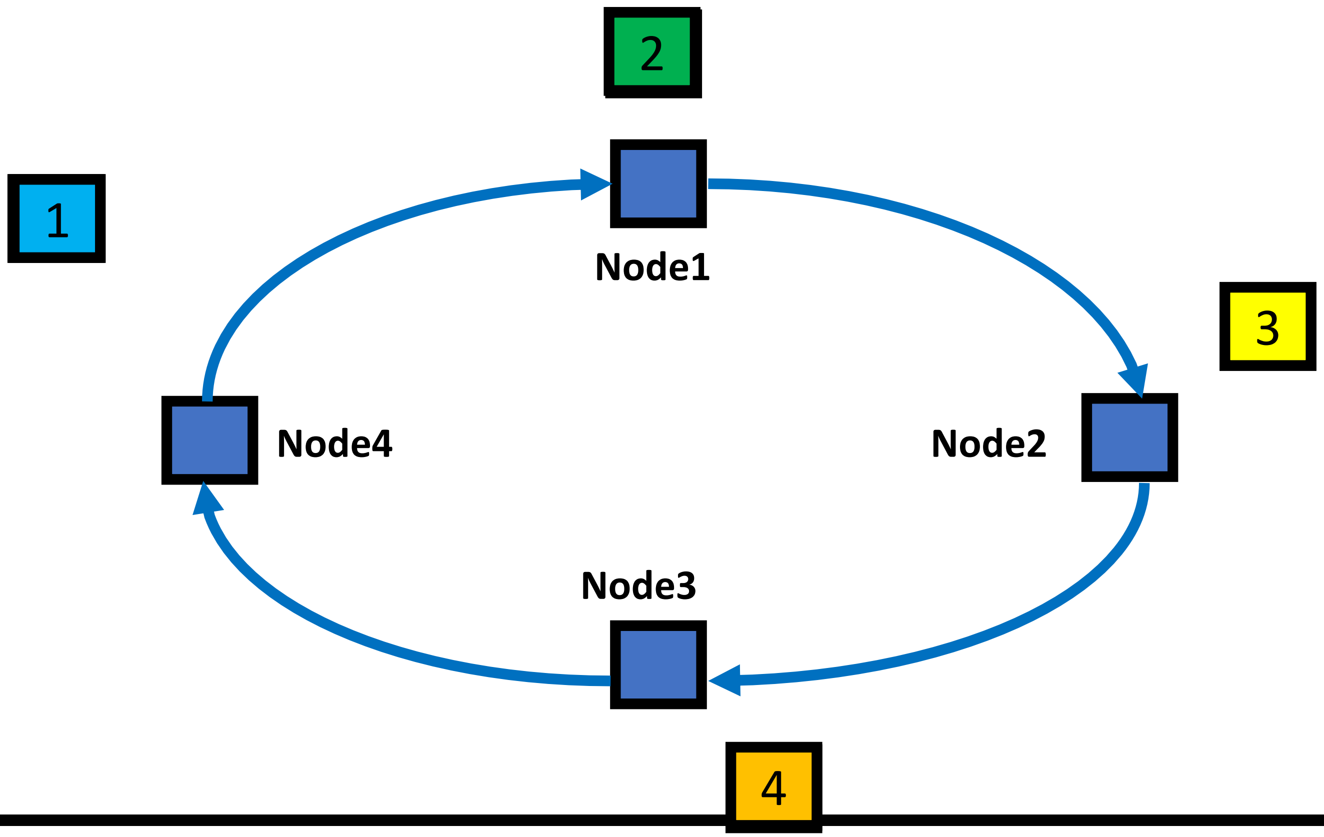


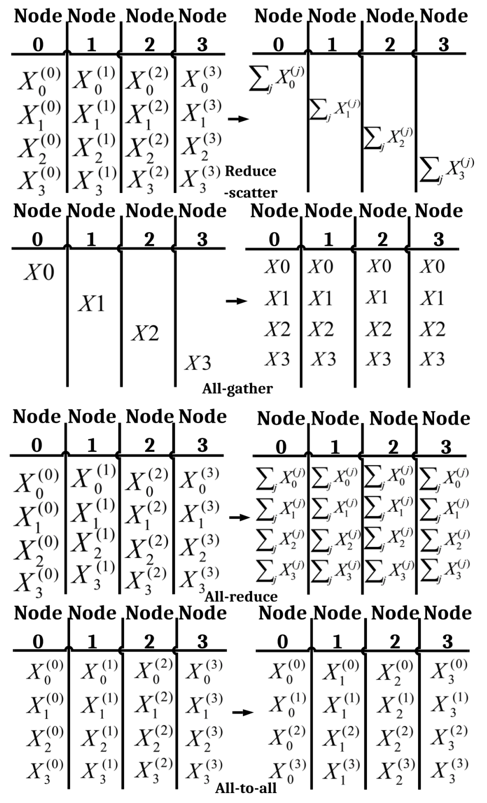
Reduce-Scatter done!



Example: Ring Based All-Reduce

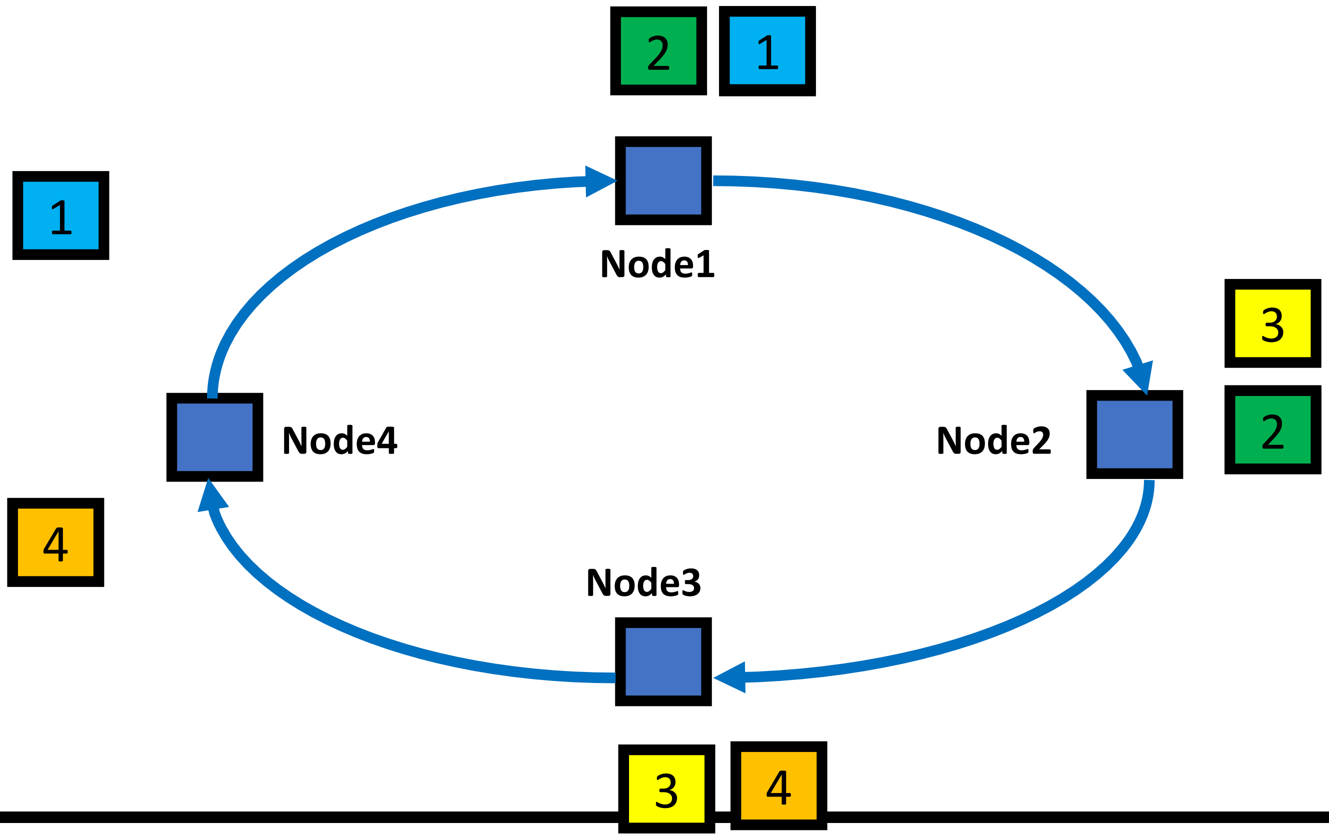
- A ring with N nodes partitions data to N messages
- Collective Communication Flow:

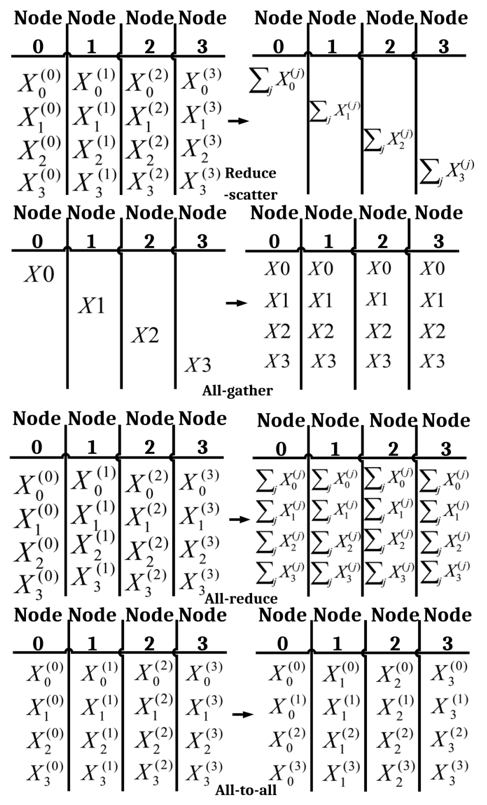




Example: Ring Based All-Reduce

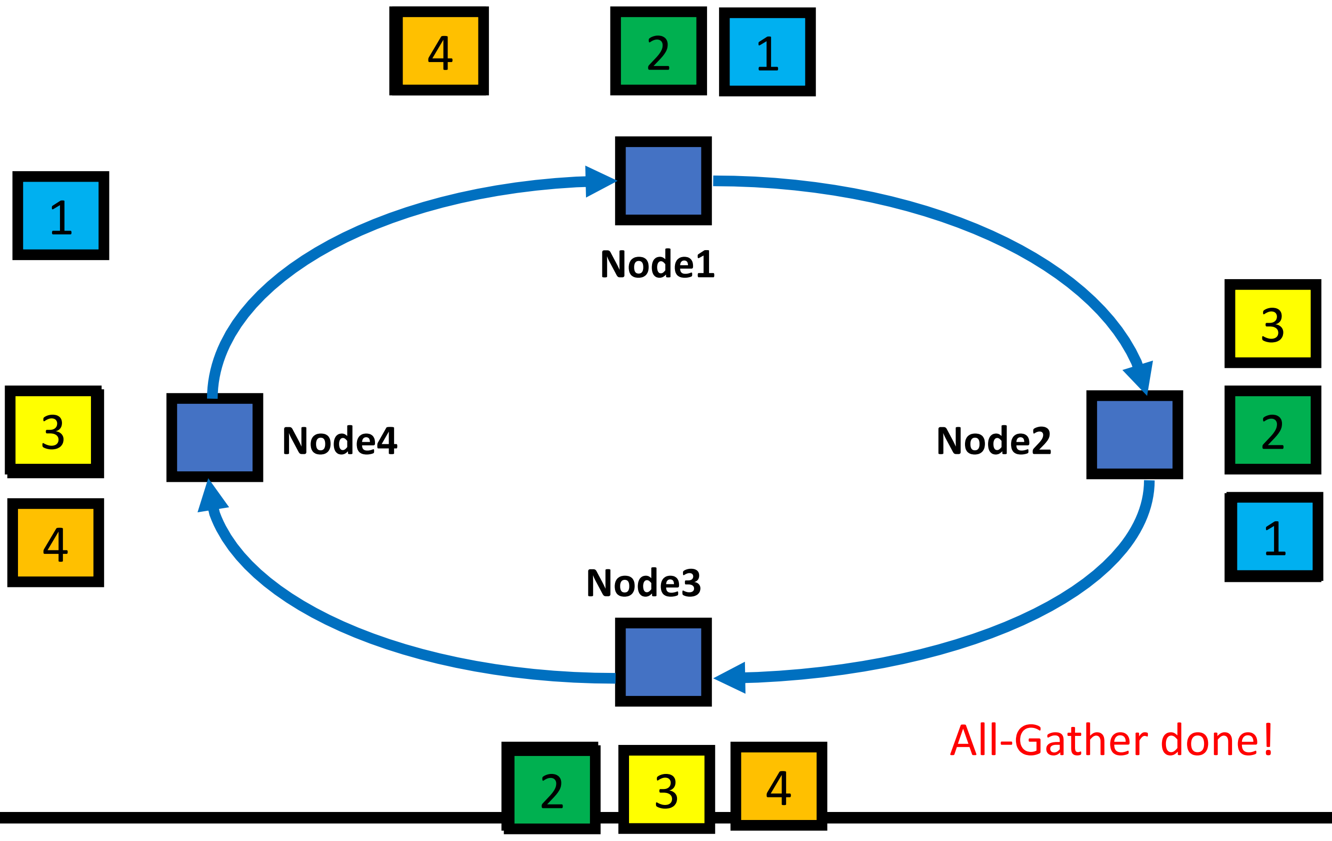
- A ring with N nodes partitions data to N messages
- Collective Communication Flow:



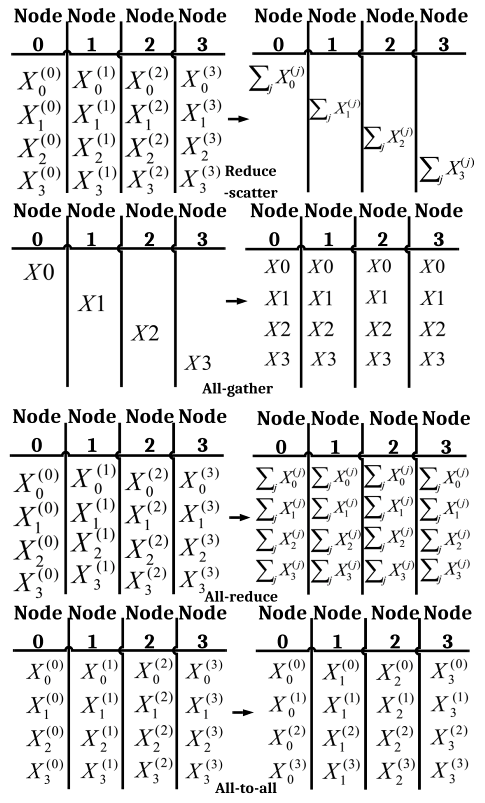


Example: Ring Based All-Reduce

- A ring with N nodes partitions data to N messages
- Collective Communication Flow:



All-Gather done!

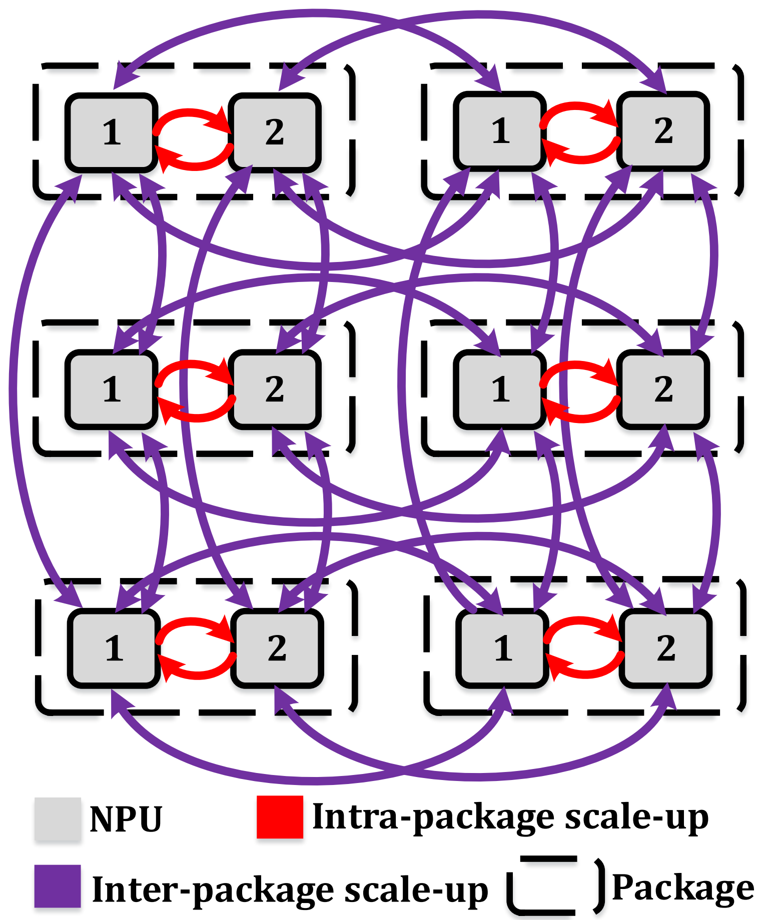


Case Studies

Heterogeneous Bandwidth

Multi-phase Collectives

Torus 3D



Similar to Google TPU

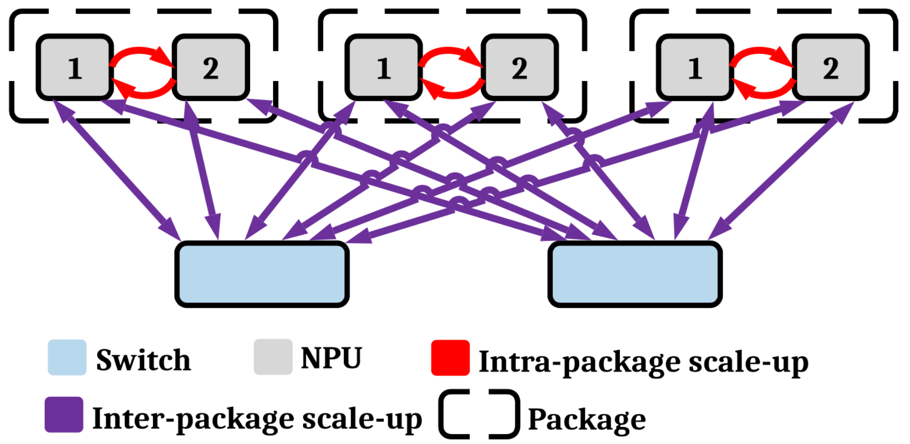
Hierarchical all-reduce:

- Reduce-scatter within package
- All-reduce across rows
- All-reduce across columns
- All-gather within package

AllToAll

Hierarchical all-reduce:

- Reduce-scatter within package
- All-reduce across switch
- All-gather within package



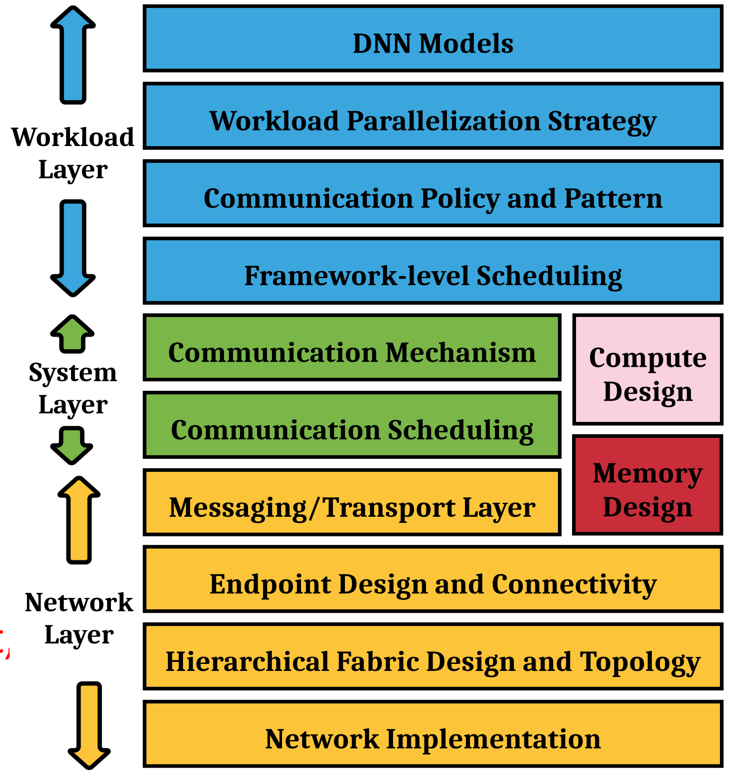
Similar to NVIDIA DGX2

How to Model and Evaluate the Communication Effect

- It is a complex problem and can be viewed as three layers :
  - 1. Workload layer (the training loop):
    - Parallelism approach
    - Compute power
    - Communication size & type and dependency order
  - 2. System layer:
    - Collective communication algorithm
    - Chunk size, schedule of collectives
  - 3. Network layer:
    - Physical topology
    - Congestion control, communication protocol
    - Link BW, latency, buffers, routing algorithm

Not too many tools cover these aspects

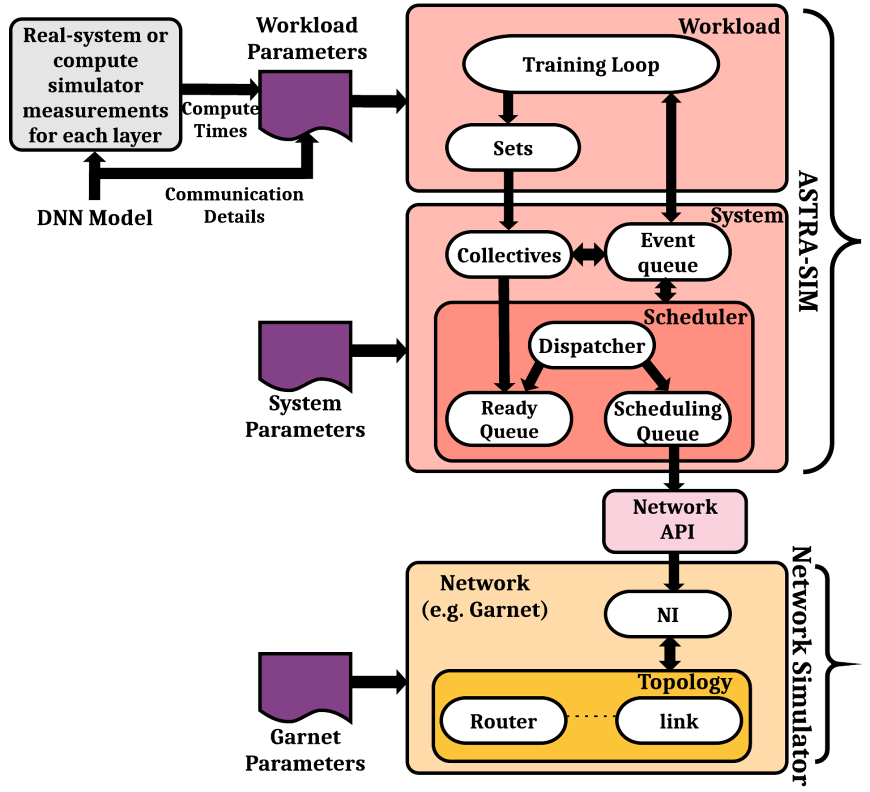
Many tools In this area (e.g., Garnet, NS3)



DNN Models
Workload Parallelization Strategy
Communication Policy and Pattern
Framework-level Scheduling
Workload Layer
Communication Mechanism
Communication Scheduling
Compute Design
Memory Design
System Layer
Messaging/Transport Layer
Endpoint Design and Connectivity
Hierarchical Fabric Design and Topology
Network Implementation
Network Layer

ASTRA-SIM Architecture: Current

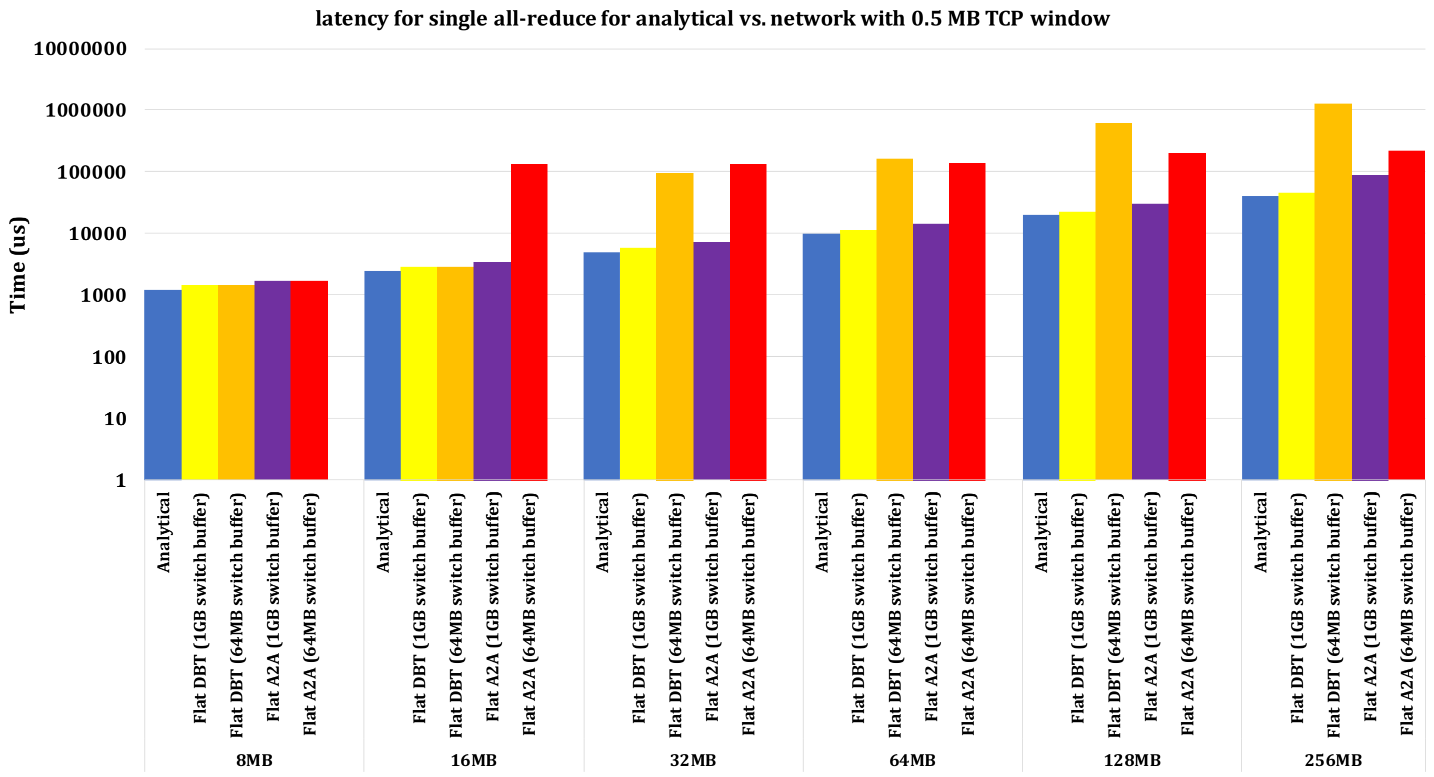
- Workload layer:
  - Supports Data-Parallel, Model-Parallel, Hybrid-Parallel training loops
  - Easy to add new arbitrary training loops
- System:
  - Ring based, Tree-based, AlltoAll based, and multi-phase collectives
  - Easy to add new collective communication
- Network:
  - Supports NS3 and GARNET Network simulator
  - NS3:
    - Supports switch-based topologies
    - Supports TCP and ROCE communication protocol
  - GARNET:
    - Supports switch-based and torus-based topologies
    - Supports credit-based flow control
  - Can add new topologies in both NS3 and GARNET



Why Simulation?

- Model systems (hardware) that do not exist today.
- Analytical modeling insufficient due to actual network congestion.
  - Consider a flat 128-node system with a single all-reduce collective.
    - Difference could be up to 50X!

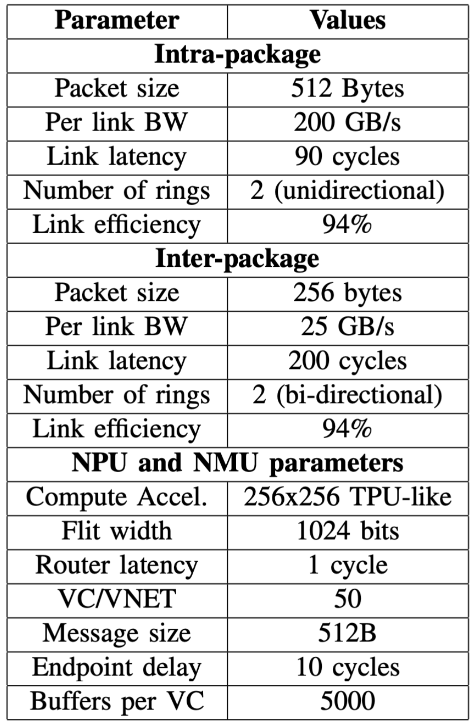
Single all-reduce example:



Simulation Methodology

- Compute model: SCALE-SIM 256*256 TPU-like systolic array simulation.
- Network model: GARNET backend with credit-based flow.
- DNN model: microbenchmark, Transformer, ResNet-50.
- Target systems: alltoall, torus 3D.

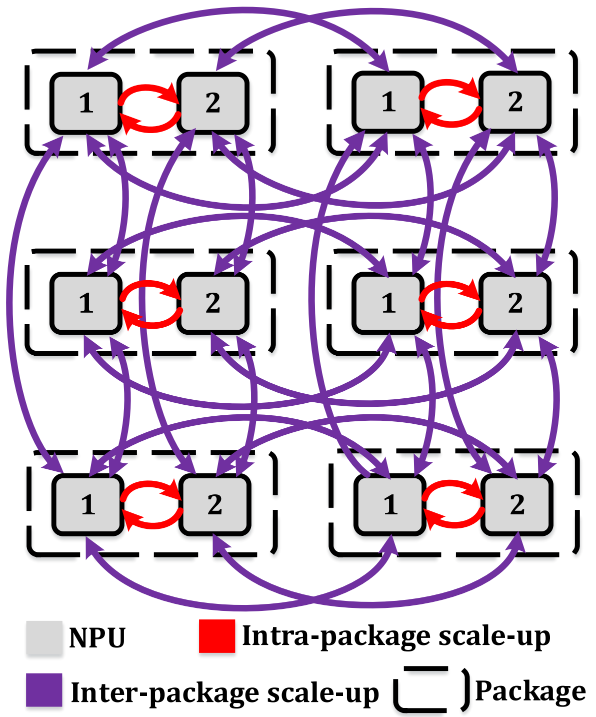
System Parameters



Parameter	**Values**
Intra-package	
Packet size	512 Bytes
Per link BW	200 GB/s
Link latency	90 cycles
Number of rings	2 (unidirectional)
Link efficiency	94%
Inter-package	
Packet size	256 bytes
Per link BW	25 GB/s
Link latency	200 cycles
Number of rings	2 (bi-directional)
Link efficiency	94%
NPU and NMU parameters	
Compute Accel.	256x256 TPU-like
Flit width	1024 bits
Router latency	1 cycle
VC/VNET	50
Message size	512B
Endpoint delay	10 cycles
Buffers per VC	5000

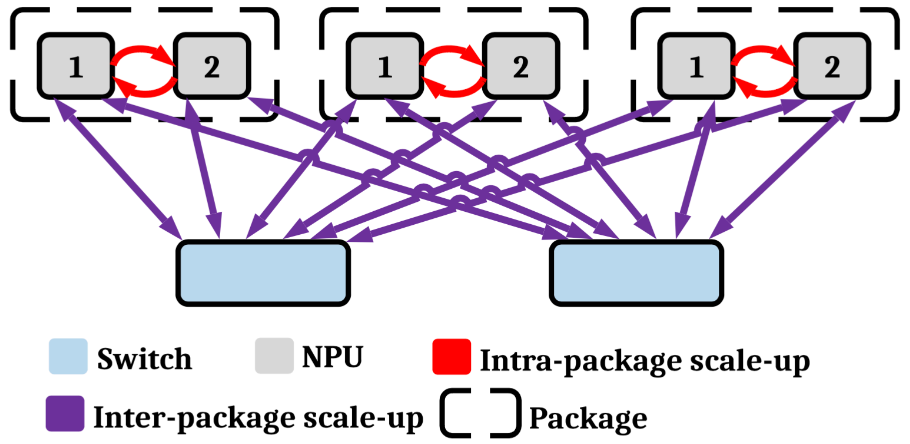
Target Systems

Torus 3D
e.g. 2*2*3 system



M*N*K dimension
M= cores within a package
N= packages in horizonal dimension
K= packages in vertical dimension

Alltoall
e.g. 2*3 system

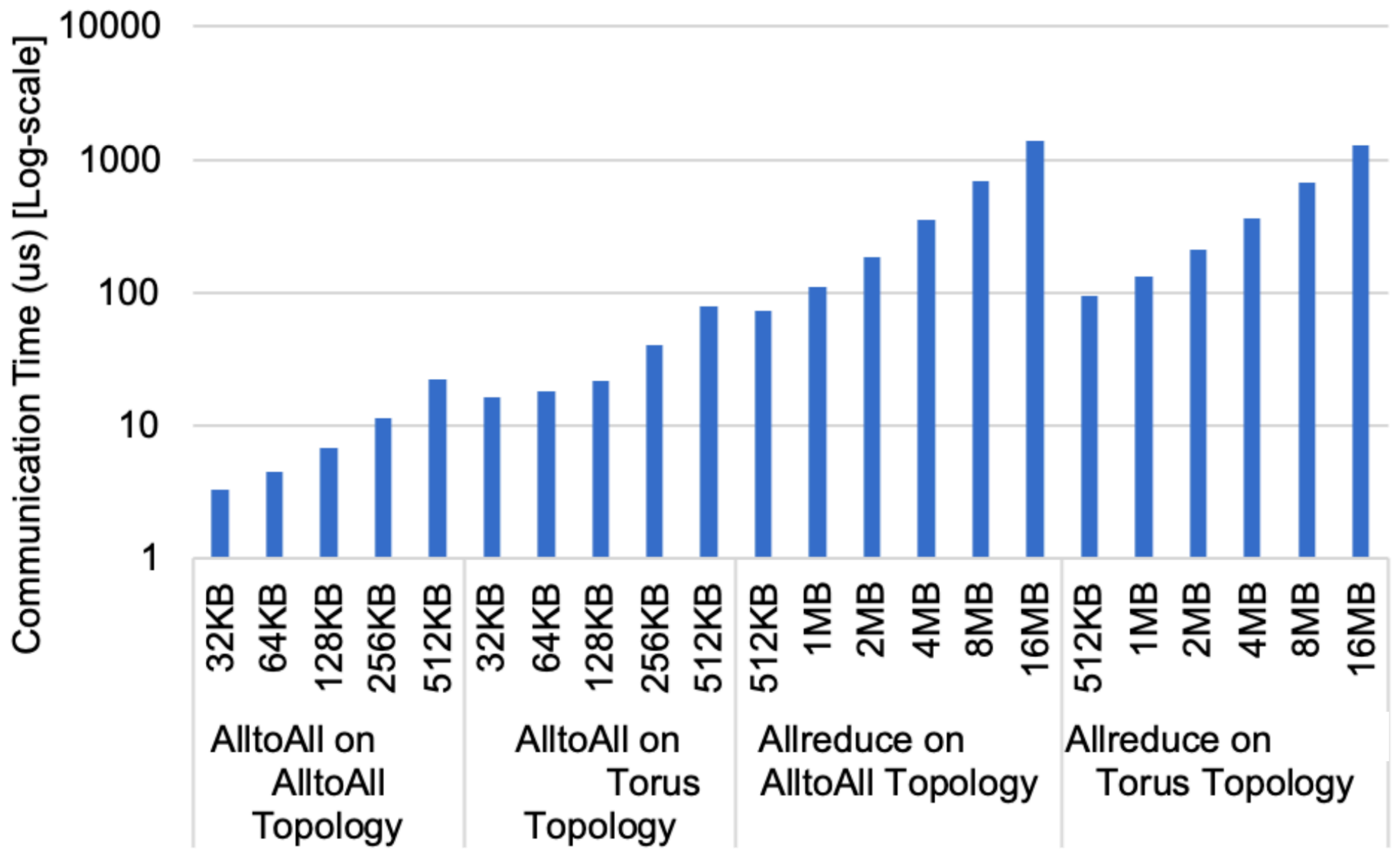


M*N dimension
M= cores within a package
N= packages in alltoall dimension

Microbenchmark: Torus Vs. AllToAll

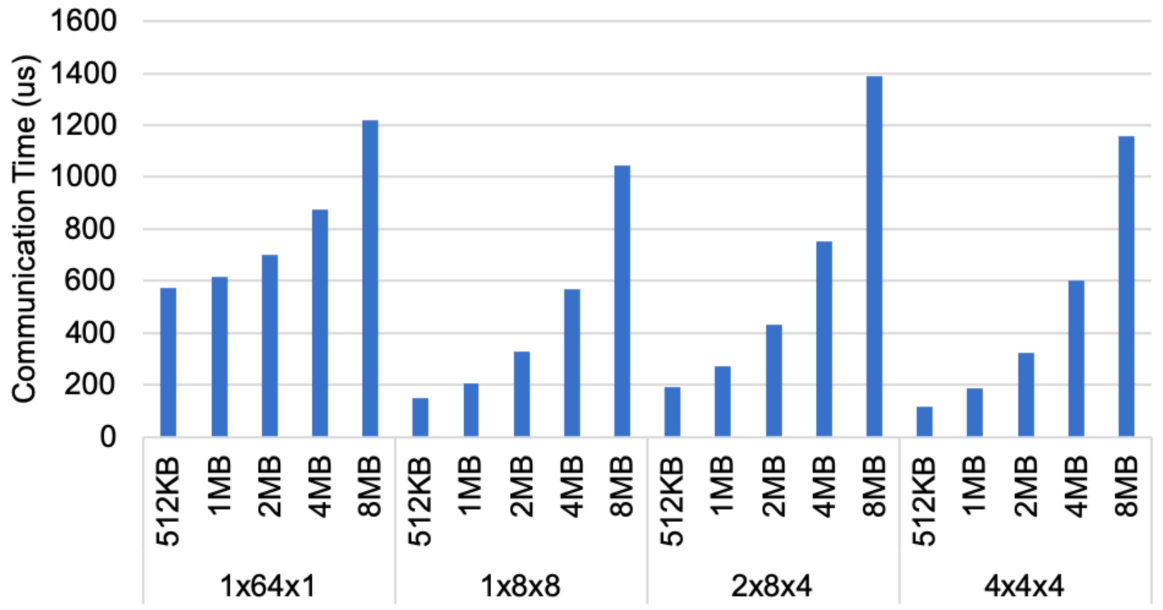
- AllToAll topology works better for all-to-all collective due to less # of steps.
- Torus 3D works better for large all-reduce sizes due to availability of inter-package 8 links compared to 7 links in AllToAll topology.

1*8*1 torus
vs.
1*8 alltoall



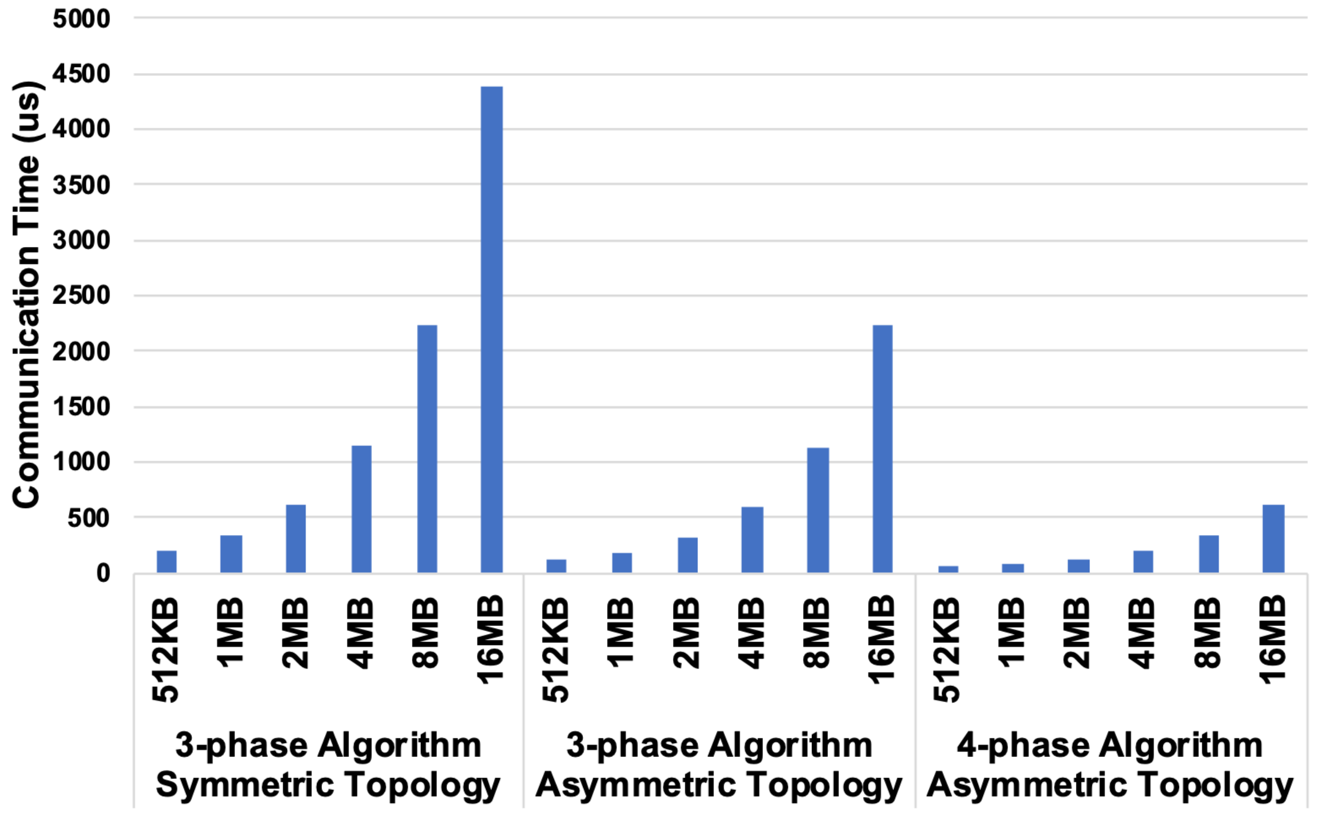
Microbenchmark: Impact of 1D/2D/3D Torus

- Adding a dimension decreases the number of steps per collective. For example, going from 1X64X1 to 1X8X8.
- Adding a dimension might increase amount of data each node sends out (depends on the algorithm). For example, going from 1X8X8 to 2X8X4.
- Hence, choosing a topology is a tradeoff between the above 2 effects.



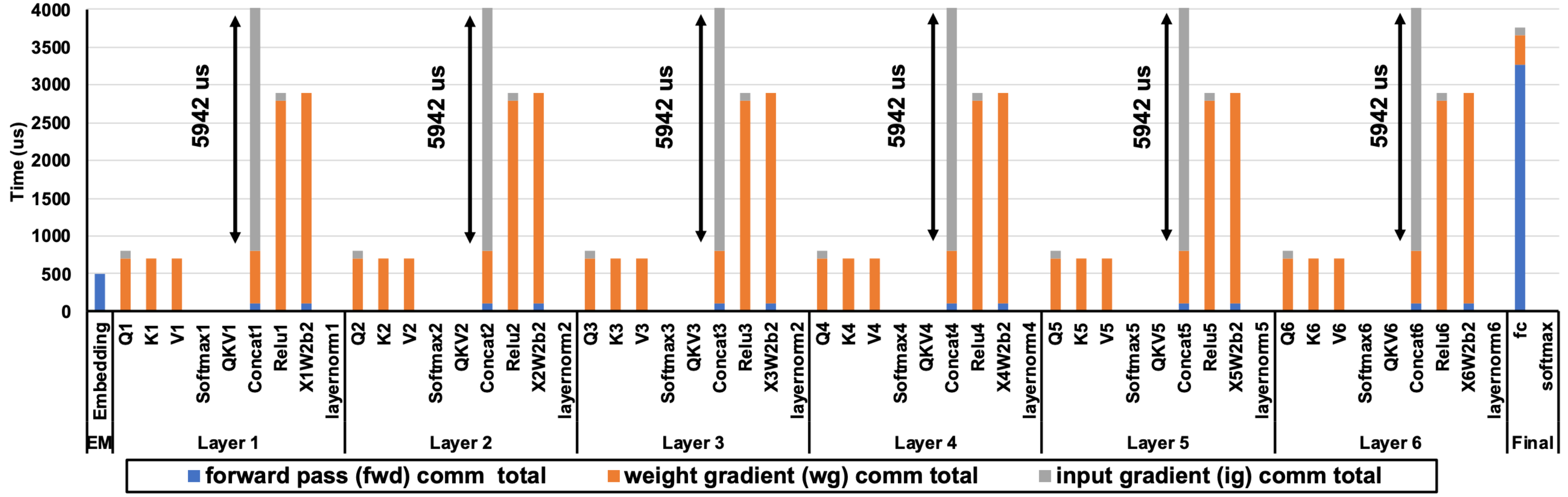
Microbenchmark: Impact of Asymmetric Hierarchical Topology

- Having higher intra-package BW improves the performance.
- We can further improve performance by changing the algorithm to leverage this asymmetric BW.



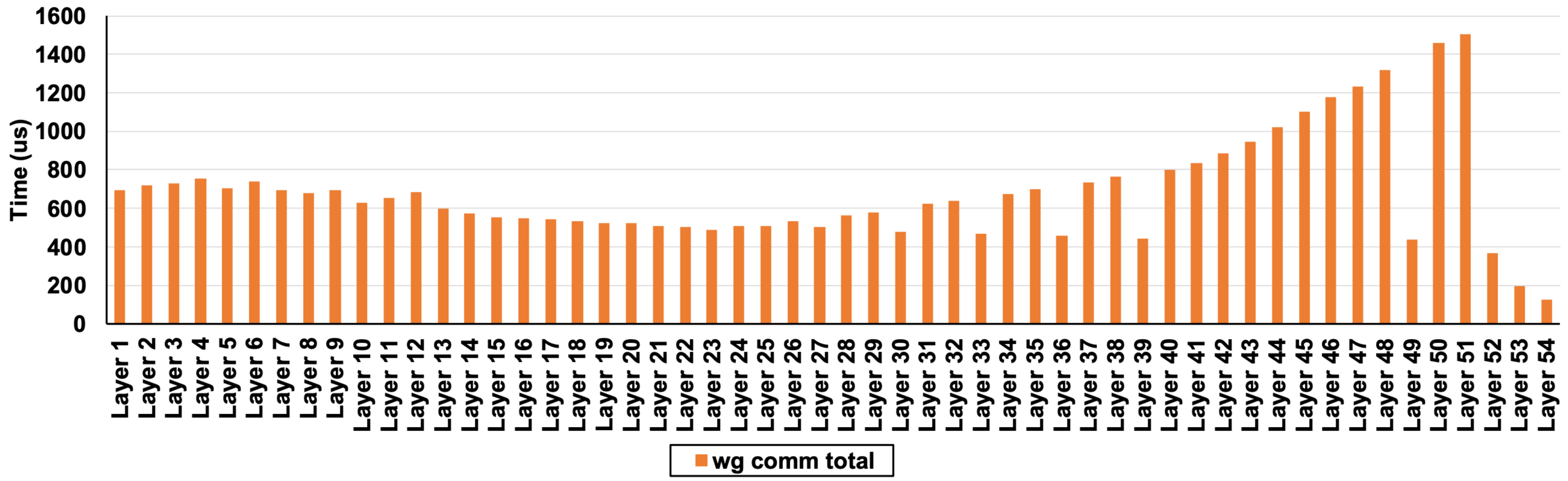
Transformer Layer-Wise Raw Comm Latency

- A Torus 3D with total of 8 (2X2X2) nodes is used.
- Hybrid parallel approach is used (model parallel across local and horizontal dimension, data parallel across vertical dimension).
- Homogeneous comm latency due to explicit ordering and blocking of most of comms in model parallel.



ResNet-50 Layer-Wise Raw Comm Latency

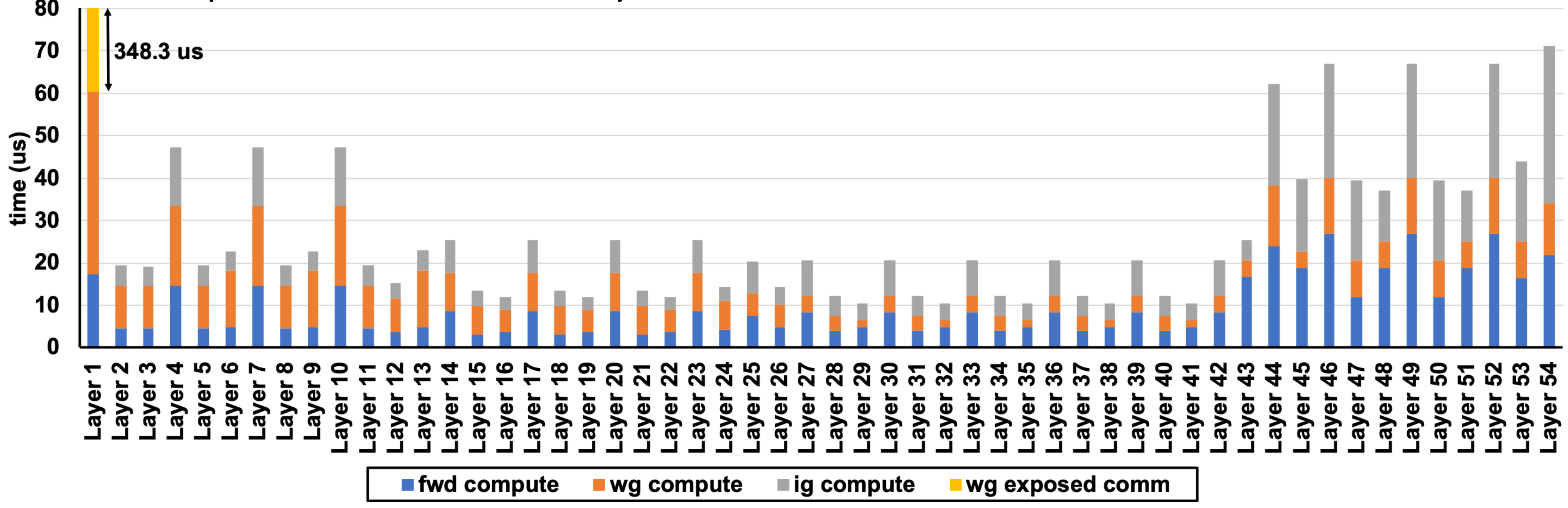
- A Torus 3D with total of 32 (2X4X4) nodes is used.
- Data parallel approach is used.
- Raw latency depends on the comm size plus the priority of each layer comm (queuing delay).



ResNet-50 Layer-Wise Compute vs. Exposed Comm Latency

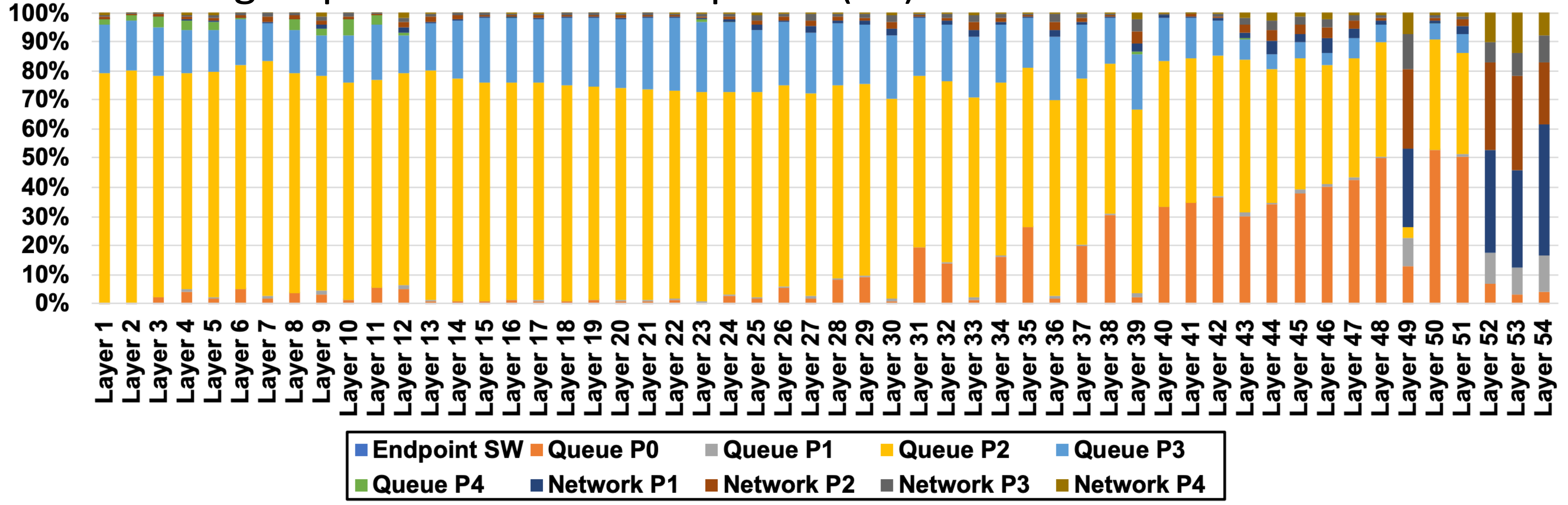
- Exposed comm latency is observed for the first layer.

because by the time we reach other layers except that. first layer, their comm is already finished.



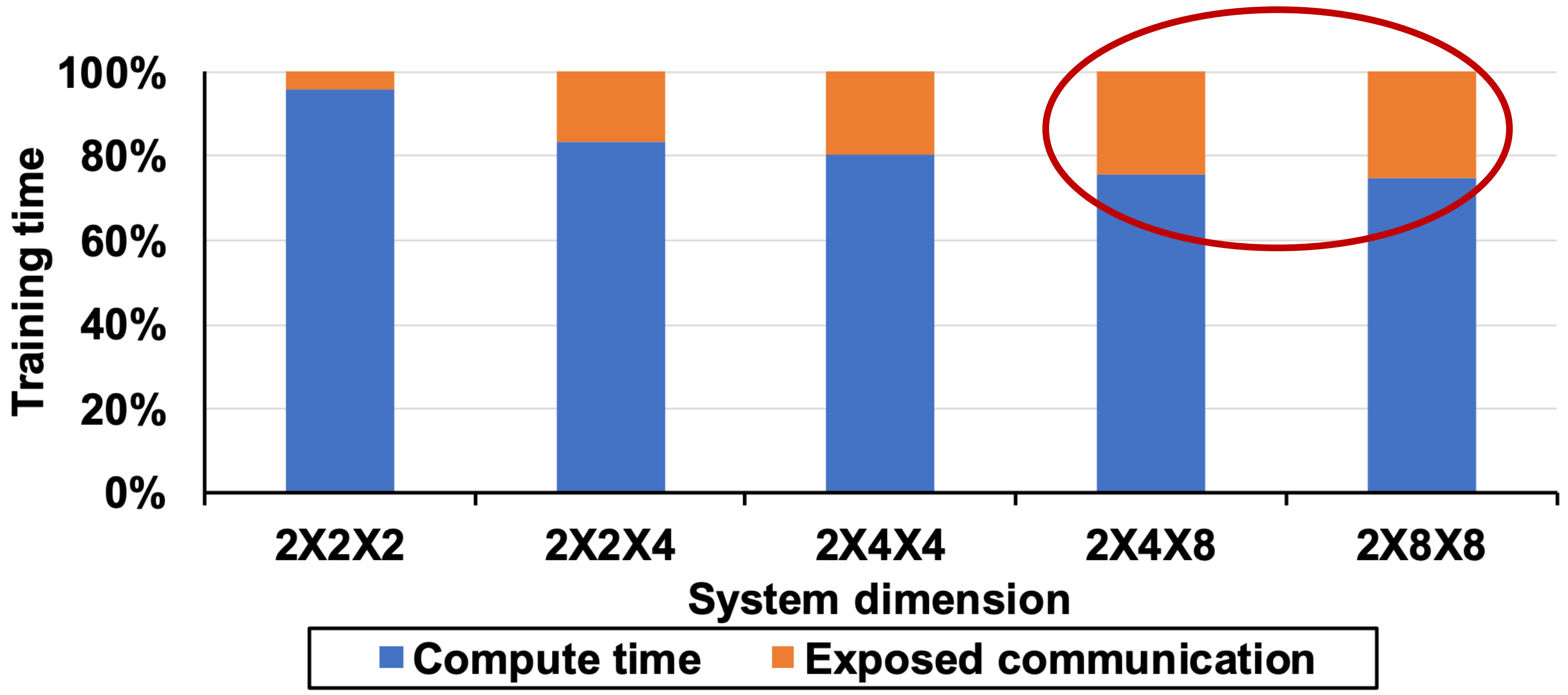
ResNet-50 Layer-Wise detailed latency

- Queue P2 is becoming the dominant factor due to very high speed of P1 (within package) that results most of the chunks get queued for the next phase (P2).



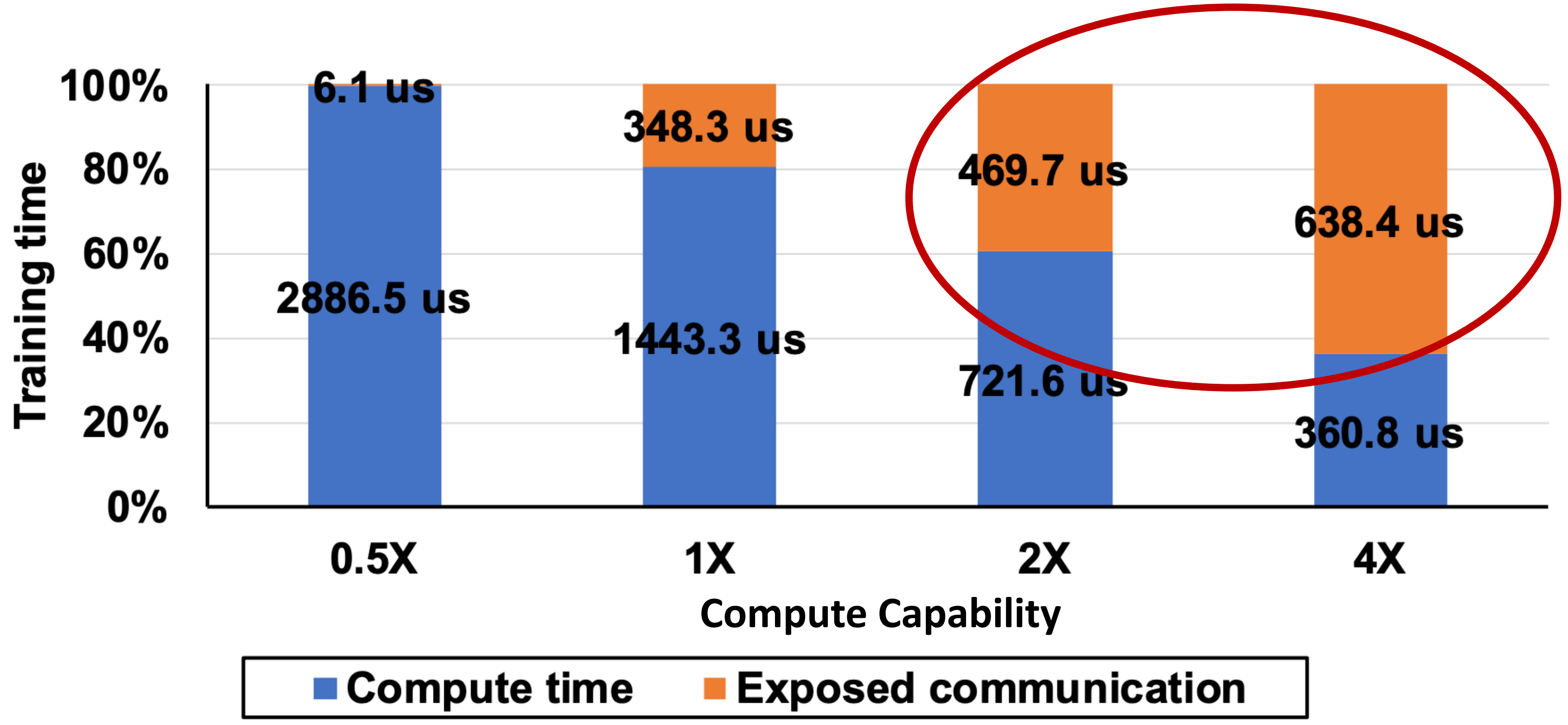
Effect of # of nodes on the Ratio of Total Compute vs Total Exposed Comm for ResNet-50

- A Torus 3D with total of 8, 16, 32, 64, 128 nodes are used.



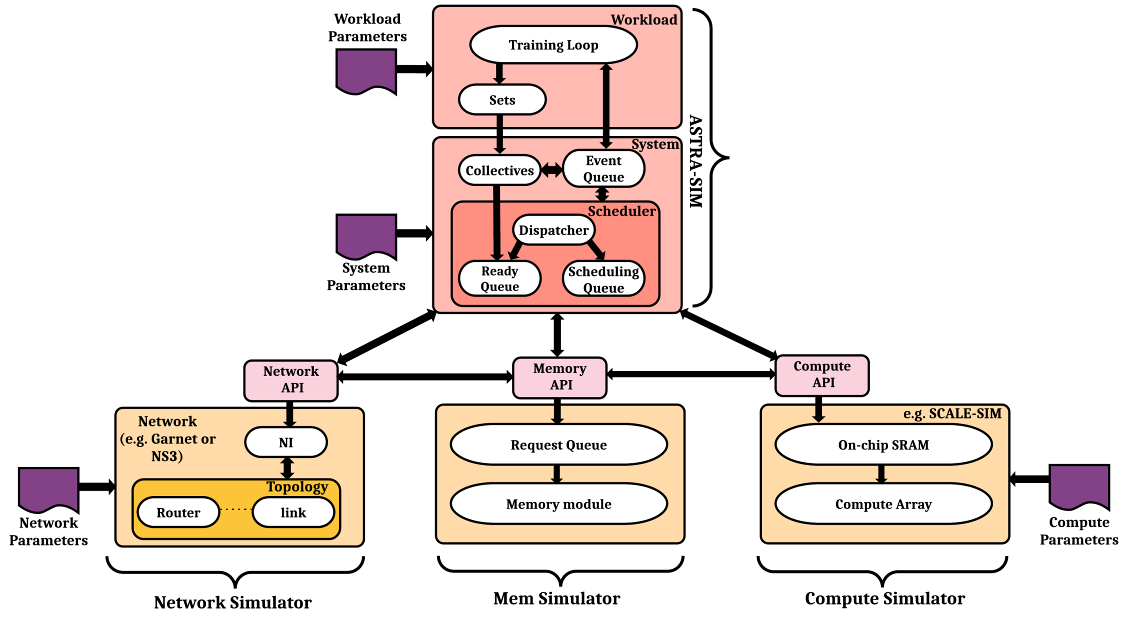
Effect of Enhanced Compute Time per Node on the Ratio of Total Compute vs Total Exposed Comm for ResNet-50

- A Torus 3D with total of 32 nodes (2X4X4) is used.



ASTRA-SIM Architecture: Next Version……

- Adding the online compute and network API interface for more accurate modeling of shared resource congestions (e.g. memory congestions) between the compute and network tasks.



https://github.com/astra-sim/astra-sim

Summary for ASTRA-Sim

- The design space to build the best HW/SW platform is quite large.
  - It is hard (or sometimes impossible) to change these parameters in real systems.
  - Analytical model can be misleading and is not accurate.
- Our simulation methodology provides a convenient way to explore this space.
- We study the performance of collectives and real-workloads across flat and hierarchical topologies in this work.
- We find that exposed communication increases as system sizes go up and as compute accelerator efficiency goes up.

Thank you!