BLINK

Fast and Generic Collectives for Distributed ML

Guanhua Wang, Shivaram Venkataraman, Amar Phanishayee

Jorgen Thelin, Nikhil R. Devanur, Ion Stoica

DNNs empower state-of-the-art results across many different applications


Image Classification

Robot Control

Hey Siri

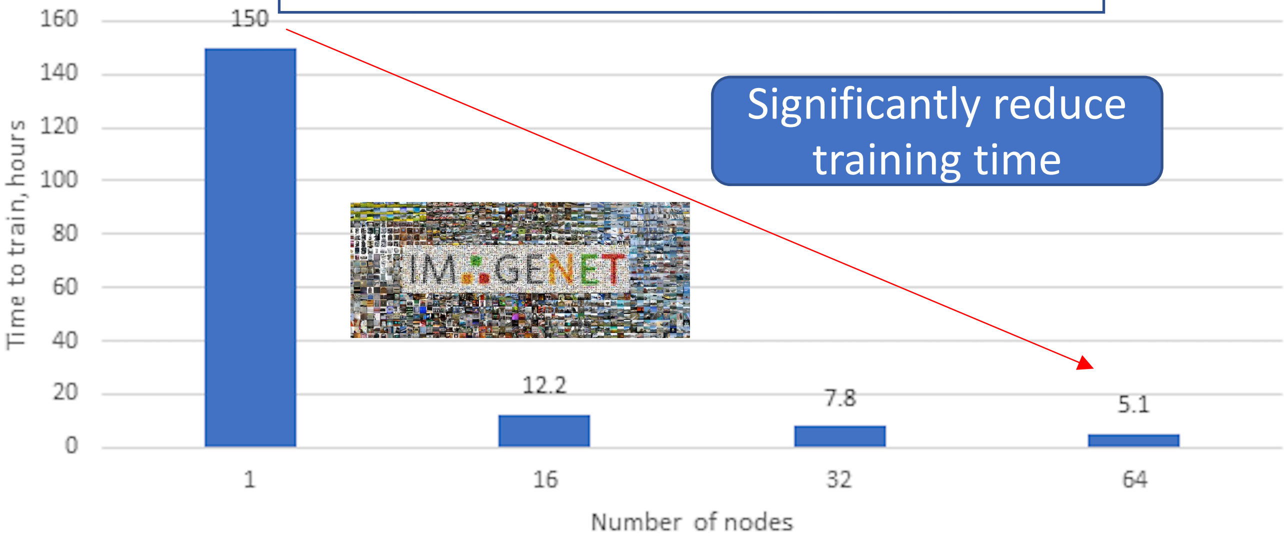
amazon alexa

Speech Recognition

Game Playing

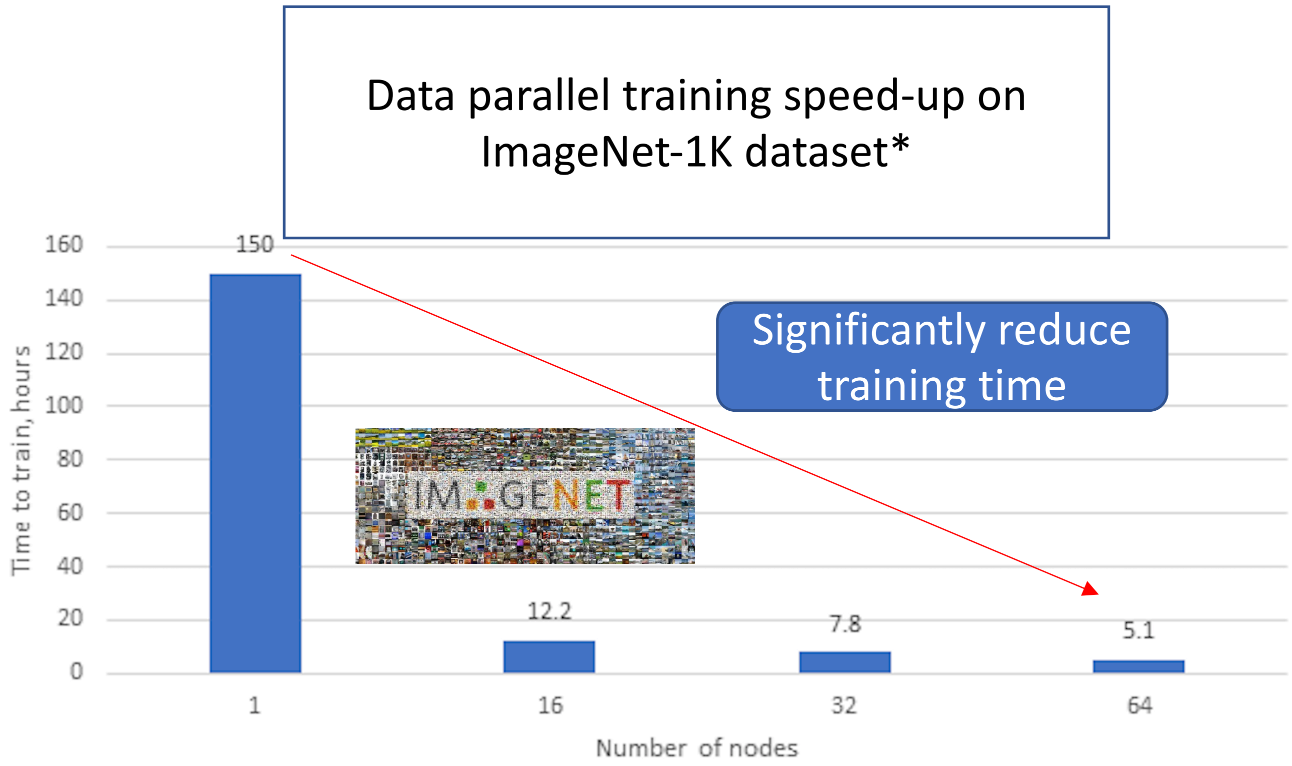
Speed-up DNN training: Data Parallelism

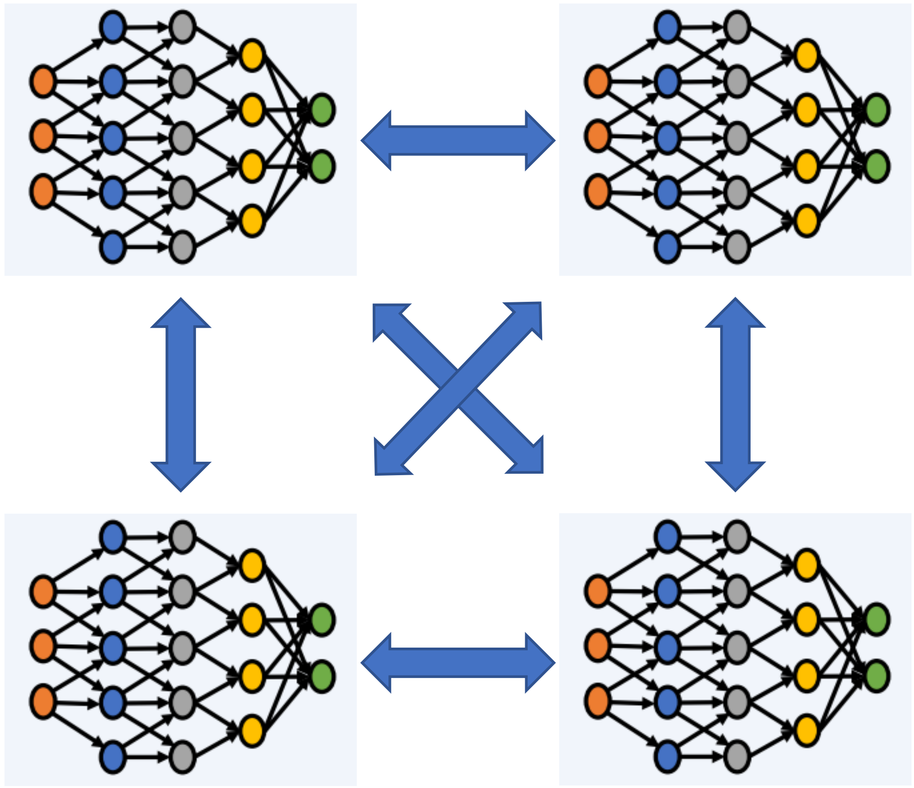
Data parallel training speed-up on ImageNet-1K dataset*



* https://software.intel.com/en-us/articles/caffe-training-on-multi-node-distributed-memory-systems

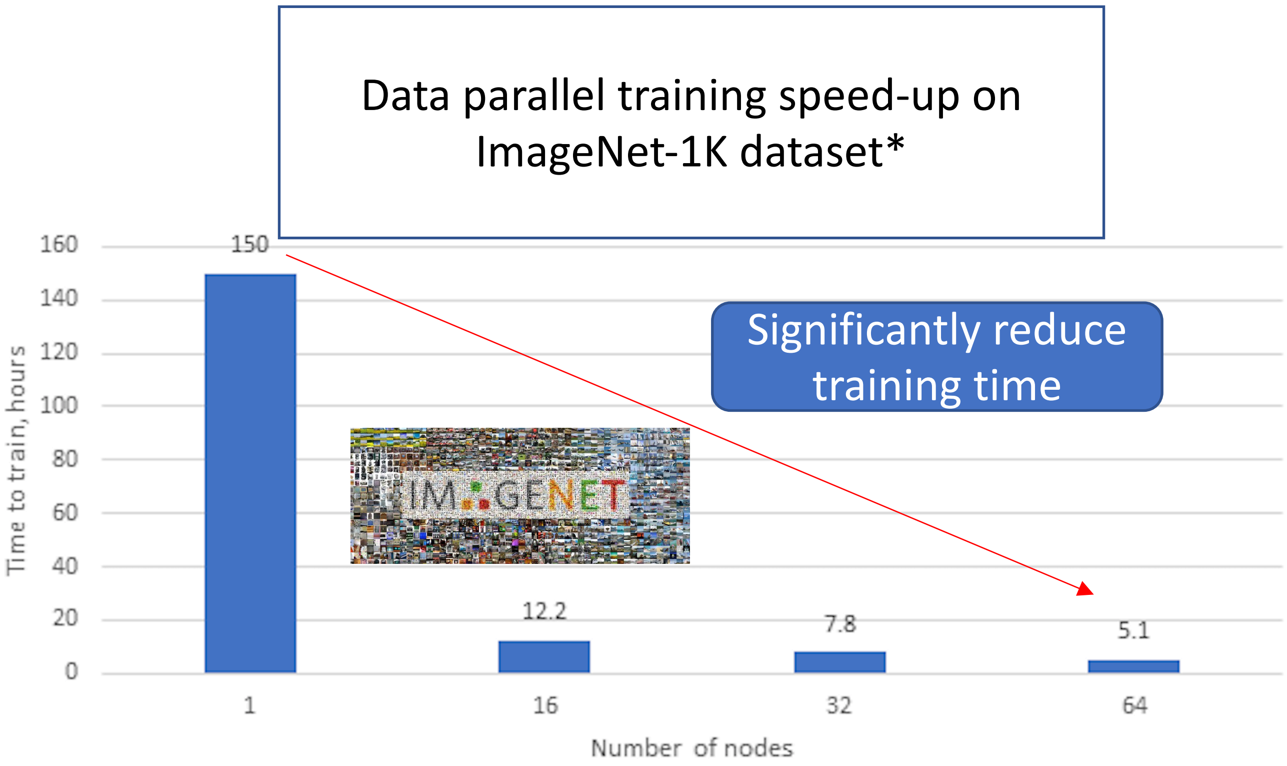
Speed-up DNN training: Data Parallelism

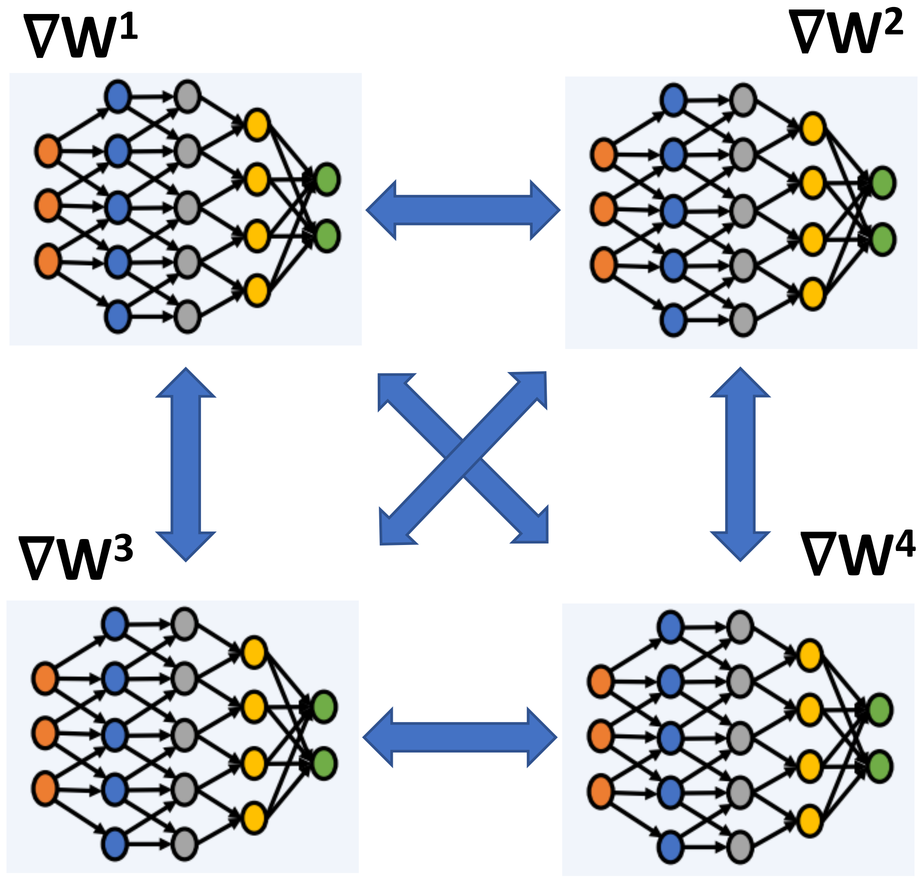




* https://software.intel.com/en-us/articles/caffe-training-on-multi-node-distributed-memory-systems

Speed-up DNN training: Data Parallelism



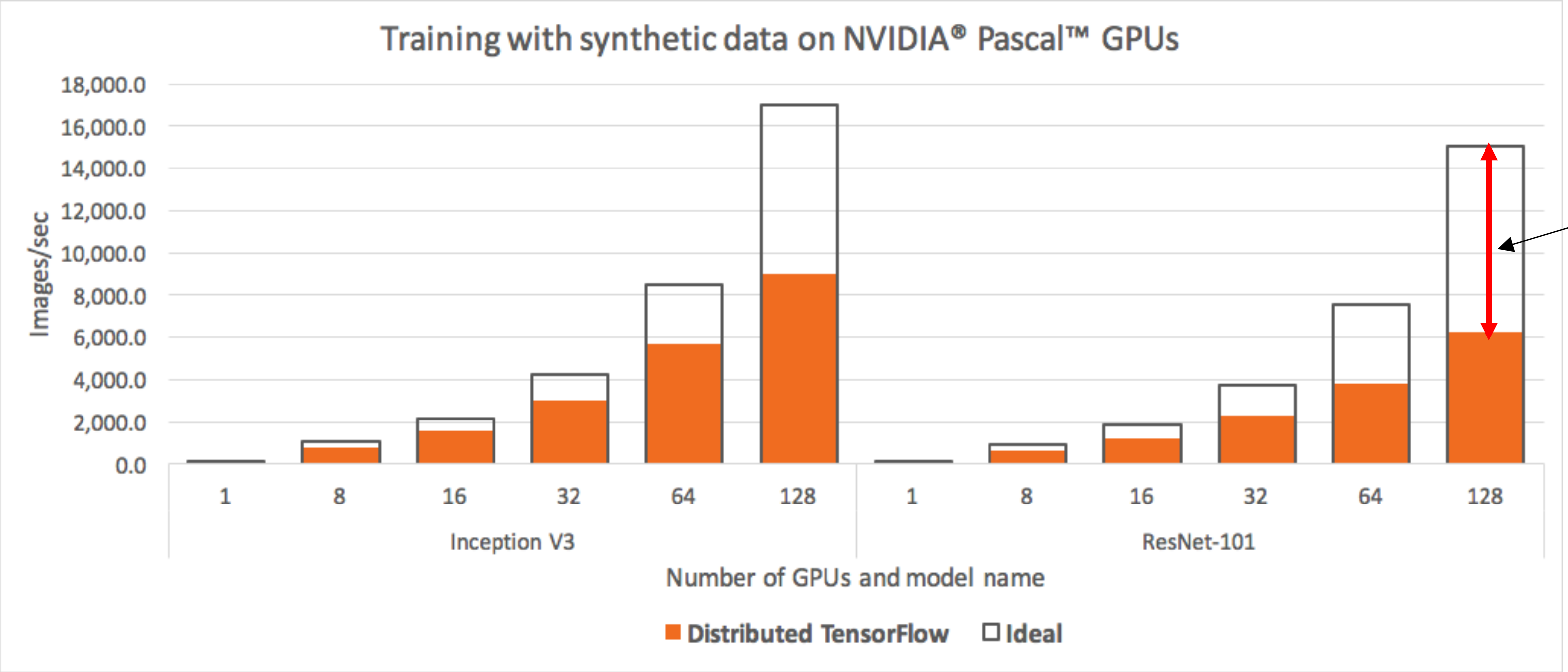


Model Synchronization

$$\nabla W = \nabla W^1 + \nabla W^2 + \cdots + \nabla W^N$$

* https://software.intel.com/en-us/articles/caffe-training-on-multi-node-distributed-memory-systems

Despite many performance optimizations, model synchronization is a big overhead in data parallel training on cloud servers

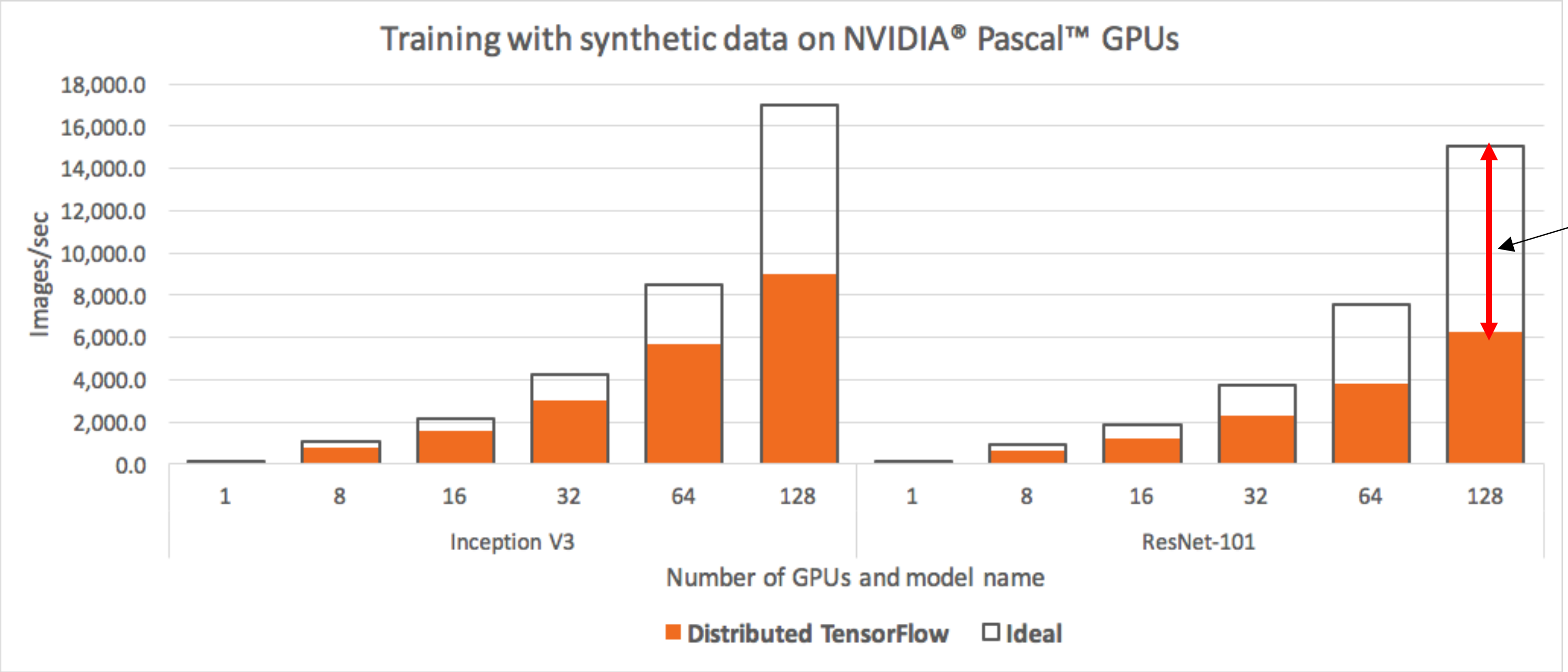


>50% communication overhead

Multi-GPU scaling performance using TensorFlow*

*Horovod: fast and easy distributed deep learning in TensorFlow, arXiv:1802.05799, 2018

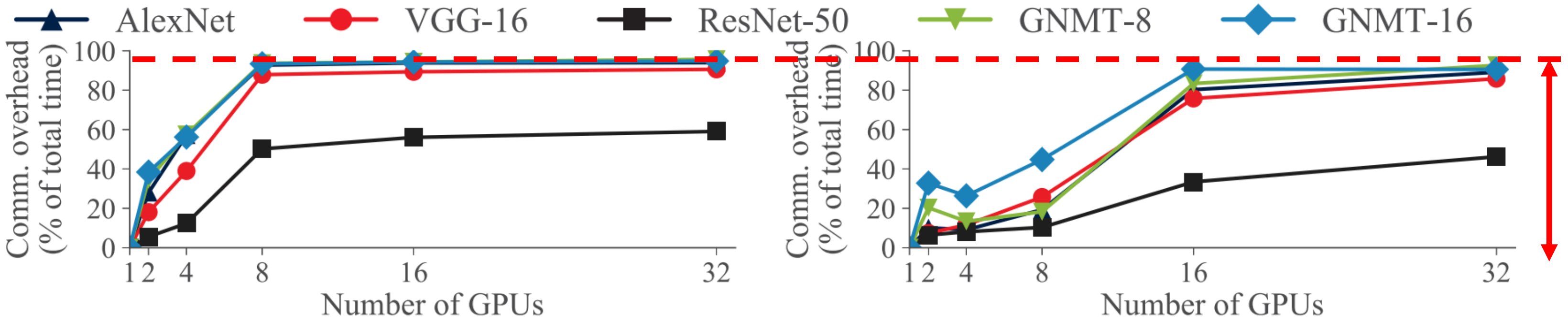
Despite many performance optimizations, model synchronization is a big overhead in data parallel training on cloud servers



>50% communication overhead

Multi-GPU scaling performance using TensorFlow*

Up to 90% communication overhead

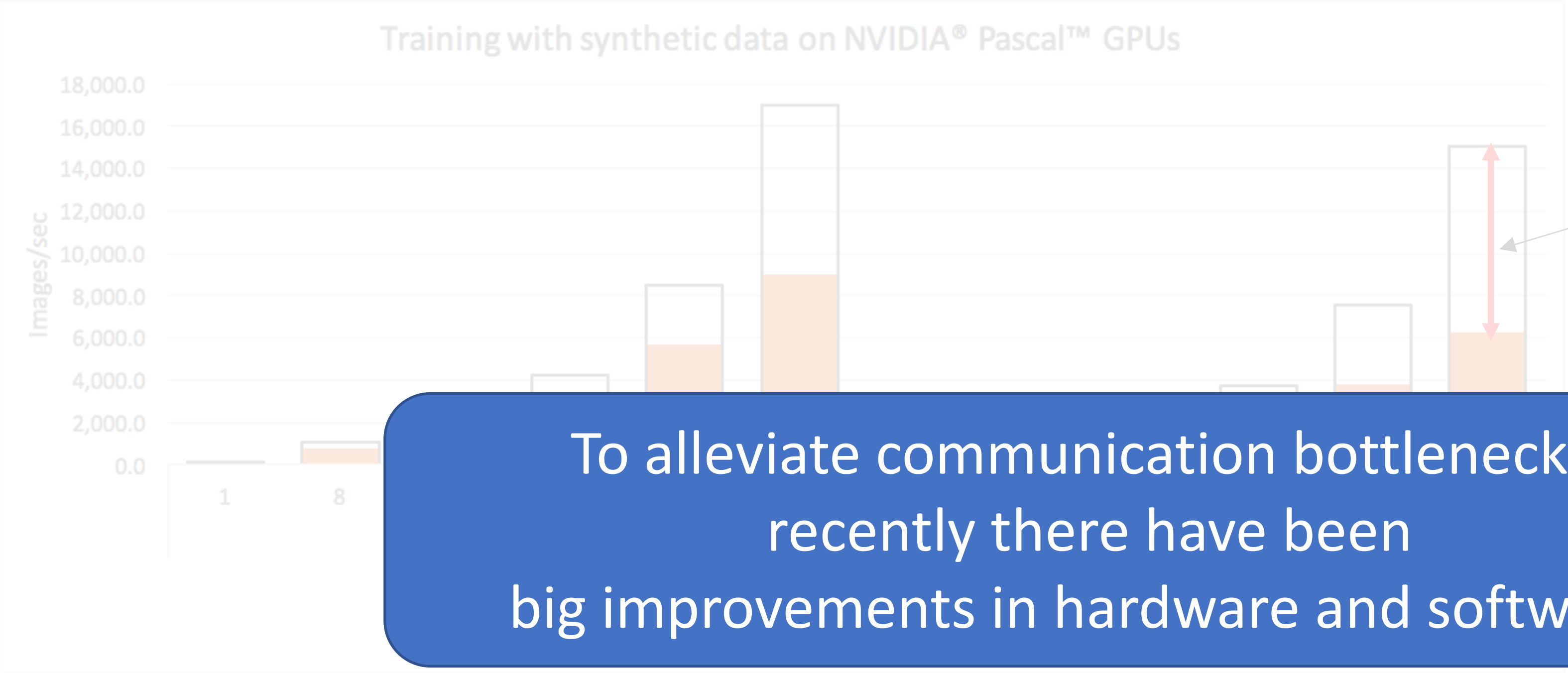


Communication overhead of data-parallel training with Multi-GPU servers using PyTorch^

^PipeDream: Generalized Pipeline Parallelism for DNN Training, SOSP 2019

*Horovod: fast and easy distributed deep learning in TensorFlow, arXiv:1802.05799, 2018

Model synchronization is a big overhead in data parallel training despite many performance optimizations



>50% communication overhead

Multi-GPU scaling performance using TensorFlow*

To alleviate communication bottlenecks, recently there have been big improvements in hardware and software.

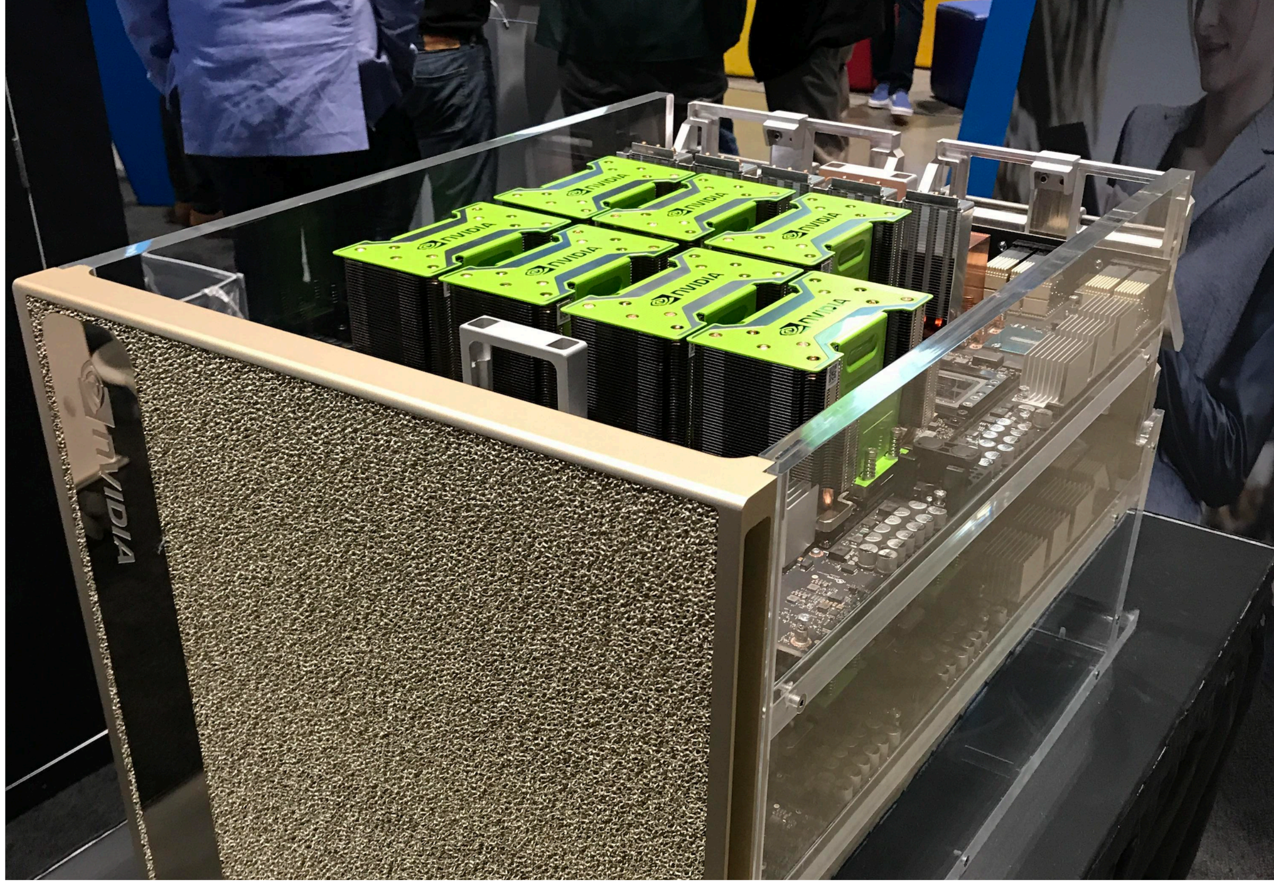
Up to 90% communication overhead

Communication overhead of data-parallel training with Multi-GPU servers using PyTorch^

^PipeDream: Generalized Pipeline Parallelism for DNN Training, SOSP 2019

*Horovod: fast and easy distributed deep learning in TensorFlow, arXiv:1802.05799, 2018

NVIDIA DGX-1



NVIDIA DGX-2

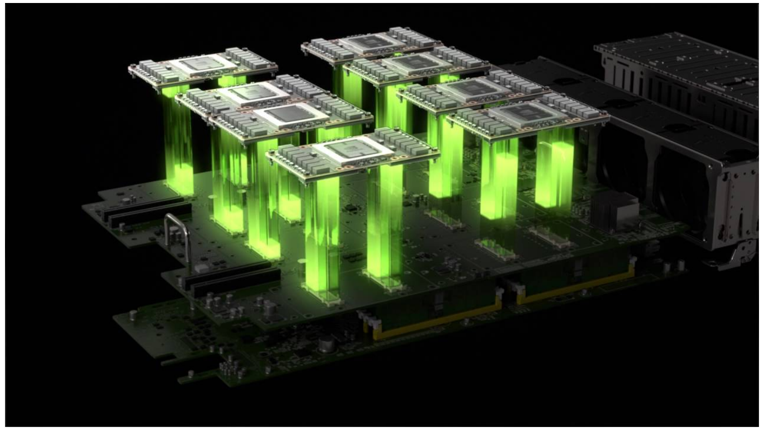
State of the art (hardware)

What is inside?

- Computation

NVIDIA P100: 5.3 Tera-FLOPs
Double Precision

NVIDIA V100: 7.8 Tera-FLOPs
Double Precision

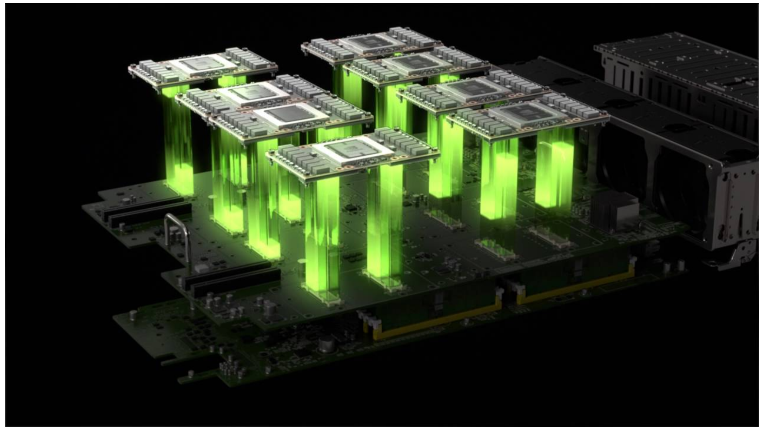


What is inside?

- Computation

NVIDIA P100: 5.3 Tera-FLOPs
Double Precision

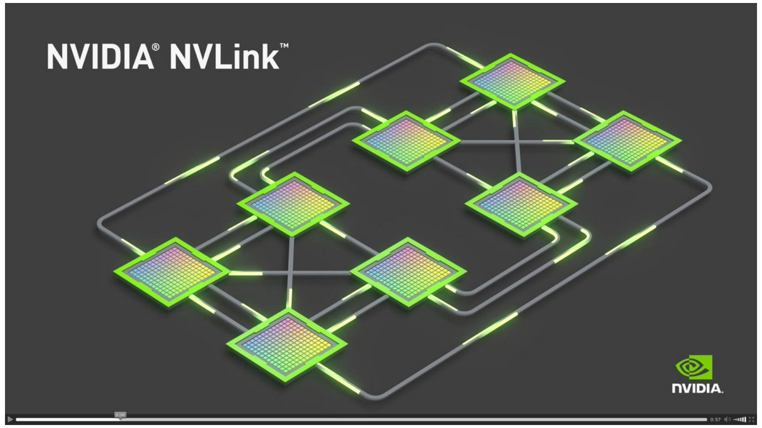
NVIDIA V100: 7.8 Tera-FLOPs
Double Precision



- Faster Interconnects

PCIe 3.0 (x16) ~10GB/s
- Shared

NVLink
- Point-to-point
- 1st Gen (P100) ~**18**GB/s
- 2nd Gen (V100) ~ **23**GB/s

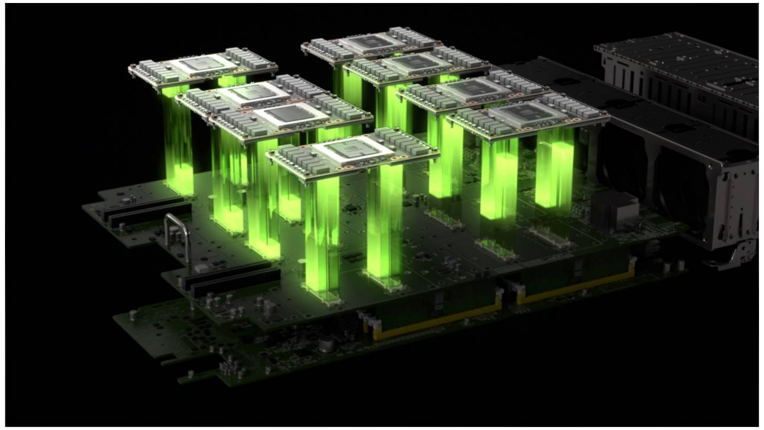


What is inside?

- Computation

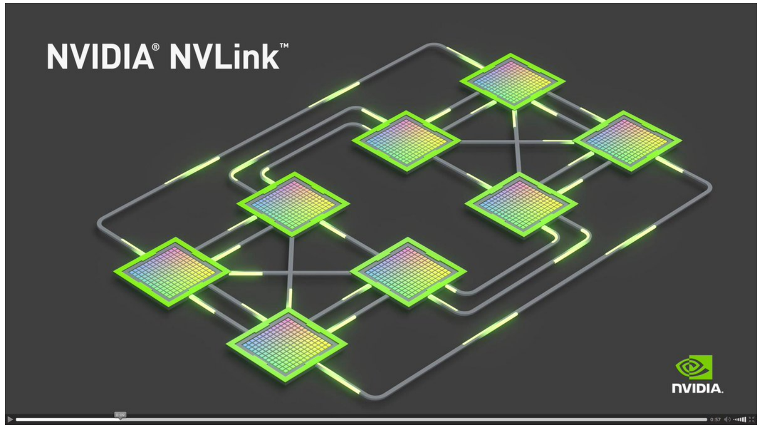
NVIDIA P100: 5.3 Tera-FLOPs
Double Precision

NVIDIA V100: 7.8 Tera-FLOPs
Double Precision



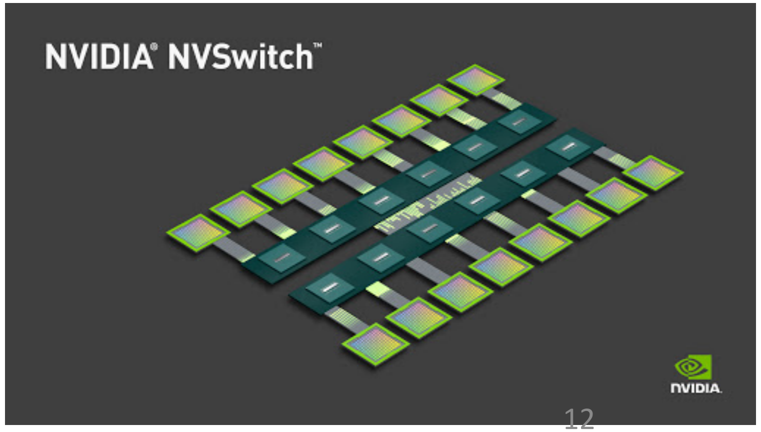
- Faster Interconnects

PCIe 3.0 (x16) ~10GB/s
- Shared



NVLink
- Point-to-point
- 1st Gen (P100) ~**18**GB/s
- 2nd Gen (V100) ~ **23**GB/s

NVSwitch
- Fully connected crossbar
- 6x NVLink 2nd Gen Bandwidth ~**130**GB/s



State of the art (software)

NCCL
(Nvidia Collective Communication Library)

HOROVOD

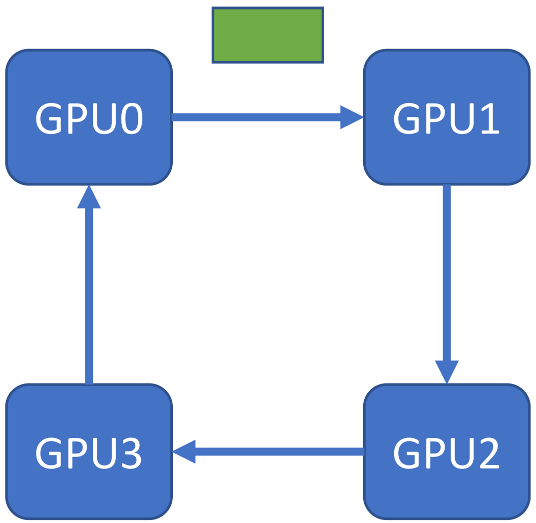
gloo

NVIDIA®

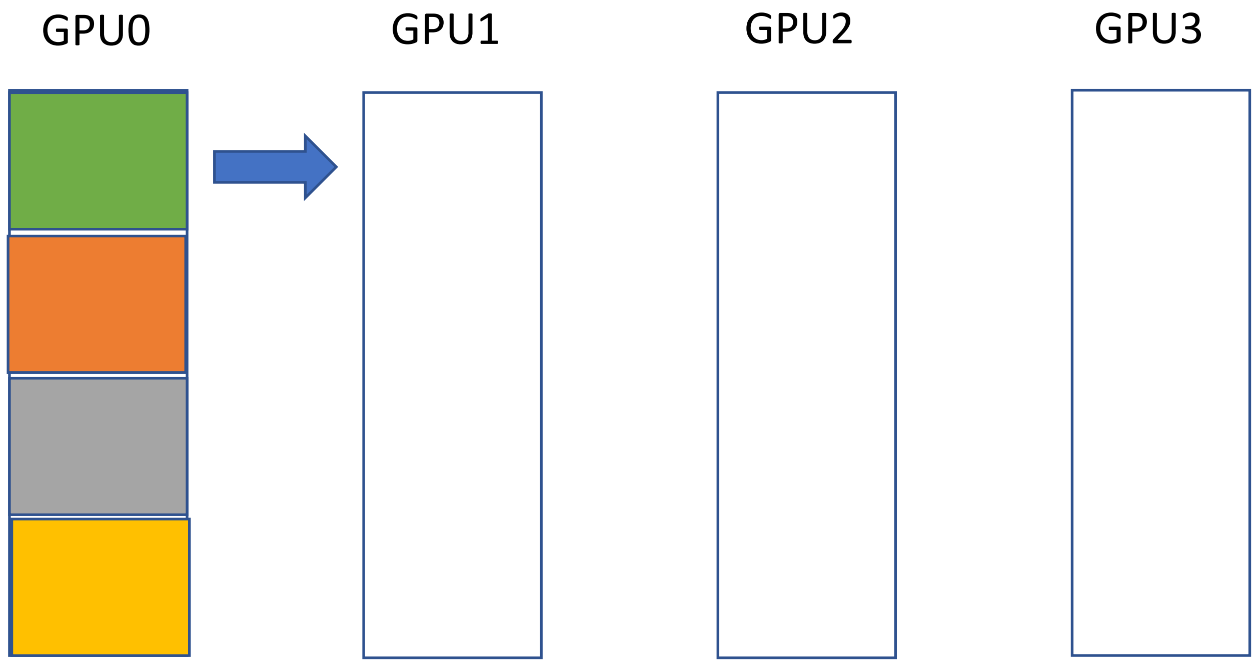
Uber

Ring-based collective communication protocols

Ring-based collectives (e.g. Broadcast)

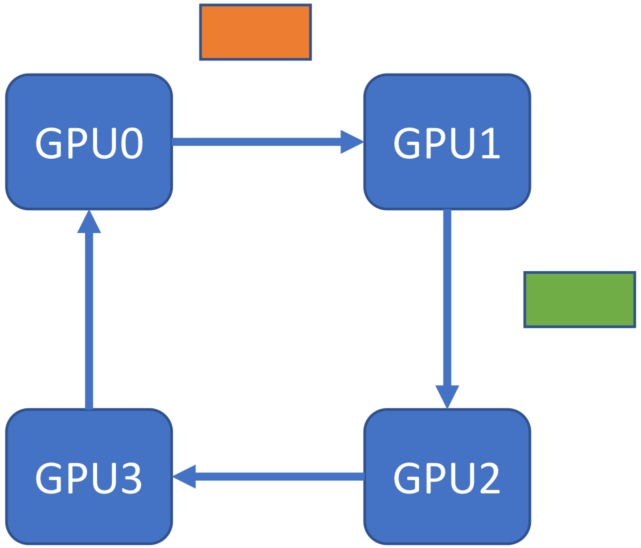


Topology

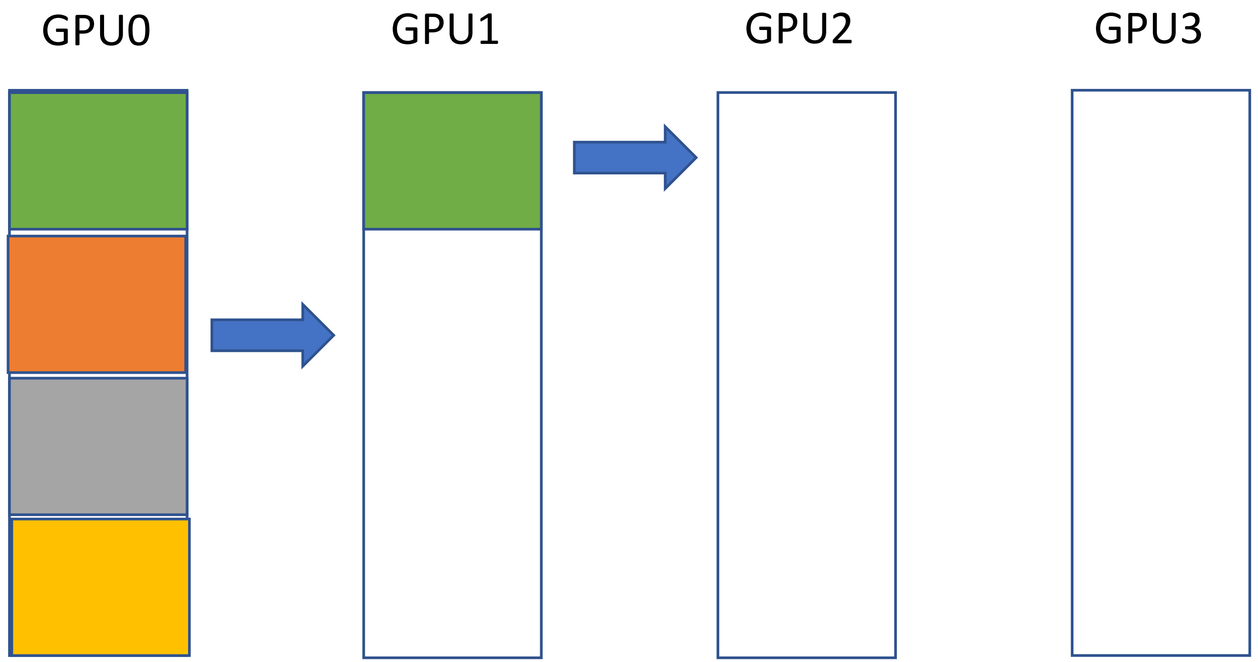


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

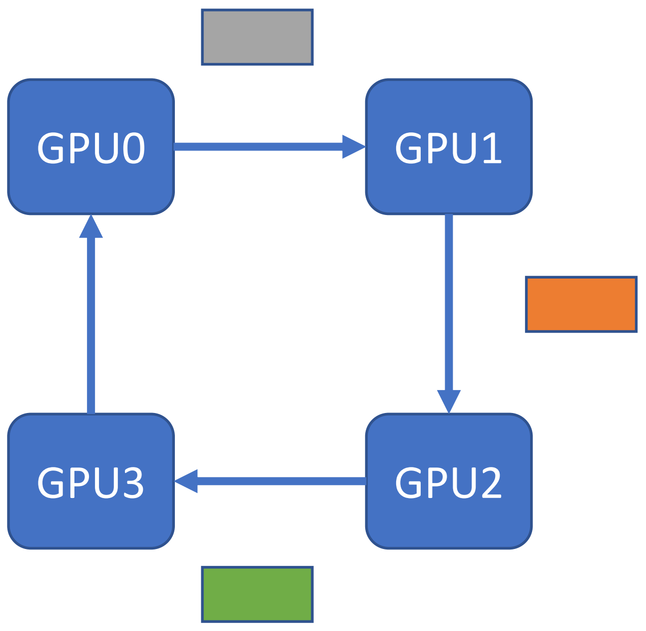


Topology

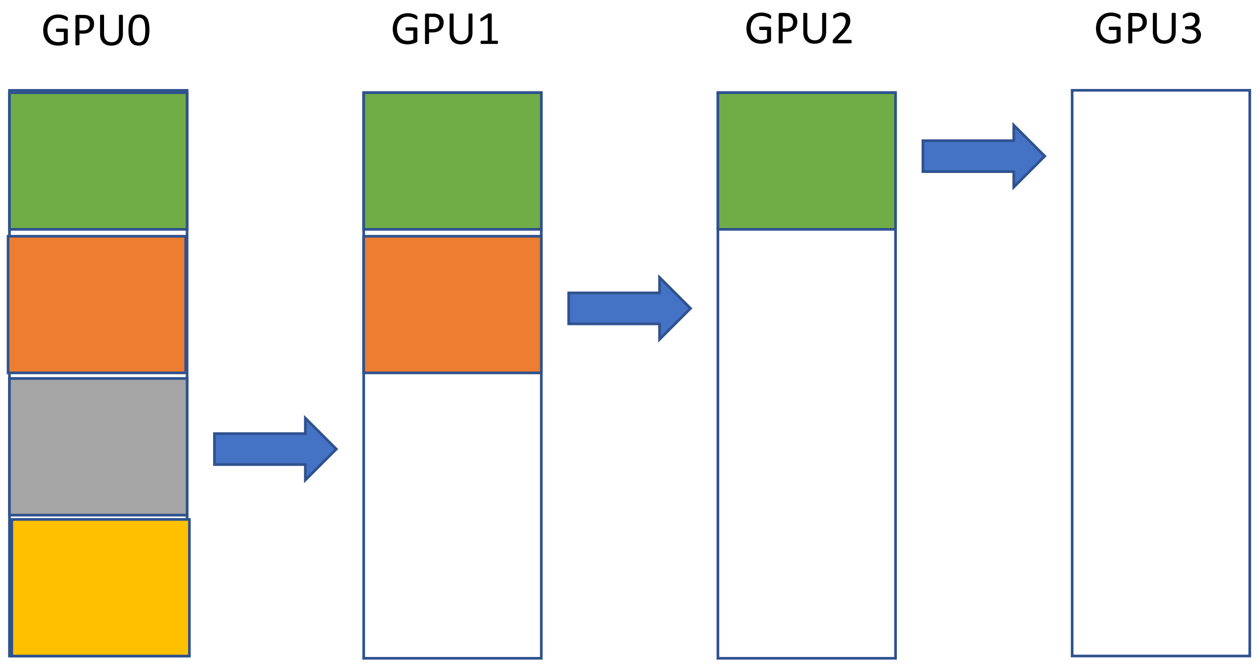


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

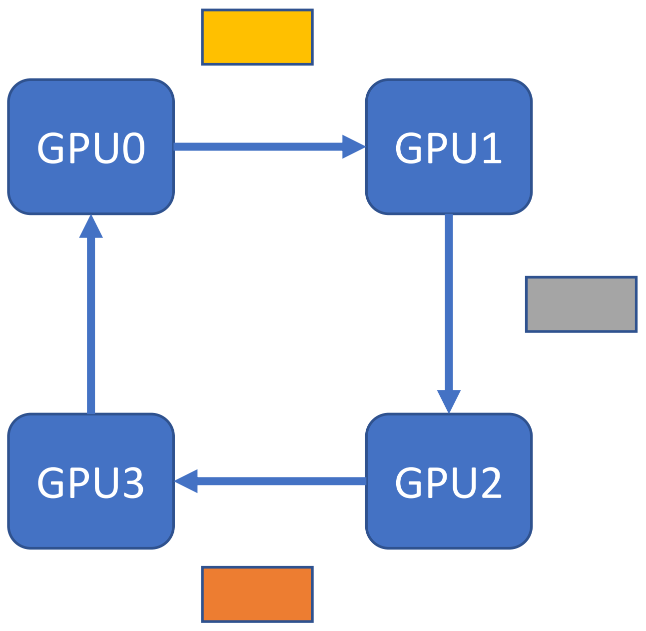


Topology

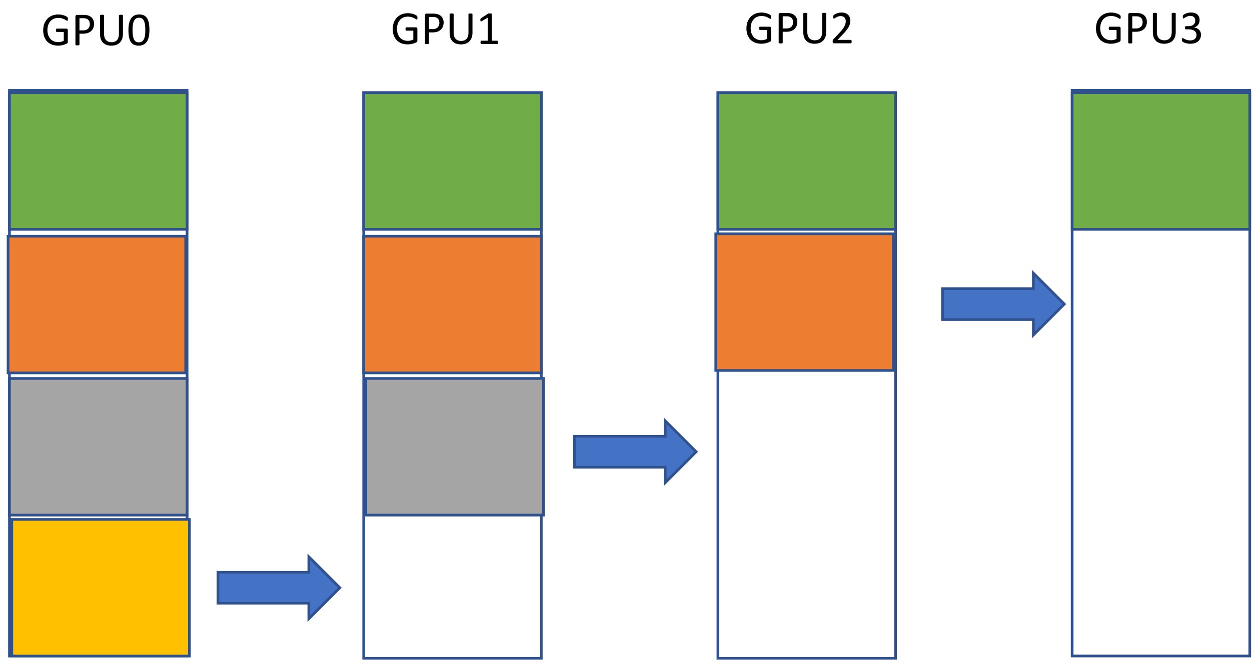


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

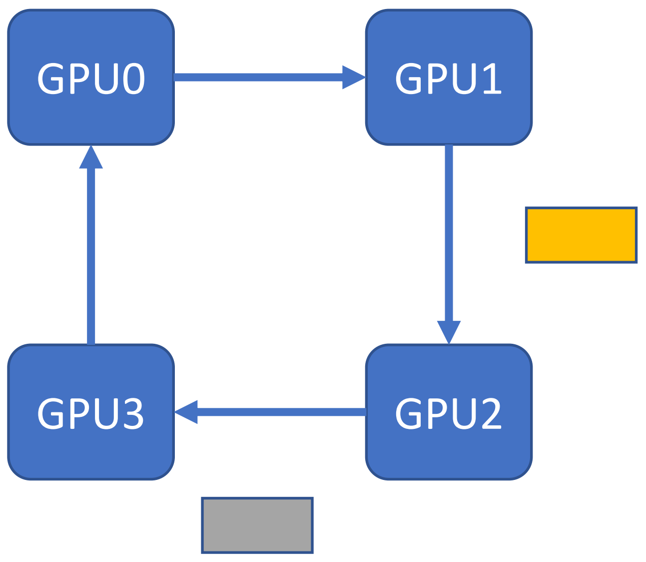


Topology

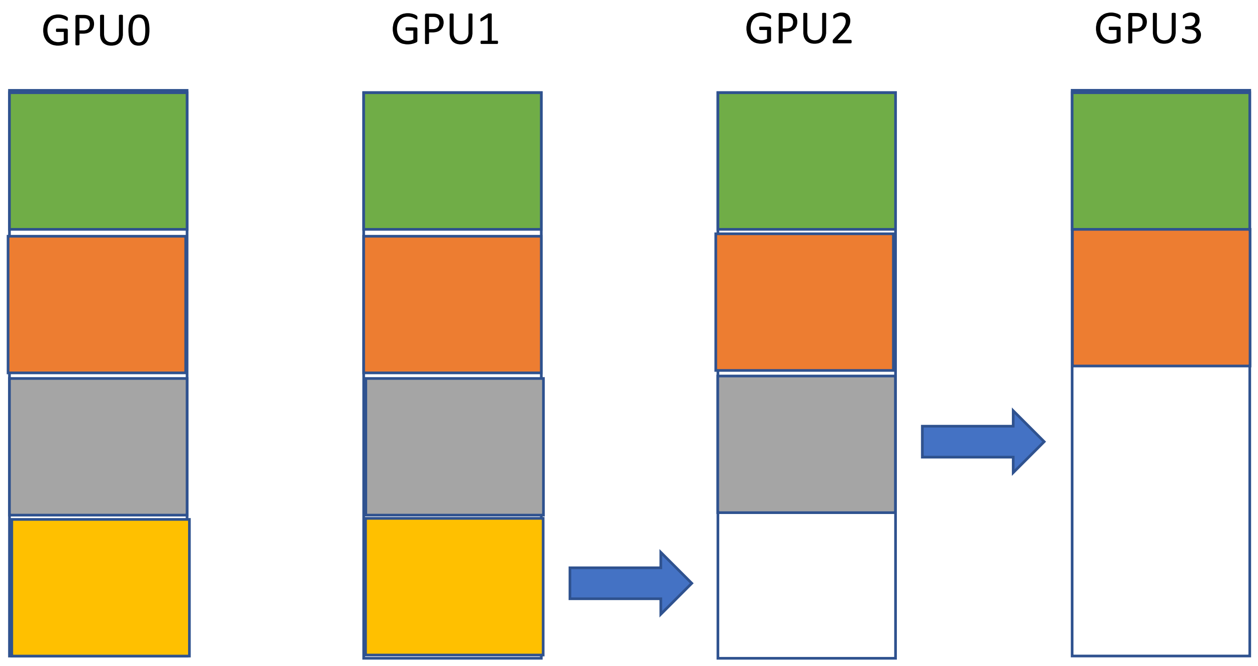


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

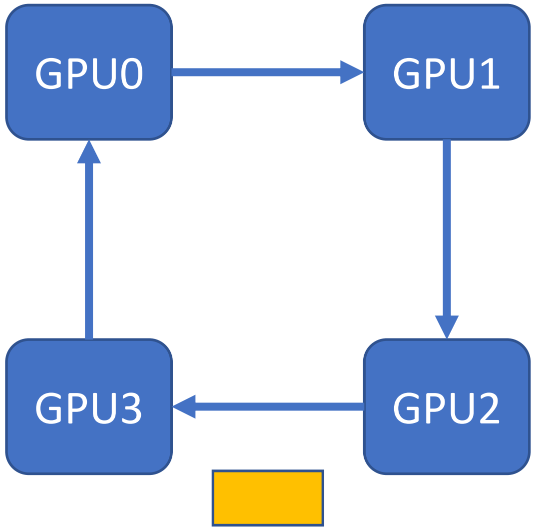


Topology

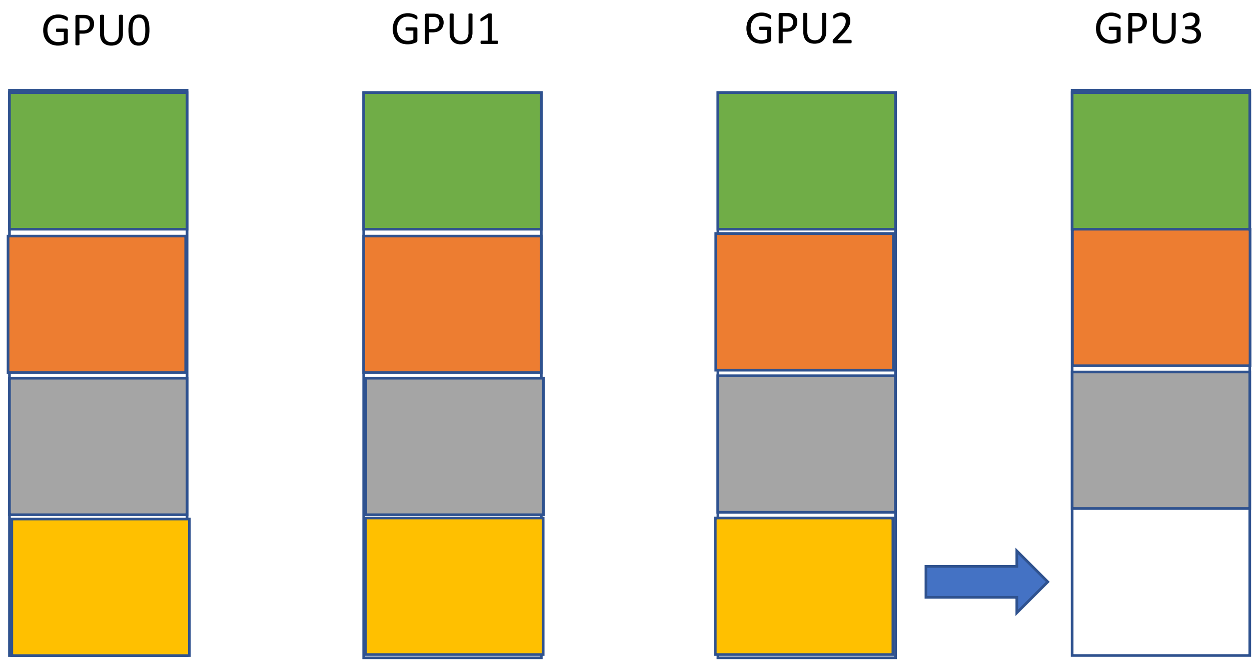


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

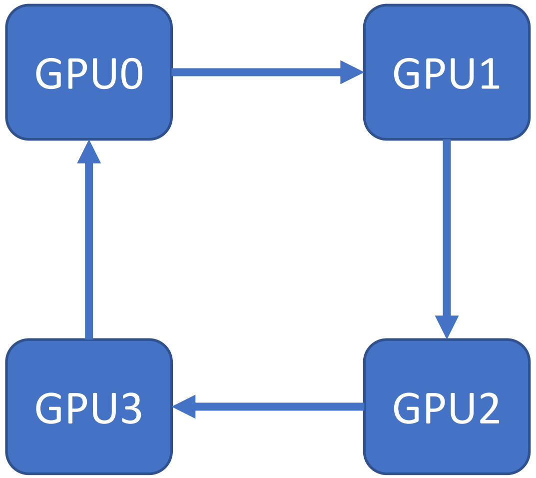


Topology

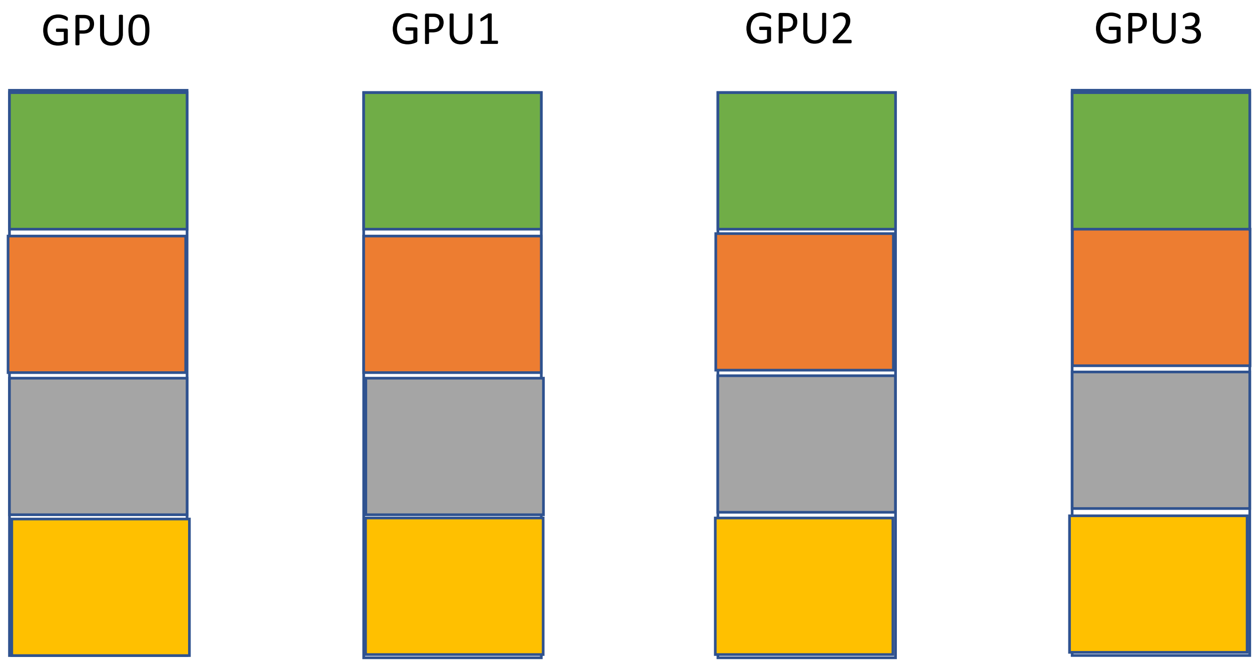


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)



Topology



Ring Broadcast (from GPU0)

State of the art (software)

NCCL

(Nvidia Collective Communication Library)

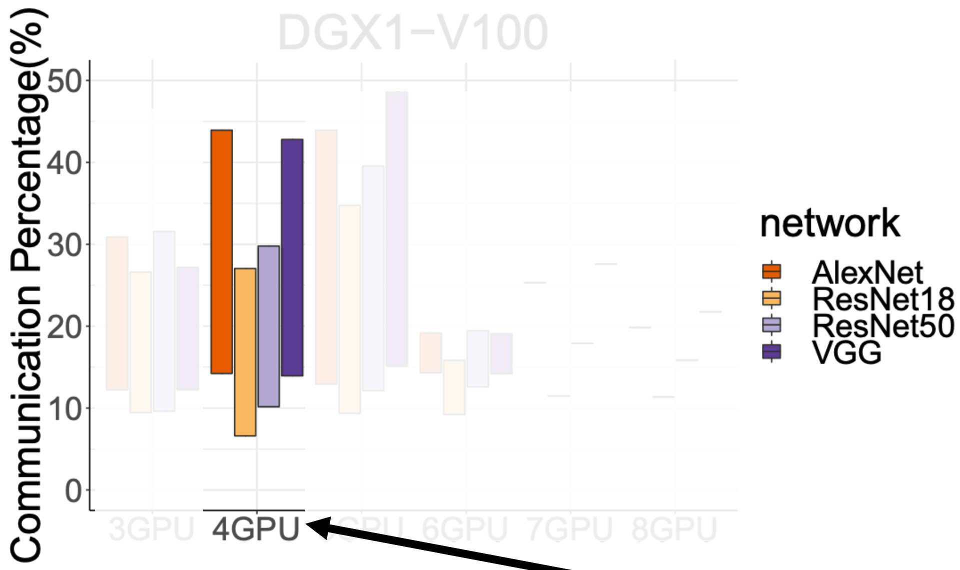
Ring-based collective communication protocols

Can these hardware & software improvements
alleviate communication bottleneck
in data-parallel training?

Can these hardware & software improvements alleviate communication bottleneck in data-parallel training?

Not Really

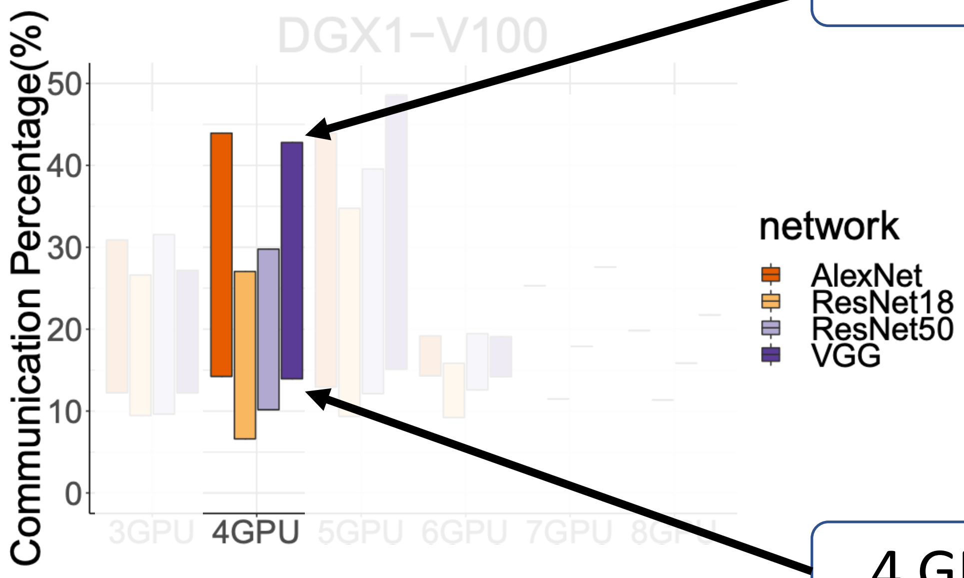
High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)



Cross-GPU communication measured as the percentage of total epoch time when running within a single 8-GPU DGX-1 box

There are many different 4 GPU allocations with a server

High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)



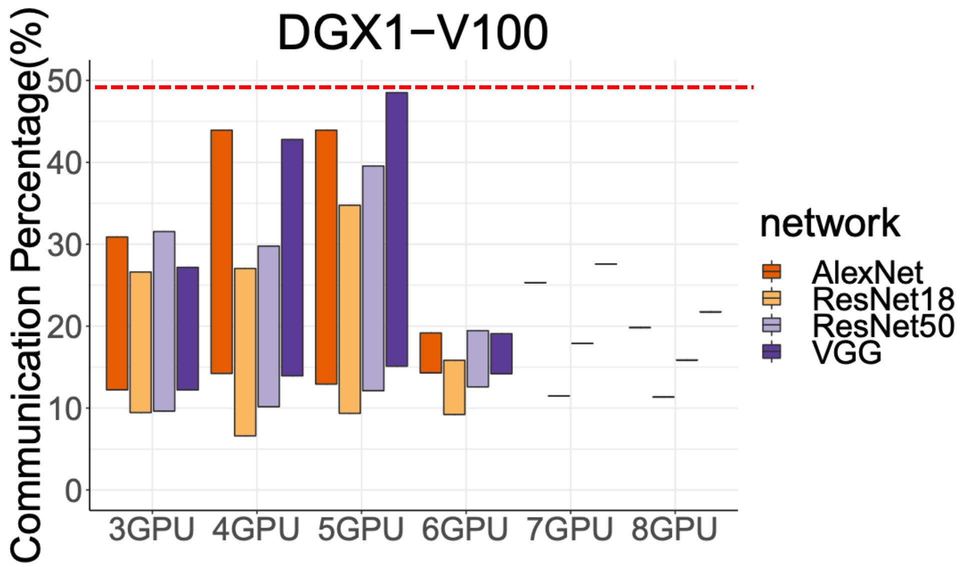
4 GPU allocation with highest overhead

High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)

4 GPU allocation with lowest overhead

Cross-GPU communication measured as the percentage of total epoch time when running within a single 8-GPU DGX-1 box

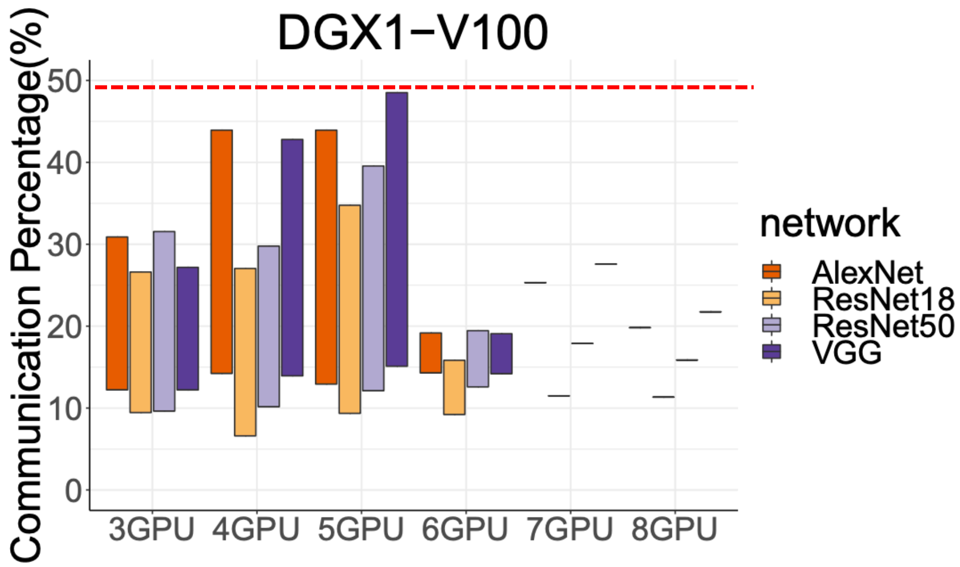
High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)



Cross-GPU communication measured as the percentage of total epoch time when running within a single 8-GPU DGX-1 box

High communication overheads is consistent across different number of workers and for a range of DNNs

High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)



Cross-GPU communication measured as the percentage of total epoch time when running within a single 8-GPU DGX-1 box

High communication overheads is consistent across different number of workers and for a range of DNNs

Communication overheads become more pronounced with increasing GPU computation power.

High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)

DGX1−V100

We need Faster Collective Communication Protocols.

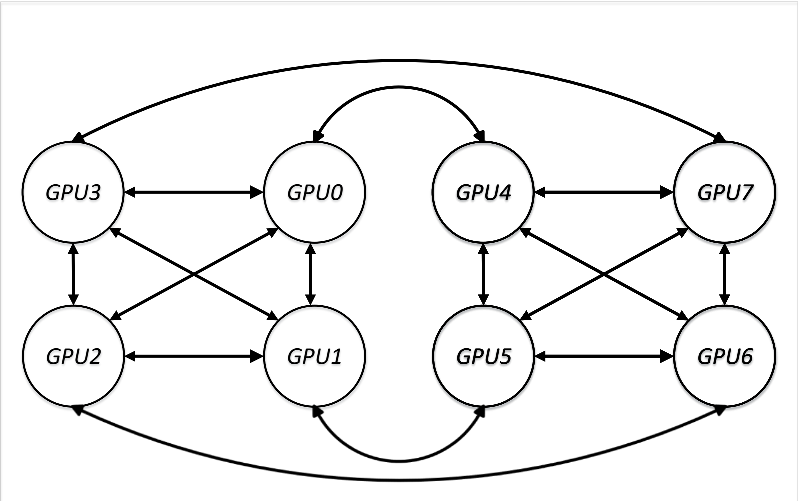
computation power.

Cross-GPU communication measured as the percentage of total epoch time when running within a single 8-GPU DGX-1 box

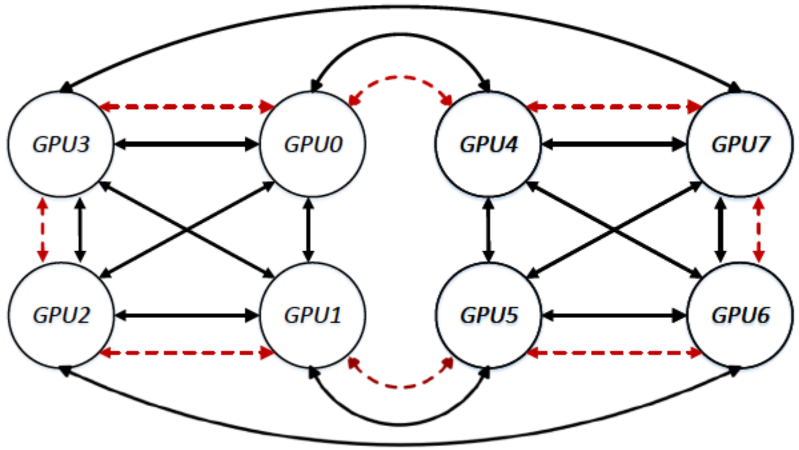
Talk Outline

- Motivation
- **Challenges to achieving faster collective communication**
- Design
- Evaluation

Challenge 1: Different server configurations

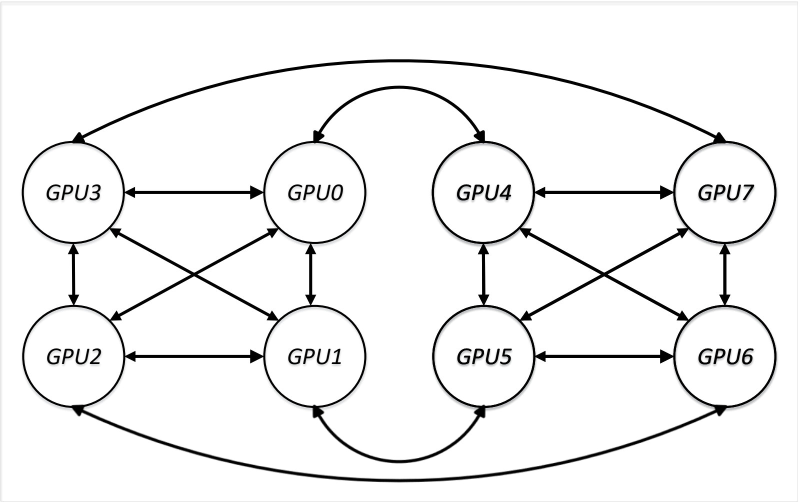


DGX1-P100 (NVLink 1st Gen, ~18GB/s)

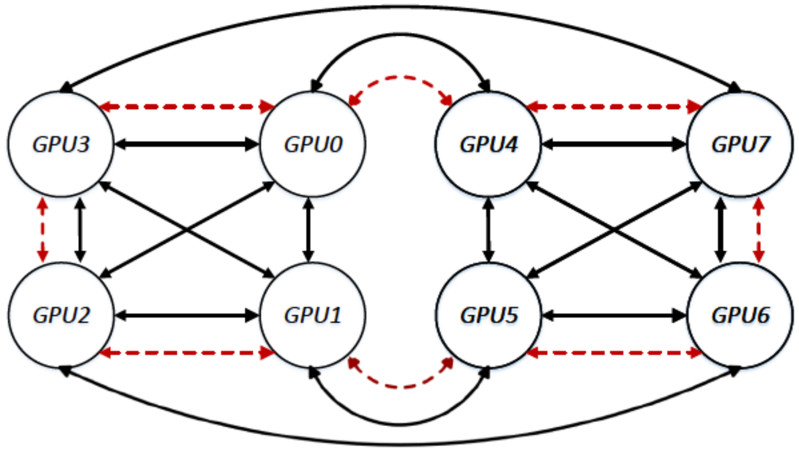


DGX1-V100 (NVLink 2nd Gen, ~23GB/s)

Challenge 1: Different server configurations



DGX1-P100 (NVLink 1st Gen, ~18GB/s)

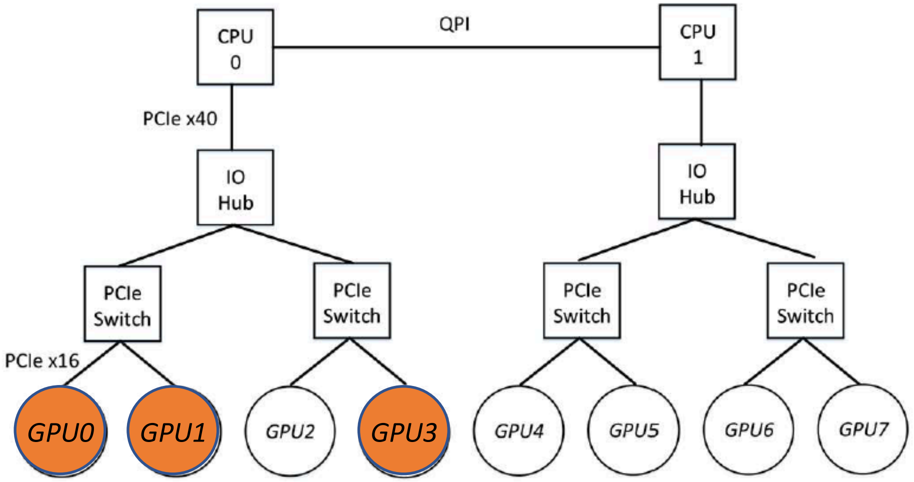


DGX1-V100 (NVLink 2nd Gen, ~23GB/s)

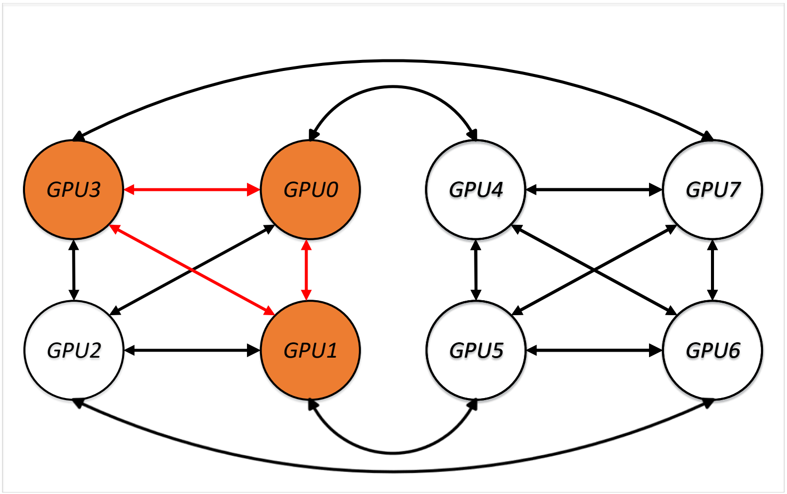
Protocols needs to be topology aware to effectively use hardware links.

Challenge 2: Link heterogeneity

PCIe topology



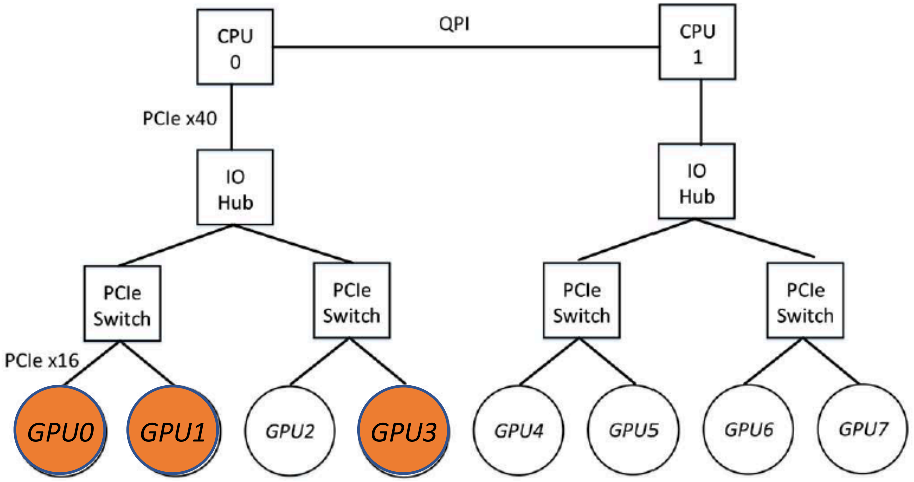
NVLink topology



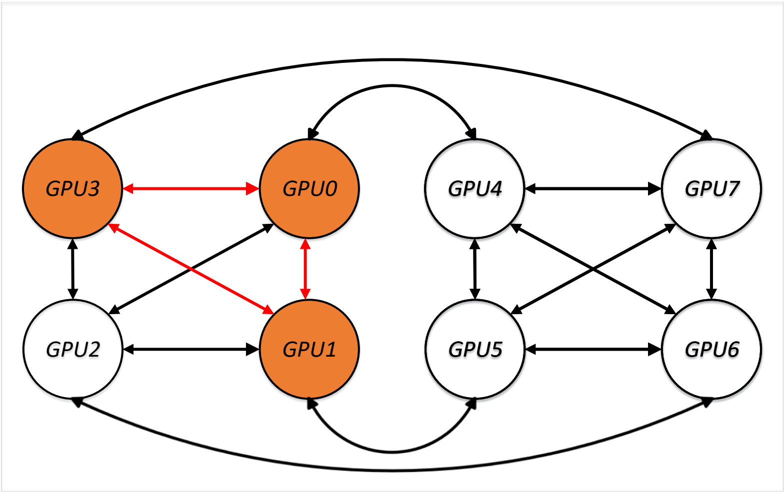
Ring-based collectives can only utilize homogeneous links.

Challenge 2: Link heterogeneity

PCIe topology



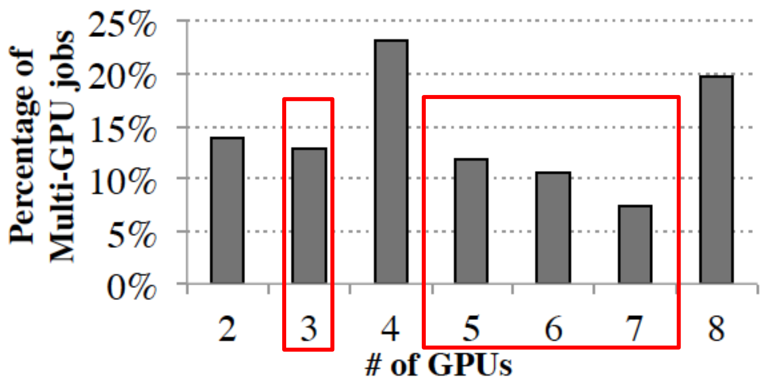
NVLink topology



Ring-based collectives can only utilize homogeneous links.

Why not heterogeneous links?
e.g. PCIe will be bottleneck if included in a NVLink ring

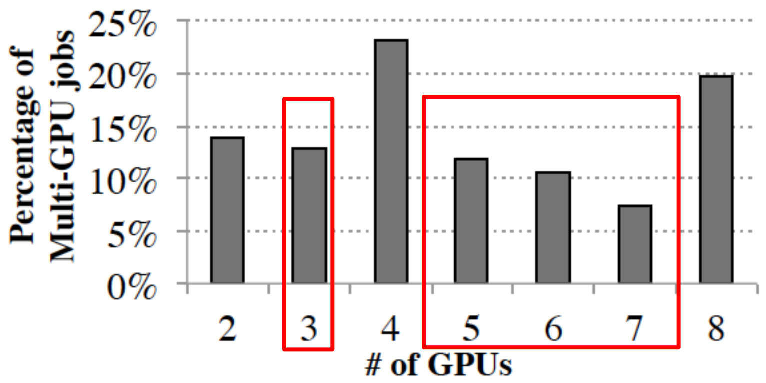
Challenge 3: Fragmentation in multi-tenant clusters



Within each 8-GPU server, # of GPUs allocated to 40,000 multi-GPU jobs at Microsoft.

Examples of fragmented allocation
(8GPU job across 2 servers)
3 + 5
2 + 6

Challenge 3: Fragmentation in multi-tenant clusters



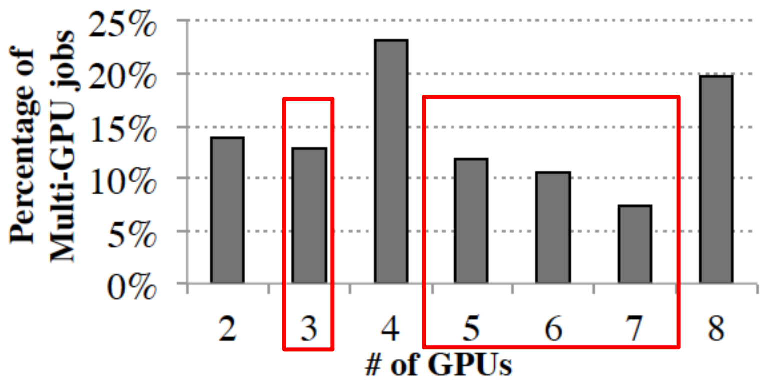
Within each 8-GPU server, # of GPUs allocated to 40,000 multi-GPU jobs at Microsoft.

Why fragmentation?

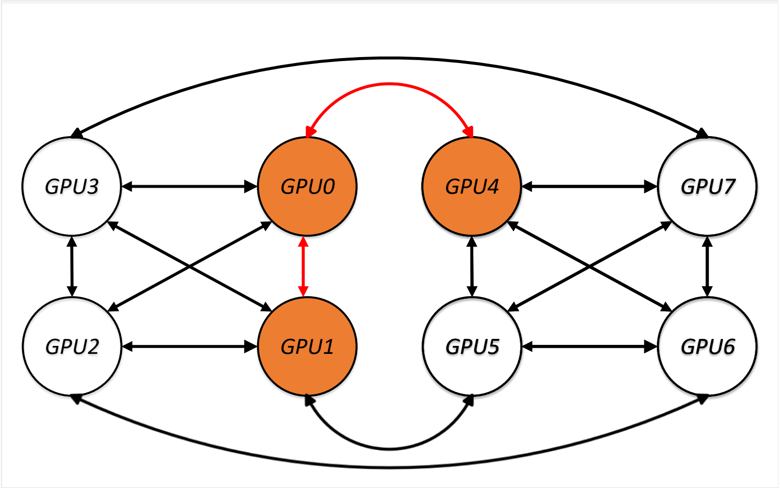
Many cluster schedulers are not topology-aware.

Without support for efficient migration, DNN jobs must embrace fragmentation to avoid queuing delays.

Challenge 3: Fragmentation in multi-tenant clusters



Within each 8-GPU server, # of GPUs allocated to 40,000 multi-GPU jobs at Microsoft.



Why fragmentation? → Irregular topo. → no ring

Many cluster schedulers are not topology-aware.

Without support for efficient migration, DNN jobs must embrace fragmentation to avoid queuing delays.

Existing solutions (NCCL) fall back to PCIe if they cannot form a NVLink ring.

Can we do better than state-of-the-art?

NVIDIA Collective Communications Library (NCCL)
Multi-GPU and multi-node collective communication primitives

HOROVOD

gloo

NVIDIA

Uber

Ring-based collective communication protocols

Topology Heterogeneity

1. Different server configurations
2. Link heterogeneity
3. Fragmentation in multi-tenant clusters

Can we do better than state-of-the-art?

BLINK

Topology Heterogeneity

1. Different server configurations
2. Link heterogeneity
3. Fragmentation in multi-tenant clusters

Talk Outline

- Motivation
- Challenges to achieving high-performance collective communication
    1. Different server configurations
    2. Link heterogeneity
    3. Fragmentation in multi-tenant clusters
- **Design**
- Evaluation

How Blink handles topology heterogeneity

Topology Heterogeneity

Blink

Different server configurations → Probe available links at job run time

How Blink handles topology heterogeneity

Topology Heterogeneity		Blink
Different server configurations	→	Probe available links at job run time
Link heterogeneity	→	Concurrent data transfer over heterogenous links

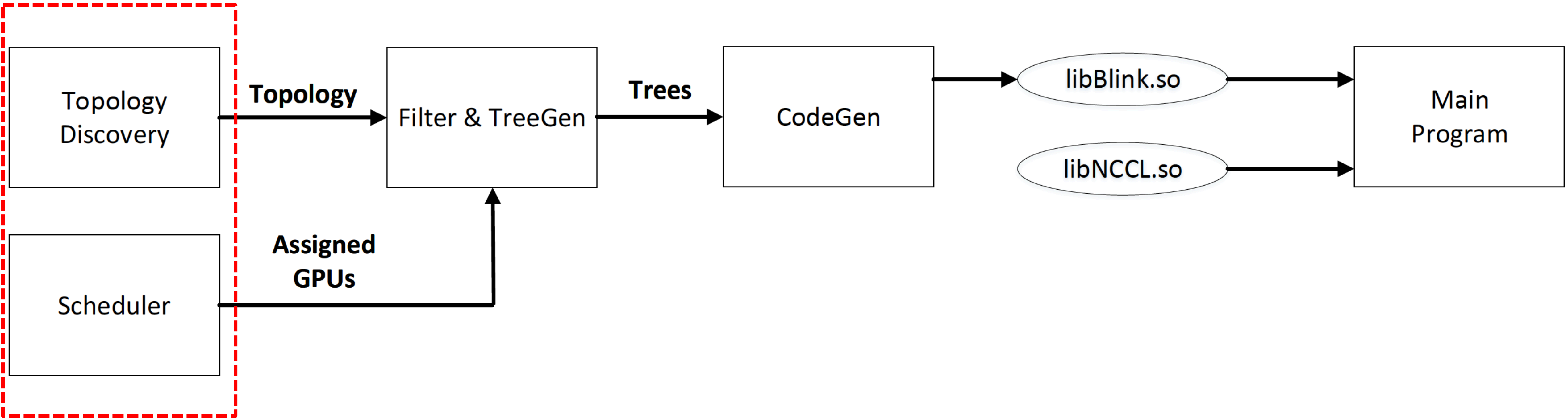
How Blink handles topology heterogeneity

Topology Heterogeneity		Blink
Different server configurations	→	Probe available links at job run time
Link heterogeneity	→	Concurrent data transfer over heterogenous links
Fragmentation in multi-tenant clusters (irregular topology)	→	Spanning trees (v.s. Rings) are more flexible and optimal.

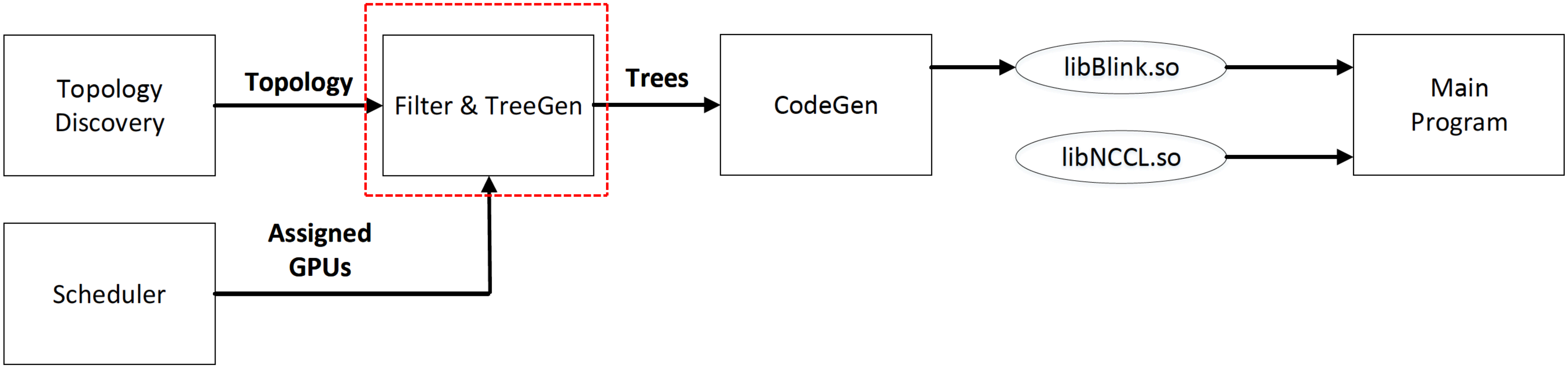
How Blink handles topology heterogeneity

Topology Heterogeneity		Blink
Different server configurations	→	Probe available links at job run time
Link heterogeneity	→	Concurrent data transfer over heterogenous links
Fragmentation in multi-tenant clusters (irregular topology)	→	Spanning trees (v.s. Rings) are more flexible and optimal.
	More →	NCCL-compatible API, seamless integration with TF, PyTorch, etc.

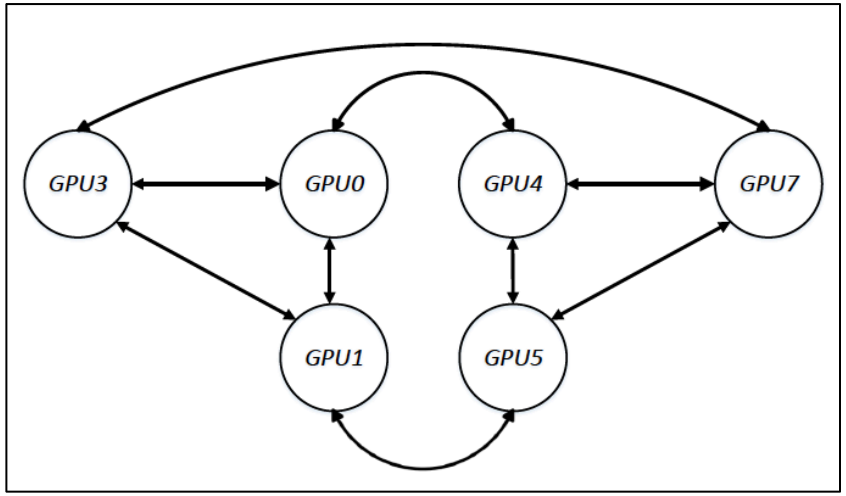
Blink workflow



Blink workflow



Broadcast comparison (Trees v.s. Rings)

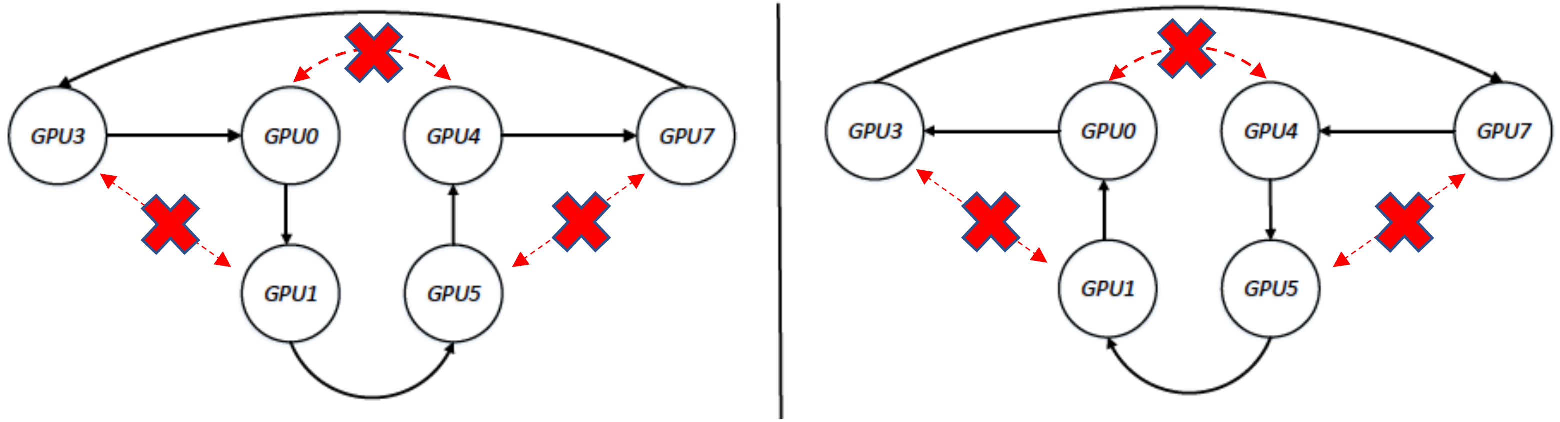


6-GPU topology

Broadcast comparison (Trees v.s. Rings)

Unused link

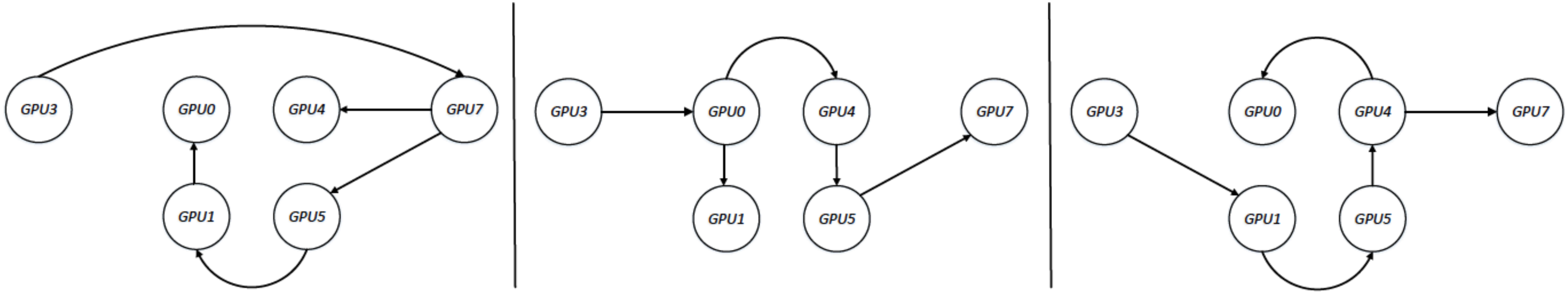
Broadcast from GPU3



NCCL 2 rings

Broadcast comparison (Trees v.s. Rings)

Broadcast from GPU3



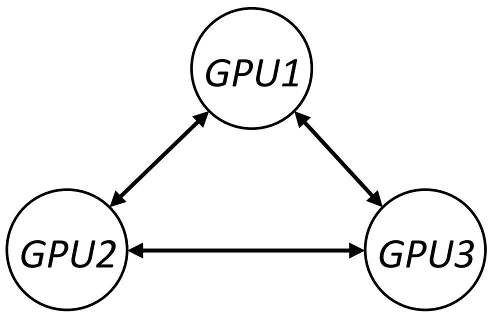
Blink 3 Spanning trees

3 Spanning trees > 2 Rings

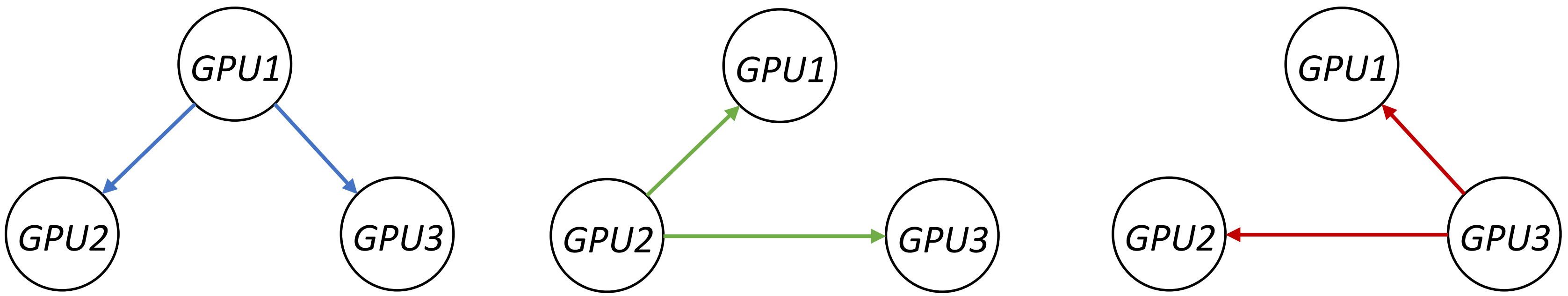
Use **All** the links available → **Optimal**

TreeGen: packing max. spanning trees

- Given available topology, pack max. unidirectional spanning trees.



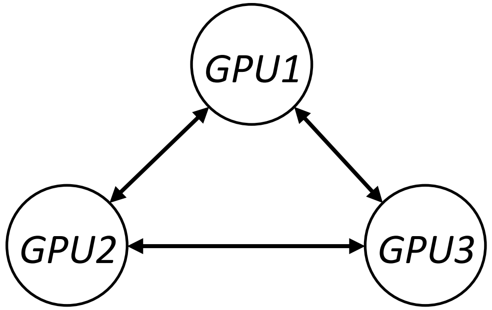
Topology



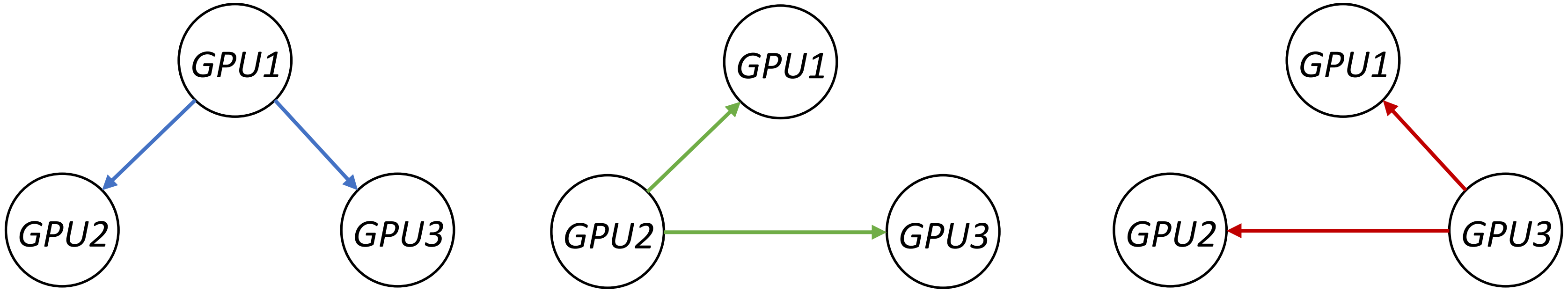
Packing unidirectional spanning trees

TreeGen: packing max. spanning trees

- Given available topology, pack max. unidirectional spanning trees.



Topology



Packing unidirectional spanning trees

Optimization problem

$$\max \sum_i w_i$$

Maximize the sum of bandwidth usage among all links

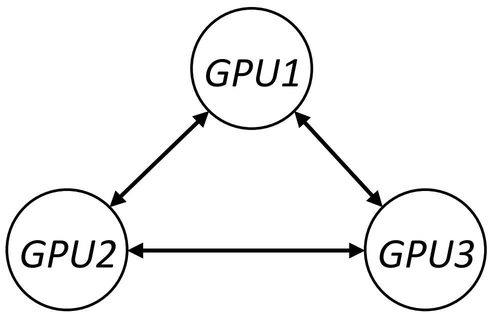
$$\text{such that } \forall e \in E, \sum_i \kappa_i * w_i < c_e$$

$$\text{where } \kappa_i = \begin{cases} 1, & \text{if } e \in T_i \\ 0, & \text{otherwise} \end{cases}$$

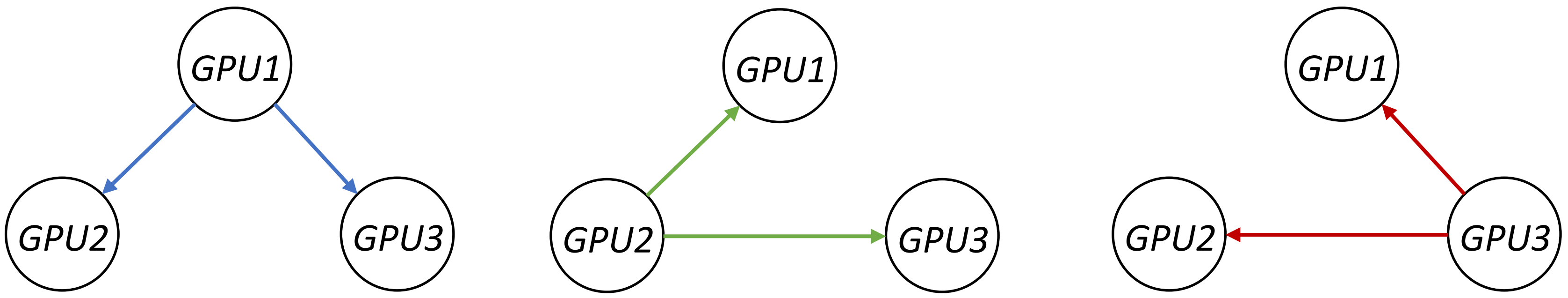
Constrain: amount of BW usage should not exceed ANY link capacity when packing multiple trees

TreeGen: packing max. spanning trees

- Given available topology, pack max. unidirectional spanning trees.



Topology



Packing unidirectional spanning trees

Optimization problem

$$\max \sum_i w_i$$

Maximize the sum of bandwidth usage among all links

$$\text{such that } \forall e \in E, \sum_i \kappa_i * w_i < c_e$$

$$\text{where } \kappa_i = \begin{cases} 1, & \text{if } e \in T_i \\ 0, & \text{otherwise} \end{cases}$$

Constrain: amount of BW usage should not exceed ANY link capacity when packing multiple trees

Too many trees!
181 spanning trees for 8-GPU DGX-1V

Data size per-tree is too small to fully saturate link BW.

TreeGen: packing max. spanning trees

- Given available topology, pack max. unidirectional spanning trees.

Optimization problem

$$\max \sum_i w_i$$

$$\text{such that } \forall e \in E, \sum_i \kappa_i * w_i < c_e$$

$$\text{where } \kappa_i = \begin{cases} 1, & \text{if } e \in T_i \\ 0, & \text{otherwise} \end{cases}$$

approximation

Approximate packing

$$\max \sum_{i=1}^{k} w_i$$

$$\text{such that } \forall e \in E, \sum_i \kappa_i * w_i < c_e$$

$$\boxed{\forall w_i \in \{0, 1\}}$$

$$\text{where } \kappa_i = \begin{cases} 1, & \text{if } e \in T_i \\ 0, & \text{otherwise} \end{cases}$$

Either a tree use ALL BW of a link, or not use it.

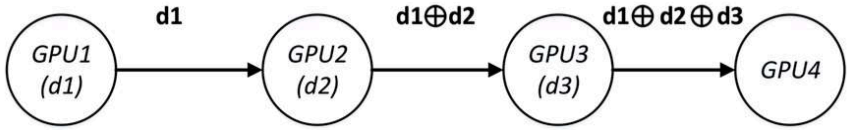
181 trees for 8GPU DGX-1V

approximation

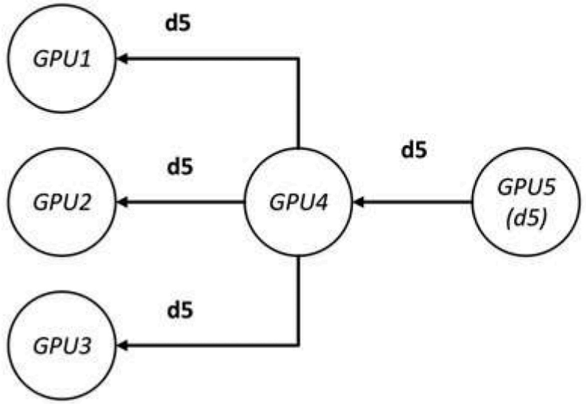
6 trees for 8GPU DGX-1V

TreeGen

- Given available topology, pack max. unidirectional spanning trees
- Direct support for one-to-many/many-to-one primitives
  - e.g. Reduce, Broadcast, etc.



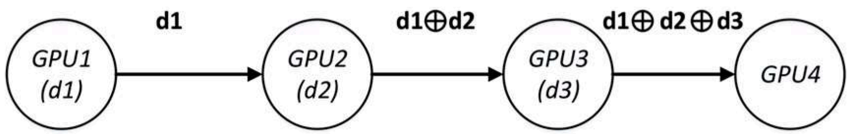
Reduce



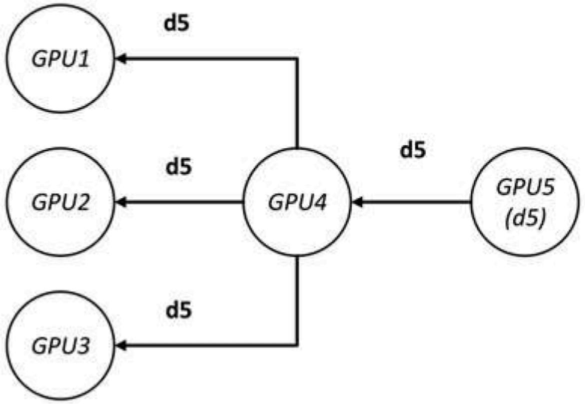
Broadcast

TreeGen

- Given available topology, pack max. unidirectional spanning trees
- Direct support for one-to-many/many-to-one primitives
  - e.g. Reduce, Broadcast, etc.



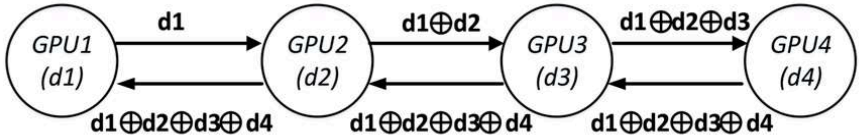
Reduce



Broadcast

- Extend to many-to-many primitives (e.g. AllReduce)
  - Pick a root node, reduce towards root, then broadcast in reverse direction.

Reduce →
Broadcast ←

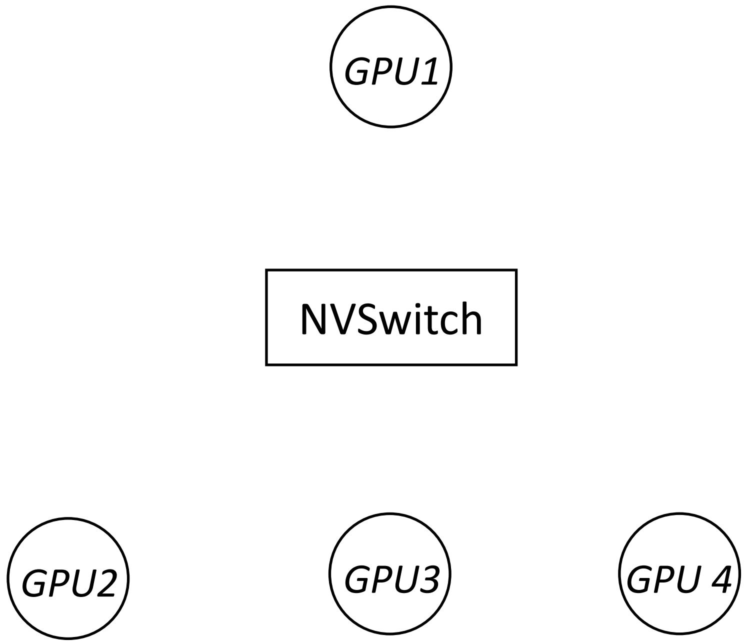


AllReduce

TreeGen for NVSwitch (DGX-2)

TreeGen for NVSwitch (DGX-2)

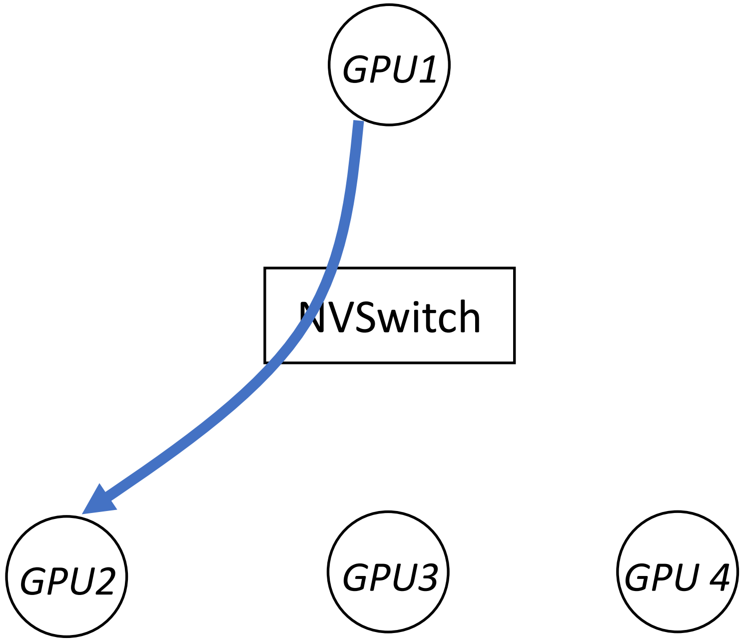
- With NVSwitch, the connectivity among any subset of GPUs is uniform
- NCCL constructs a multi-hop ring.



4GPU Reduce (G1->G4)

TreeGen for NVSwitch (DGX-2)

- With NVSwitch, the connectivity among any subset of GPUs is uniform
- NCCL constructs a multi-hop ring.



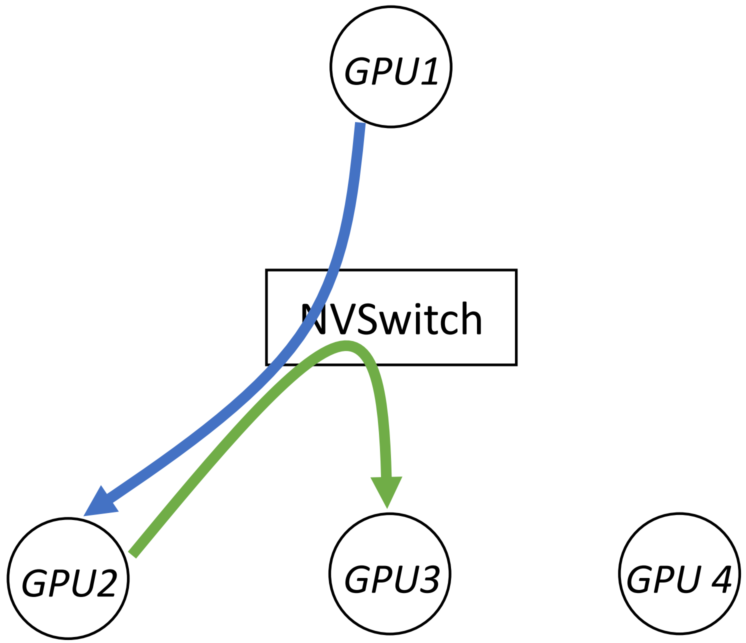
Hop Count

1

4GPU Reduce (G1->G4)

TreeGen for NVSwitch (DGX-2)

- With NVSwitch, the connectivity among any subset of GPUs is uniform
- NCCL constructs a multi-hop ring.



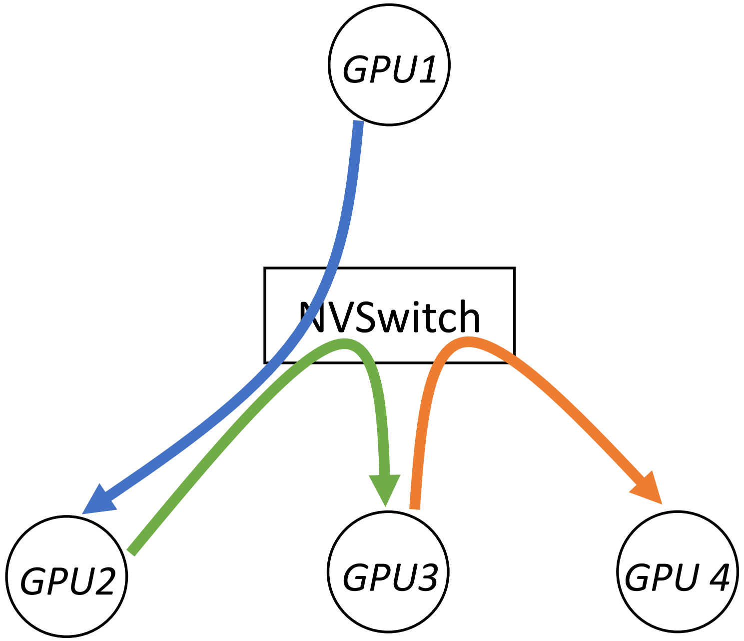
Hop Count

2

4GPU Reduce (G1->G4)

TreeGen for NVSwitch (DGX-2)

- With NVSwitch, the connectivity among any subset of GPUs is uniform
- NCCL constructs a multi-hop ring.



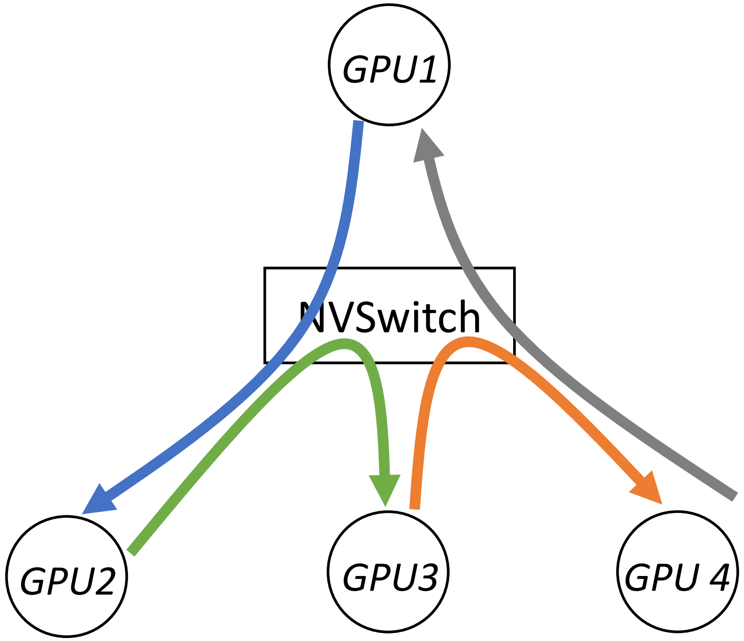
Hop Count

3

4GPU Reduce (G1->G4)

TreeGen for NVSwitch (DGX-2)

- With NVSwitch, the connectivity among any subset of GPUs is uniform
- NCCL constructs a multi-hop ring.



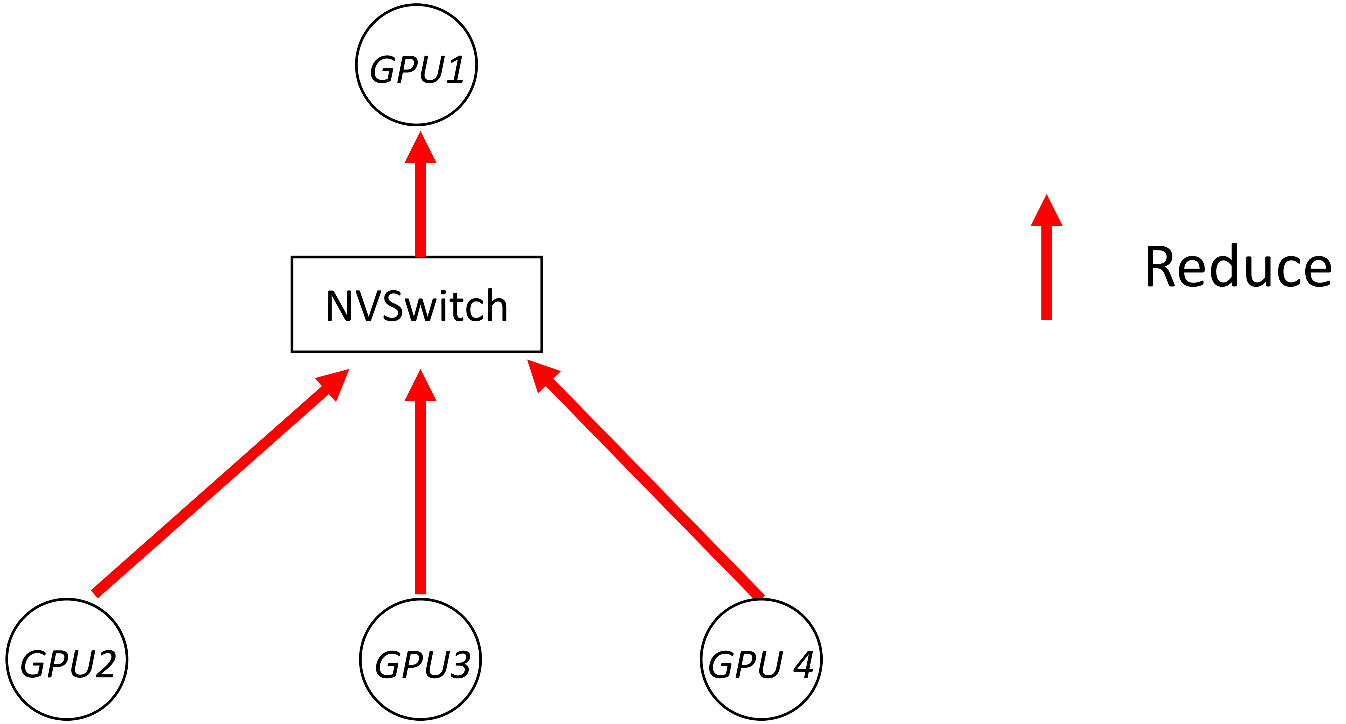
Hop Count

4

4GPU Reduce (G1->G4)

TreeGen for NVSwitch (DGX-2)

- DGX-2 single-hop tree

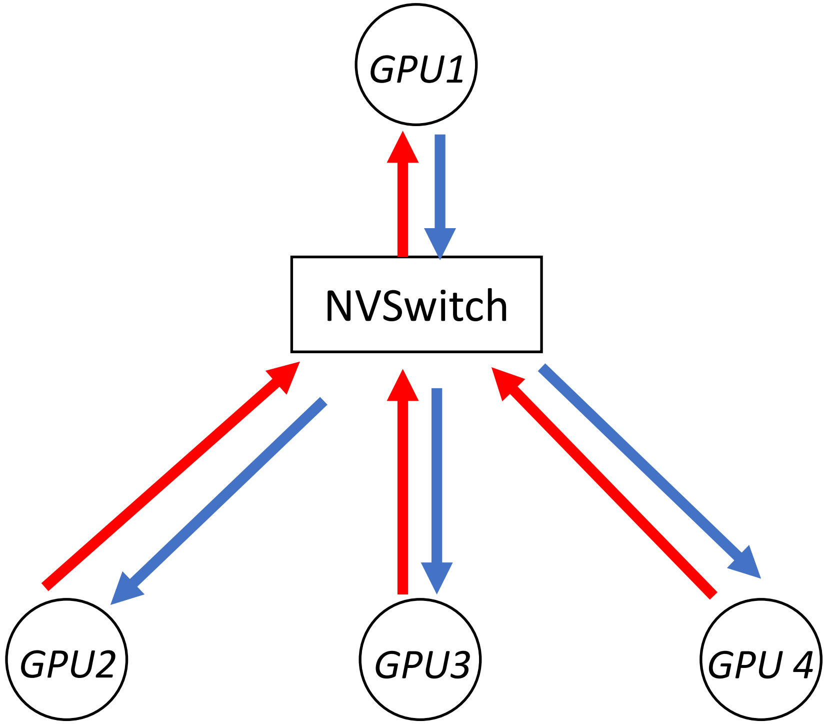


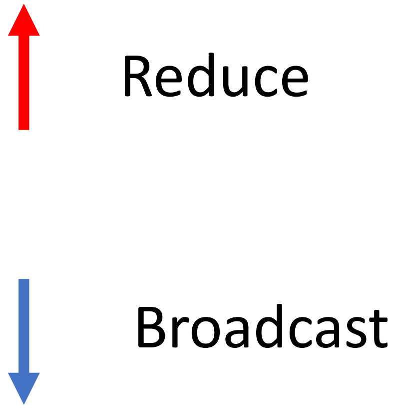
1-hop tree introduces Min. latency

4GPU Reduce

TreeGen for NVSwitch (DGX-2)

- DGX-2 single-hop tree



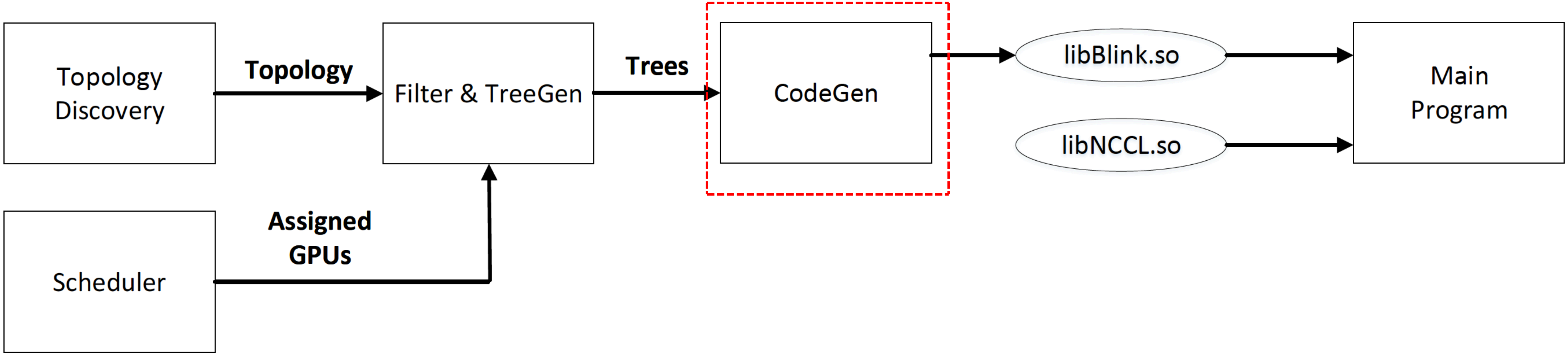


1-hop tree introduces Min. latency

AllReduce → Reduce, Broadcast

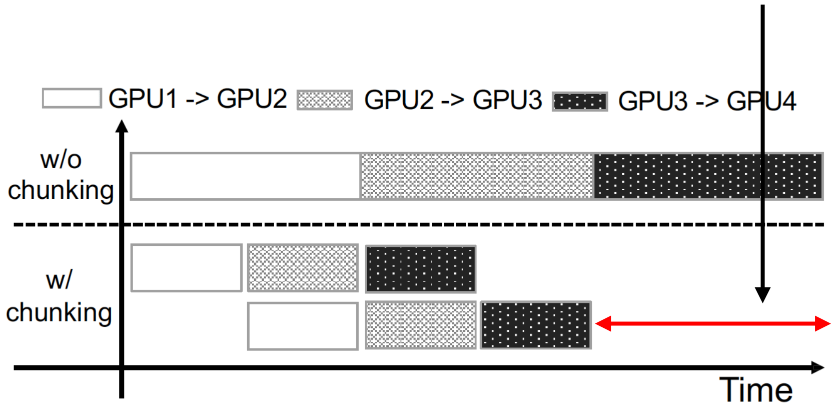
For N GPUs,
N 1-hop trees, with each tree responsible for 1/N data.

Blink workflow



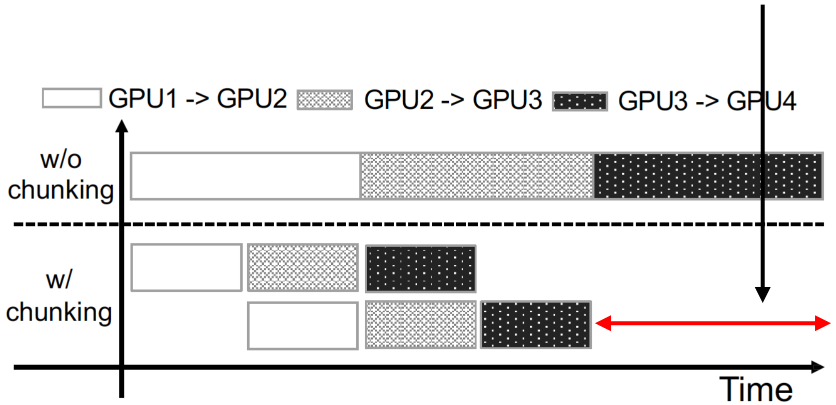
CodeGen

- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency



CodeGen

- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency

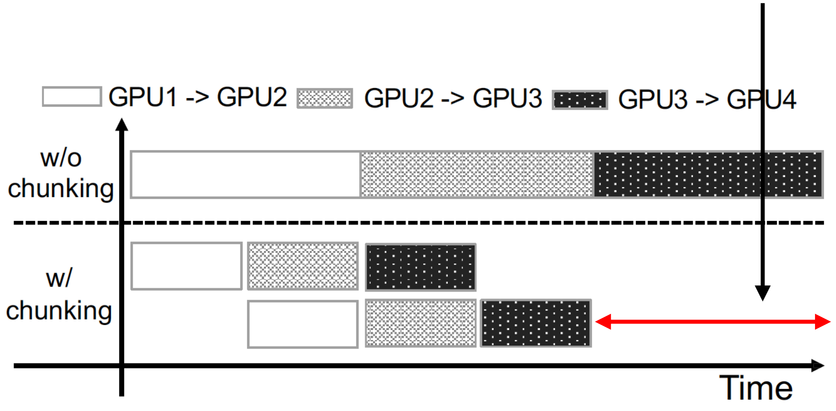


What chunk size to use?

- Too small, cannot fully utilize BW
- Too big, high latency

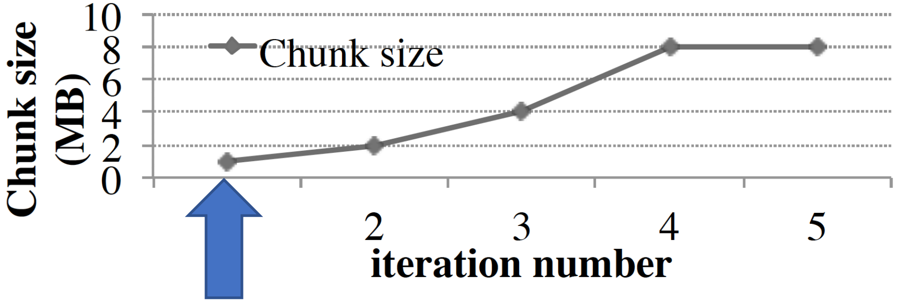
CodeGen

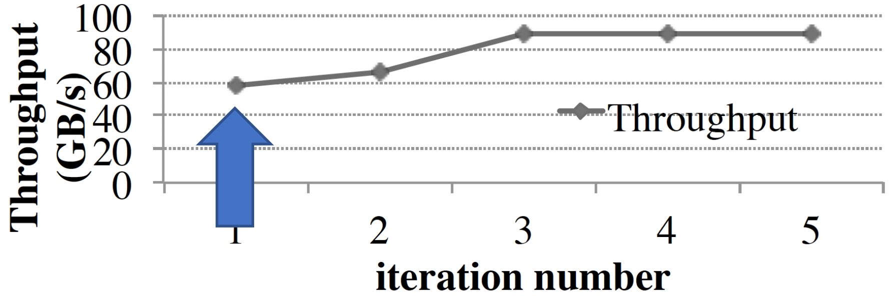
- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency



Automatic chunk size selection

MIAD
(multiple-increase, additive-decrease)



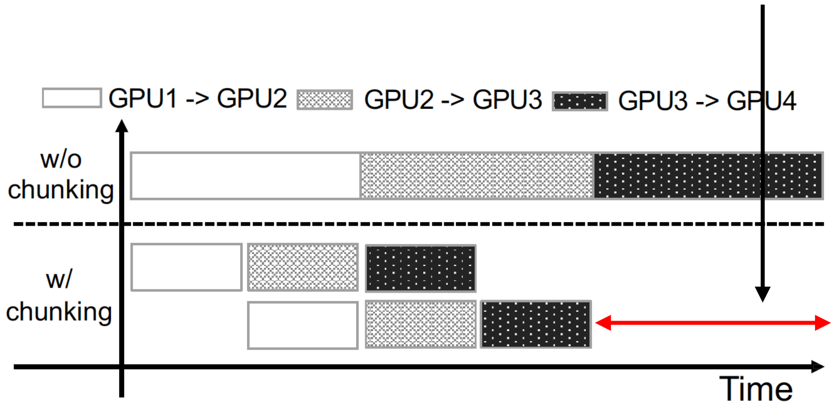


What chunk size to use?

- Too small, cannot fully utilize BW
- Too big, high latency

CodeGen

- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency



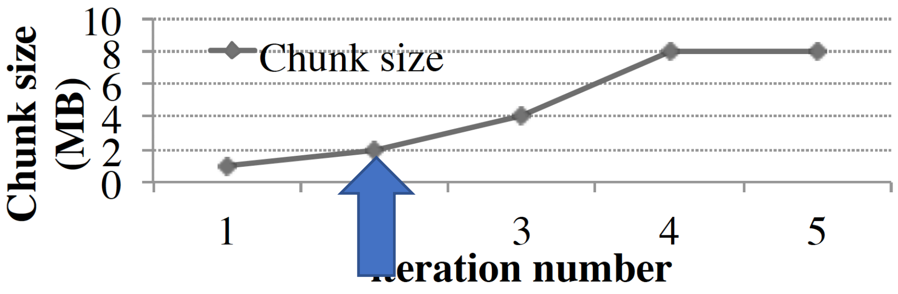
What chunk size to use?

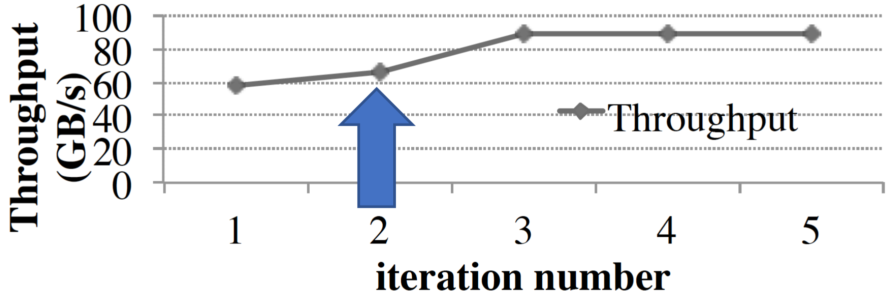
- Too small, cannot fully utilize BW
- Too big, high latency

Automatic chunk size selection

MIAD

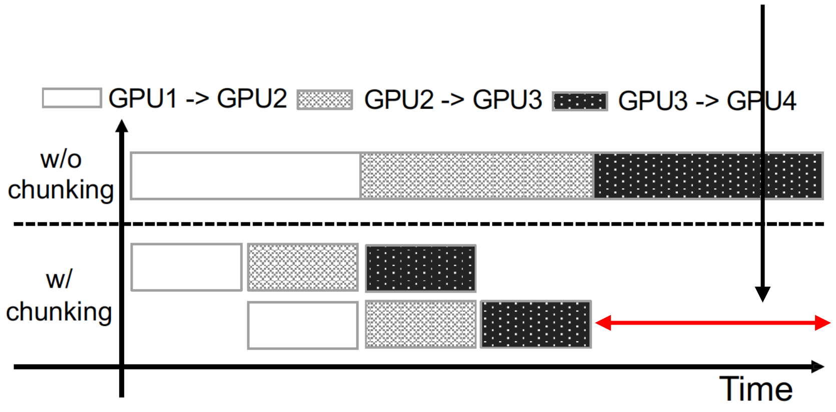
(multiple-increase, additive-decrease)





CodeGen

- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency

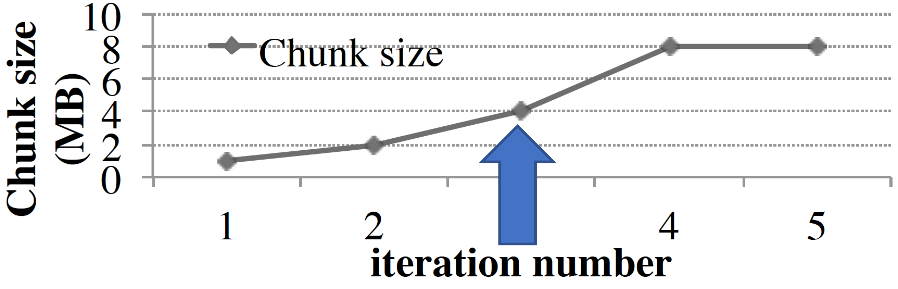


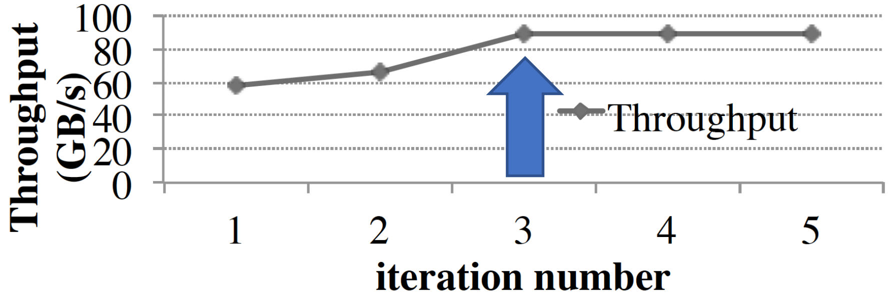
What chunk size to use?

- Too small, cannot fully utilize BW
- Too big, high latency

Automatic chunk size selection

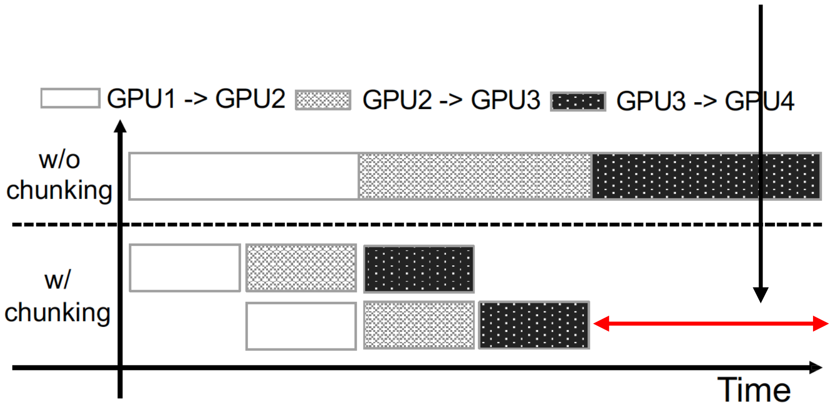
MIAD
(multiple-increase, additive-decrease)





CodeGen

- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency

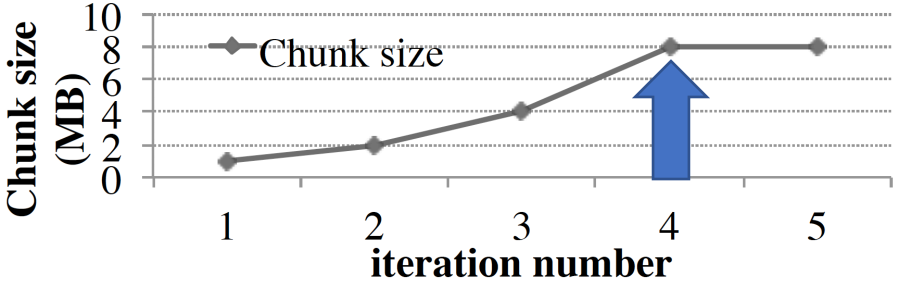


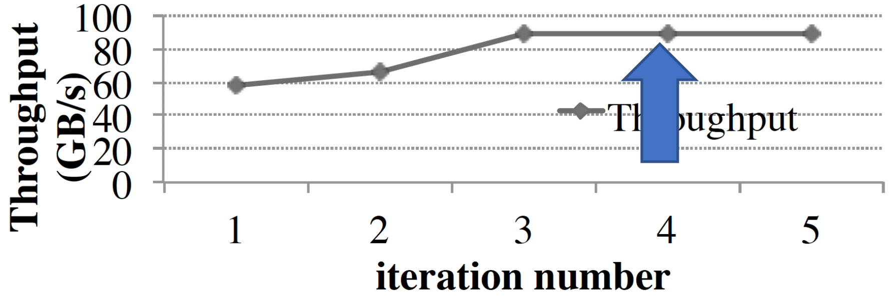
What chunk size to use?

- Too small, cannot fully utilize BW
- Too big, high latency

Automatic chunk size selection

MIAD
(multiple-increase, additive-decrease)





Blink design recap

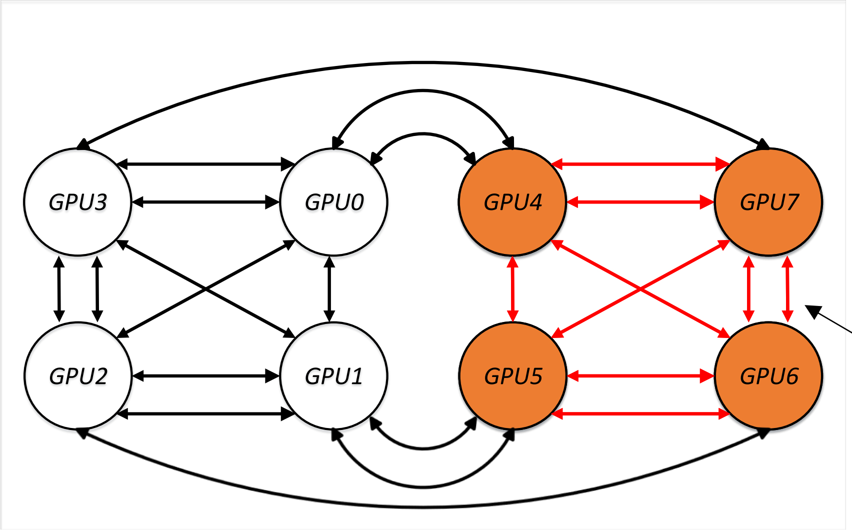
- Packing spanning trees while minimizing trees
  - Single hop trees for DGX-2 (NVSwitch)
- Chunking, pipelining transfers for max link utilization
  - Auto chunk size selection with MIAD
- GPU stream reuse for fair sharing of links
- PCIe + NVLink Hybrid transfers
- Support for multi-machine collectives

Drop-in NCCL replacement (load-time, no code recompile)

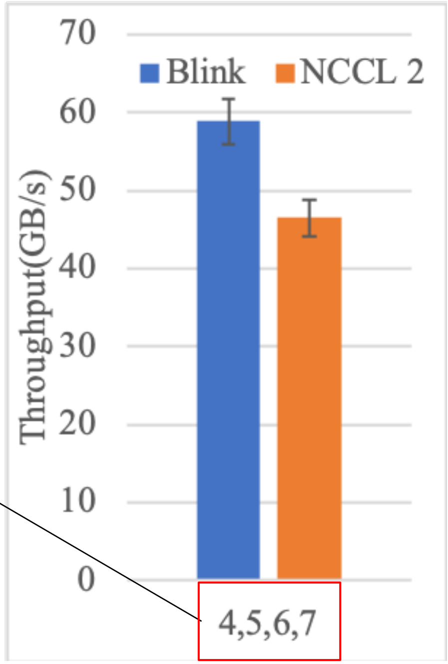
Talk Outline

- Motivation
- Challenges to achieving high-performance collective communication
    1. Different server configurations
    2. Link heterogeneity
    3. Fragmentation in multi-tenant clusters
- Design
- **Evaluation**
    - **AllReduce and Broadcast Microbenchmarks**
    - **End-to-end improvements**
    - **Benefits of One-Hop Trees over Rings or Double Binary trees**
    - **Rest of the extensive evaluation → refer to the paper**

Microbenchmarks (DGX-1V)

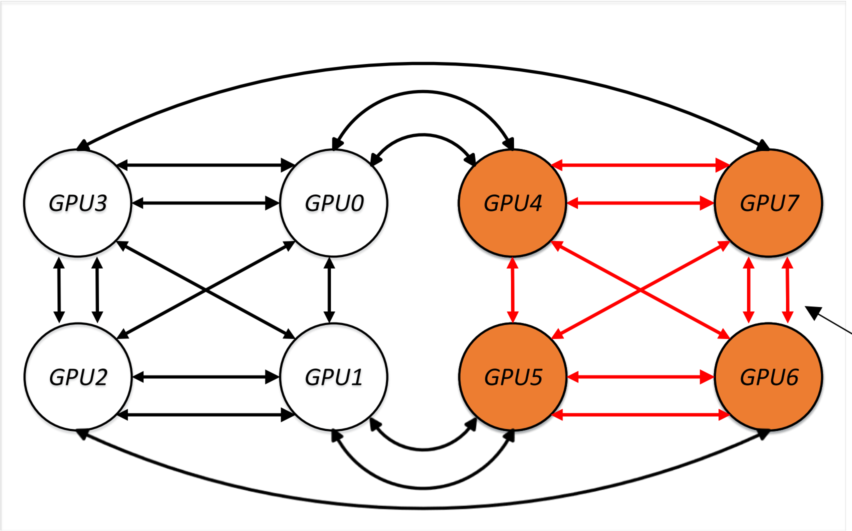


Topology

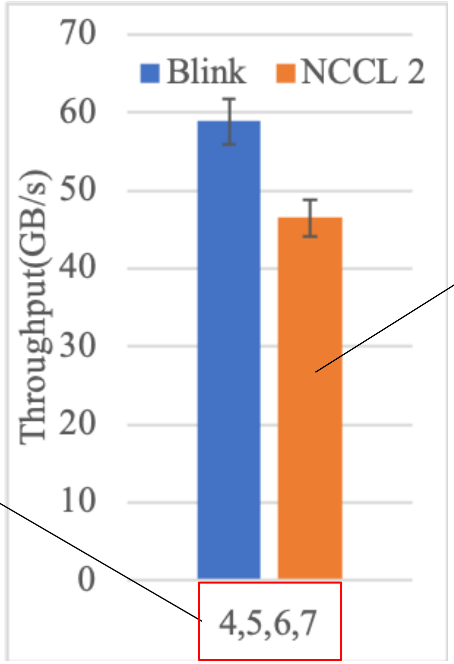


AllReduce

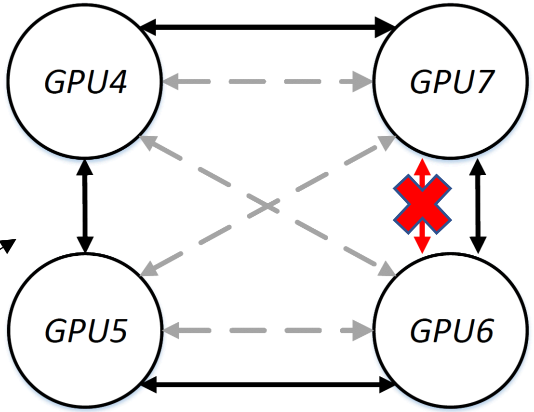
Microbenchmarks (DGX-1V)



Topology

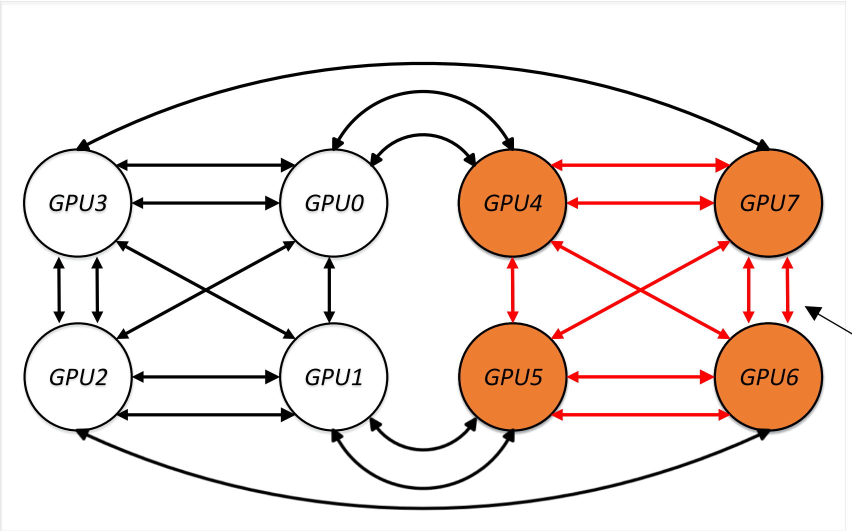


AllReduce

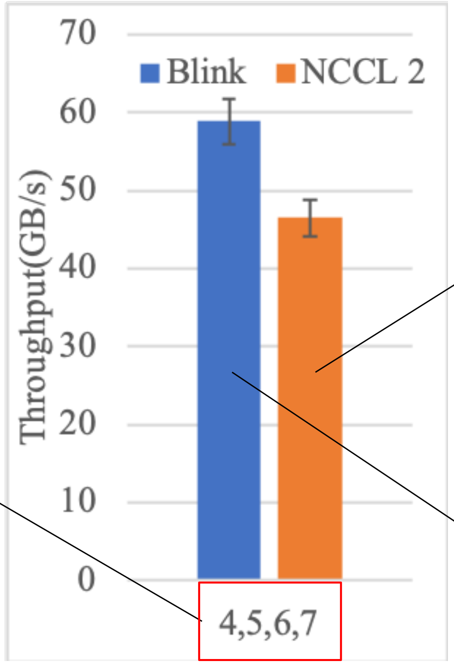


NCCL2 (2 rings)

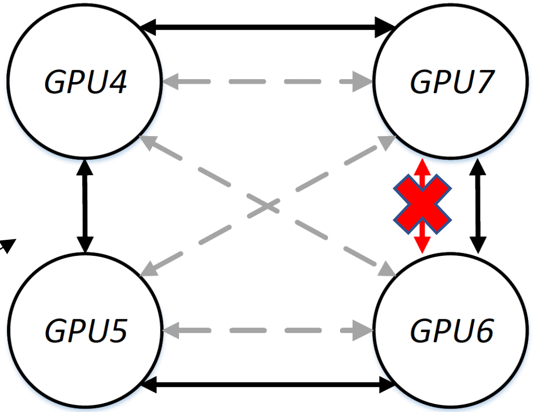
Microbenchmarks (DGX-1V)



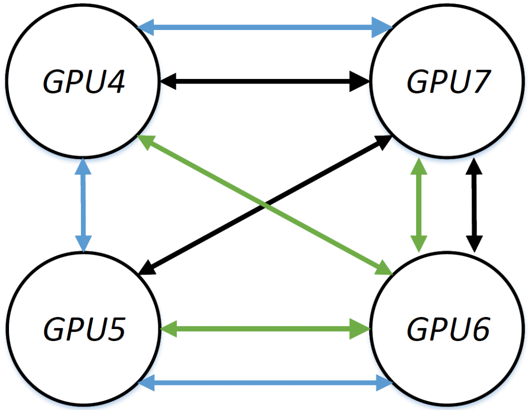
Topology



AllReduce

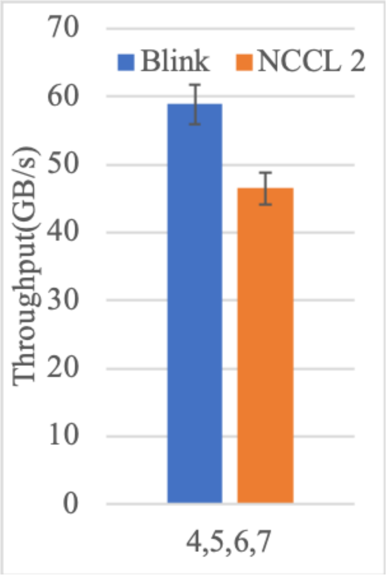


NCCL2 (2 rings)



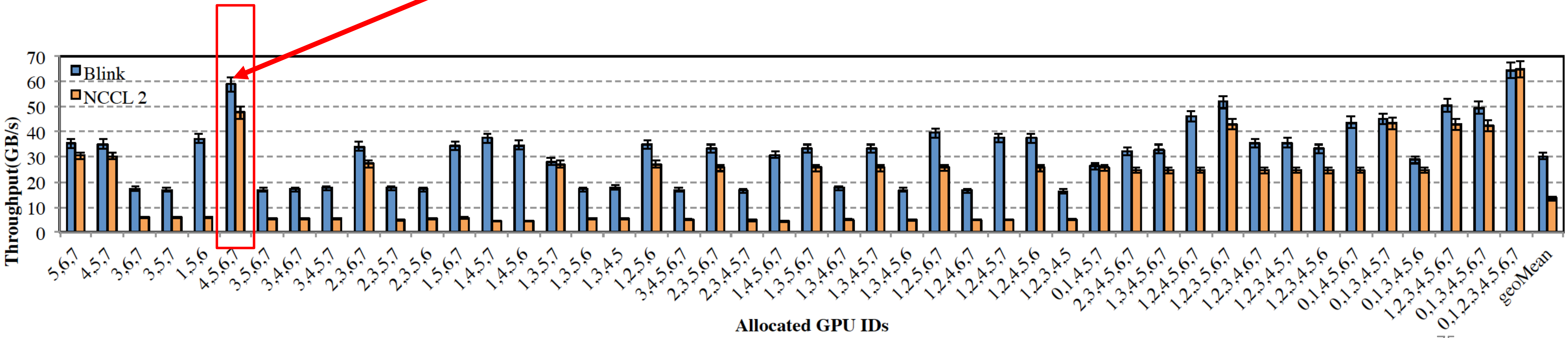
Blink (3 spanning trees)

Microbenchmarks (DGX-1V)

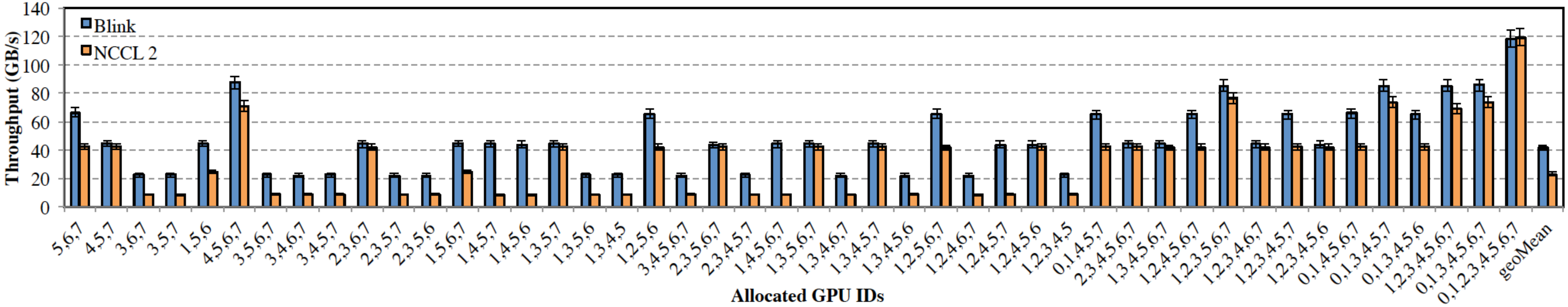


AllReduce

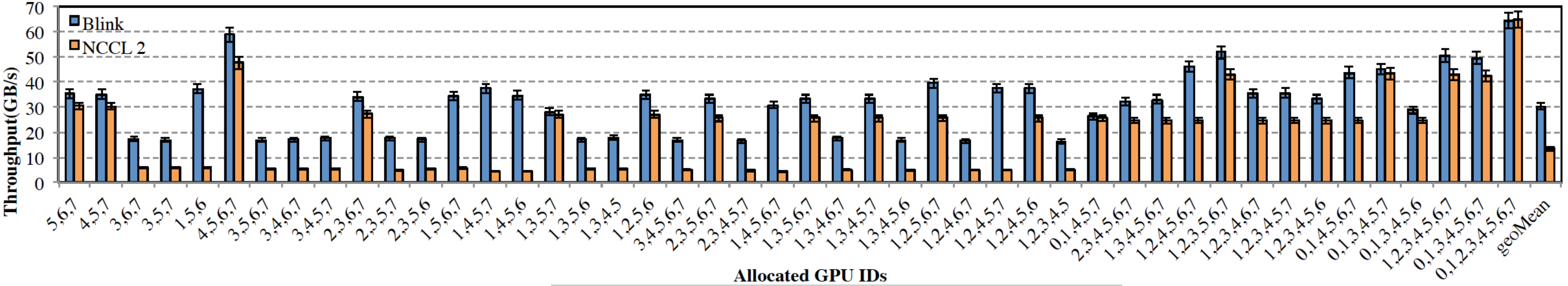
(up to 8x speed-up, 2x geo-mean)



Microbenchmarks (DGX-1V)

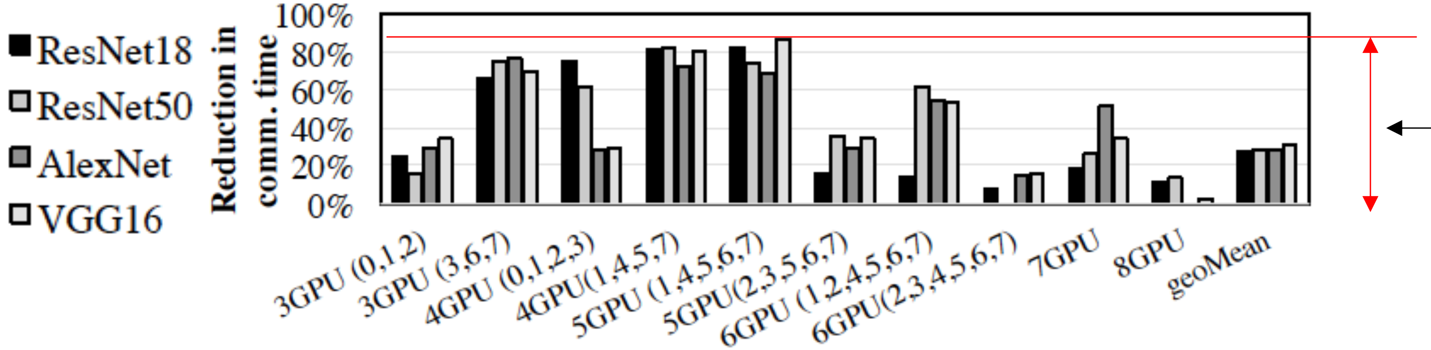


Broadcast (up to 6x speed-up, 2x geo-mean)



AllReduce (up to 8x speed-up, 2x geo-mean)

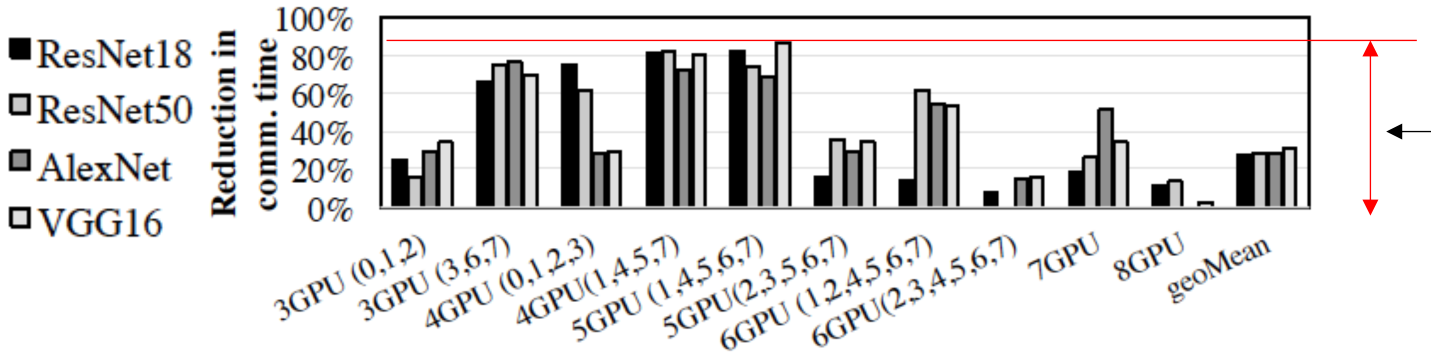
End-to-end Benchmarks (DGX-1V)



Blink end-to-end Communication time reduction (ImageNet1K)

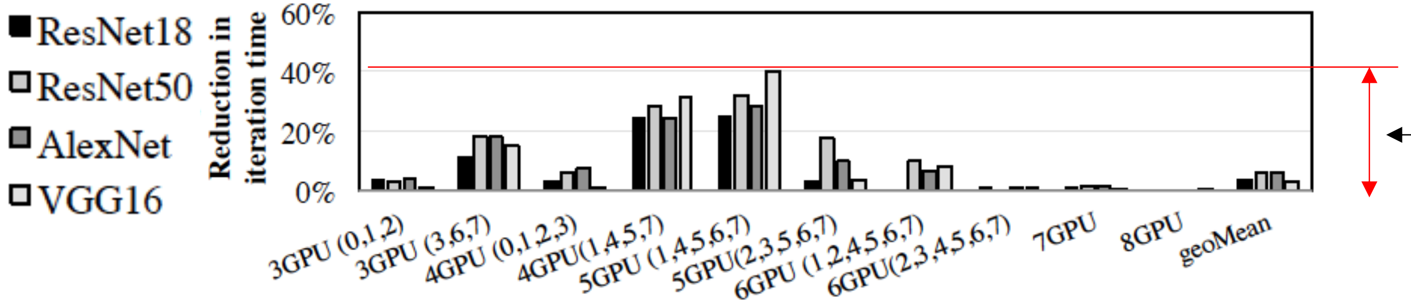
up to 87% Communication time reduction (51% avg.)

End-to-end Benchmarks (DGX-1V)



Blink end-to-end Communication time reduction (ImageNet1K)

up to 87% Communication time reduction (51% avg.)

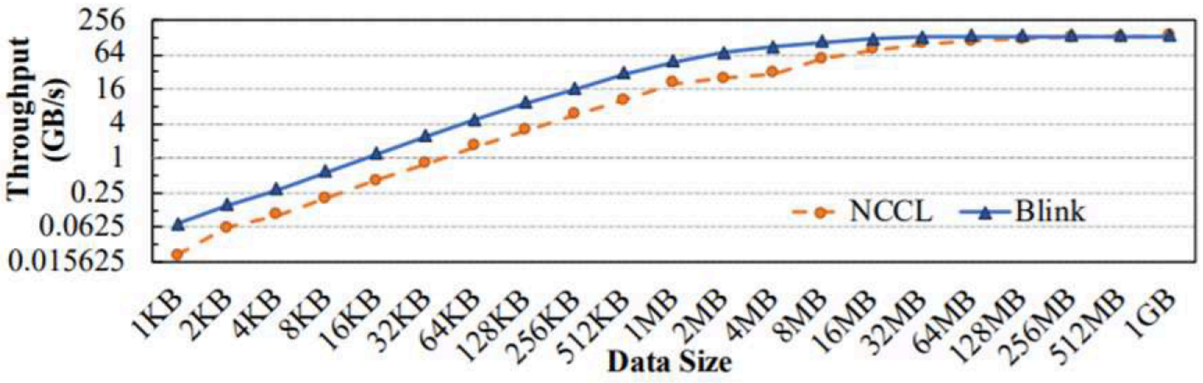


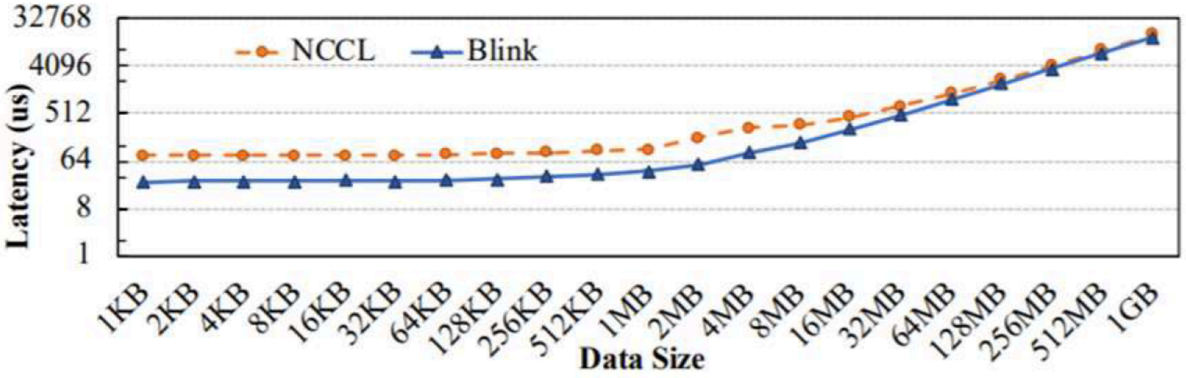
Blink end-to-end training time reduction (ImageNet1K)

up to 40% end-to-end training time reduction

Microbenchmarks (DGX-2)

16 GPU AllReduce





Throughput
(up to 3.5x speed-up)

Latency
(Up to 3.32x reduction)

Biggest win in small chunk sizes because our 1-hop tree achieve min. latency.

BLINK

Guanhua Wang guanhua@cs.berkeley.edu

- Topology heterogeneity results in link underutilization for collectives.

- Blink packs spanning trees for optimal link utilization

- Auto-generates one-to-all, all-to-one, all-to-all collectives
  - Broadcast, AllReduce, etc.

- Faster collective communication than NCCL
  - Up to 6x faster Broadcast (2x geo-mean)
  - Up to 8x faster AllReduce (2x geo-mean)
  - Up to 7.7x (2x geo-mean) communication time reduction in E2E data-parallel training on DGX-1 machines.

Back-ups

TreeGen

- Handle hybrid communication (e.g. PCIe & NVLink)
  - Balance amount of data transfer over different link types based on link bandwidth.
  - Take link type switching (i.e. *disable_peer_access*) latency into account.

$$
\begin{aligned}
&\text{Objective } T_{PCIe} + T_{dpa} = T_{NVL} \\
\implies D_{PCIe} =& \frac{D_{total} \times BW_{PCIe}}{BW_{PCIe} + BW_{NVL}} - \\
& \frac{T_{dpa} \times BW_{PCIe} \times BW_{NVL}}{BW_{PCIe} + BW_{NVL}} \\
D_{NVL} =& D_{total} - D_{PCIe}
\end{aligned}
$$

TreeGen

- Multi-server transfers

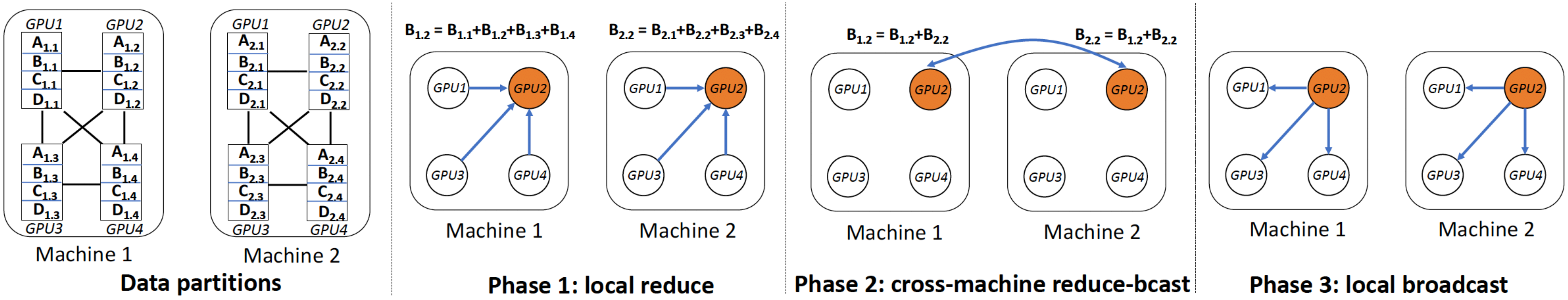
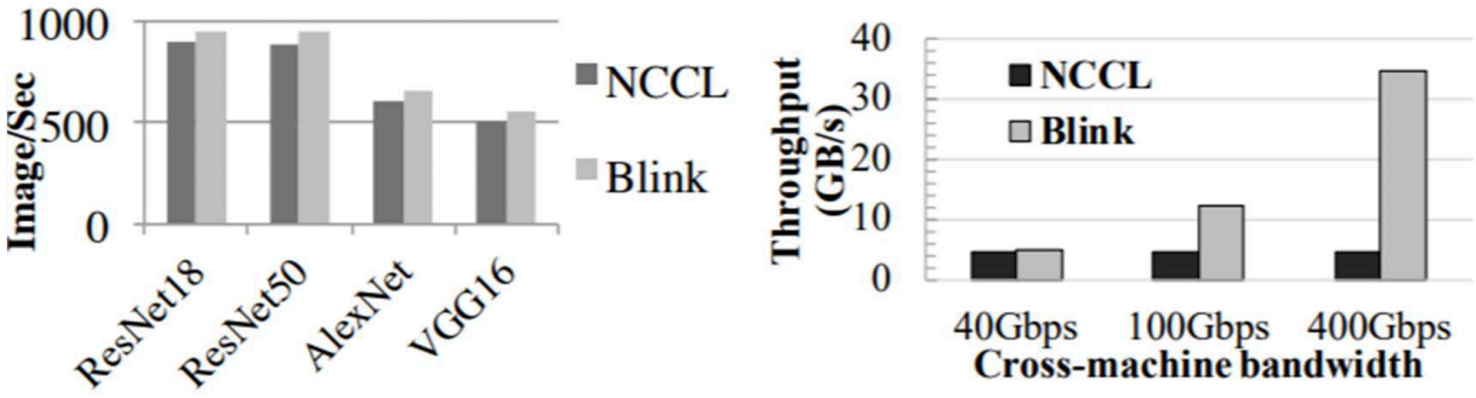


Figure 10: Three-phase AllReduce protocol for cross-machine settings. Data item $X_{m.g}$ refers to data partition X on server m and GPU g. Each data partition has a distinct server-local root. The figure above shows the reduction (function is denoted as $+$) for partition B which has a root at $GPU2$. Similar protocol is followed for other data partitions.

Multiple DGX-1s DNN Training



(a) Using 2 DGX-1Vs (b) AllReduce Projections

- 8-GPU job on 2 DGX-1V machines (5-3 GPU placement)
- Inter-server tput (40Gb/s) < Intra-server tput (40GB/s)
- Projection with 100/400 Gbps inter-server bandwidth, highlight Blink's advantage.