Disruptive Research on Distributed ML Systems

Guanhua Wang

Advisor: Prof. Ion Stoica

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

Image Classification

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

Image Classification

Hey Siri

amazon alexa

Speech Recognition

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

Image Classification

Robot Control

Hey Siri

amazon alexa

Speech Recognition

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

Image Classification

Robot Control

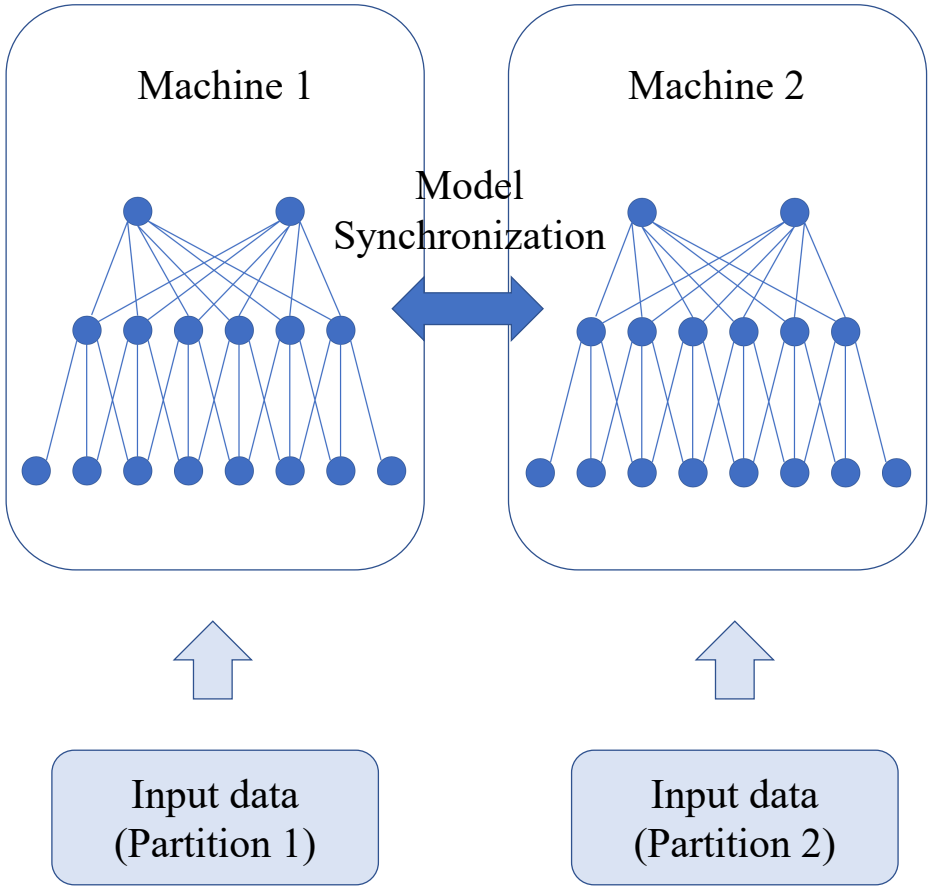
Hey Siri

amazon alexa

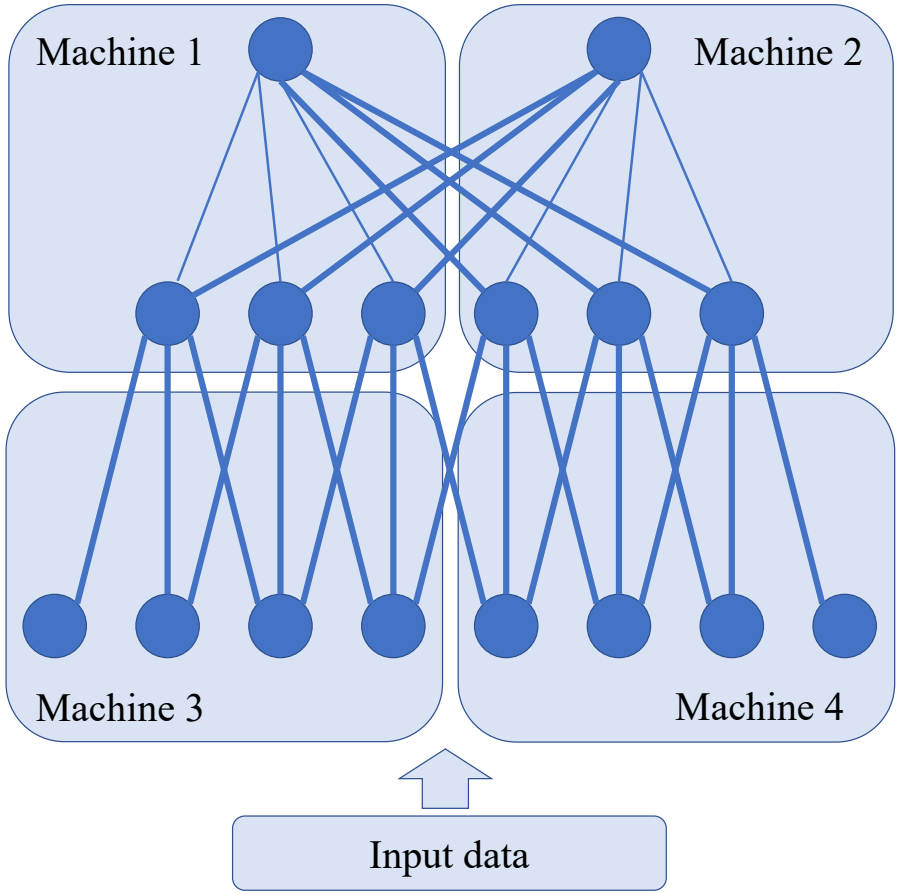
Speech Recognition

Game Playing

Speed-up DNN training and inference: Data and Model Parallelism



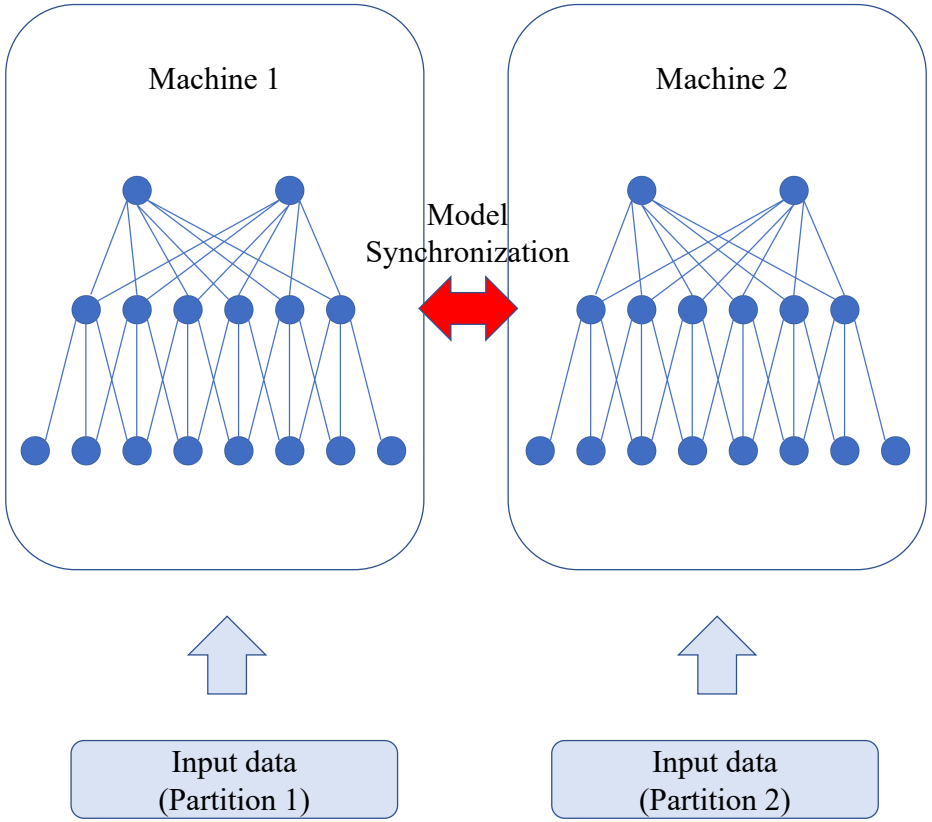
Data Parallelism



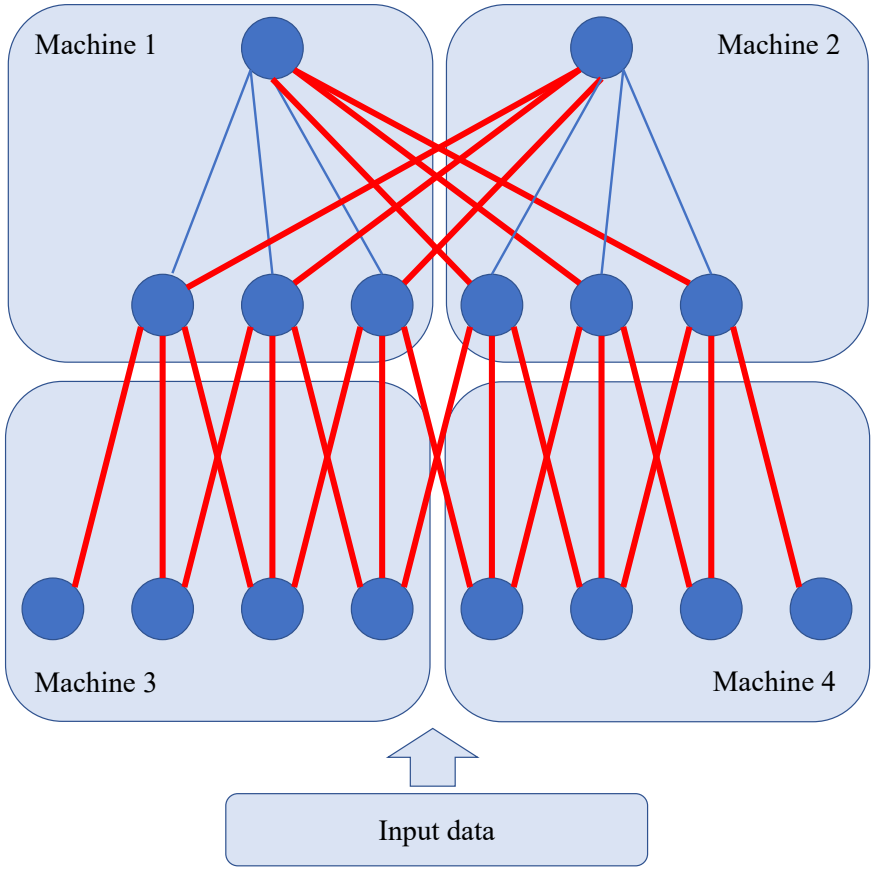
Model Parallelism

Communication and Memory are the main causes for system inefficiency in distributed ML

- Communication



Data Parallelism

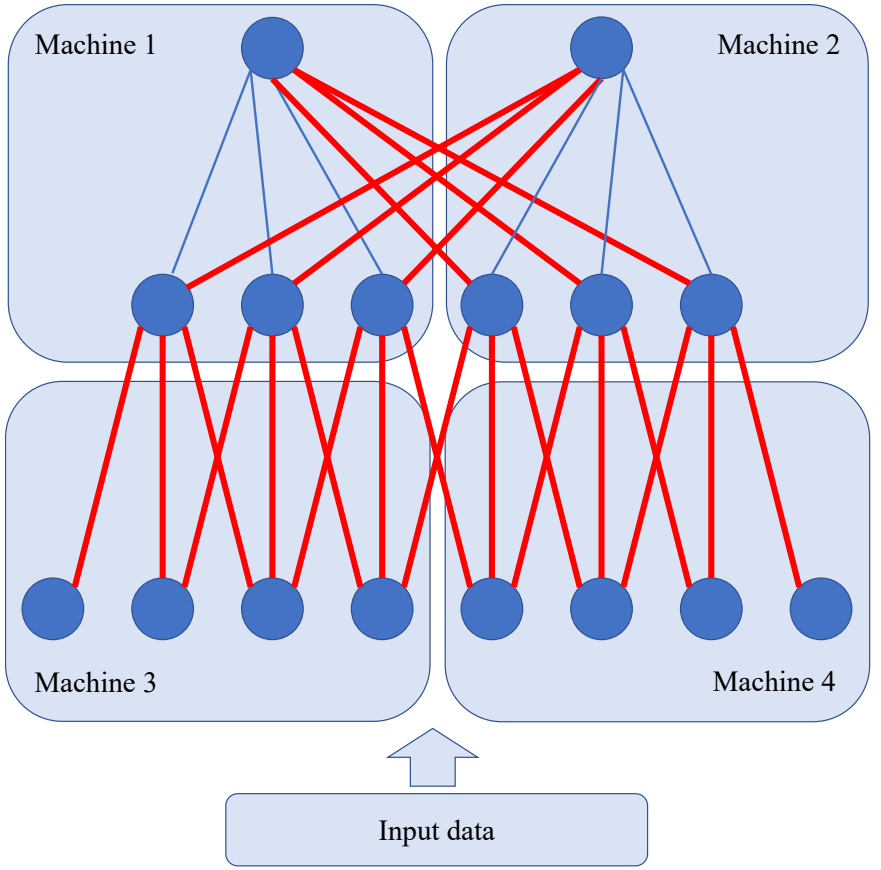


Model Parallelism

Communication and Memory are the main causes for system inefficiency in distributed ML

- Communication

Data Parallelism

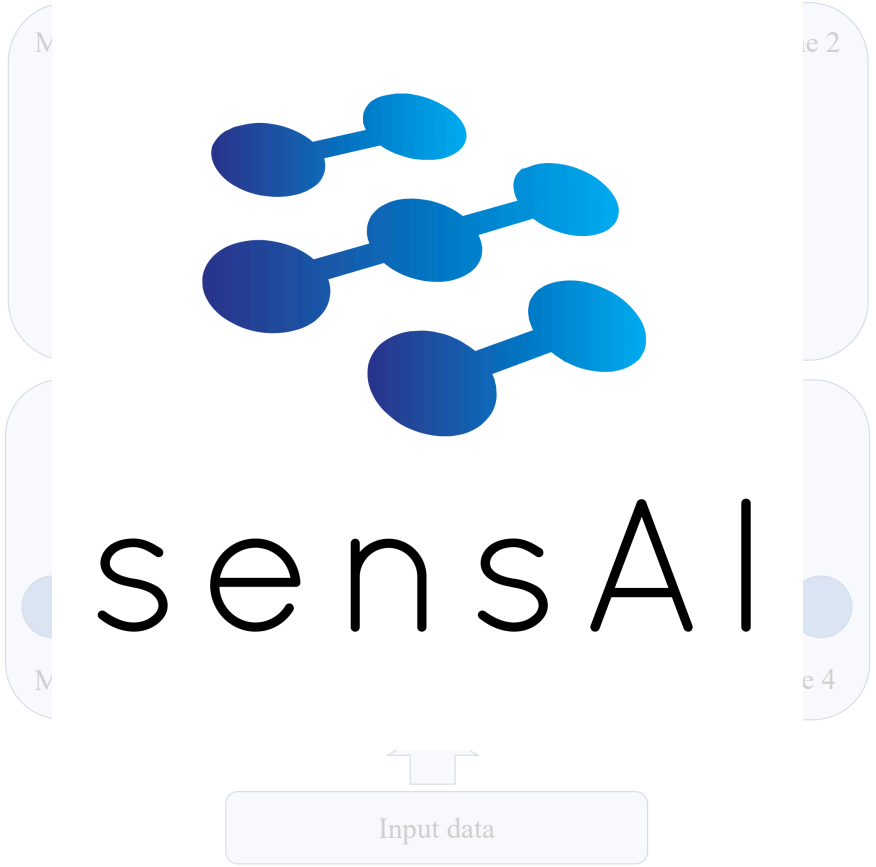


Model Parallelism

Communication and Memory are the main causes for system inefficiency in distributed ML

- Communication

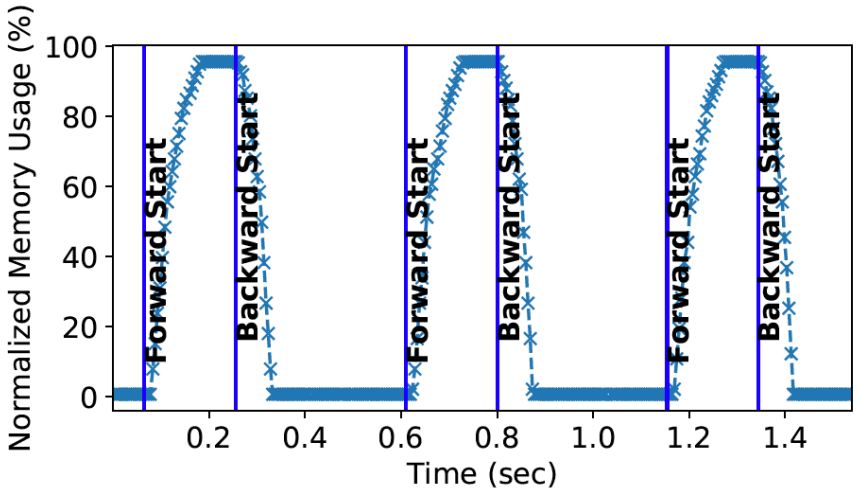
Data Parallelism

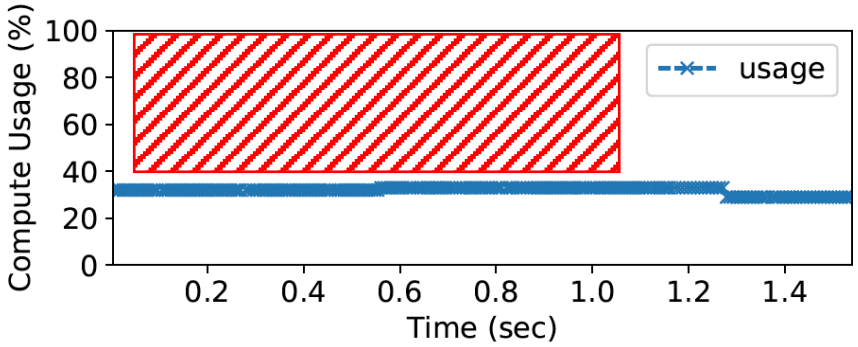


Model Parallelism

Communication and Memory are the main causes for system inefficiency in distributed ML

- Memory





Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Communication and Memory are the main causes for system inefficiency in distributed ML

- Memory

Wavelet

Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Three works

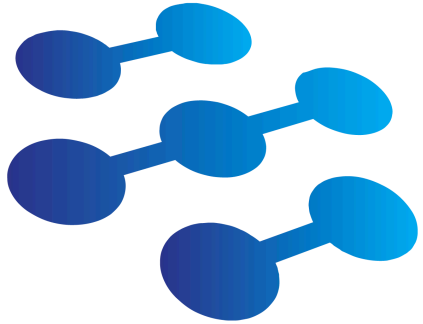
BLINK

Optimal model synchronization scheme in Data Parallelism

Three works

BLINK

Optimal model synchronization scheme in Data Parallelism



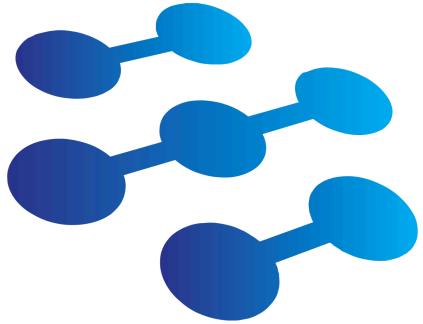
sensAI

Eliminate communication in Model Parallelism

Three works

BLINK

Optimal model synchronization scheme in Data Parallelism



sensAI

Eliminate communication in Model Parallelism

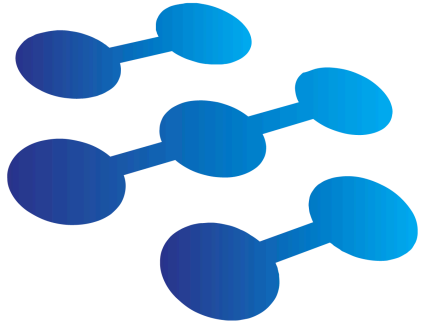
Wavelet

More efficient on-device memory usage

Three works

BLINK

Optimal model synchronization scheme in Data Parallelism



sensAI

Eliminate communication in Model Parallelism

Communication

Wavelet

More efficient on-device memory usage

Memory

BLINK

Fast and Generic Collectives for Distributed ML

MLSys 2020

Guanhua Wang, Shivaram Venkataraman, Amar Phanishayee

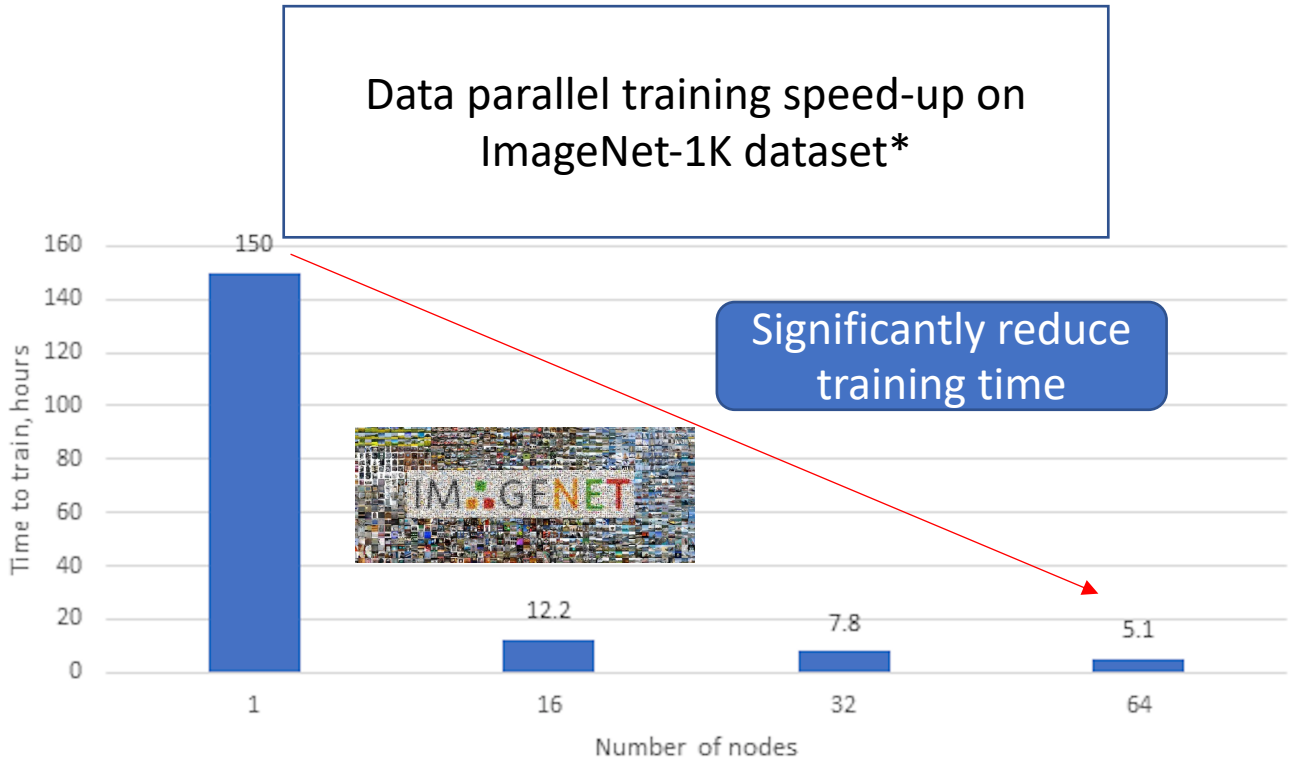
Jorgen Thelin, Nikhil R. Devanur, Ion Stoica

riselab UC Berkeley

WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

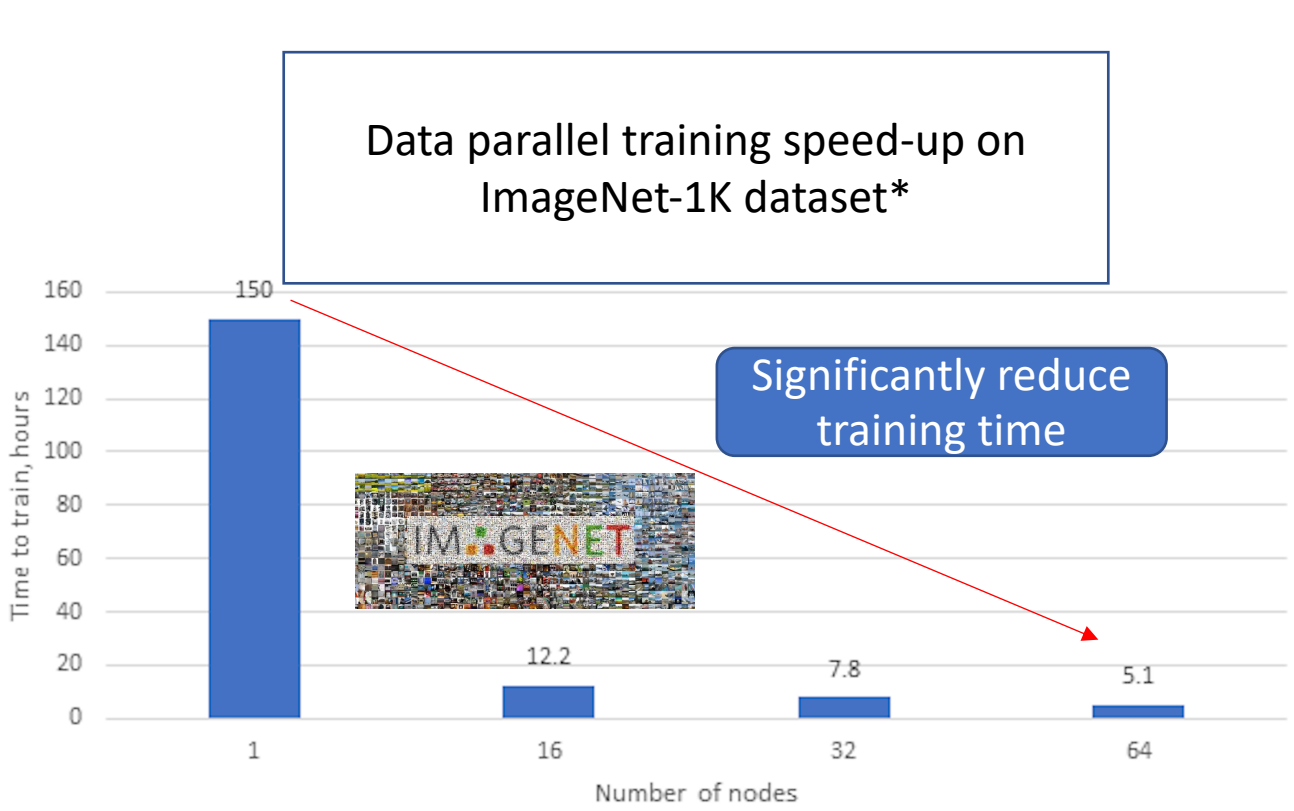
Microsoft Research

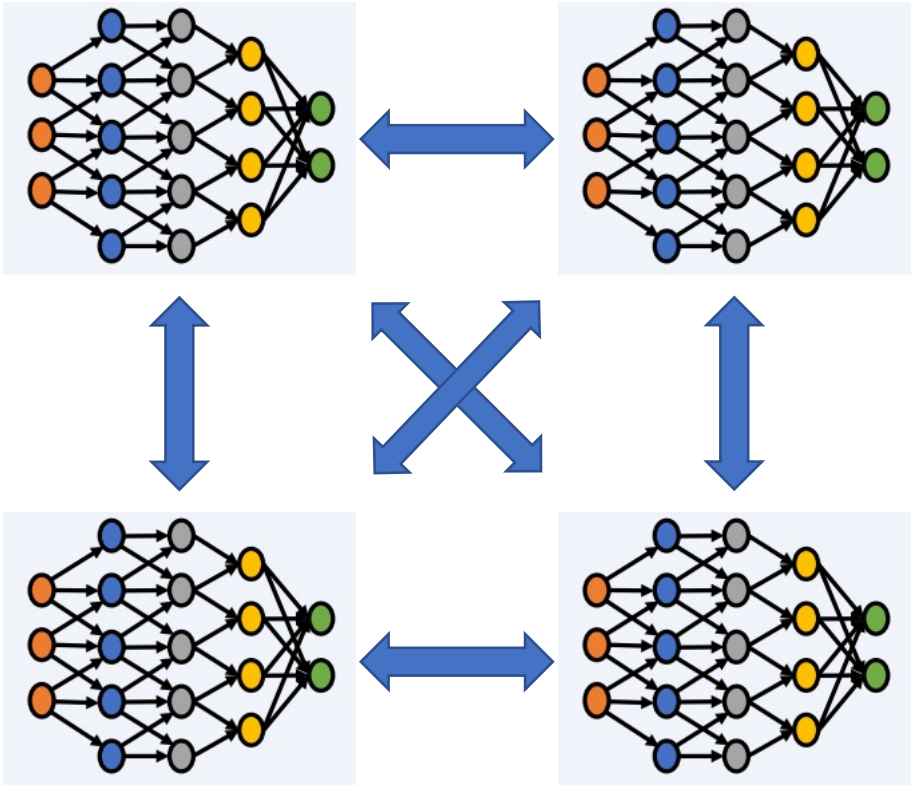
Speed-up DNN training: Data Parallelism



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

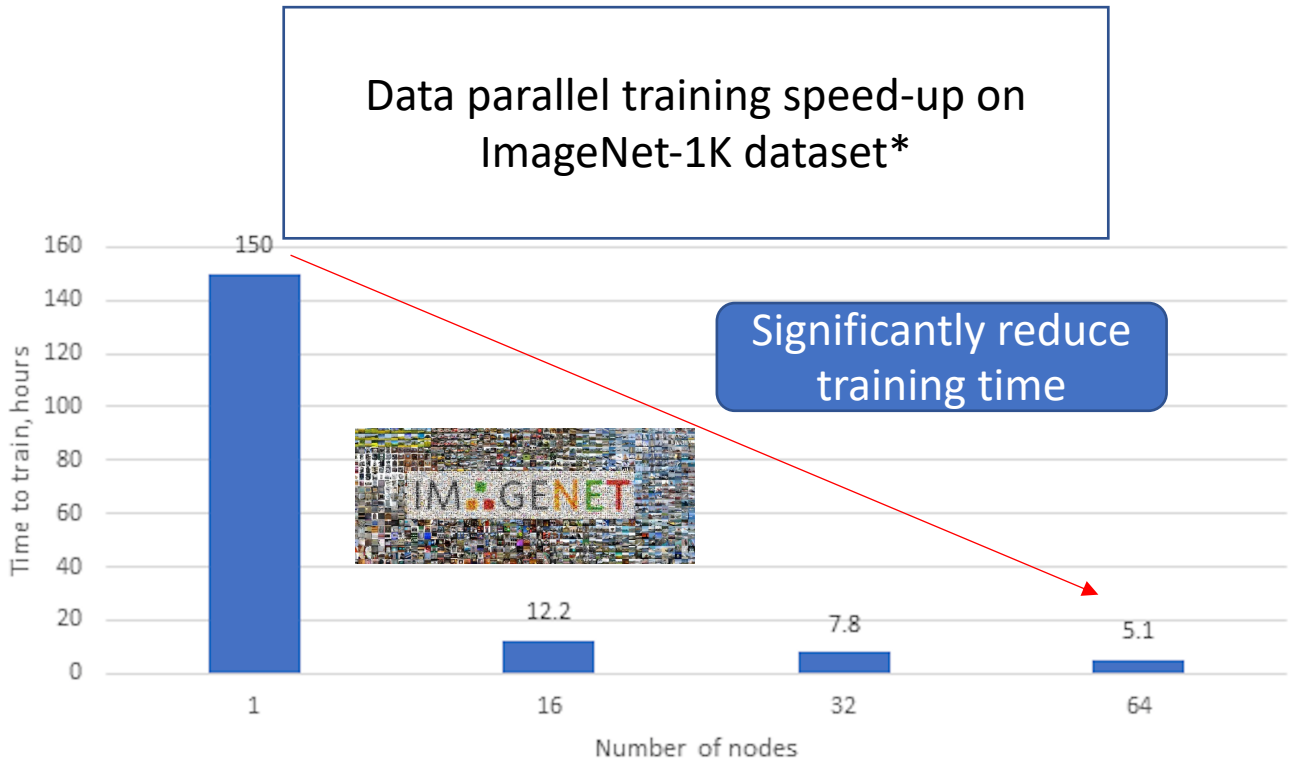
Speed-up DNN training: Data Parallelism

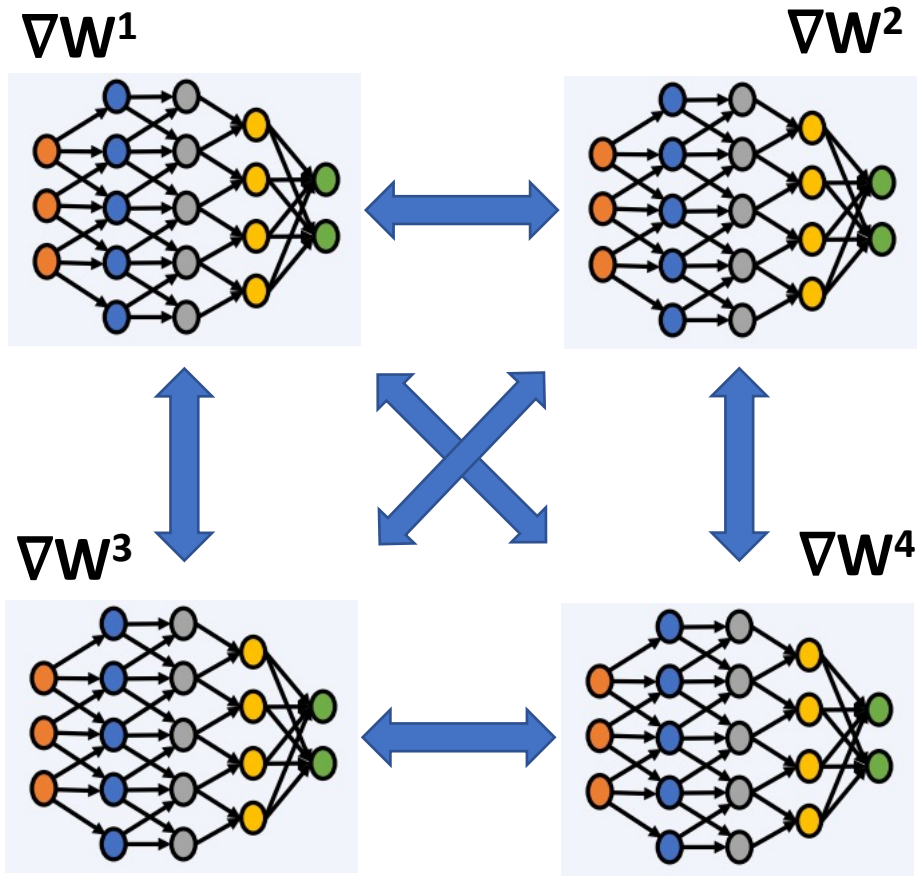




* https://software.intel.com/en-us/articles/caffe-training-on-multi-node-distributed-memory-systems

Speed-up DNN training: Data Parallelism



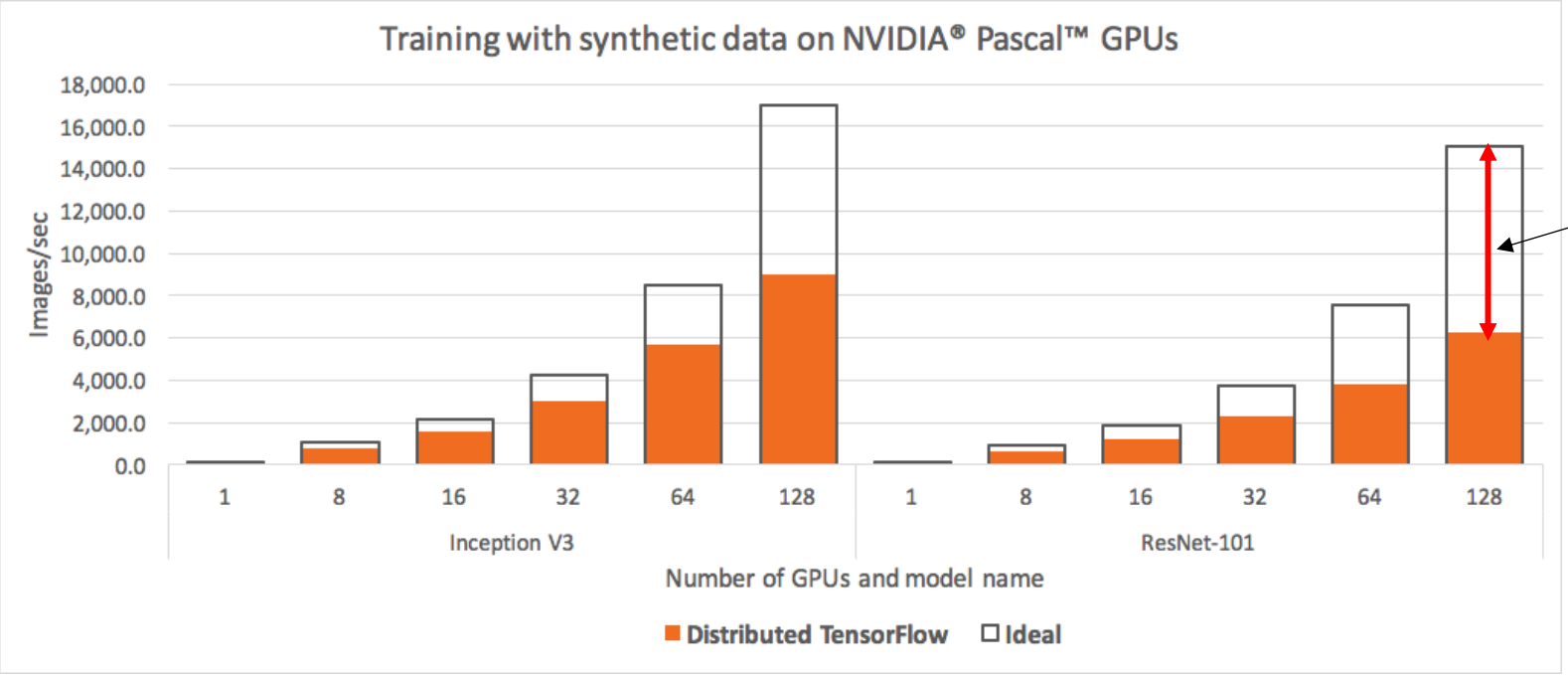


Model Synchronization

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

* 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

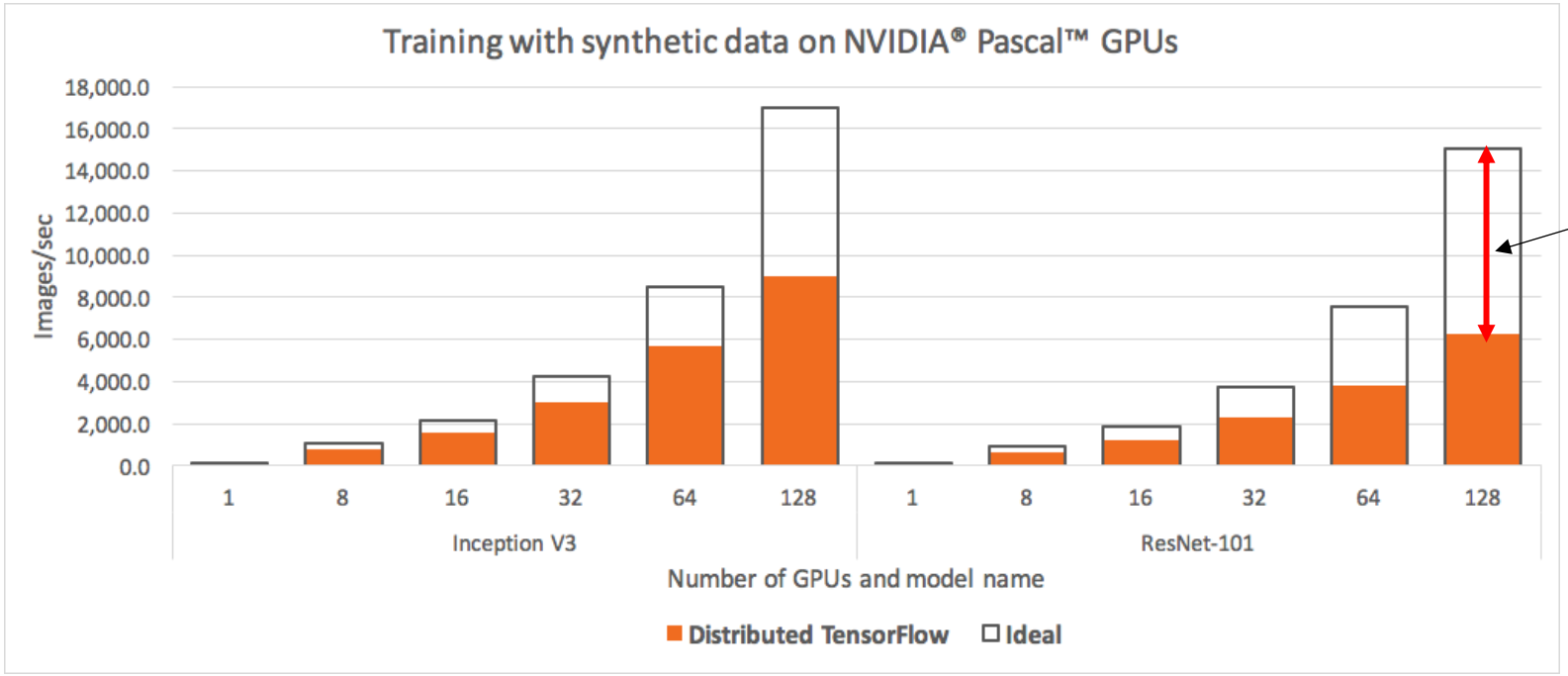


>50% communication overhead

Multi-GPU scaling performance using TensorFlow*

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

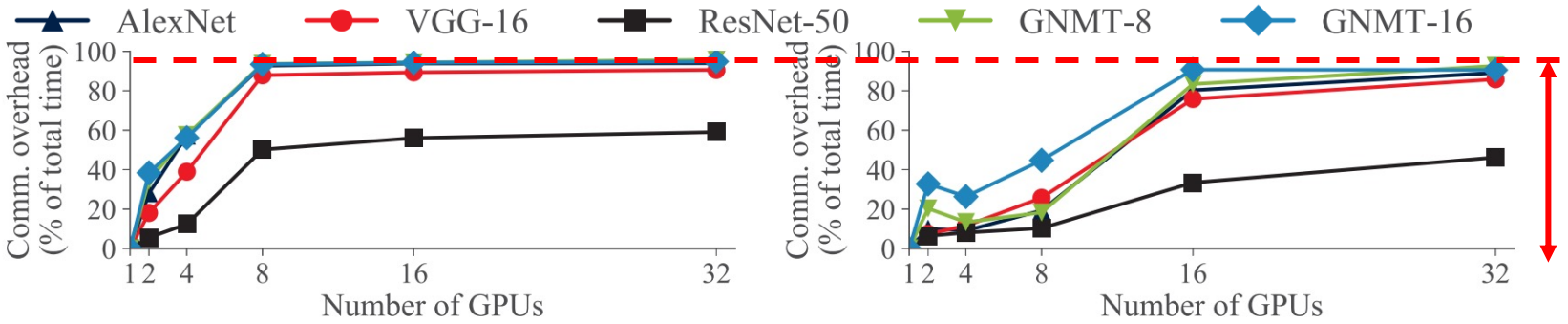
Despite many performance optimizations, model synchronization is a big overhead in data parallel training on cloud servers



>50% communication overhead

Multi-GPU scaling performance using TensorFlow*

Up to 90% communication overhead

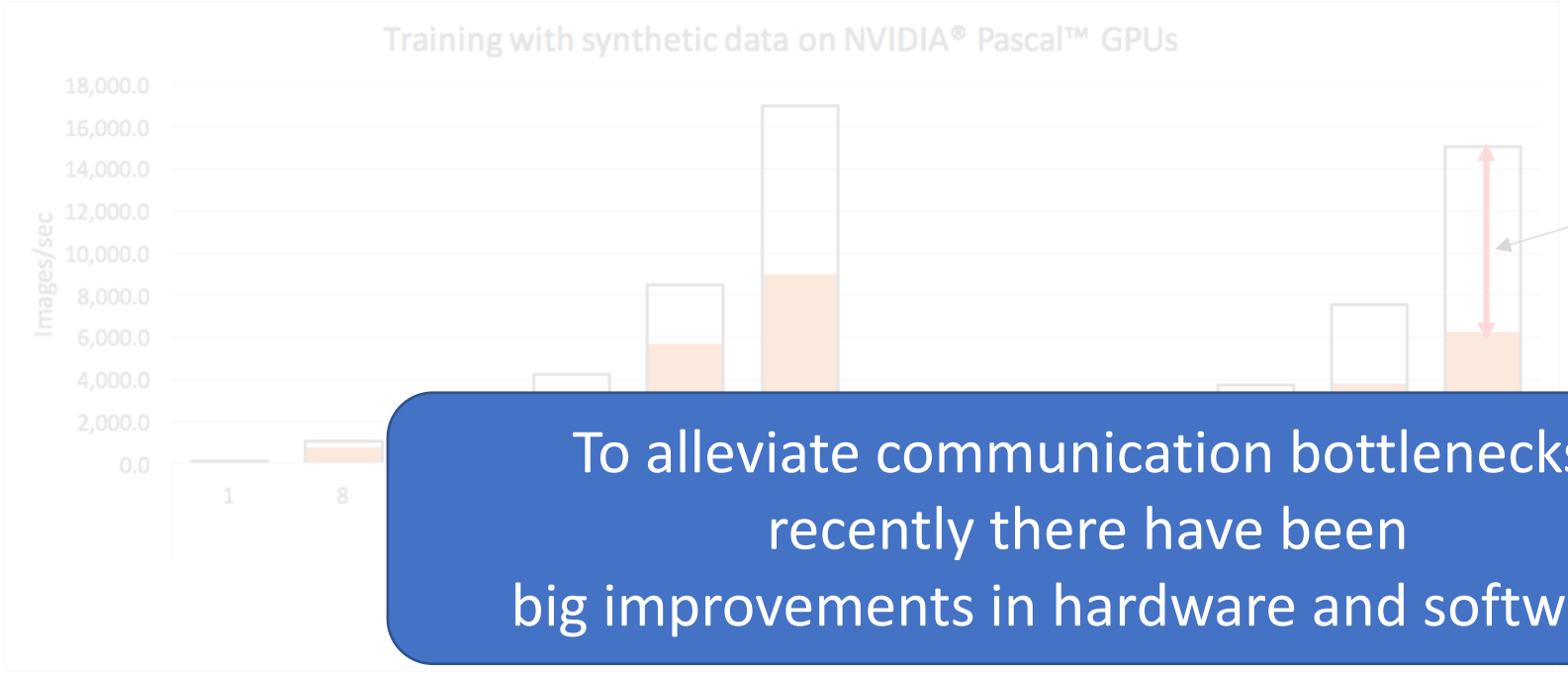


Communication overhead of data-parallel training with Multi-GPU servers using PyTorch^

^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.

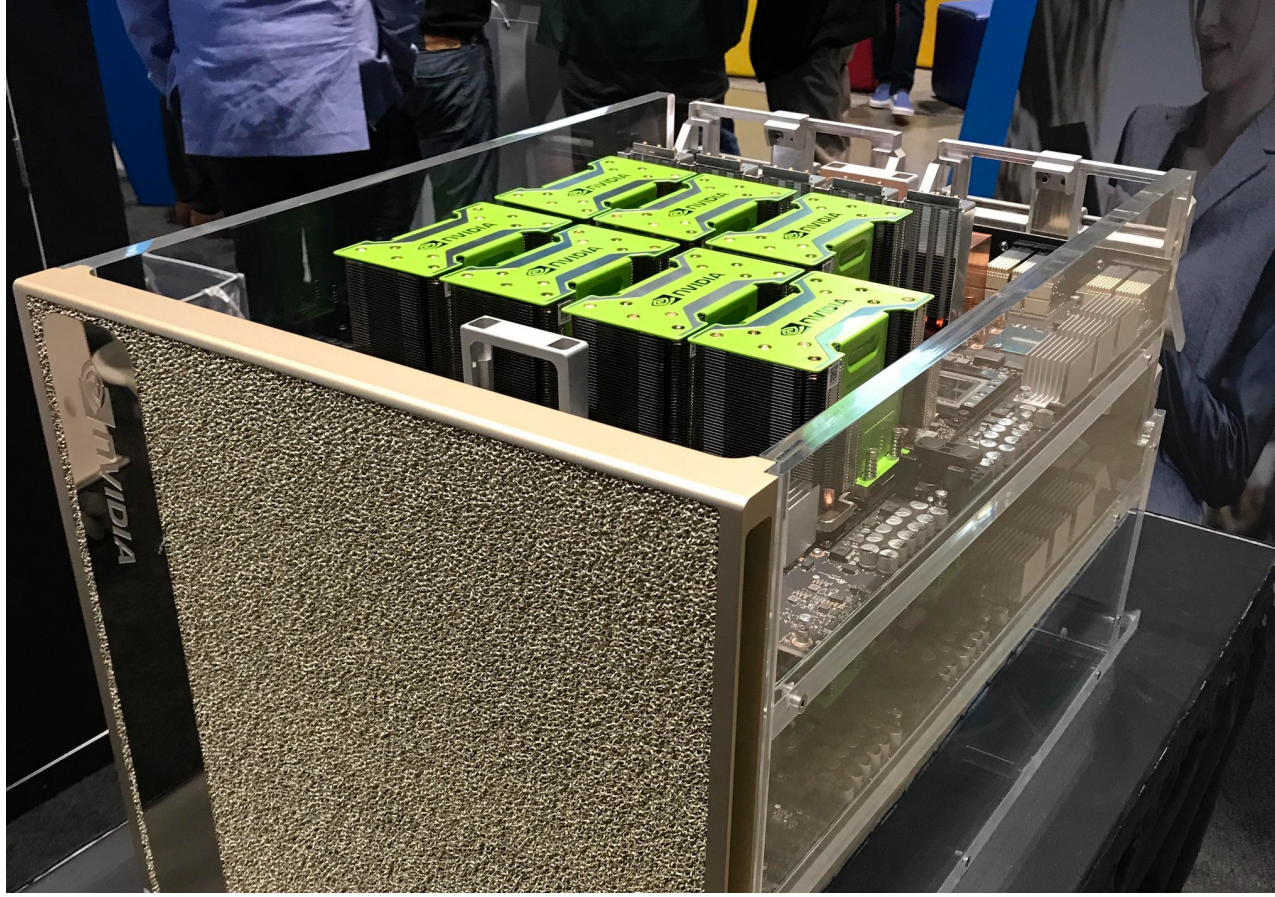
Up to 90% communication overhead

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

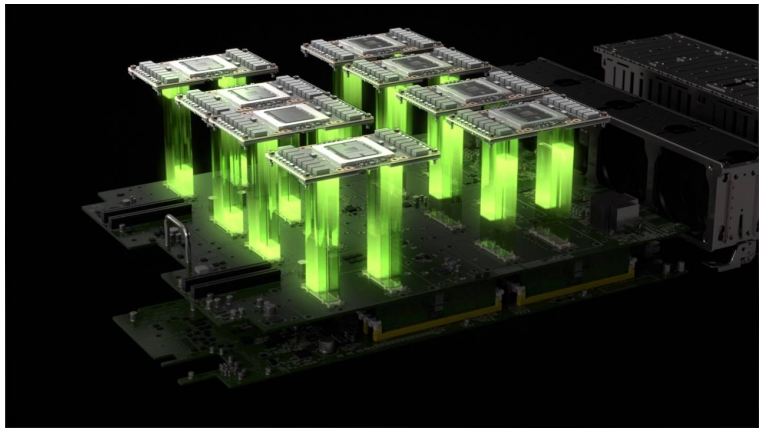
State of the art (hardware)

What is inside?

- Computation

NVIDIA P100: 5.3 Tera-FLOPs
Double Precision

NVIDIA V100: 7.8 Tera-FLOPs
Double Precision

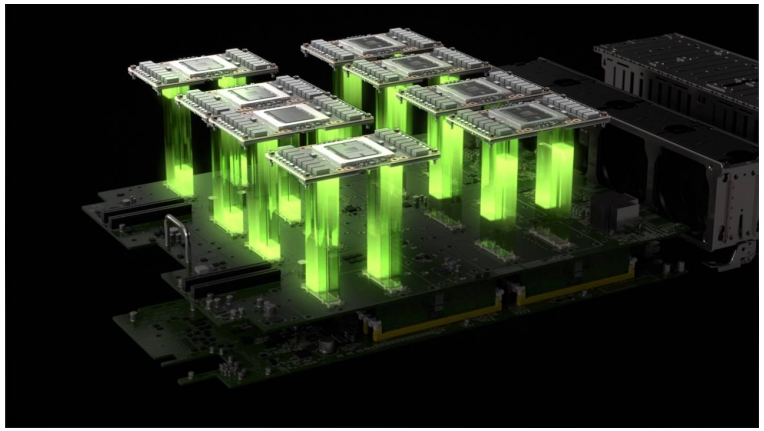


What is inside?

- Computation

NVIDIA P100: 5.3 Tera-FLOPs
Double Precision

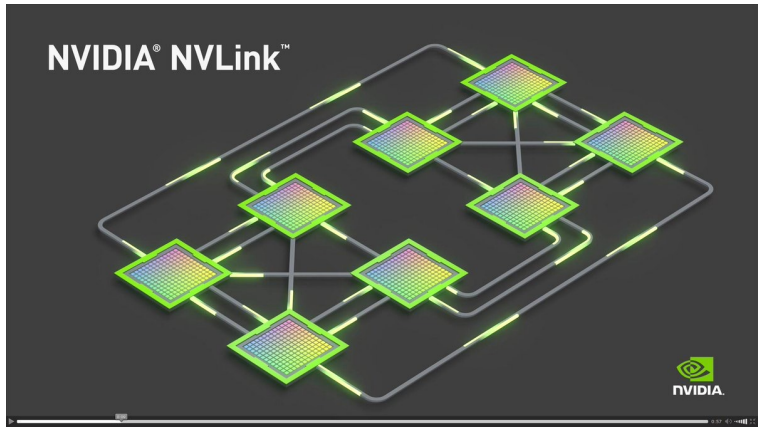
NVIDIA V100: 7.8 Tera-FLOPs
Double Precision



- 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

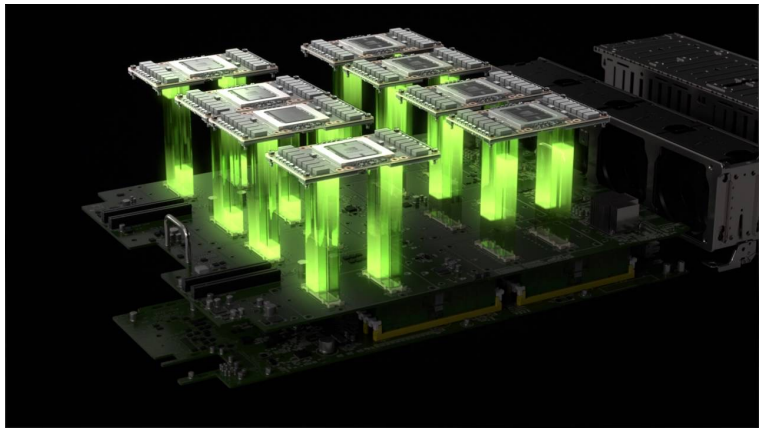


What is inside?

- Computation

NVIDIA P100: 5.3 Tera-FLOPs
Double Precision

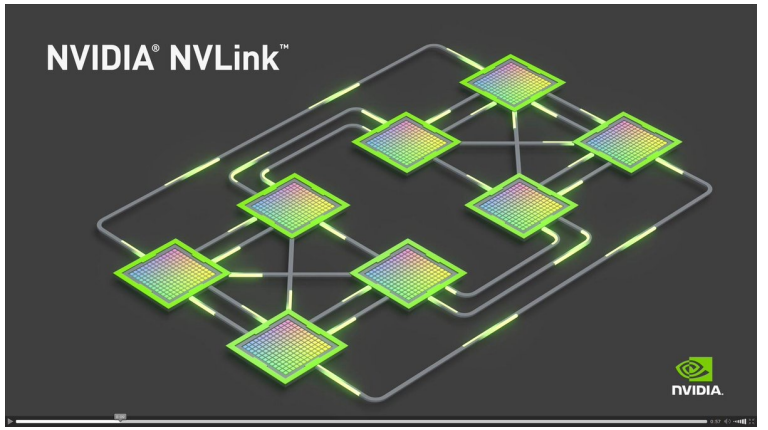
NVIDIA V100: 7.8 Tera-FLOPs
Double Precision



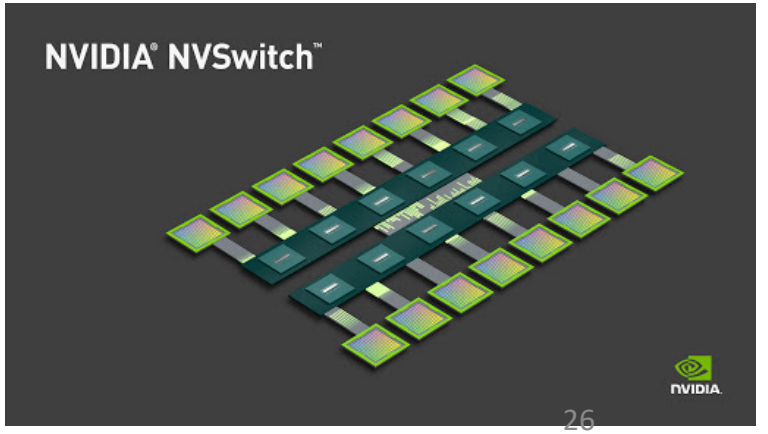
- Faster Interconnects

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

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

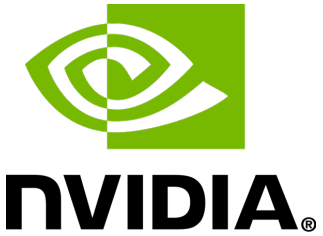


NVSwitch
- Fully connected crossbar
- 6x NVLink 2nd Gen Bandwidth ~**130**GB/s



State of the art (software)

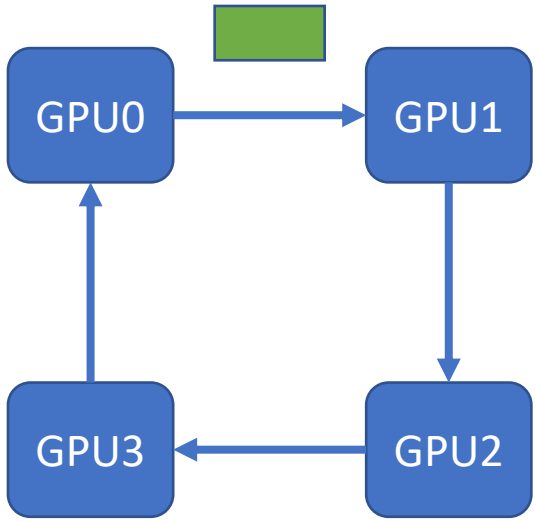
NCCL
(Nvidia Collective Communication Library)



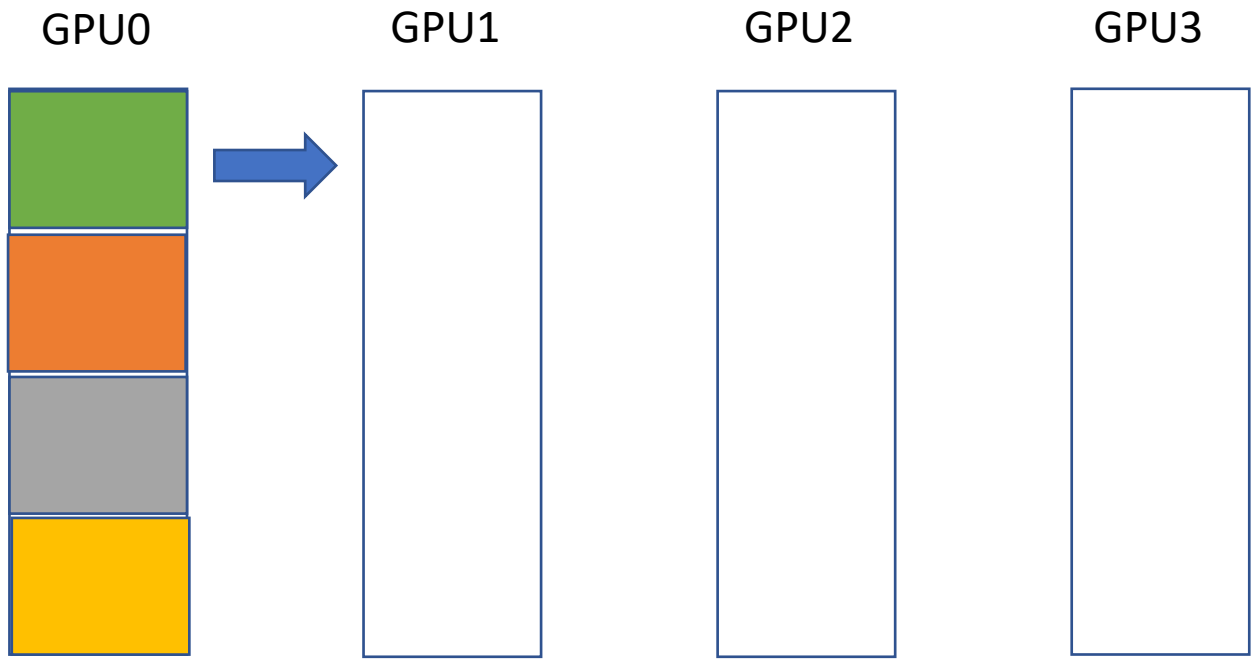
gloo

Ring-based collective communication protocols

Ring-based collectives (e.g. Broadcast)

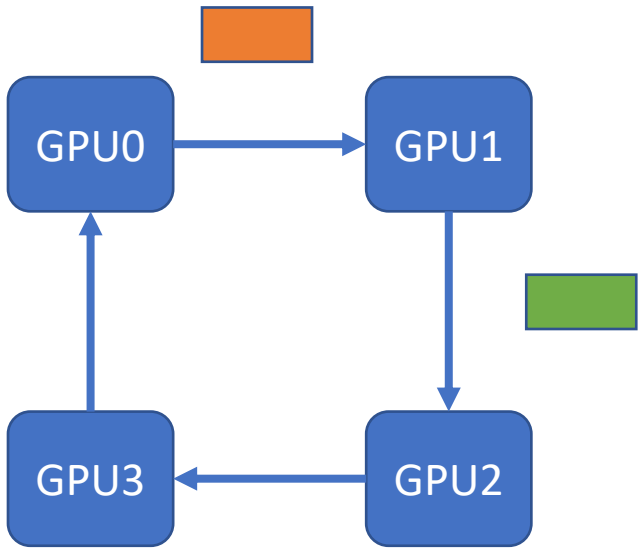


Topology

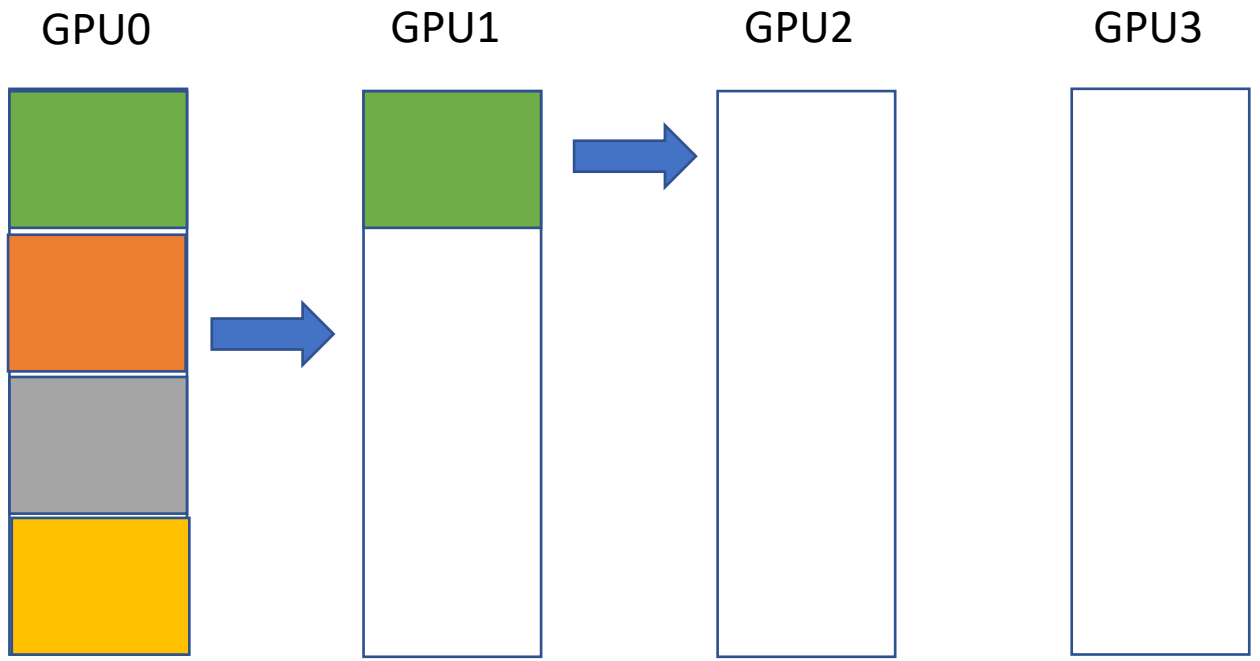


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

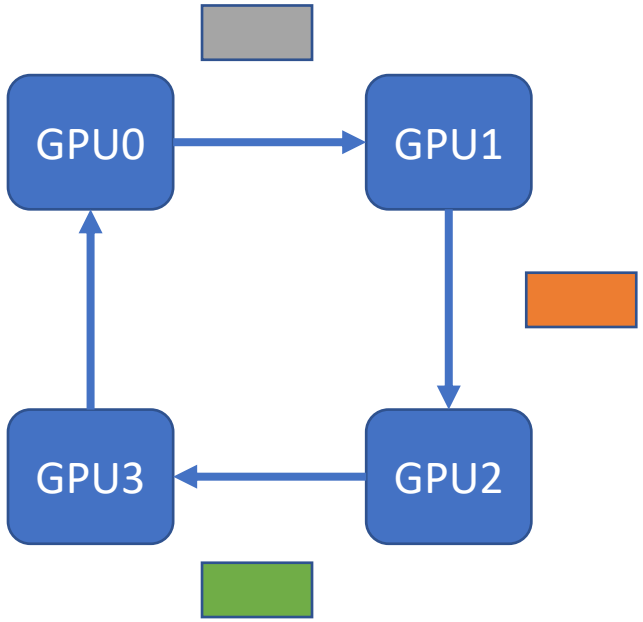


Topology

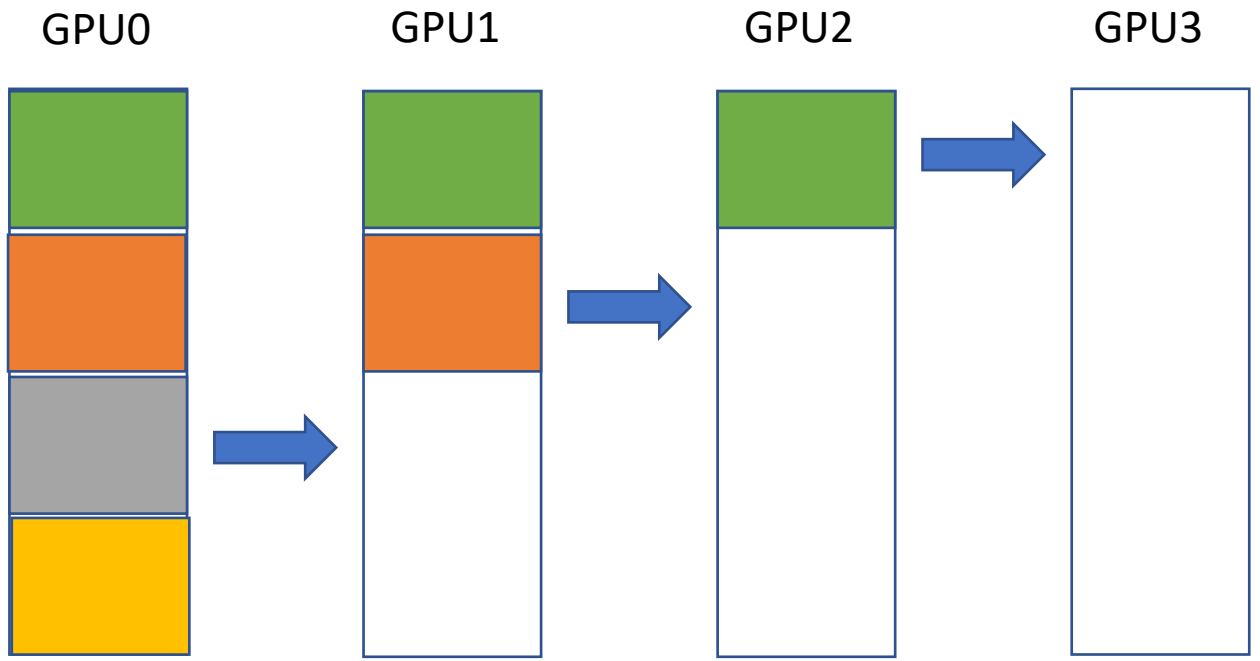


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

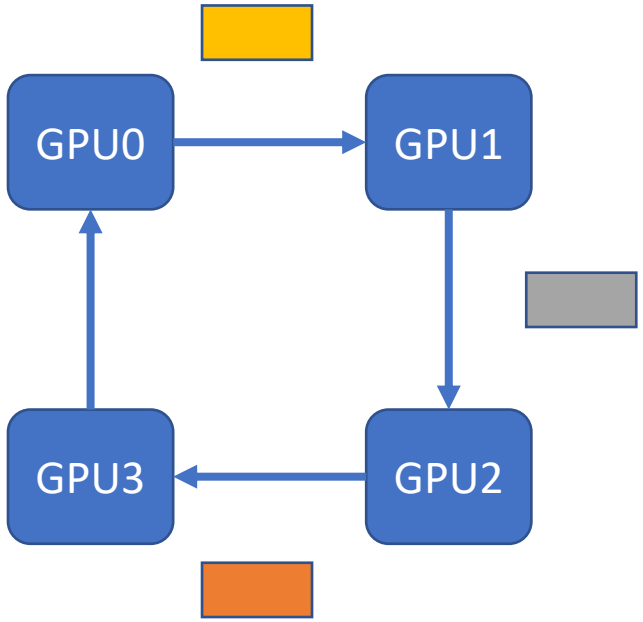


Topology

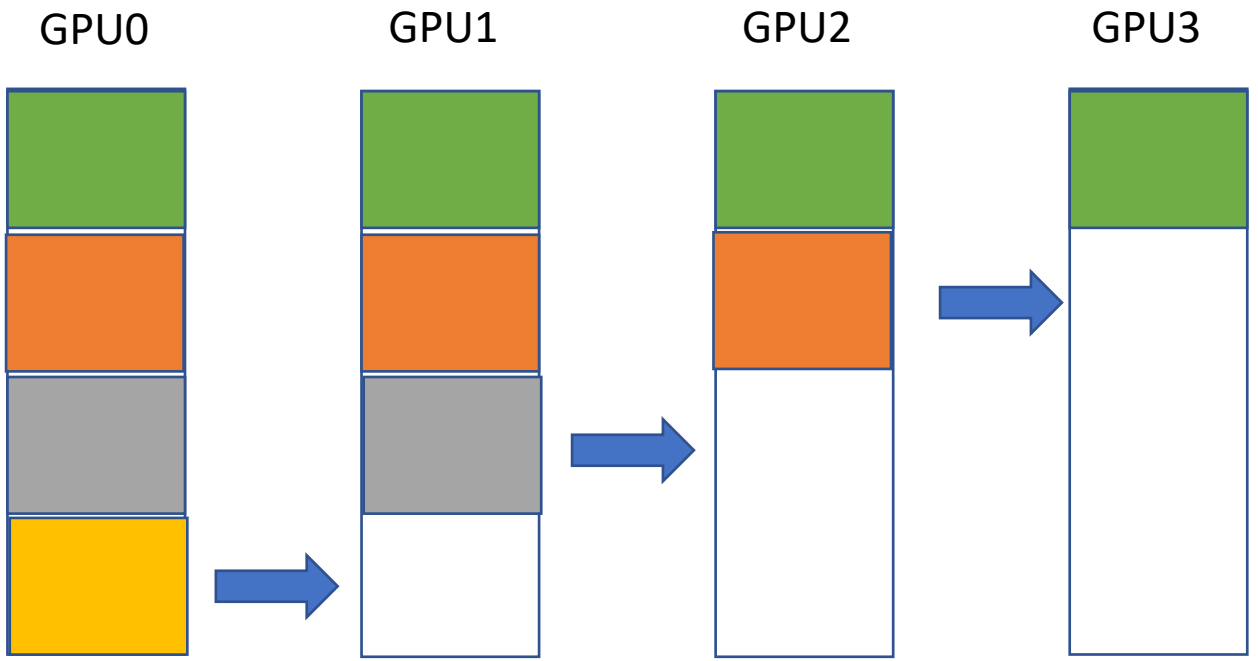


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

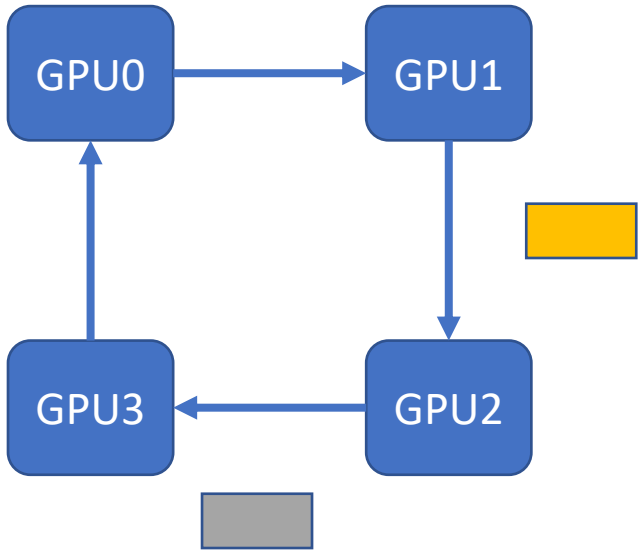


Topology

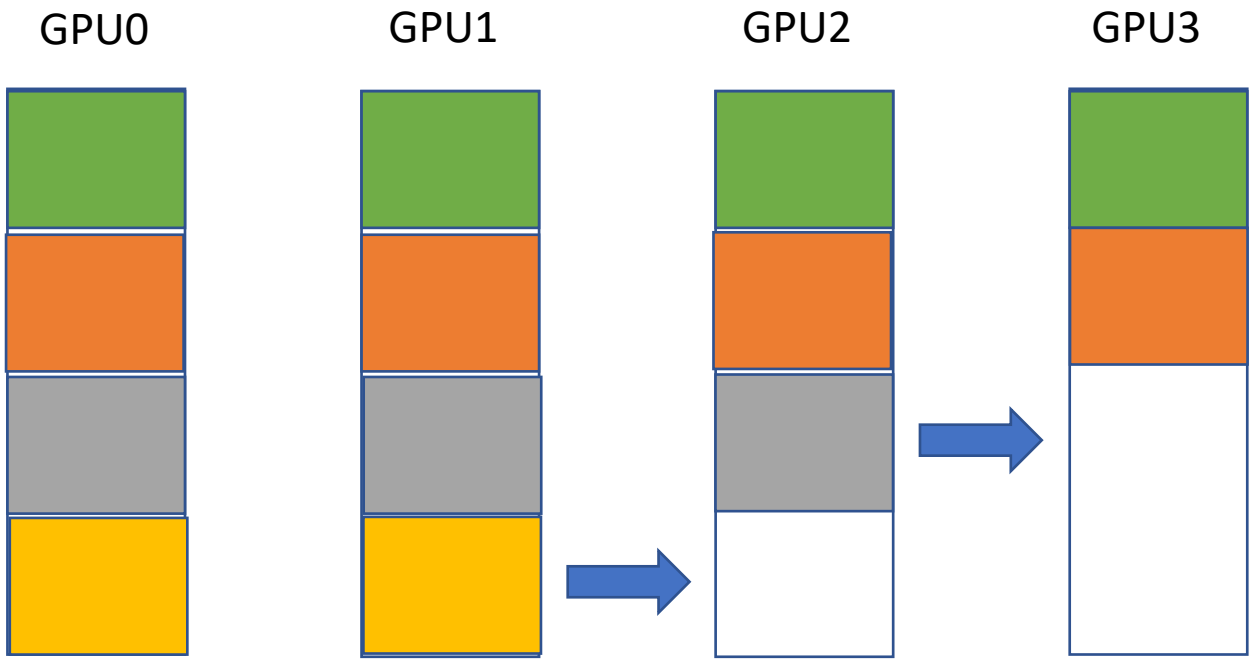


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

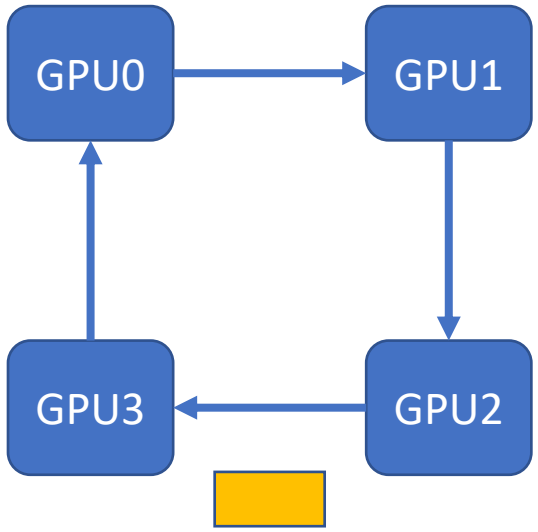


Topology

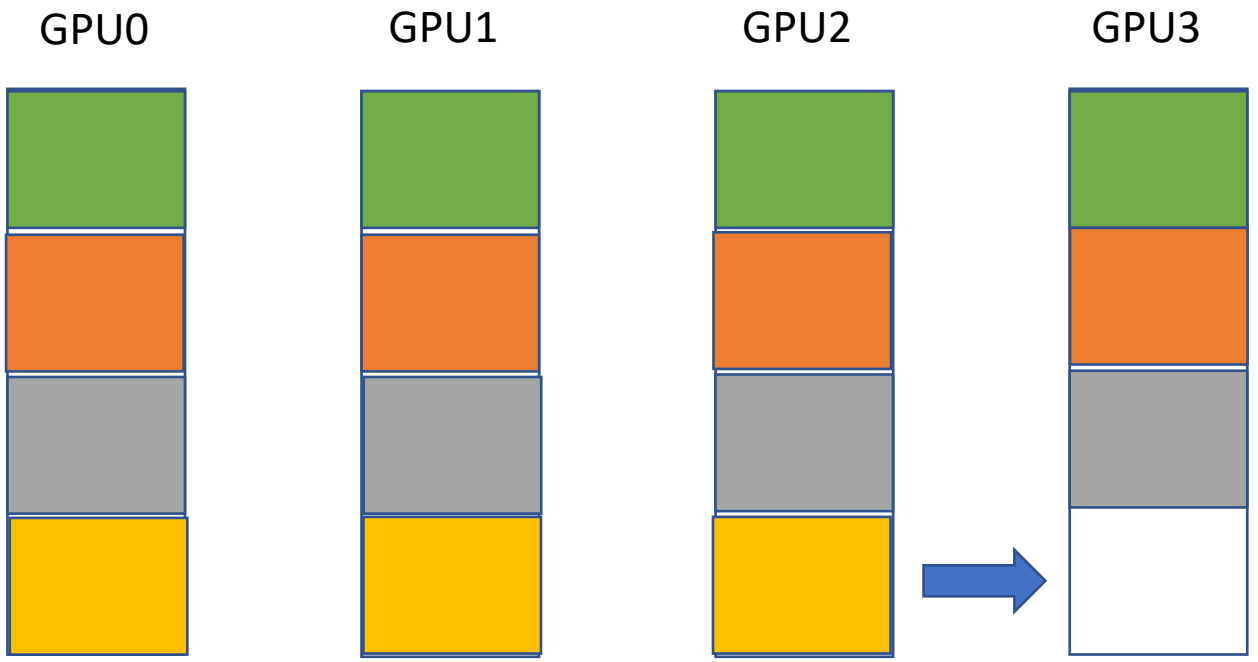


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)

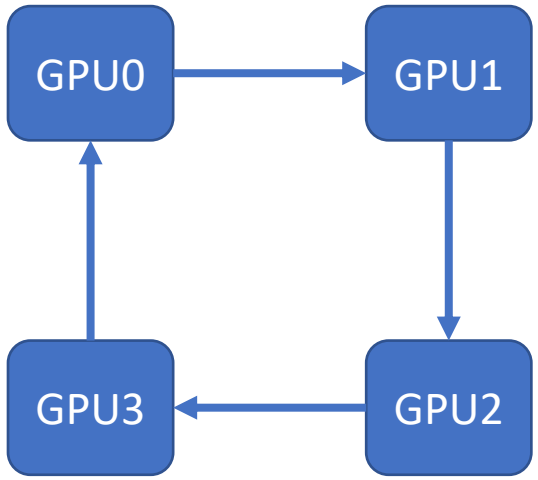


Topology

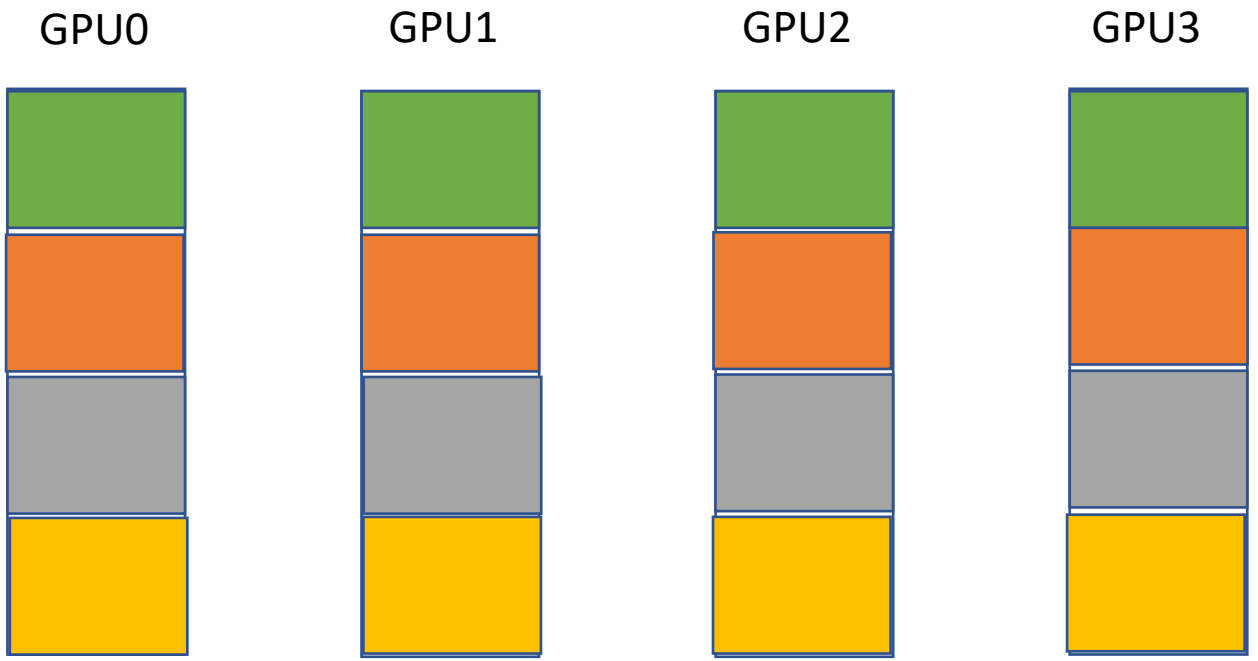


Ring Broadcast (from GPU0)

Ring-based collectives (e.g. Broadcast)



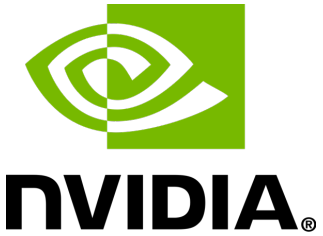
Topology



Ring Broadcast (from GPU0)

State of the art (software)

NCCL
(Nvidia Collective Communication Library)



gloo

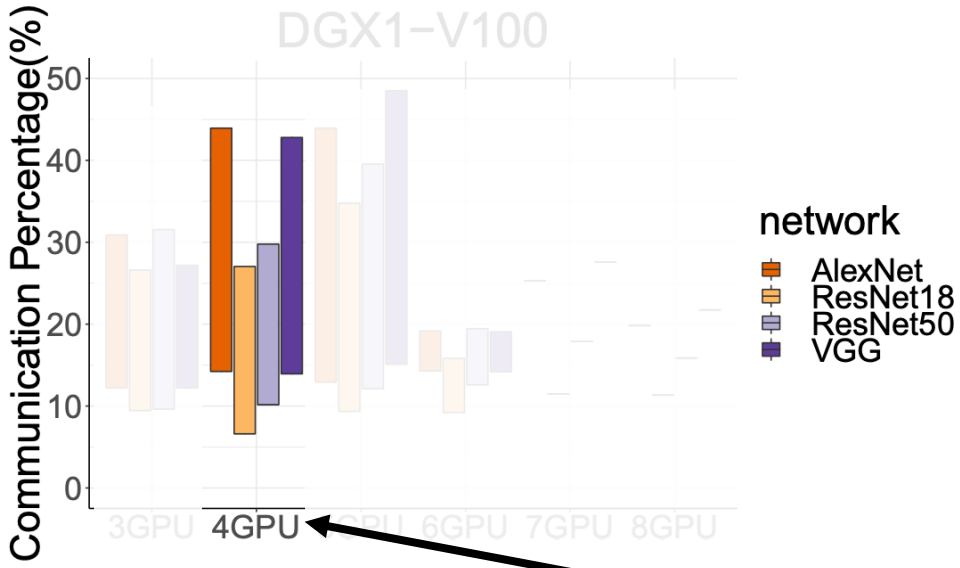
Ring-based collective communication protocols

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

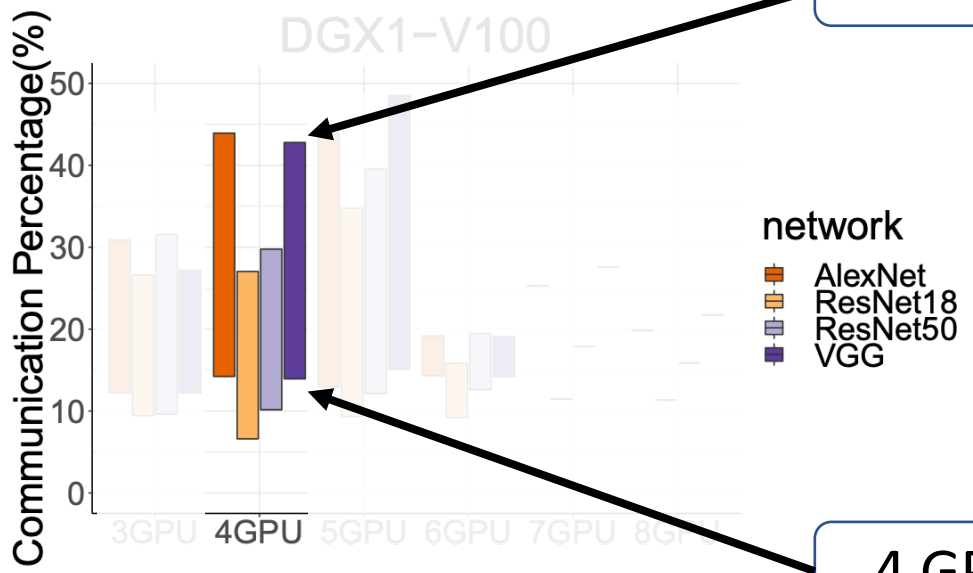
High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)



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)



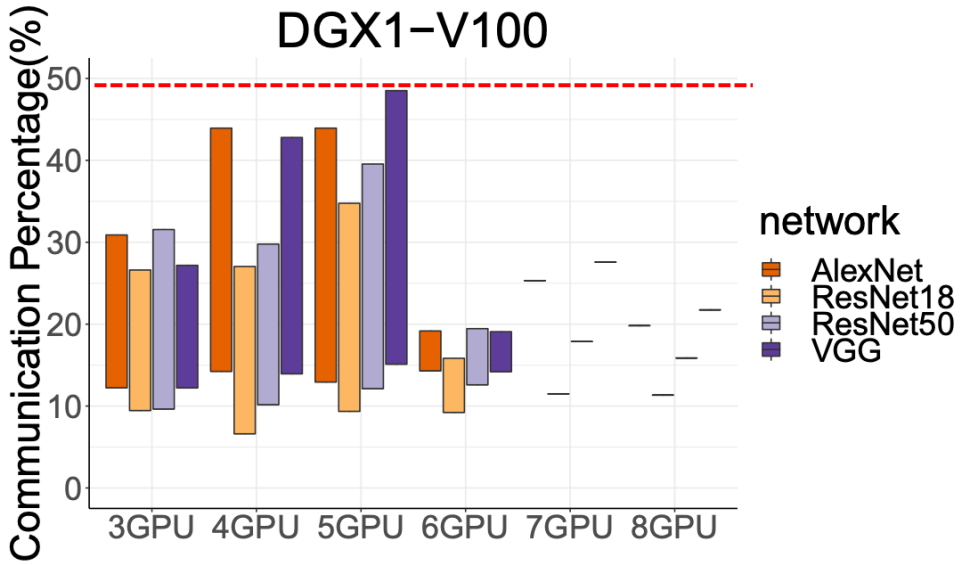
4 GPU allocation with highest overhead

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

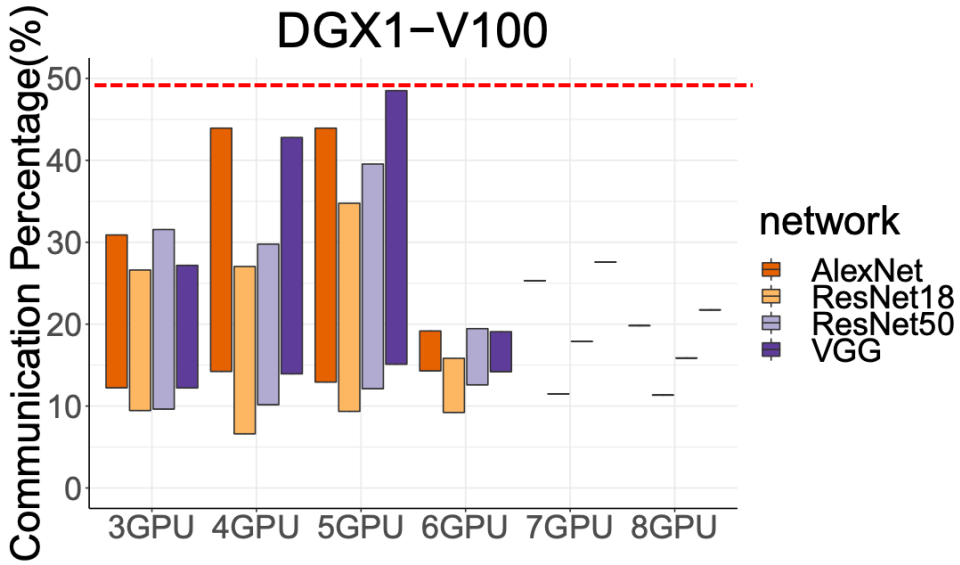
High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)



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

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

Communication overheads become more pronounced with increasing GPU computation power.

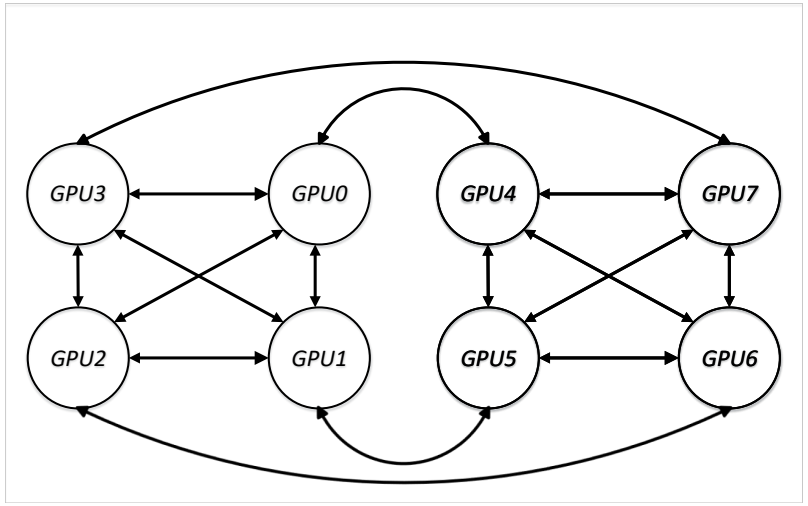
High communication overheads even with state-of-the-art hardware (NVLink) and software (NCCL)

We need Faster Collective Communication Protocols.

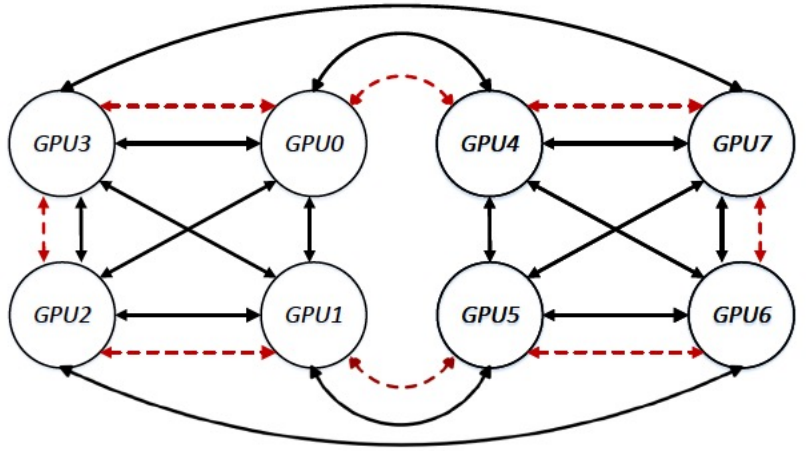
Talk Outline

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

Challenge 1: Different server configurations

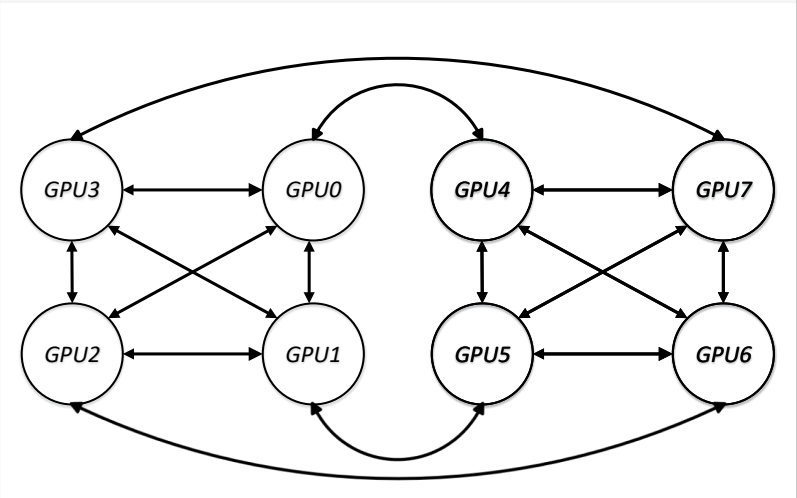


DGX1-P100 (NVLink 1st Gen, ~18GB/s)

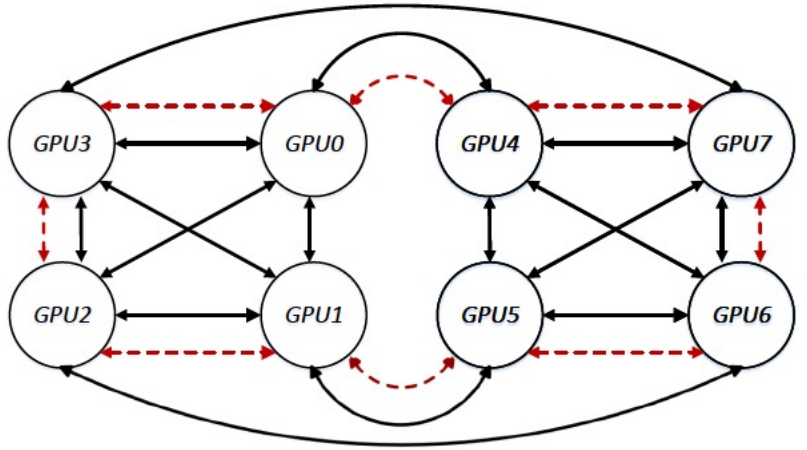


DGX1-V100 (NVLink 2nd Gen, ~23GB/s)

Challenge 1: Different server configurations



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

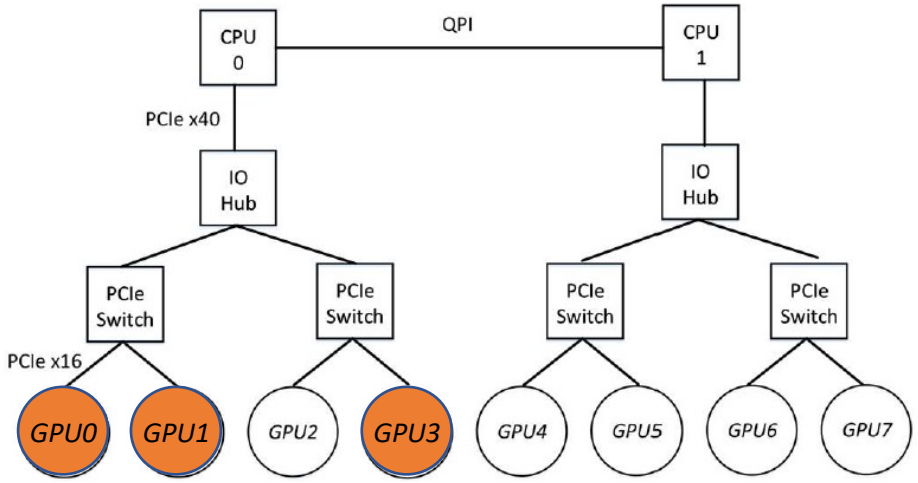


DGX1-V100 (NVLink 2nd Gen, ~23GB/s)

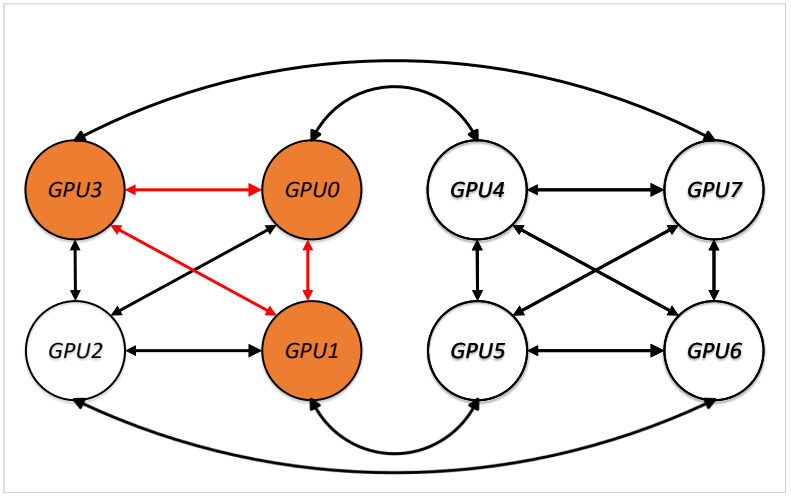
Protocols needs to be topology aware to effectively use hardware links.

Challenge 2: Link heterogeneity

PCIe topology



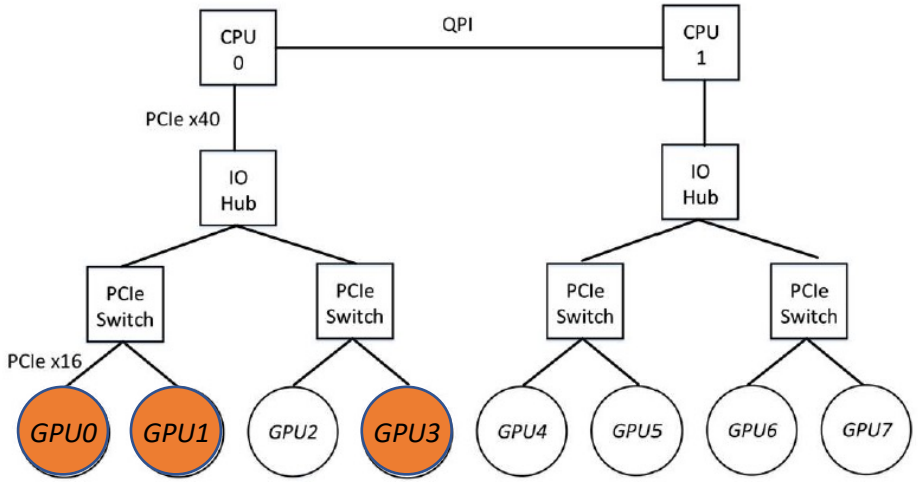
NVLink topology



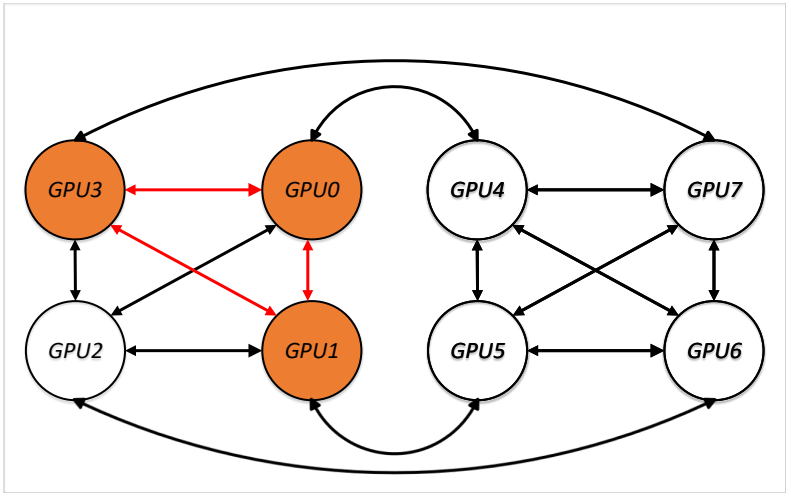
Ring-based collectives can only utilize homogeneous links.

Challenge 2: Link heterogeneity

PCIe topology



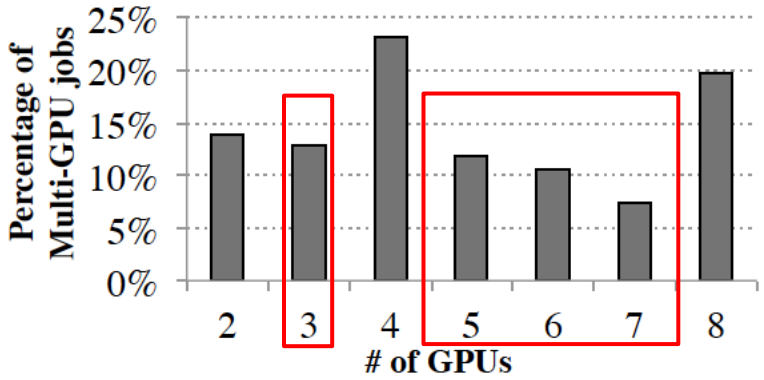
NVLink topology



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



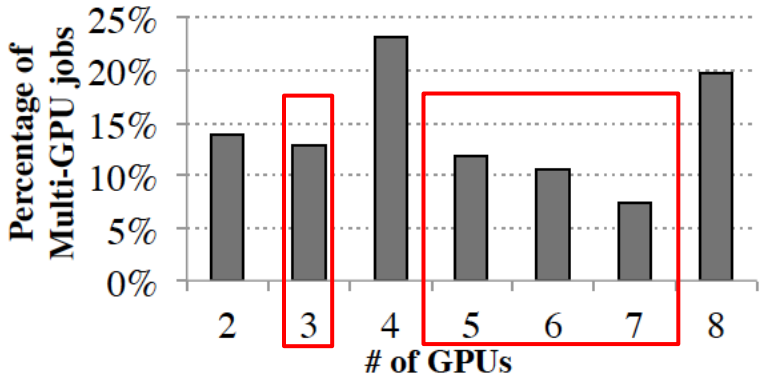
Within each 8-GPU server, # of GPUs allocated to 40,000 multi-GPU jobs at Microsoft.

Examples of fragmented allocation
(8GPU job across 2 servers)

3 + 5

2 + 6

Challenge 3: Fragmentation in multi-tenant clusters



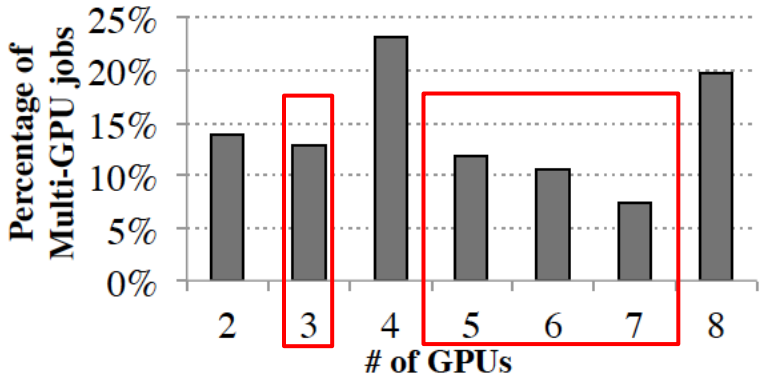
Within each 8-GPU server, # of GPUs allocated to 40,000 multi-GPU jobs at Microsoft.

Why fragmentation?

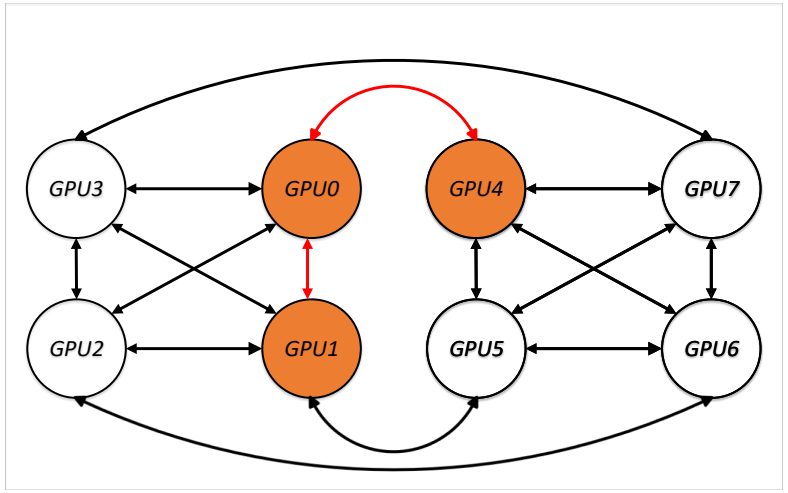
Many cluster schedulers are not topology-aware.

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?

gloo

NVIDIA

Uber

Topology Heterogeneity

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

Ring-based collective communication protocols

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

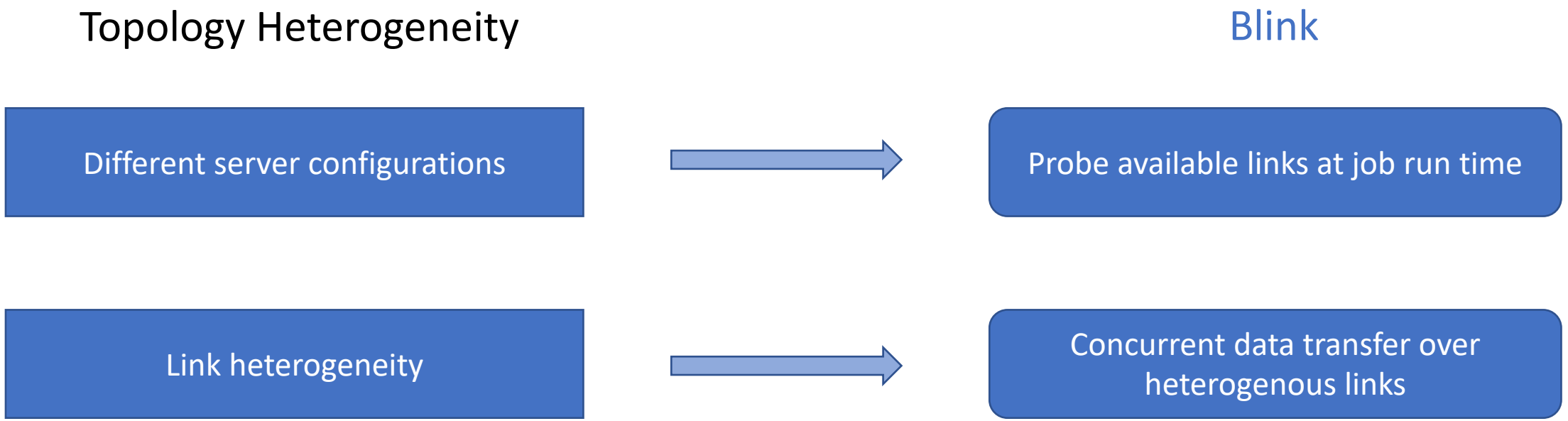
Topology Heterogeneity

Different server configurations

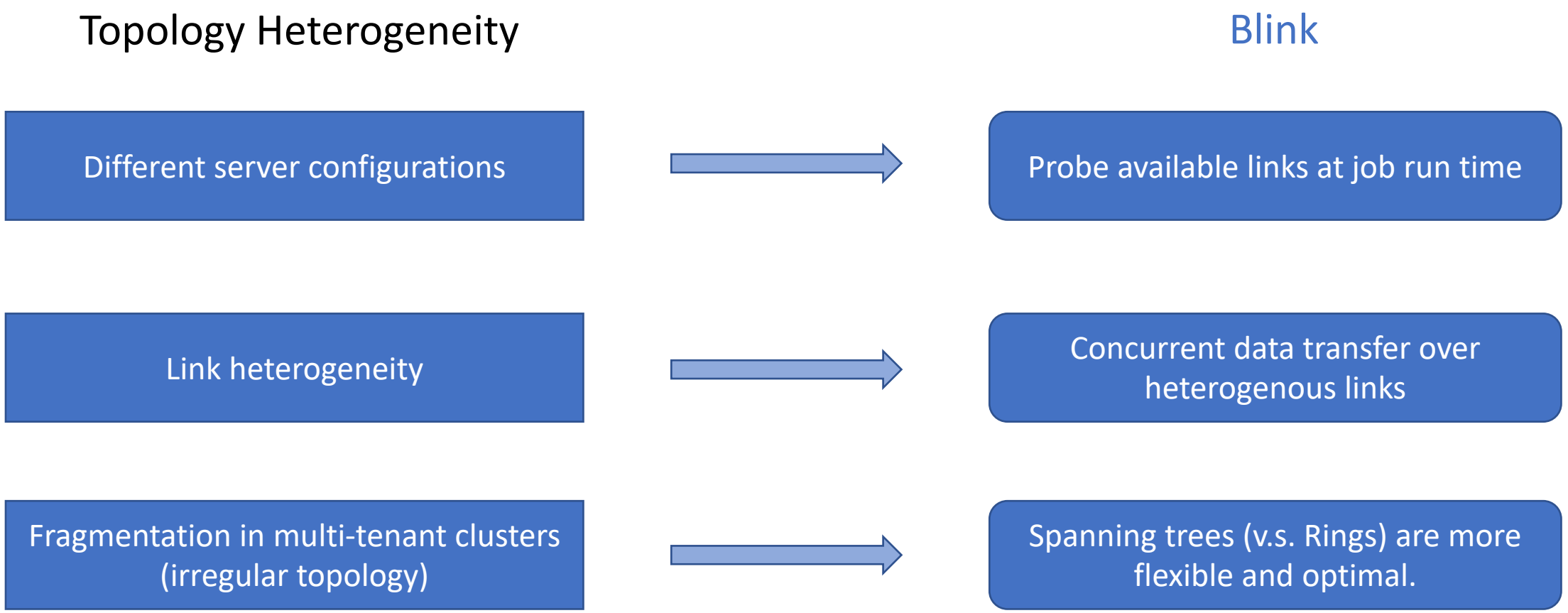
Blink

Probe available links at job run time

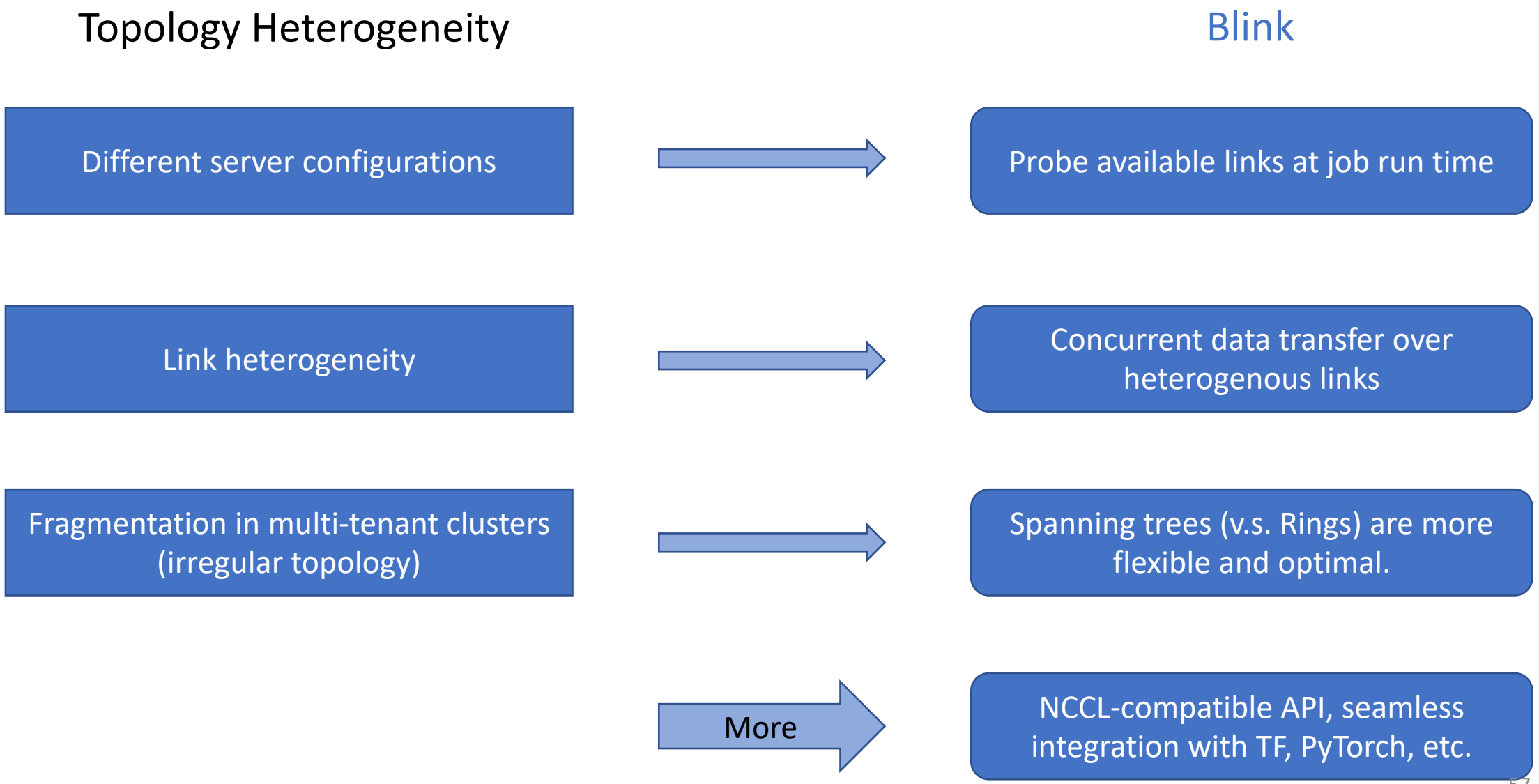
How Blink handles topology heterogeneity



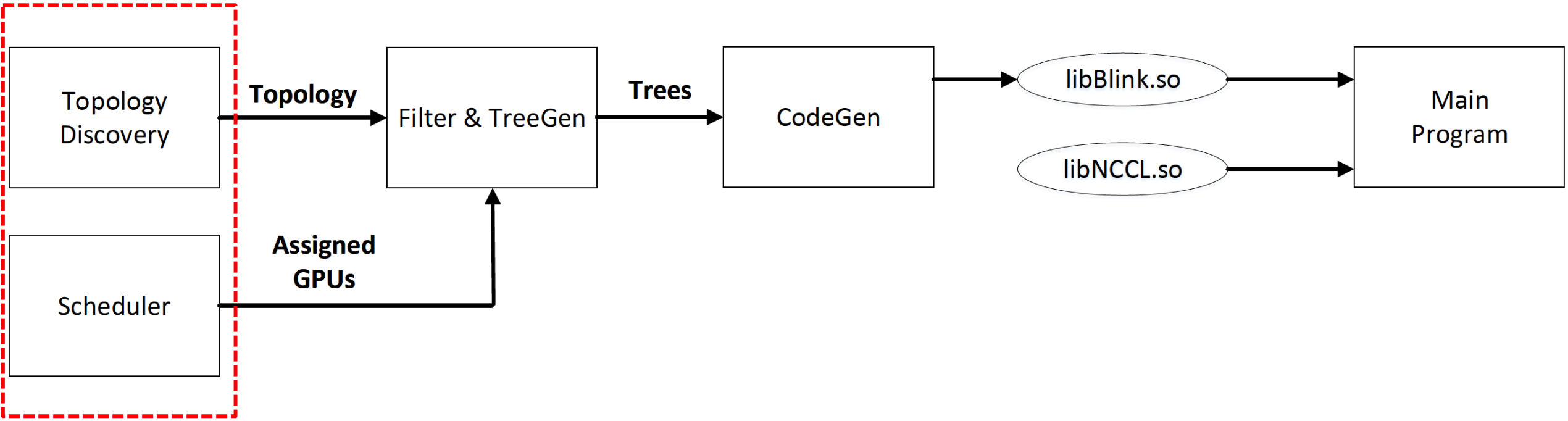
How Blink handles topology heterogeneity



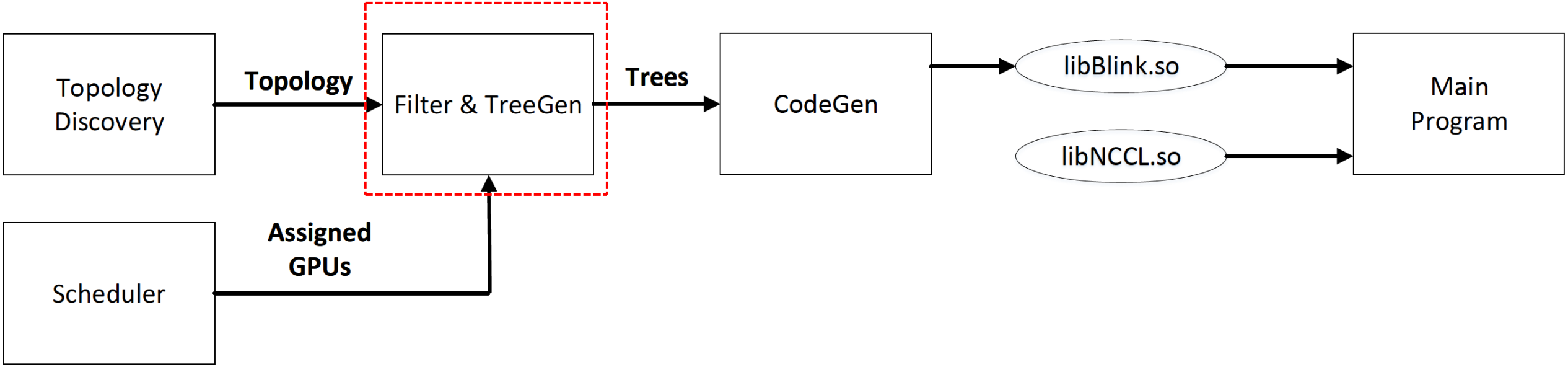
How Blink handles topology heterogeneity



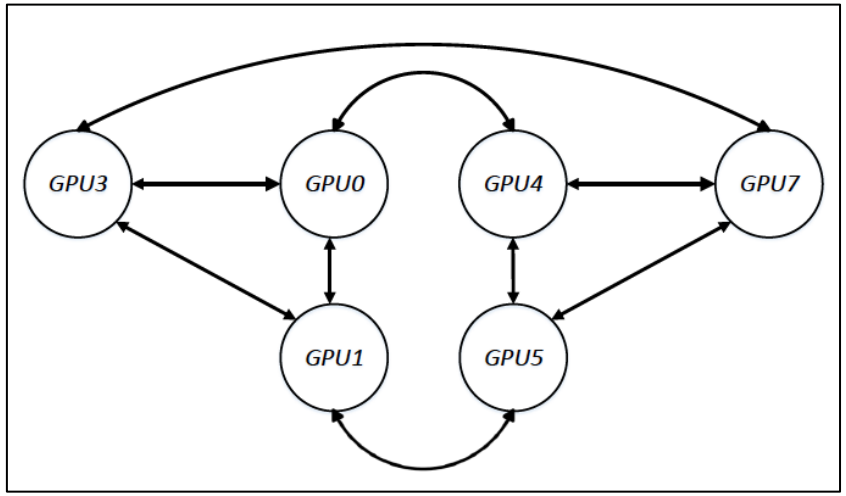
Blink workflow



Blink workflow

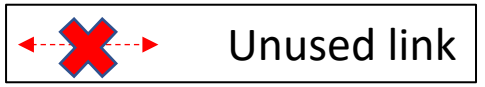


Broadcast comparison (Trees v.s. Rings)

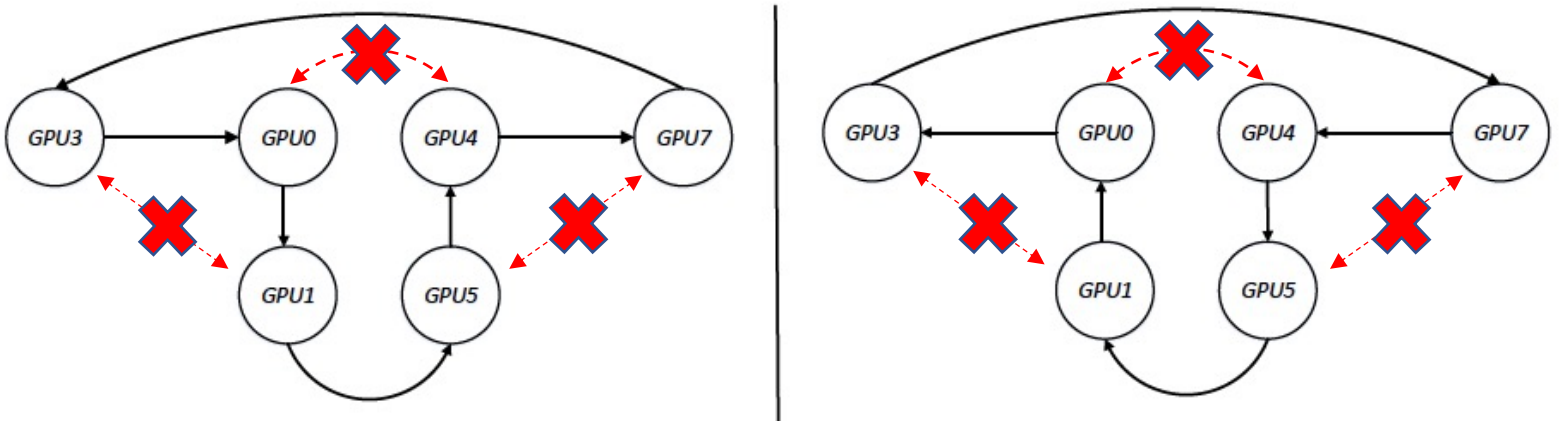


6-GPU topology

Broadcast comparison (Trees v.s. Rings)



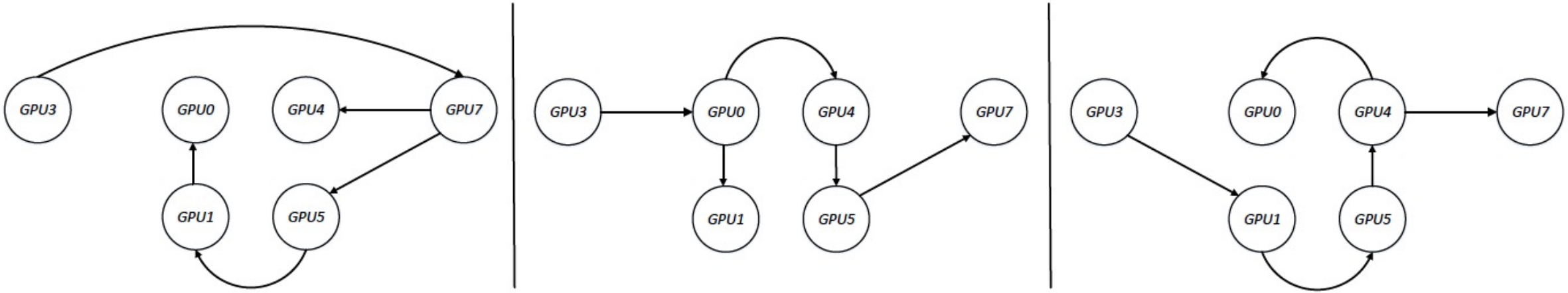
Broadcast from GPU3



NCCL 2 rings

Broadcast comparison (Trees v.s. Rings)

Broadcast from GPU3



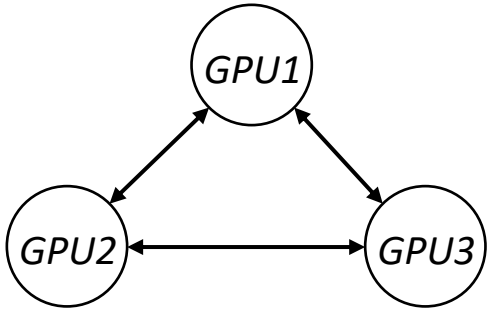
Blink 3 Spanning trees

3 Spanning trees > 2 Rings

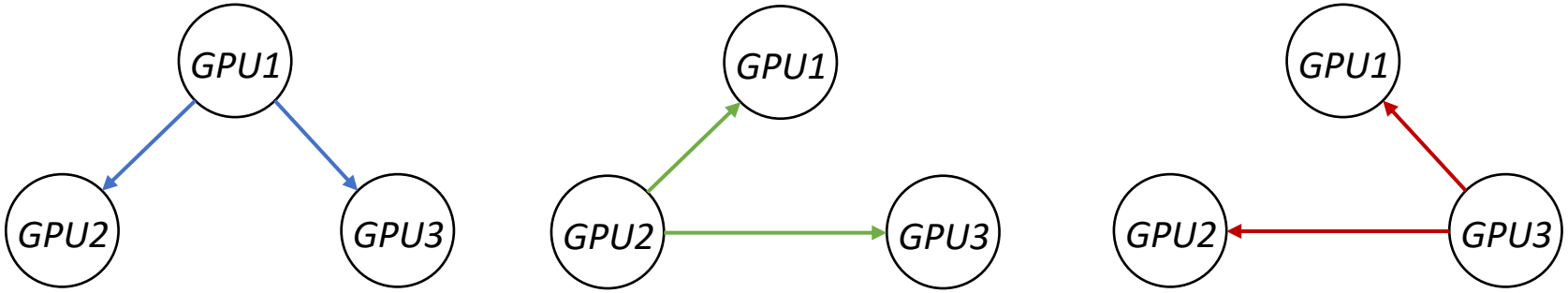
Use **All** the links available → **Optimal**

TreeGen: packing max. spanning trees

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



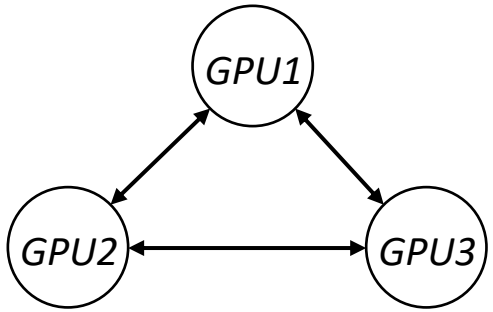
Topology



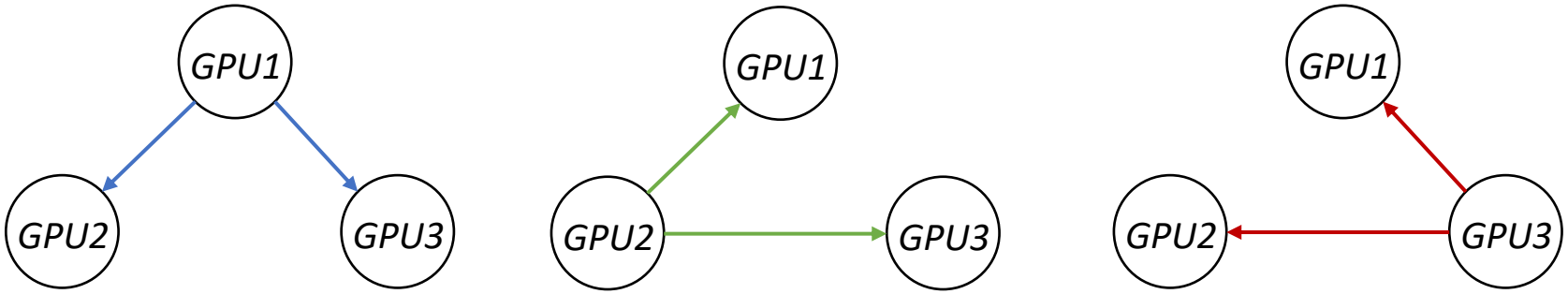
Packing unidirectional spanning trees

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

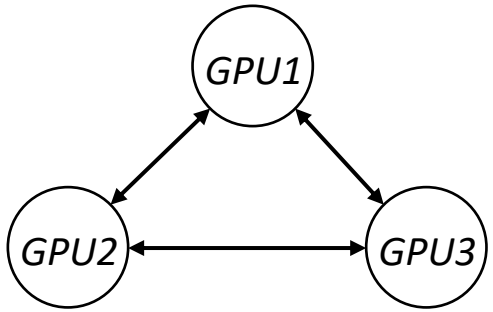
$$\text{such that } \forall e \in E, \sum_i \kappa_i * w_i < c_e$$

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

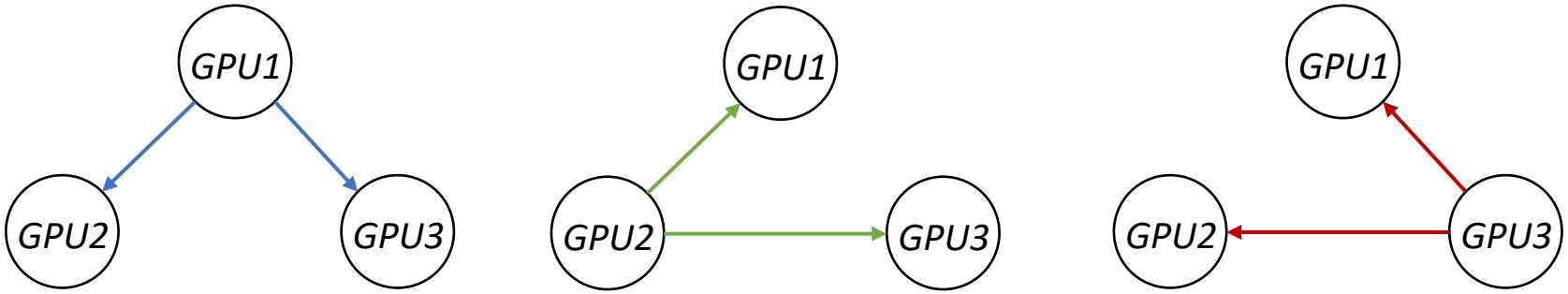
Constrain: amount of BW usage should not exceed ANY link capacity when packing multiple trees

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

$$\max \sum_i w_i$$

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

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

approxi mation

Approximate packing

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

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

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

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

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

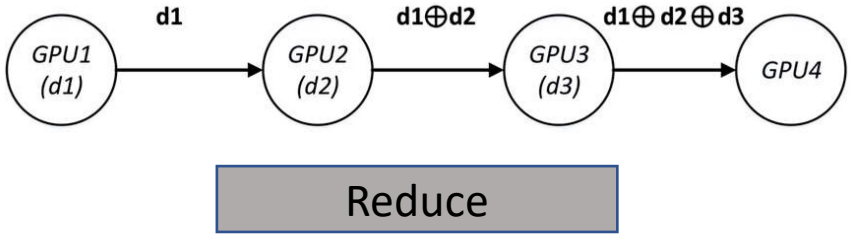
181 trees for 8GPU DGX-1V

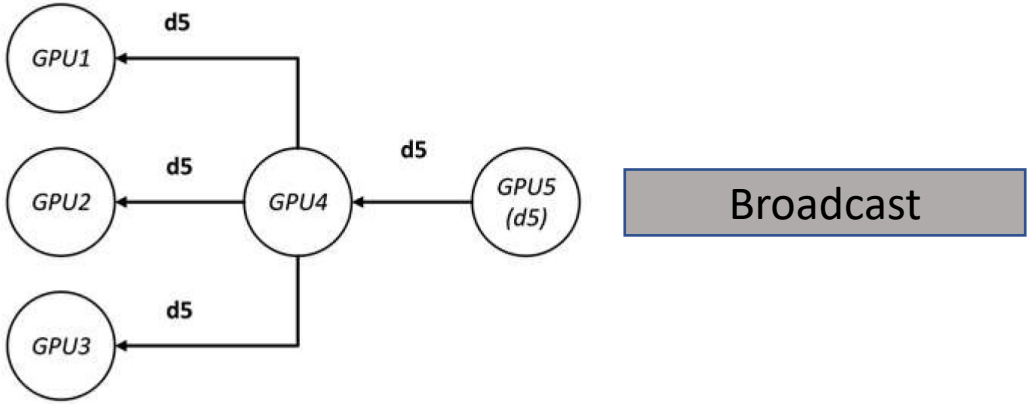
approximation

6 trees for 8GPU DGX-1V

TreeGen

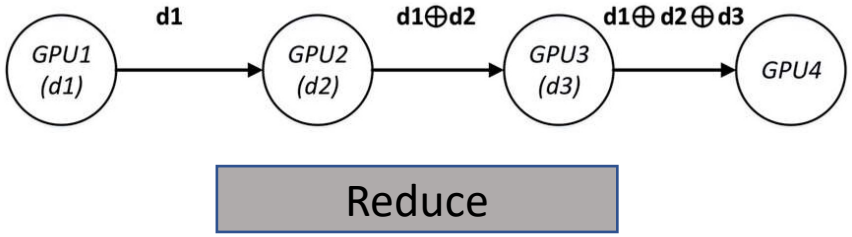
- Given available topology, pack max. unidirectional spanning trees
- Direct support for one-to-many/many-to-one primitives
  - e.g. Reduce, Broadcast, etc.

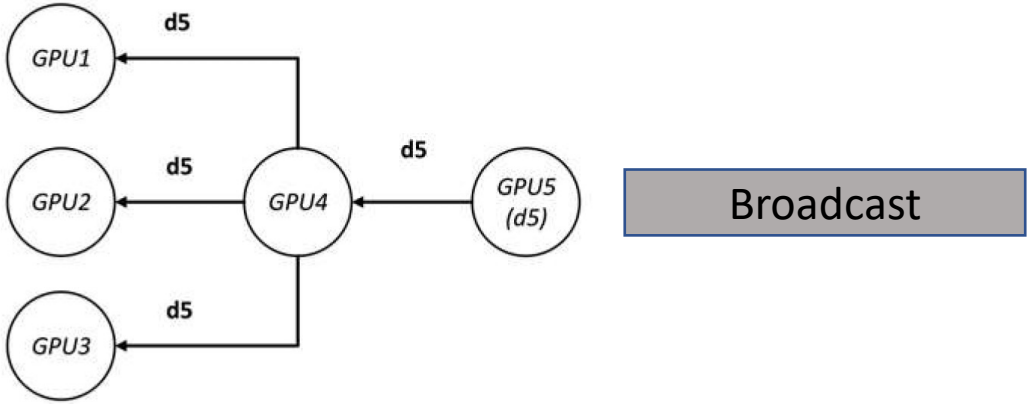




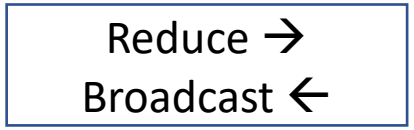
TreeGen

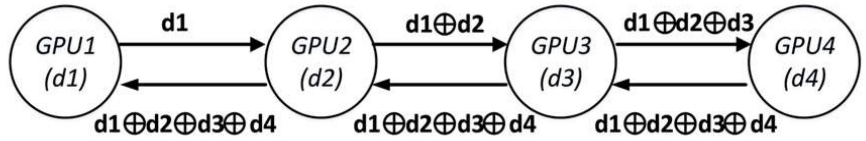
- Given available topology, pack max. unidirectional spanning trees
- Direct support for one-to-many/many-to-one primitives
  - e.g. Reduce, Broadcast, etc.





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

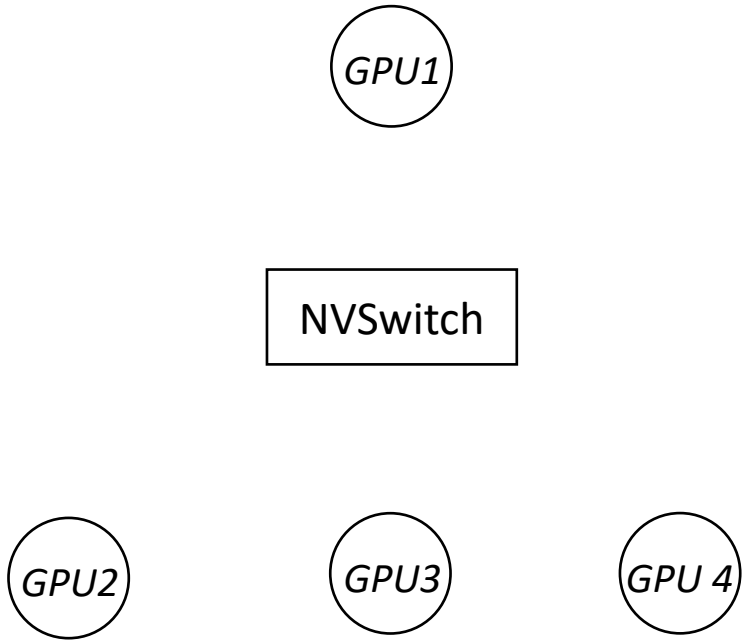




TreeGen for NVSwitch (DGX-2)

TreeGen for NVSwitch (DGX-2)

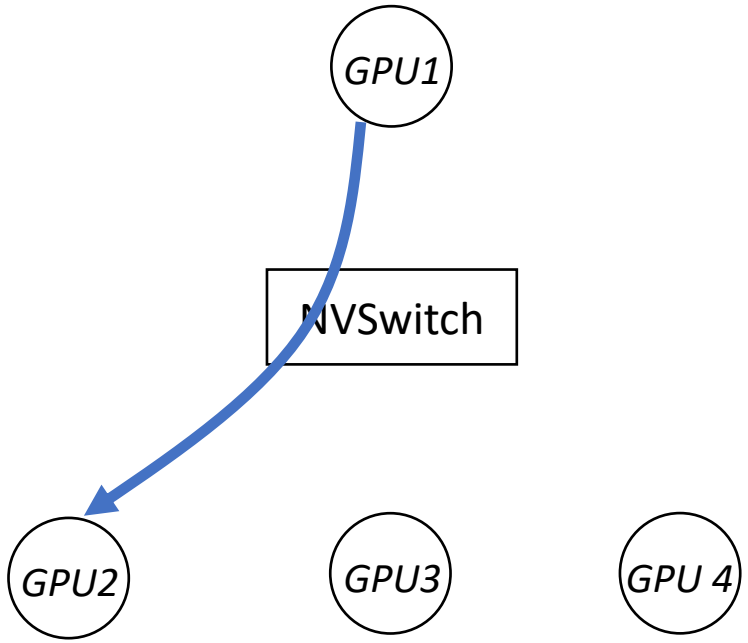
- With NVSwitch, the connectivity among any subset of GPUs is uniform
- NCCL constructs a multi-hop ring.



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.



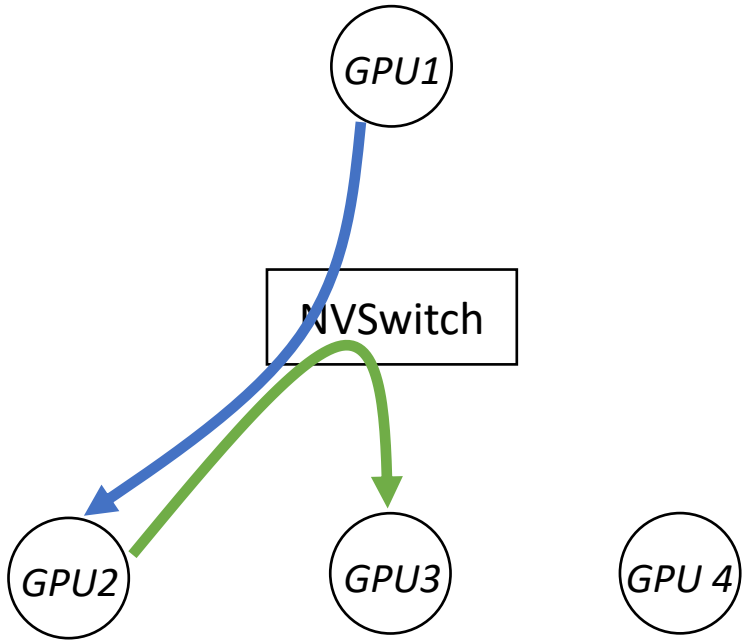
Hop Count

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.



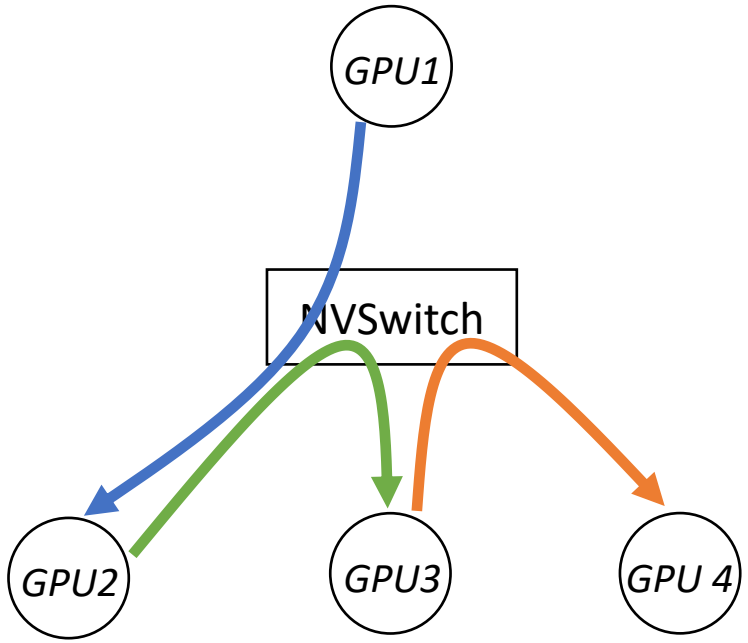
Hop Count

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.



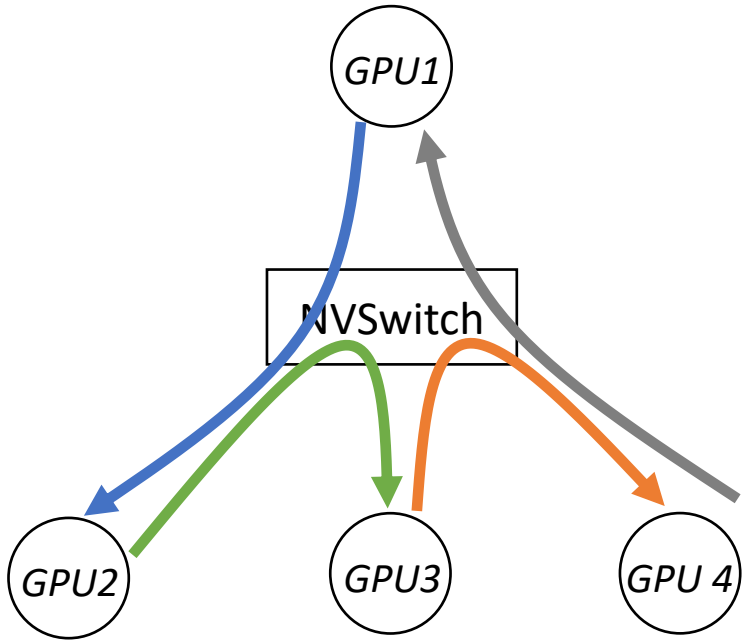
Hop Count

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.



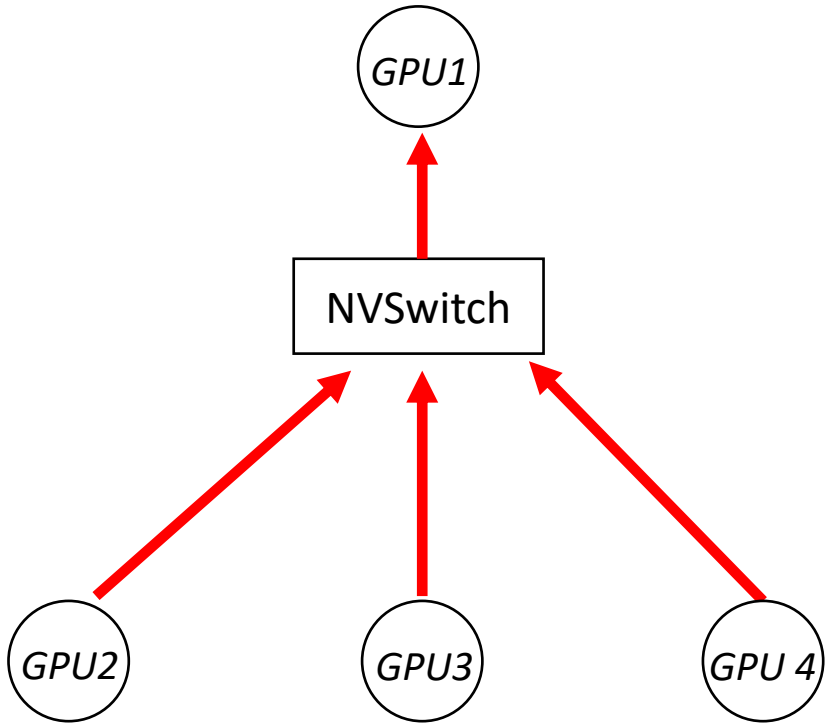
Hop Count

4

4GPU Reduce (G1->G4)

TreeGen for NVSwitch (DGX-2)

- DGX-2 single-hop tree



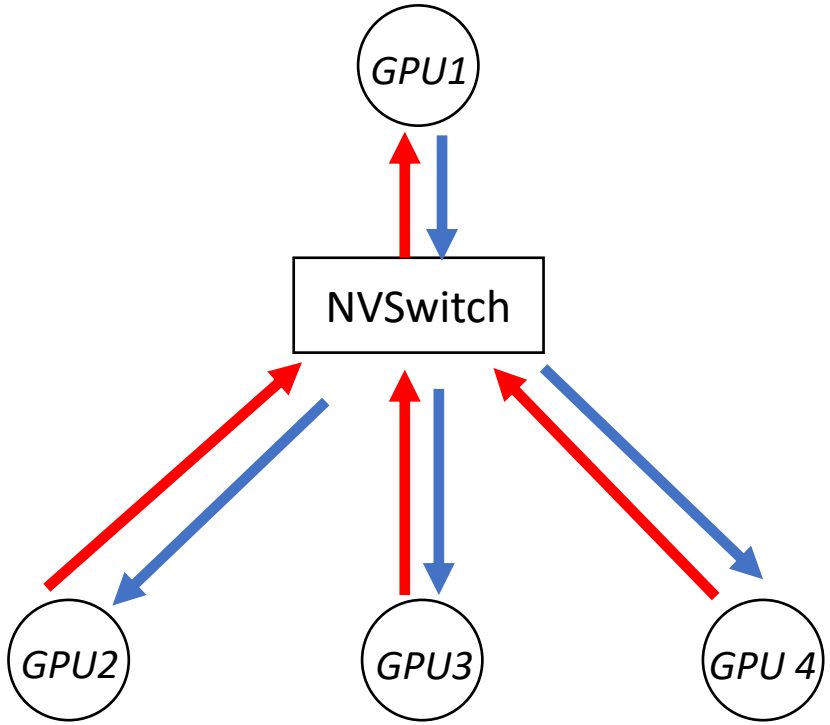
Reduce

1-hop tree introduces Min. latency

4GPU Reduce

TreeGen for NVSwitch (DGX-2)

- DGX-2 single-hop tree



Reduce

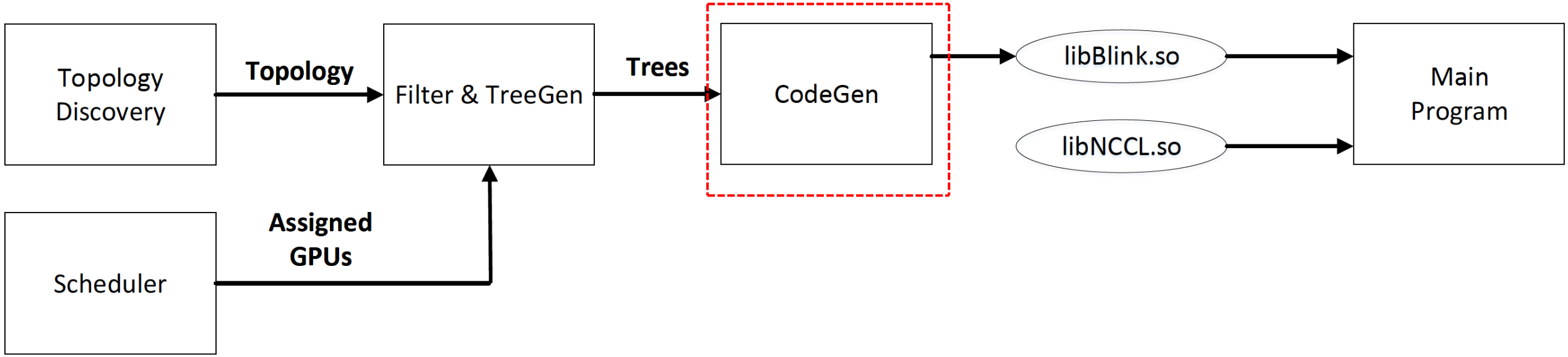
Broadcast

1-hop tree introduces Min. latency

AllReduce → Reduce, Broadcast

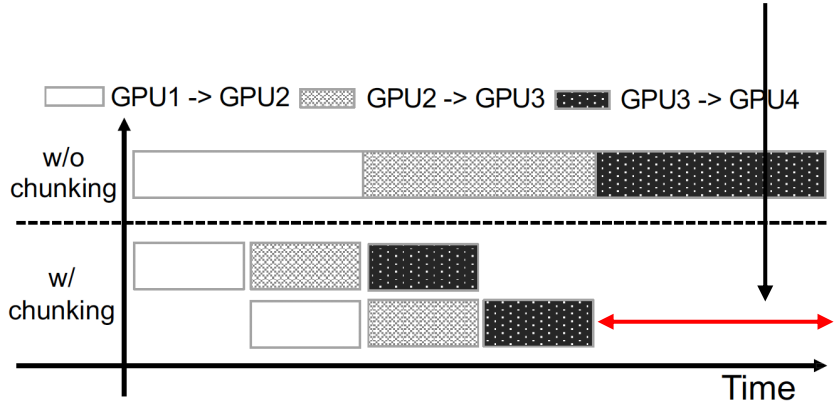
For N GPUs,
N 1-hop trees, with each tree responsible for 1/N data.

Blink workflow



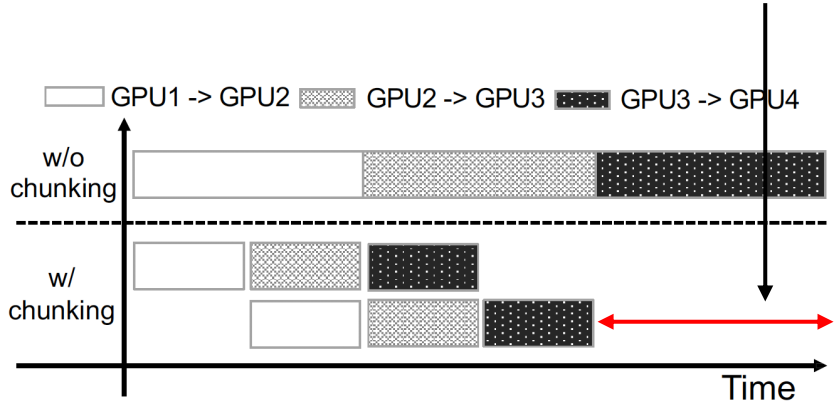
CodeGen

- 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

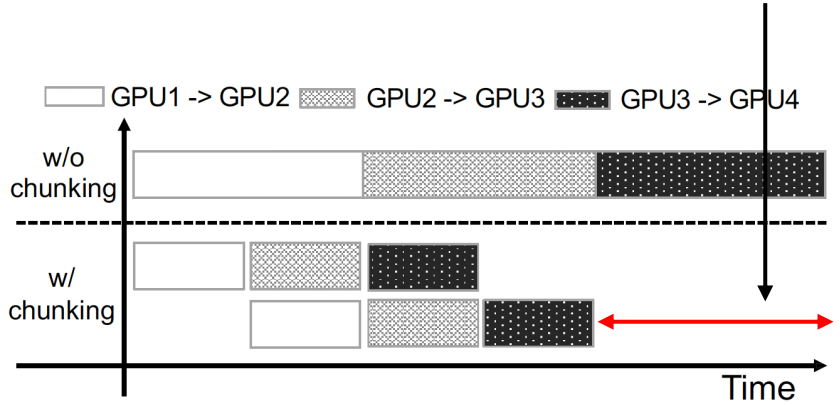


What chunk size to use?

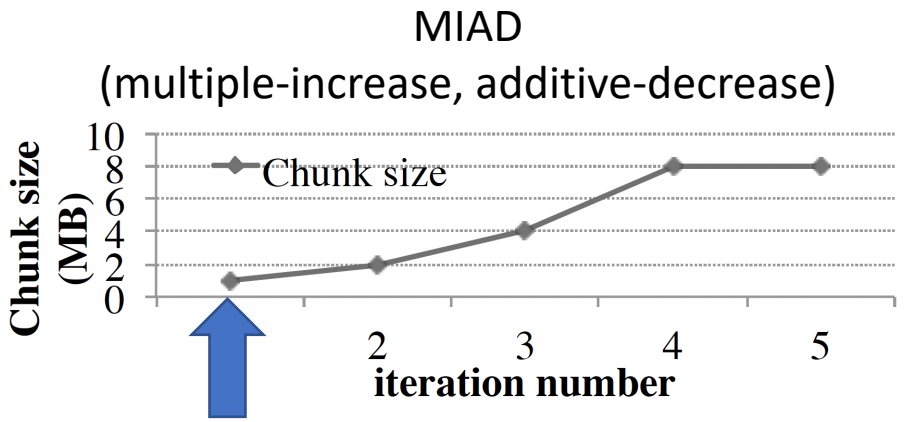
- Too small, cannot fully utilize BW
- Too big, high latency

CodeGen

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

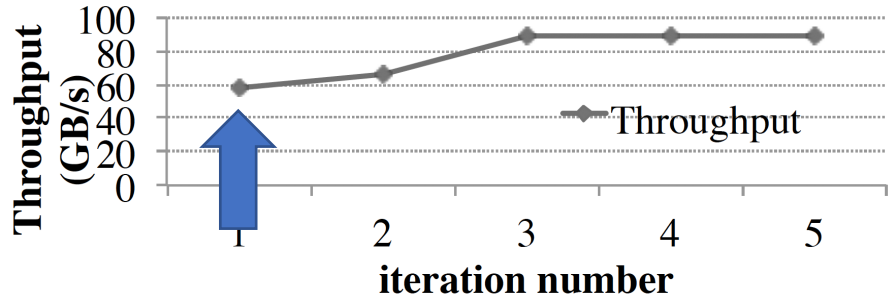


Automatic chunk size selection



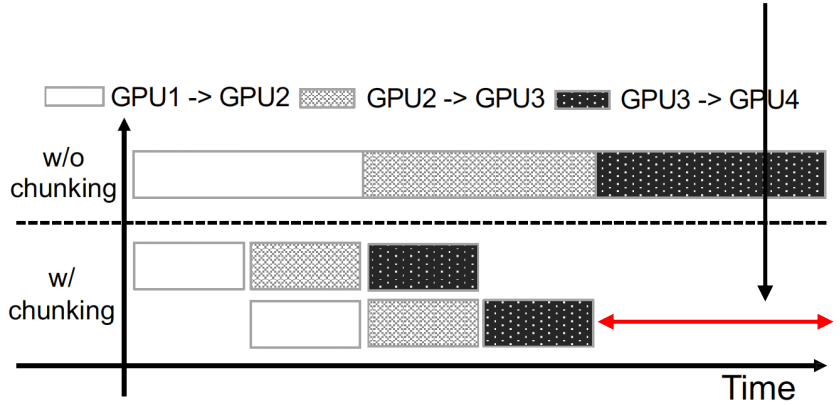
What chunk size to use?

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



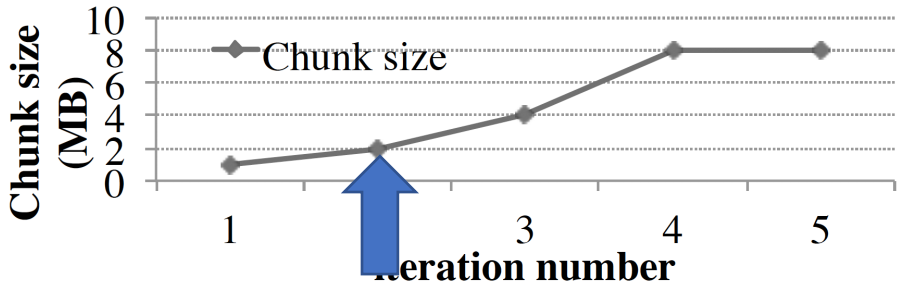
CodeGen

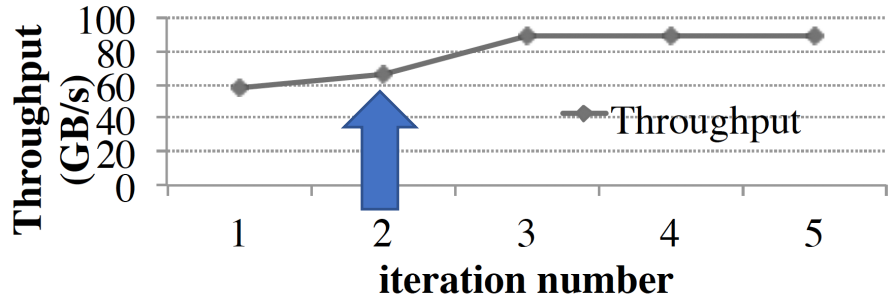
- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency



Automatic chunk size selection

MIAD
(multiple-increase, additive-decrease)



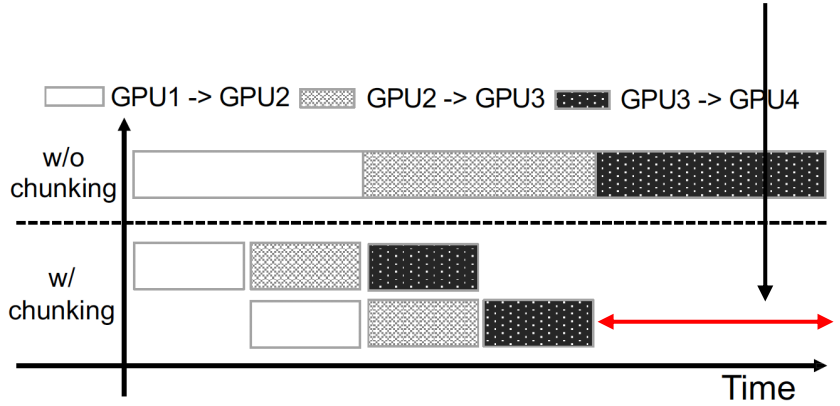


What chunk size to use?

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

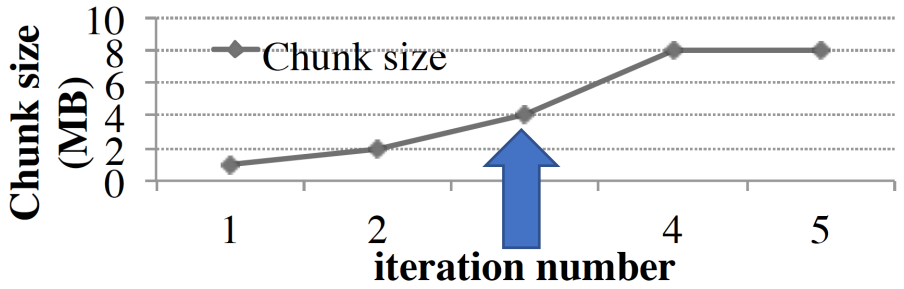
CodeGen

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



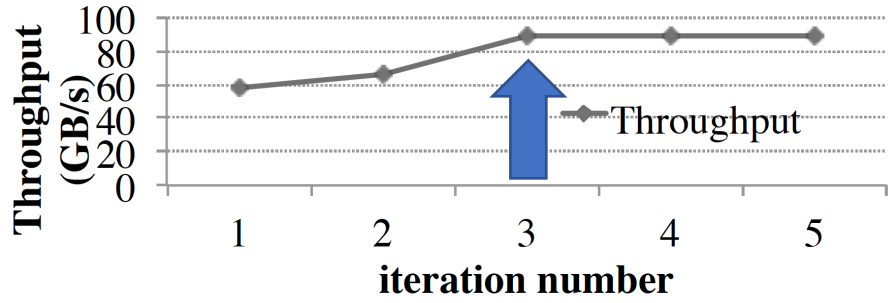
Automatic chunk size selection

MIAD
(multiple-increase, additive-decrease)



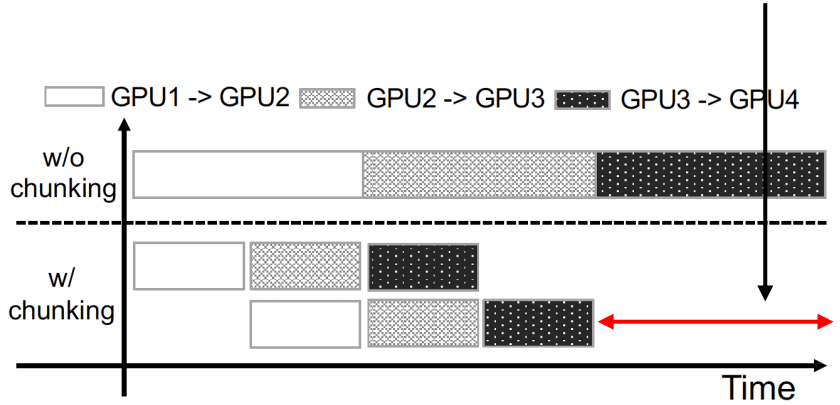
What chunk size to use?

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



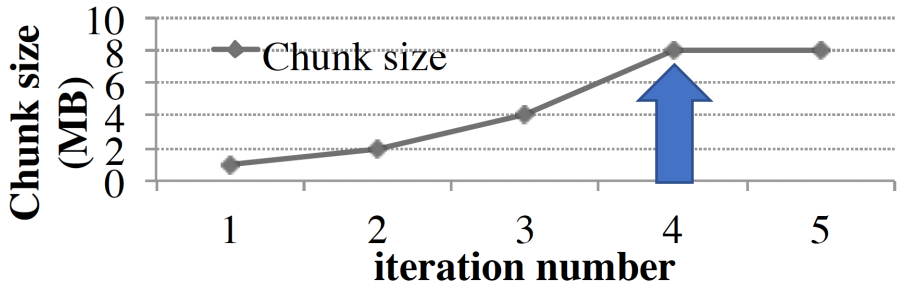
CodeGen

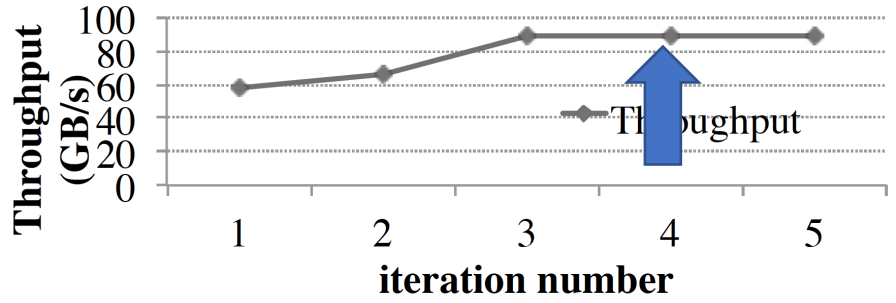
- Translate TreeGen output (spanning trees) into real data transfer commands
- CodeGen optimizations:
  - Pipelining data chunks to reduce latency



Automatic chunk size selection

MIAD
(multiple-increase, additive-decrease)





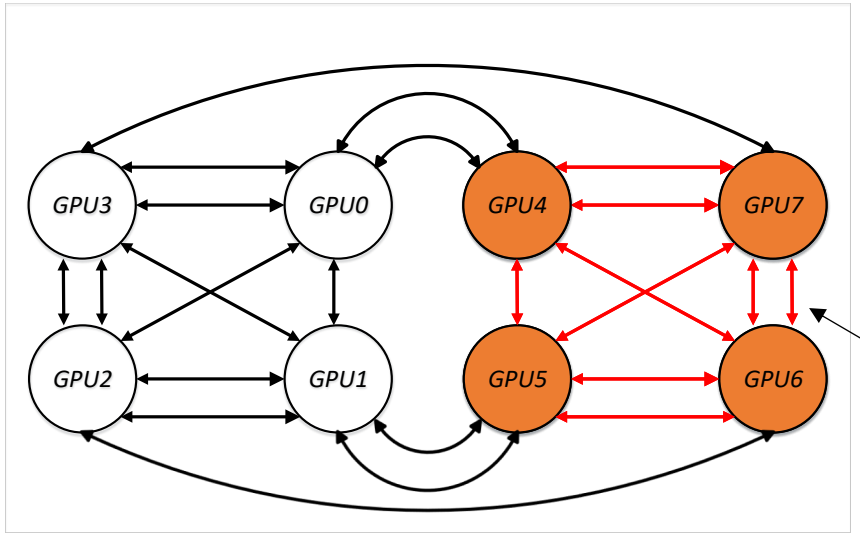
What chunk size to use?

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

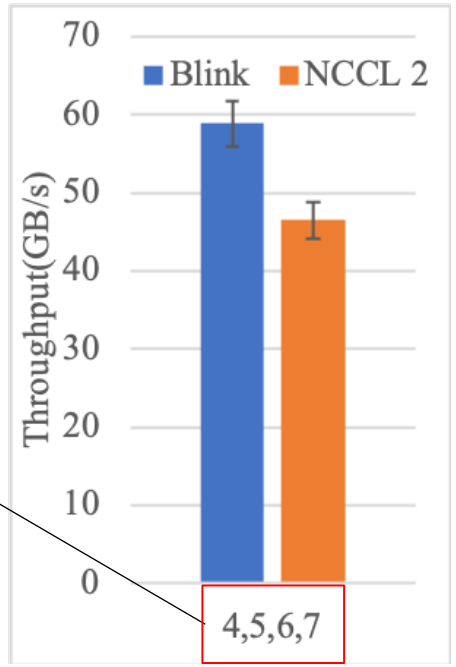
Talk Outline

- 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)

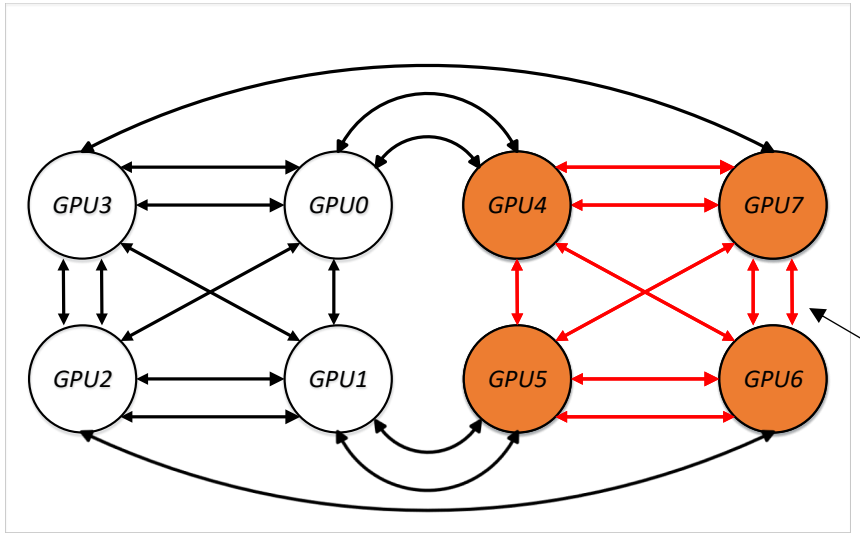


Topology

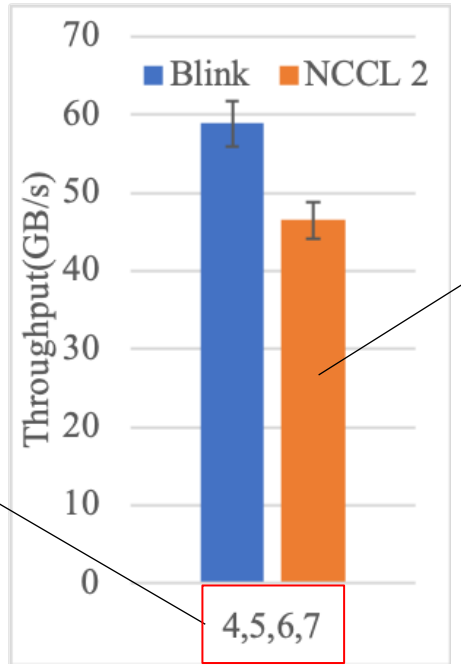


AllReduce

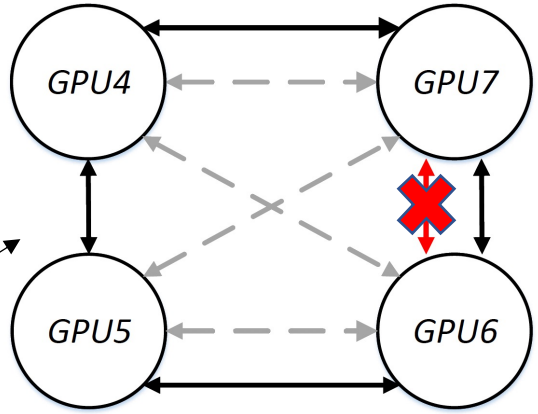
Microbenchmarks (DGX-1V)



Topology

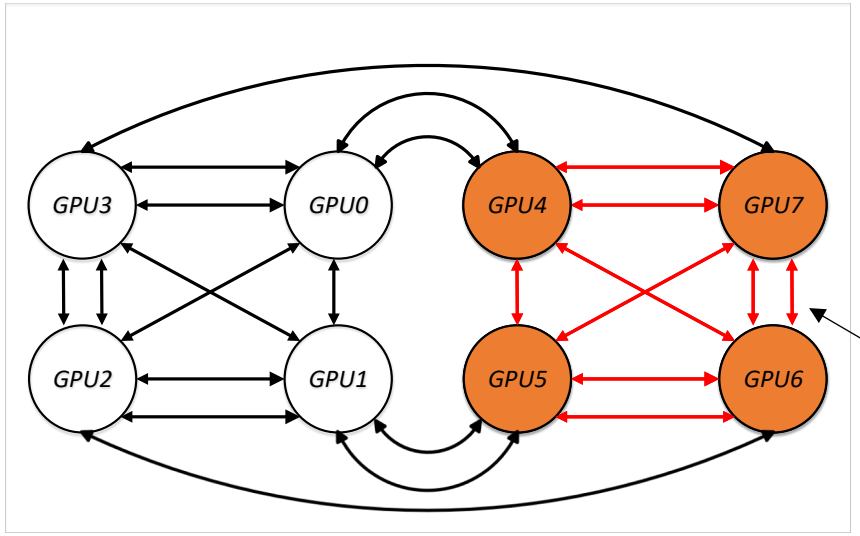


AllReduce

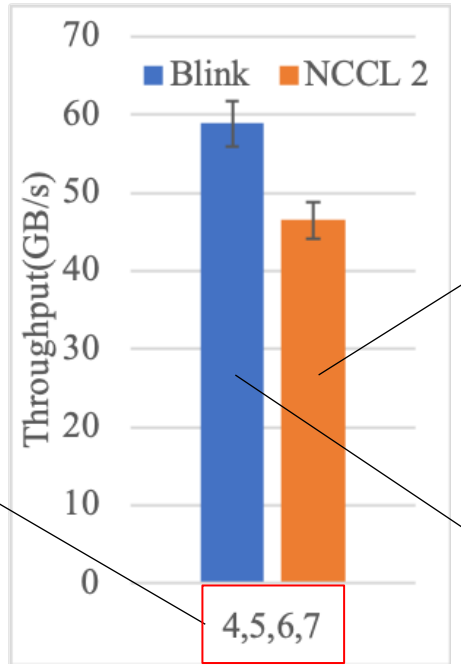


NCCL2 (2 rings)

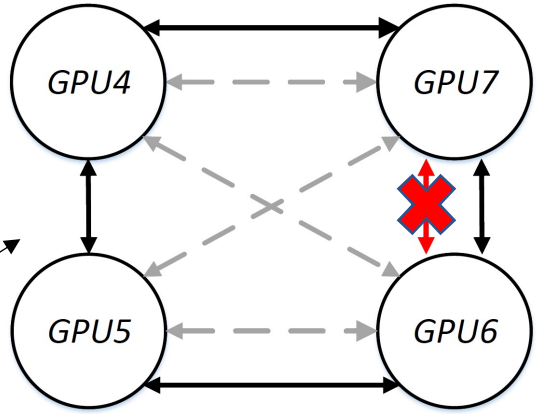
Microbenchmarks (DGX-1V)



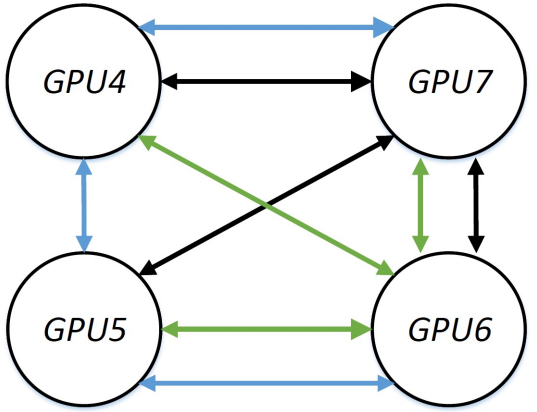
Topology



AllReduce

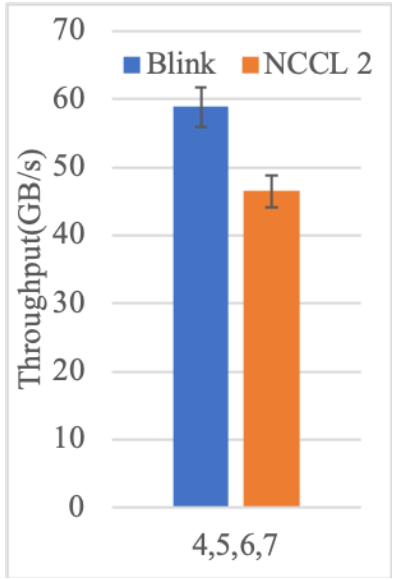


NCCL2 (2 rings)



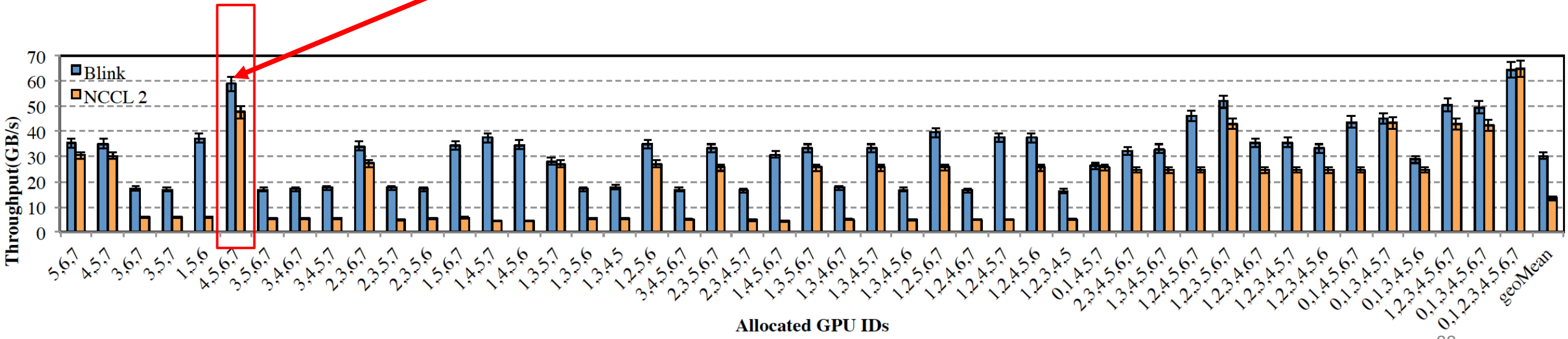
Blink (3 spanning trees)

Microbenchmarks (DGX-1V)

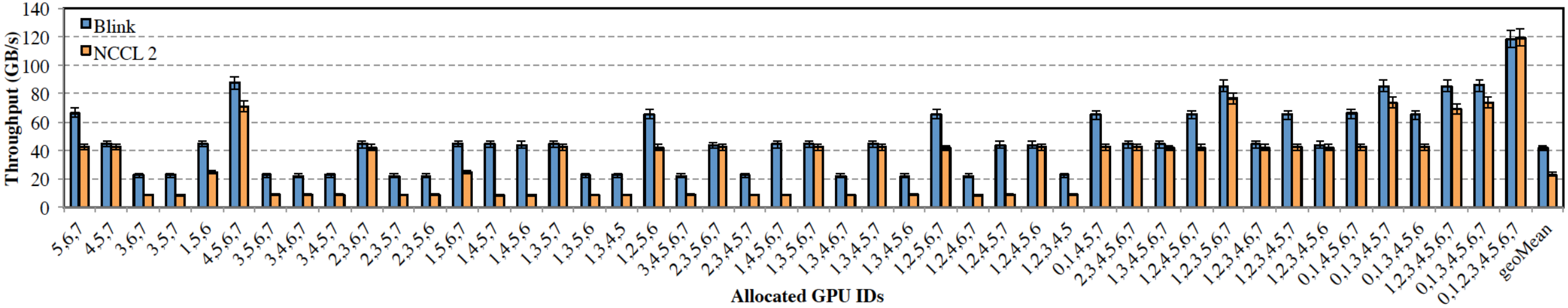


AllReduce

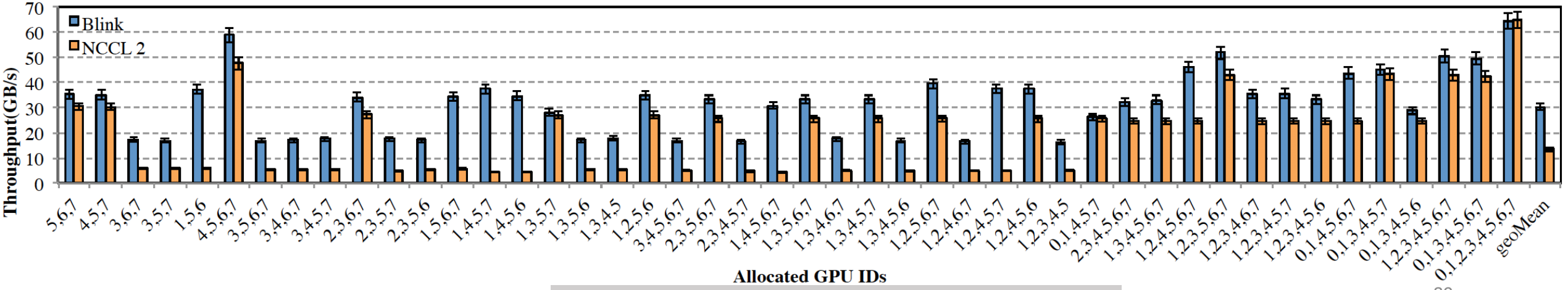
(up to 8x speed-up, 2x geo-mean)



Microbenchmarks (DGX-1V)

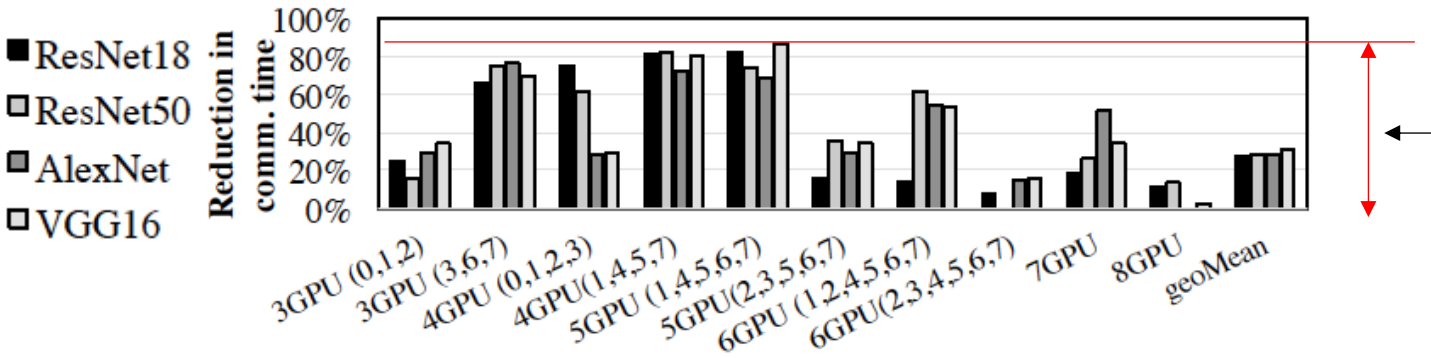


Broadcast (up to 6x speed-up, 2x geo-mean)



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

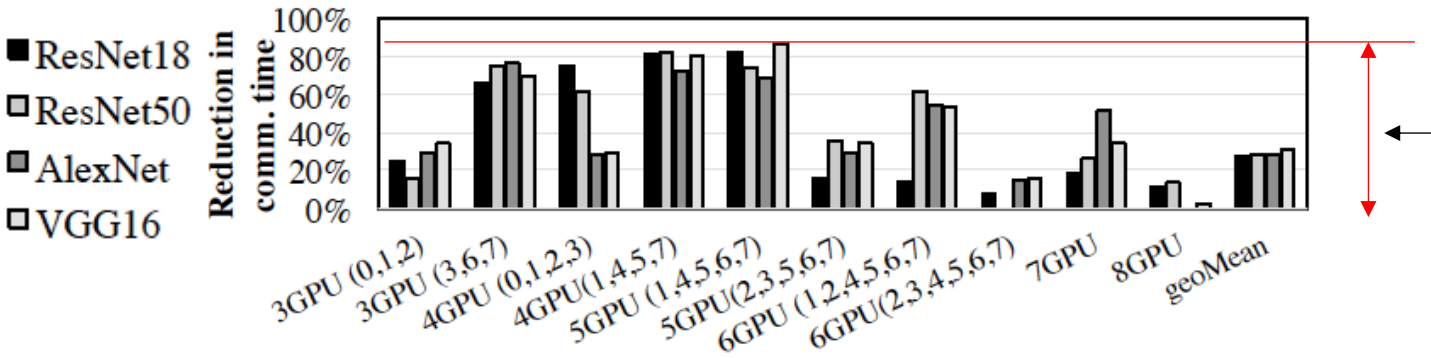
End-to-end Benchmarks (DGX-1V)



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

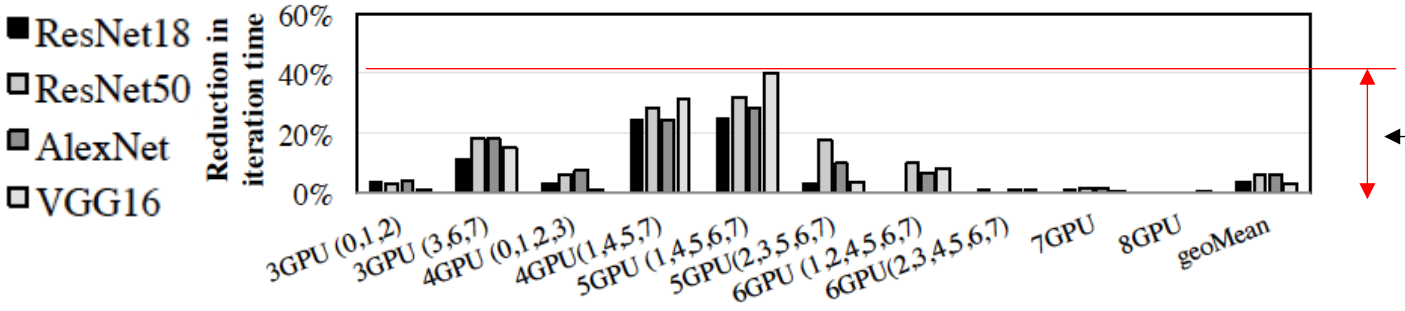
Blink end-to-end Communication time reduction (ImageNet1K)

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



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

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

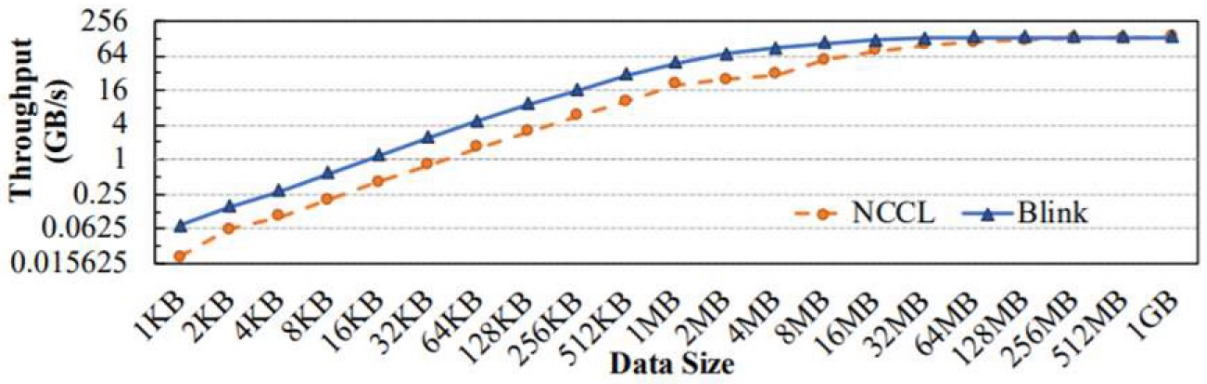


up to 40% end-to-end training time reduction

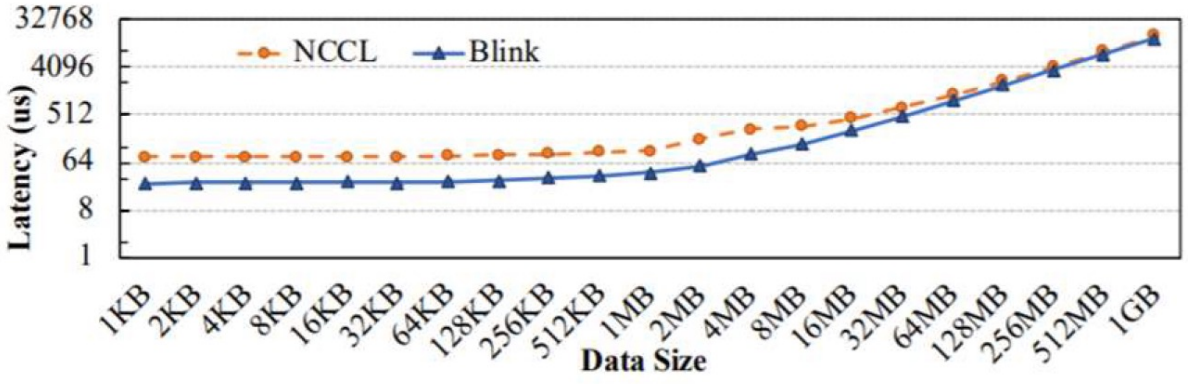
Blink end-to-end training time reduction (ImageNet1K)

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 Summary

- 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.

BLINK

Microsoft

Patent No. US 2020/0160171 A1

BLINK

Microsoft

Patent No. US 2020/0160171 A1

facebook

PyTorch

ByteDance

TikTok

BLINK

Microsoft

Patent No. US 2020/0160171 A1

facebook

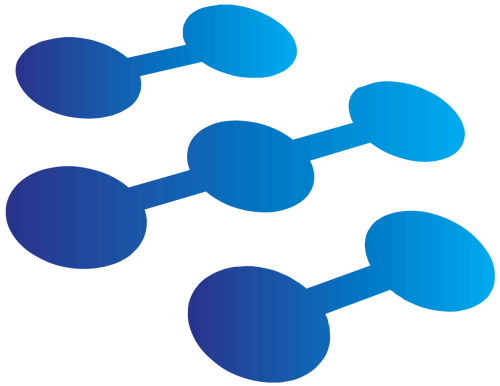
PyTorch

ByteDance

TikTok

Baidu 百度

Tencent



sensAI

ConvNets Decomposition via Class Parallelism for Fast Inference on Live Data

MLSys 2021

CNNs enable Computers to Excel on Vision Learning Tasks

Dog: 96%
Cat: 29%
Duck: 2%
Bird: 0%

Dog: 36%
Cat: 94%
Duck: 2%
Bird: 1%

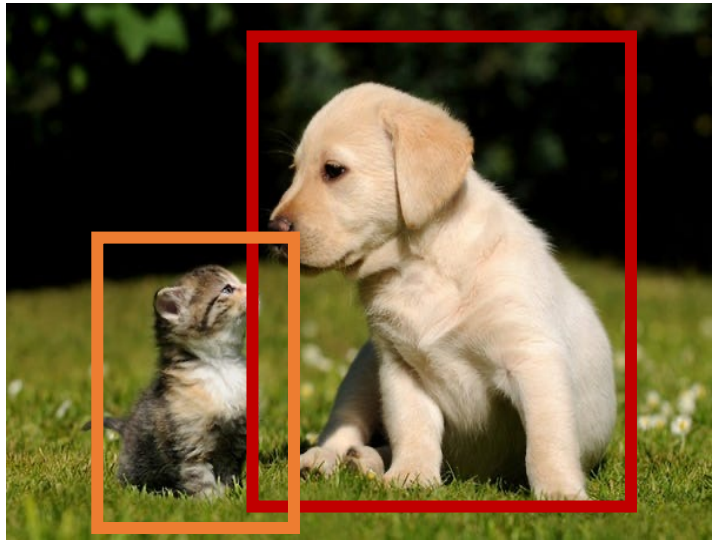
Image Classification

CNNs enable Computers to Excel on Vision Learning Tasks

Dog: 96%
Cat: 29%
Duck: 2%
Bird: 0%

Dog: 36%
Cat: 94%
Duck: 2%
Bird: 1%

Image Classification



Object Detection

CNNs enable Computers to Excel on Vision Learning Tasks

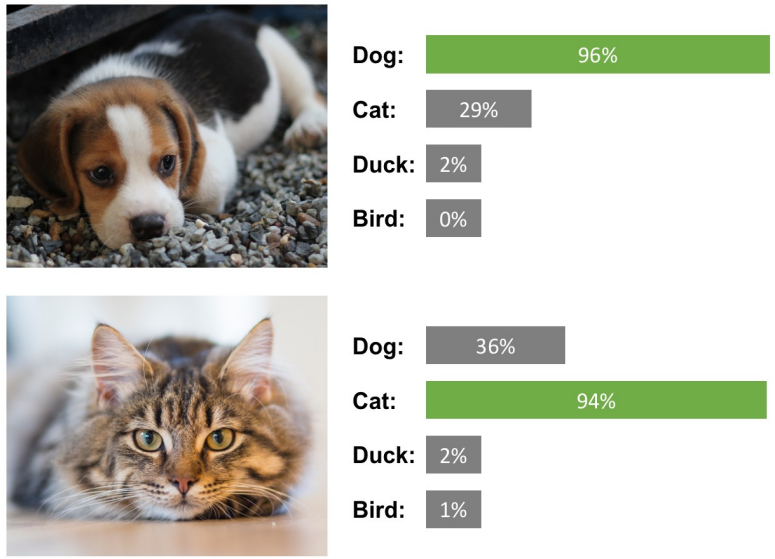
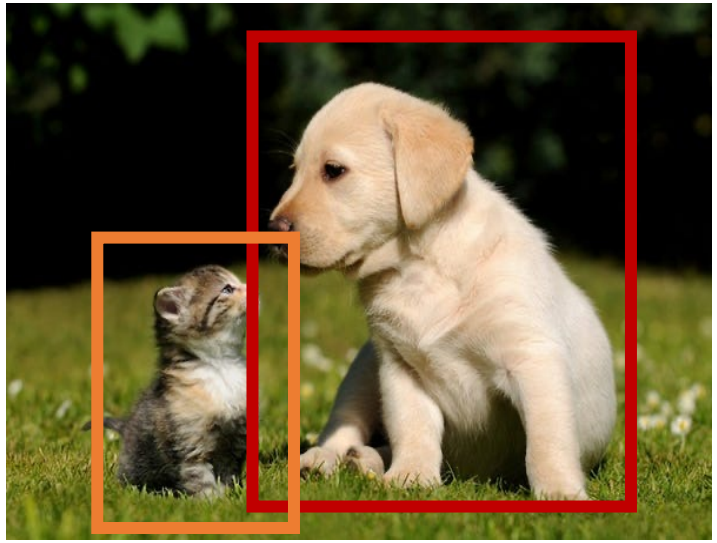
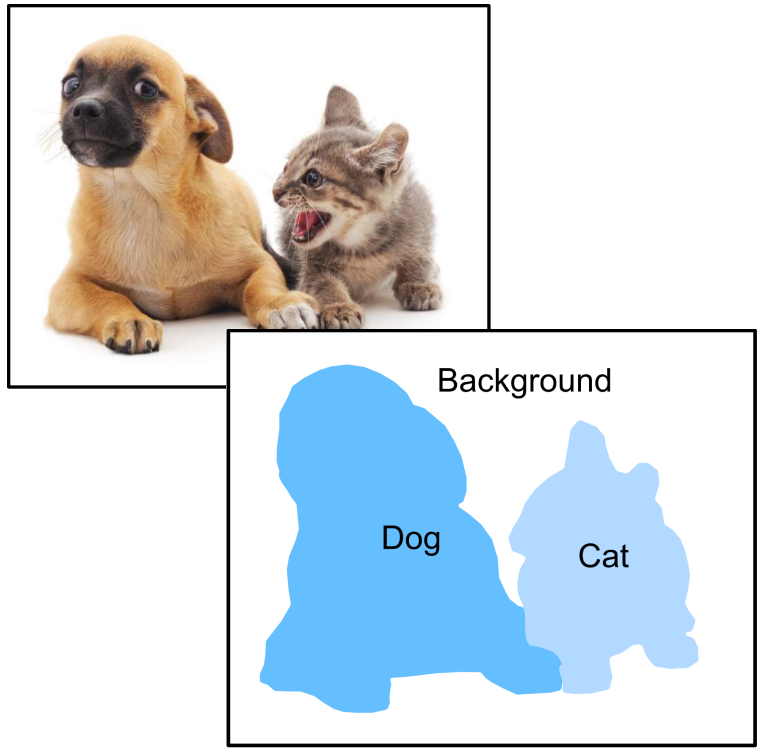


Image Classification

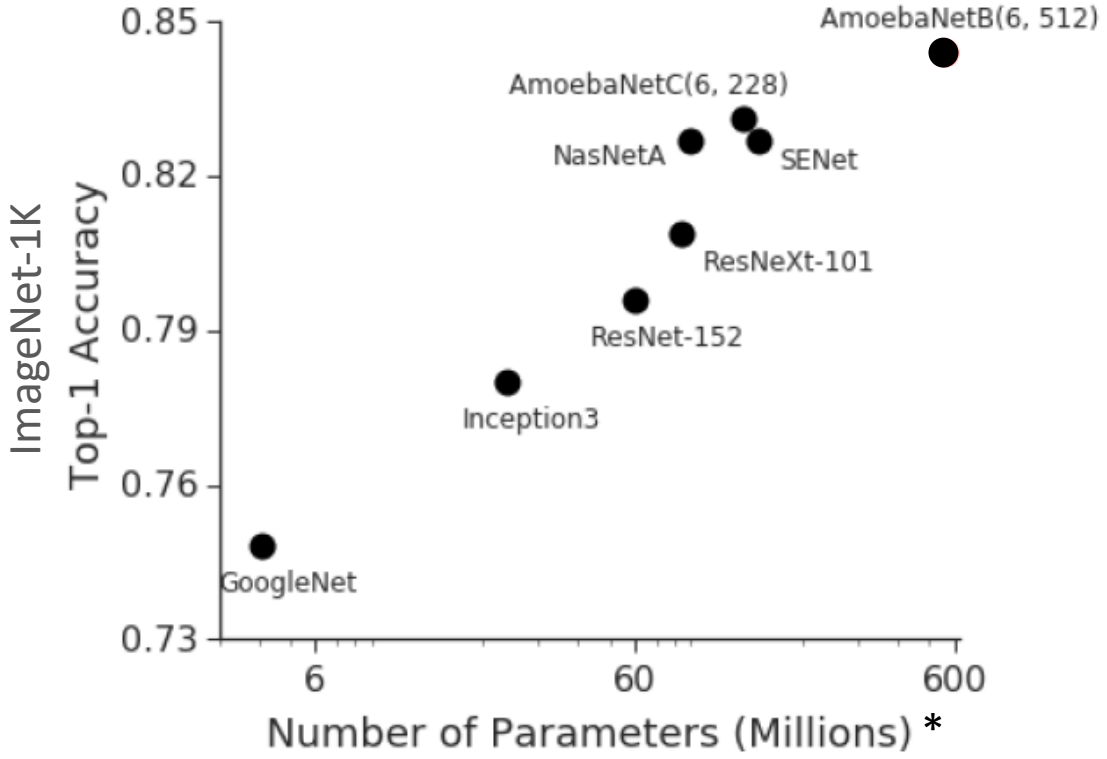


Object Detection



Sematic Segmentation

Models and Input Data are Growing Drastically



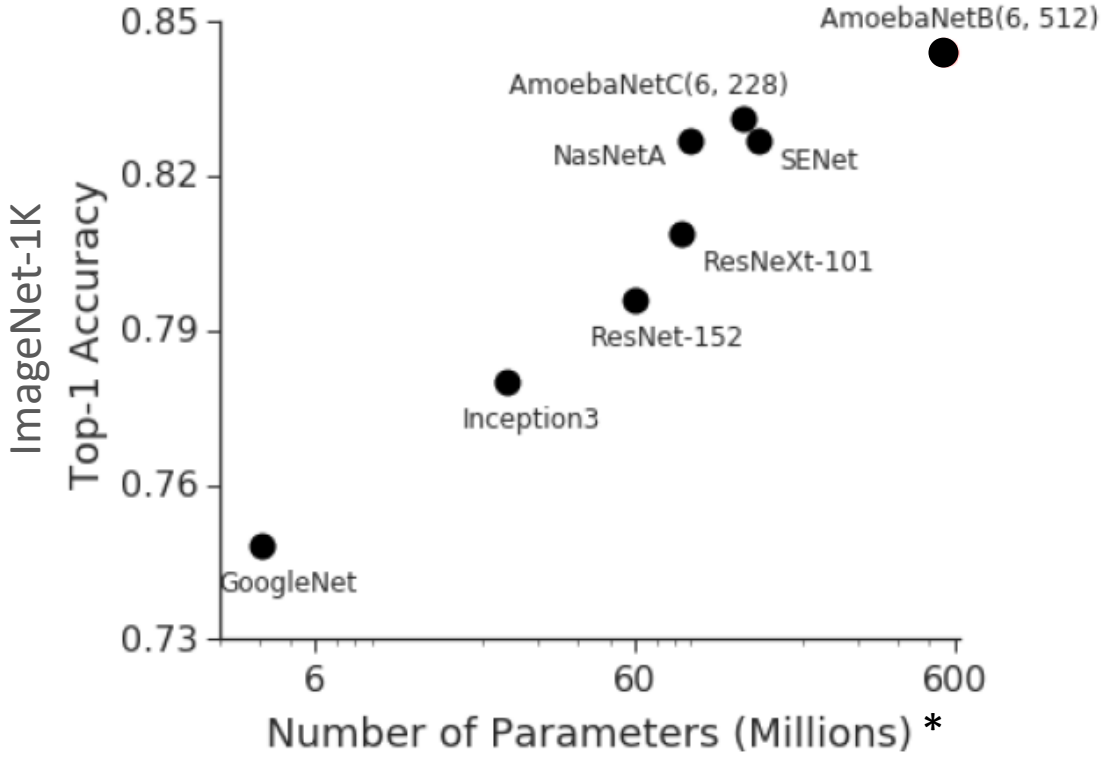
Better Serving Accuracy

Models grow in

- Depth (layer)
- Width (channel)
- Connections
  (e.g. ResNet->DenseNet)

*Introducing Gpipe open-source library, Google AI Blog 2019

Models and Input Data are Growing Drastically



Better Serving Accuracy

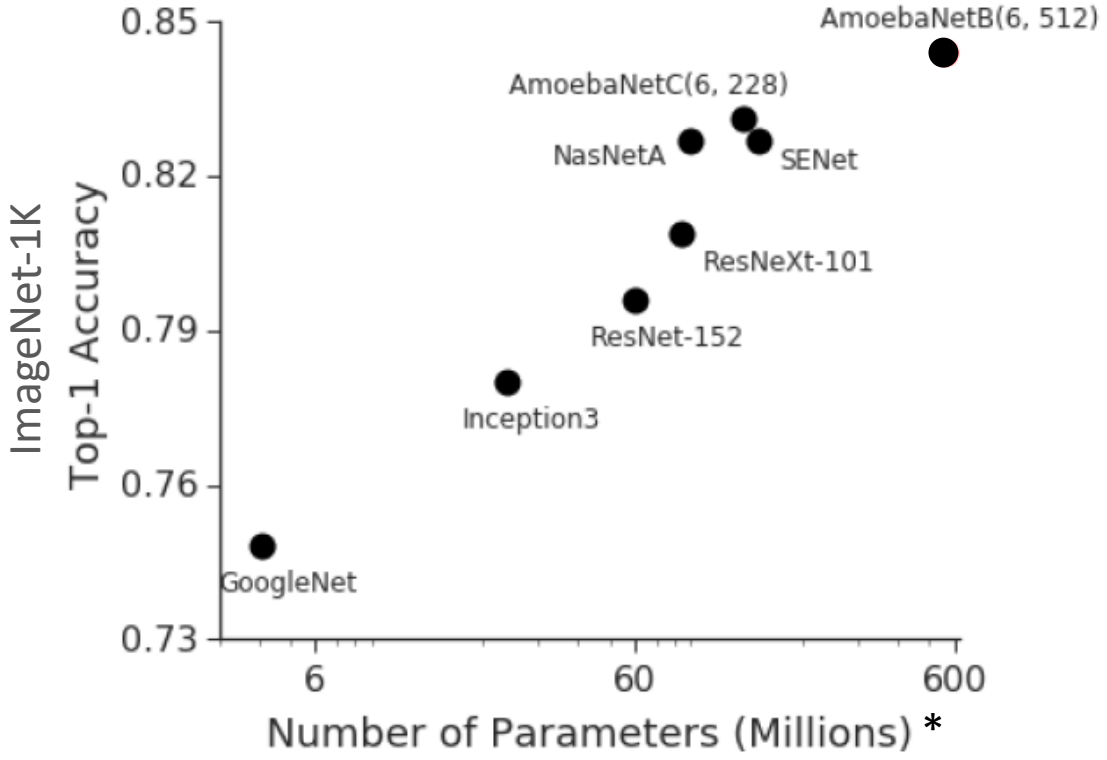
Models grow in

- Depth (layer)
- Width (channel)
- Connections
  (e.g. ResNet->DenseNet)

More Parameters
More FLOPs
More Memory Footprints

*Introducing Gpipe open-source library, Google AI Blog 2019

Models and Input Data are Growing Drastically



Models grow in

- Depth (layer)
- Width (channel)
- Connections
  (e.g. ResNet->DenseNet)

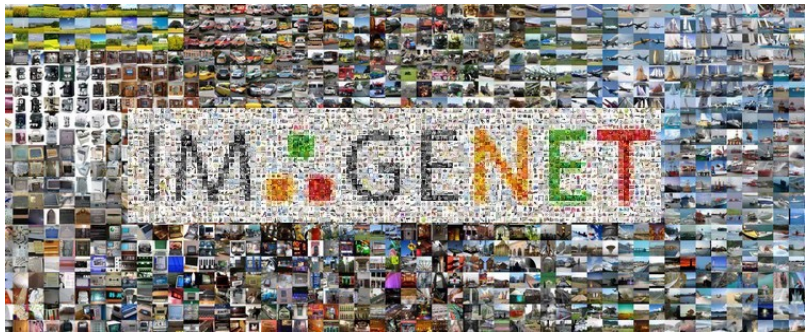
More Parameters
More FLOPs
More Memory Footprints

Better Serving Accuracy → Higher Latency

*Introducing Gpipe open-source library, Google AI Blog 2019

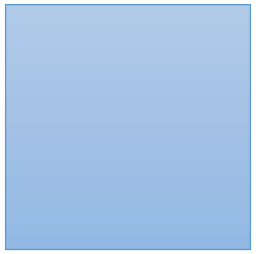
Models and Input Data are Growing Drastically

CIFAR-10/100
(32x32)



ImageNet-1K
(256x256)

COCO
(640x480)



Larger Image => Higher Latency

Making **Faster** Decision on Live Data is Important

WAYMO

cruise

Making **Faster** Decision on Live Data is Important

WAYMO

cruise

J.P.Morgan

Goldman Sachs

Making **Faster** Decision on Live Data is Important

WAYMO

cruise

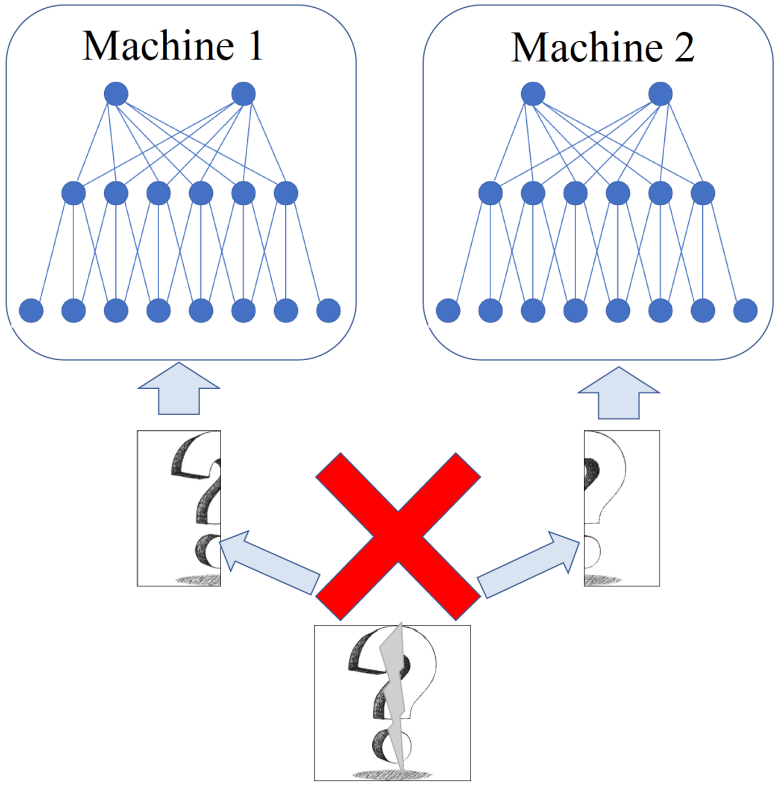
J.P.Morgan

Goldman Sachs

System view: make faster decision on a single data item

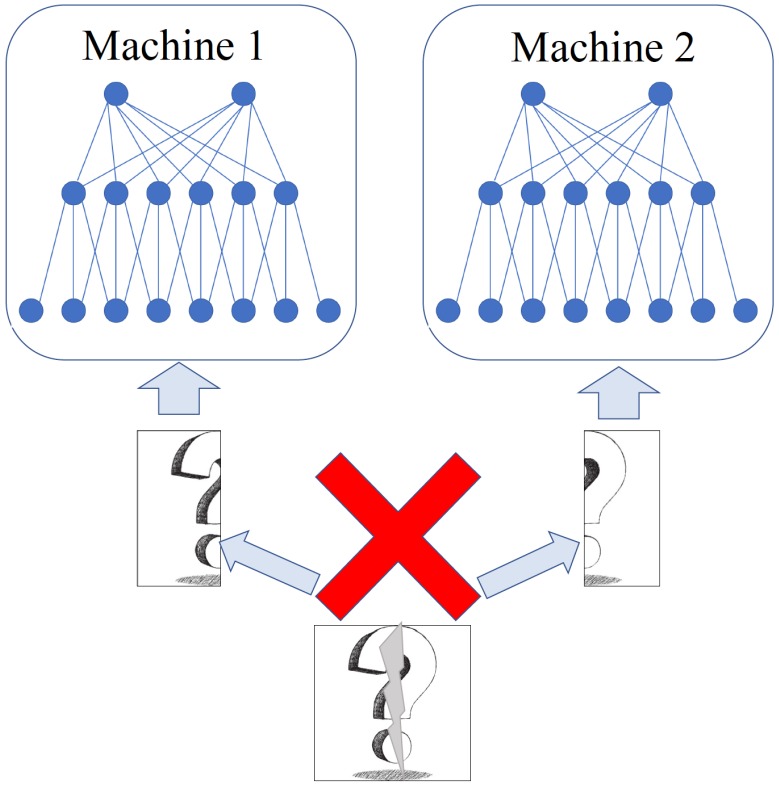
e.g. 1 image, 1 stock's instantaneous price

Neither Data Parallelism nor Model Parallelism can Reduce Live Data Serving Latency

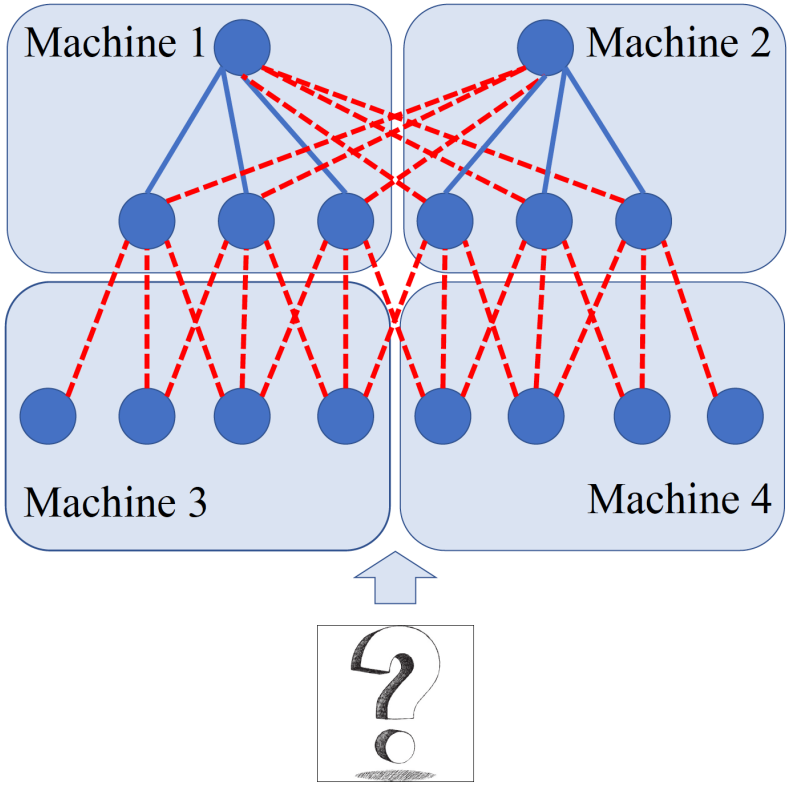


Data Parallelism

Neither Data Parallelism nor Model Parallelism can Reduce Live Data Serving Latency

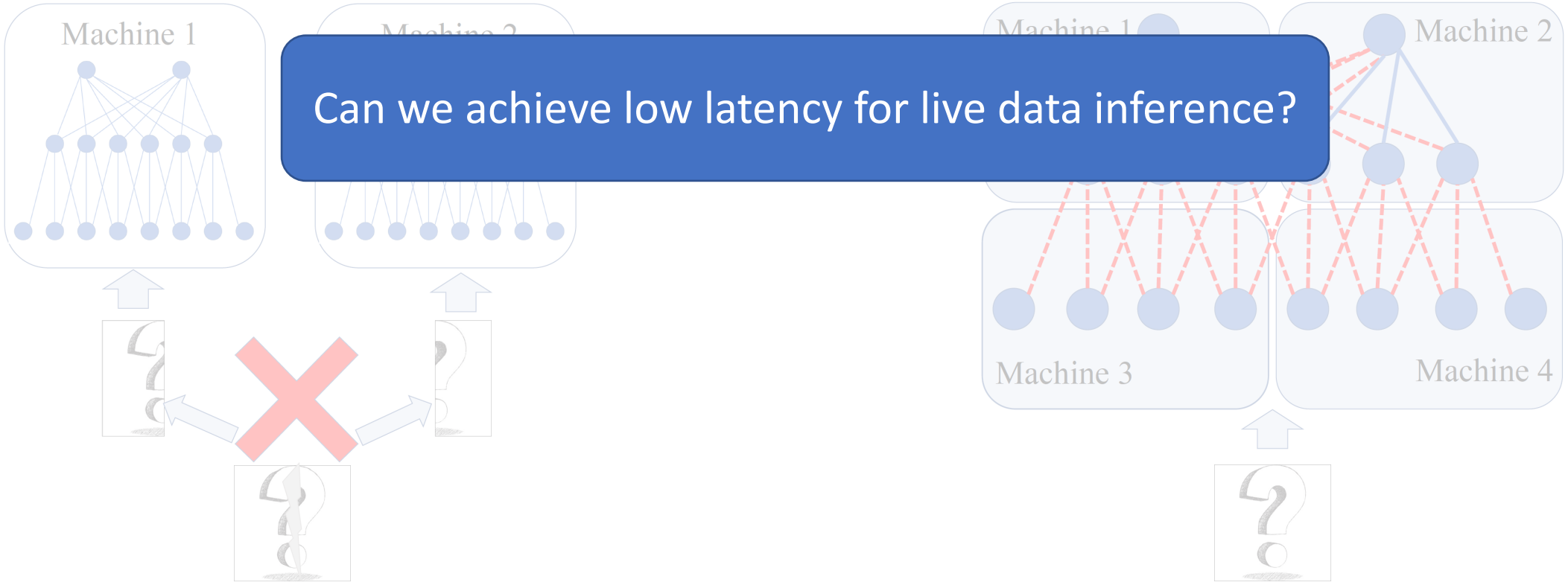


Data Parallelism



Model Parallelism

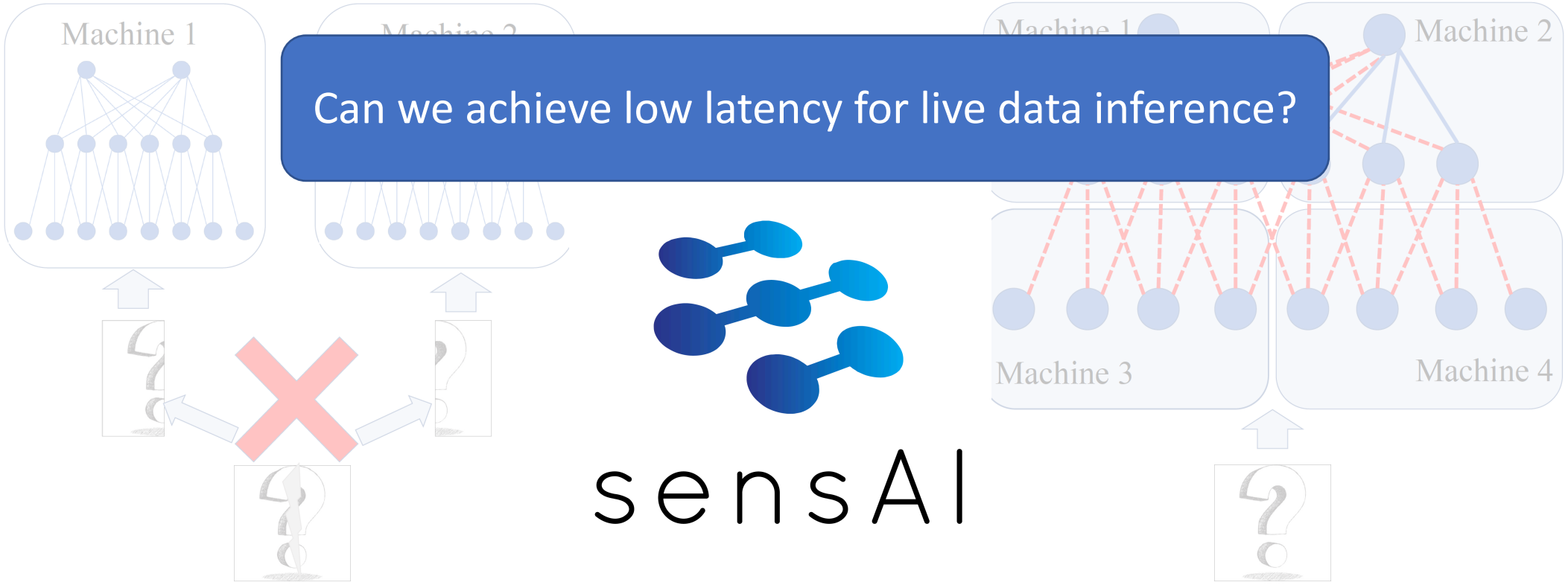
Neither Data Parallelism nor Model Parallelism can Reduce Live Data Serving Latency



Data Parallelism

Model Parallelism

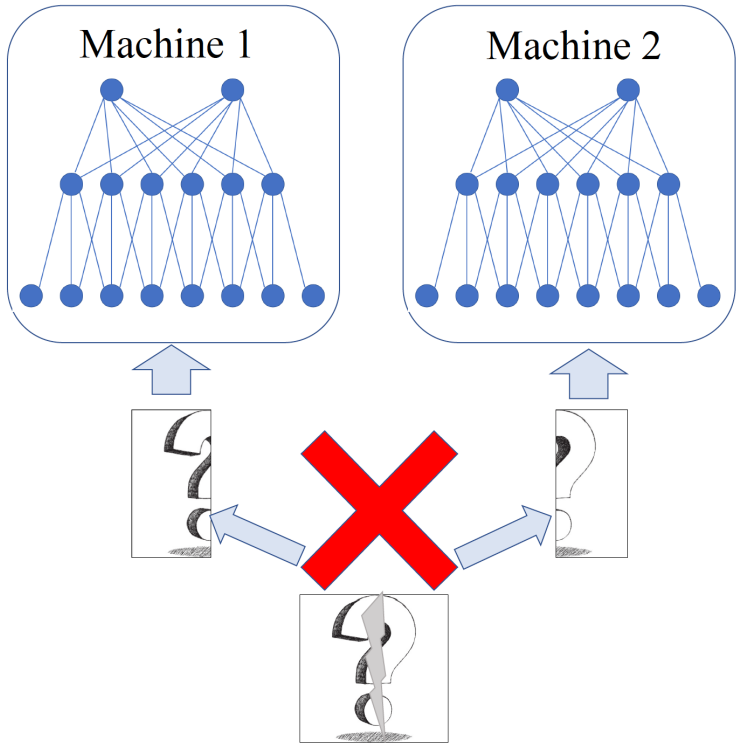
Neither Data Parallelism nor Model Parallelism can Reduce Live Data Serving Latency



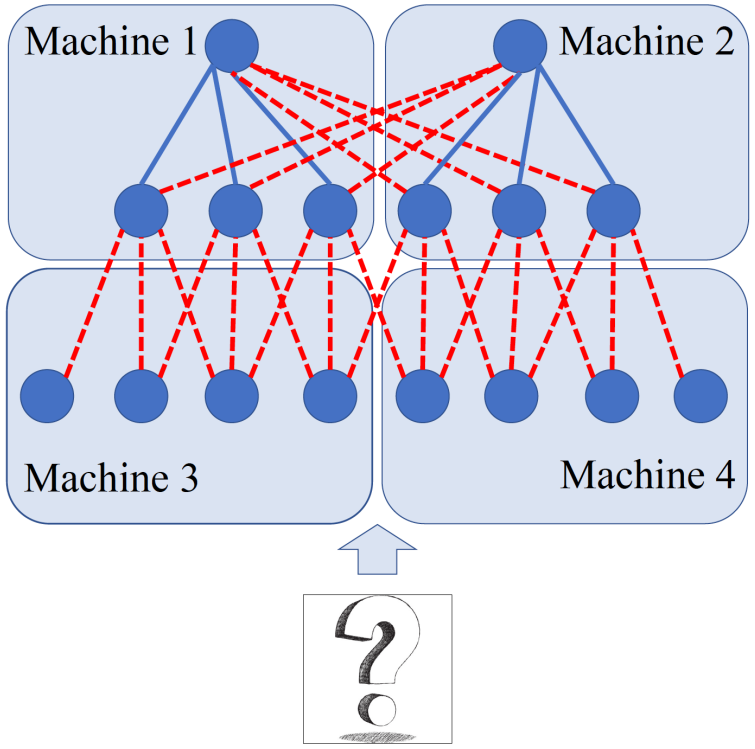
Data Parallelism

Model Parallelism

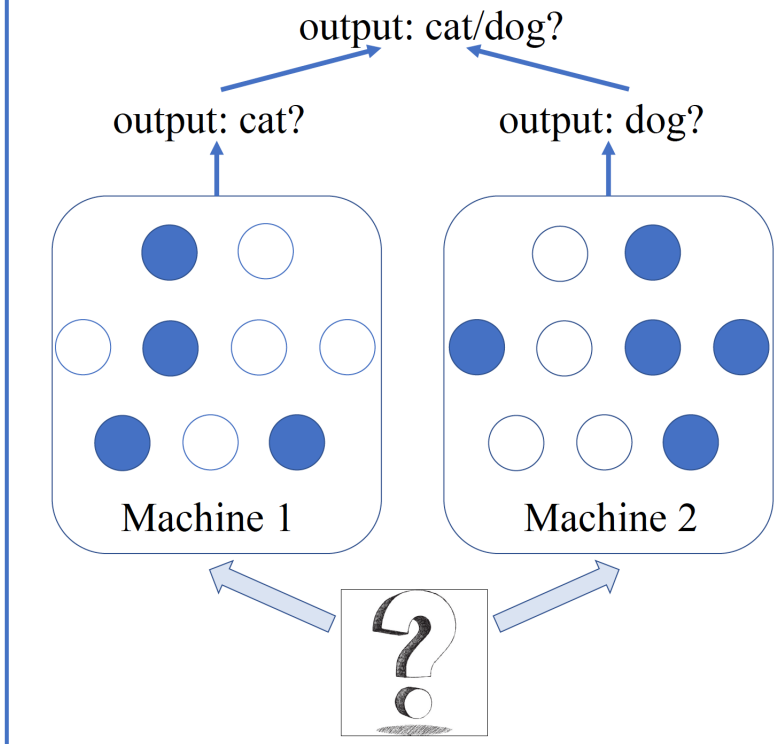
Class Parallelism in sensAI



Data Parallelism



Model Parallelism

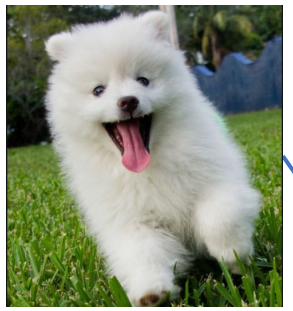


Class Parallelism

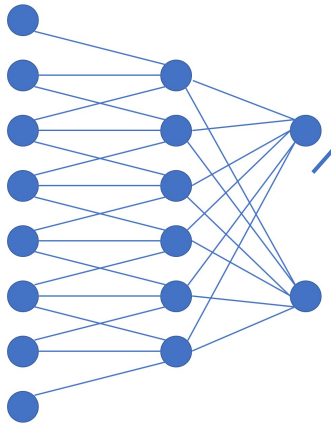
Talk Outline

- Motivation
- **Design**
- Evaluation
- Extensions

sensAI workflow



Fully-trained
CNN



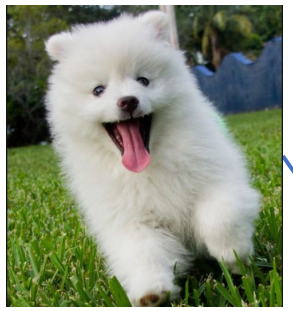
Class-specific
pruning

Pruning

Active neurons

Inactive/pruned neurons

sensAI workflow



Fully-trained CNN

Class-specific pruning

Dog classifier

retrain

Cat classifier

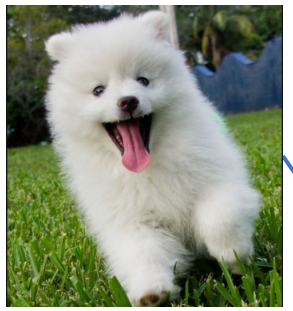
Pruning

Retraining

Active neurons

Inactive/pruned neurons

sensAI workflow



Fully-trained CNN

Class-specific pruning

Dog classifier

retrain

Cat classifier

dog?

0.2

cat?

0.6

SoftMax

Decision: Cat

Active neurons

Inactive/pruned neurons

Pruning

Retraining

Inference

Talk Outline

- Motivation

- Design
  - **Class-specific Pruning**
  - **Retraining**
  - **Combine results back to the original N-way Predictions**

- Evaluation

- Extensions

Talk Outline

- Motivation

- Design
  - **Class-specific Pruning**
  - **Retraining**
  - **Combine results back to the original N-way Predictions**

- Evaluation

- Extensions

Class-specific Pruning

- Binary Classifiers
  - Feed in training images of 1 class to a fully trained model, collect activations per neuron

Class-specific Pruning

- Binary Classifiers
  - Feed in training images of 1 class to a fully trained model, collect activations per neuron
  - Activation based pruning criteria
  - hybrid policy

Class-specific Pruning

- Binary Classifiers
  - Feed in training images of 1 class to a fully trained model
  - Activation based pruning criteria
  - hybrid policy

$$\Psi(N_{i,j}^c) = \frac{\sum^{D_c} \Phi(A_{i,j}^c < \theta_1)}{D_c}$$

Step 1: Generate initial pruning candidates

Average % of activation values $< \theta_1$

Class-specific Pruning

- Binary Classifiers
  - Feed in training images of 1 class to a fully trained model
  - Activation based pruning criteria
  - hybrid policy

$$\Psi(N_{i,j}^c) = \frac{\sum^{D_c} \Phi(A_{i,j}^c < \theta_1)}{D_c}$$

$$where\ \Theta(A_{i,j}^c) < \theta_2$$

Step 1: Generate initial pruning candidates

Average % of activation values $< \theta_1$

Step 2: Exclude neurons from the pruning list

if mean absolute values of its activation $> \theta_2$

Class-specific Pruning

- Binary Classifiers
  - Feed in training images of 1 class to a fully trained model
  - Activation based pruning criteria
  - hybrid policy

$$\Psi(N_{i,j}^c) = \frac{\sum^{D_c} \Phi(A_{i,j}^c < \theta_1)}{D_c}$$

Step 1: Generate initial pruning candidates

Average % of activation values $< \theta_1$

$$where\ \Theta(A_{i,j}^c) < \theta_2$$

Step 2: Exclude neurons from the pruning list

if mean absolute values of its activation $> \theta_2$

To avoid pruning neurons with activations that
1. high % of near-zero ($<\theta_1$) values
2. a few large non-zero ($>\theta_2$) values

Class-specific Pruning

- Binary Classifiers
- Grouped Classifiers
  - Why grouping?

Class-specific Pruning

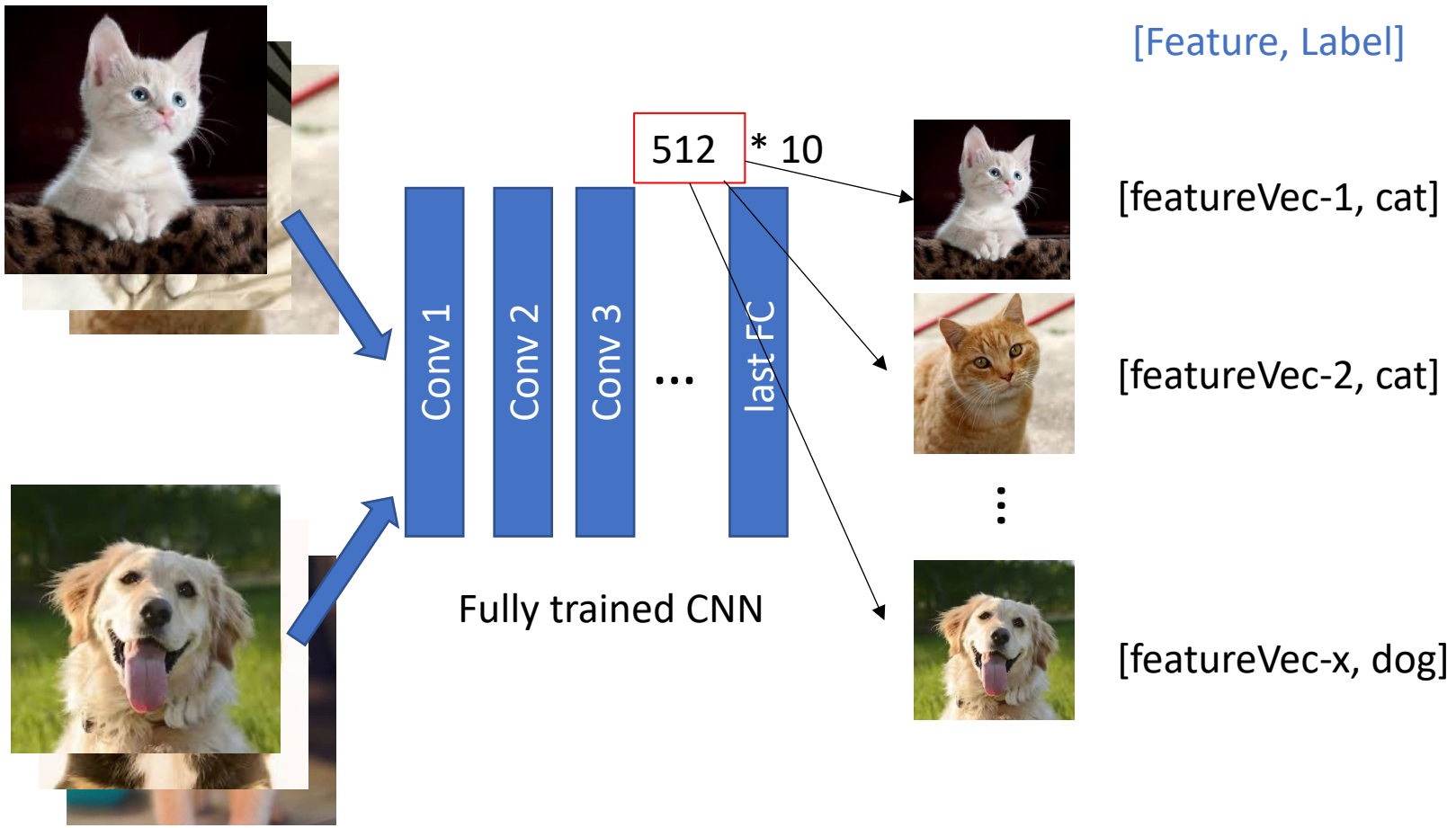
- Binary Classifiers
- Grouped Classifiers
  - Why grouping?
    1. Too many classes in the dataset (e.g. ImageNet-1K)
    2. Achieving arbitrary levels of parallelism

Class-specific Pruning

- Binary Classifiers
- Grouped Classifiers
  - Class grouping, how?

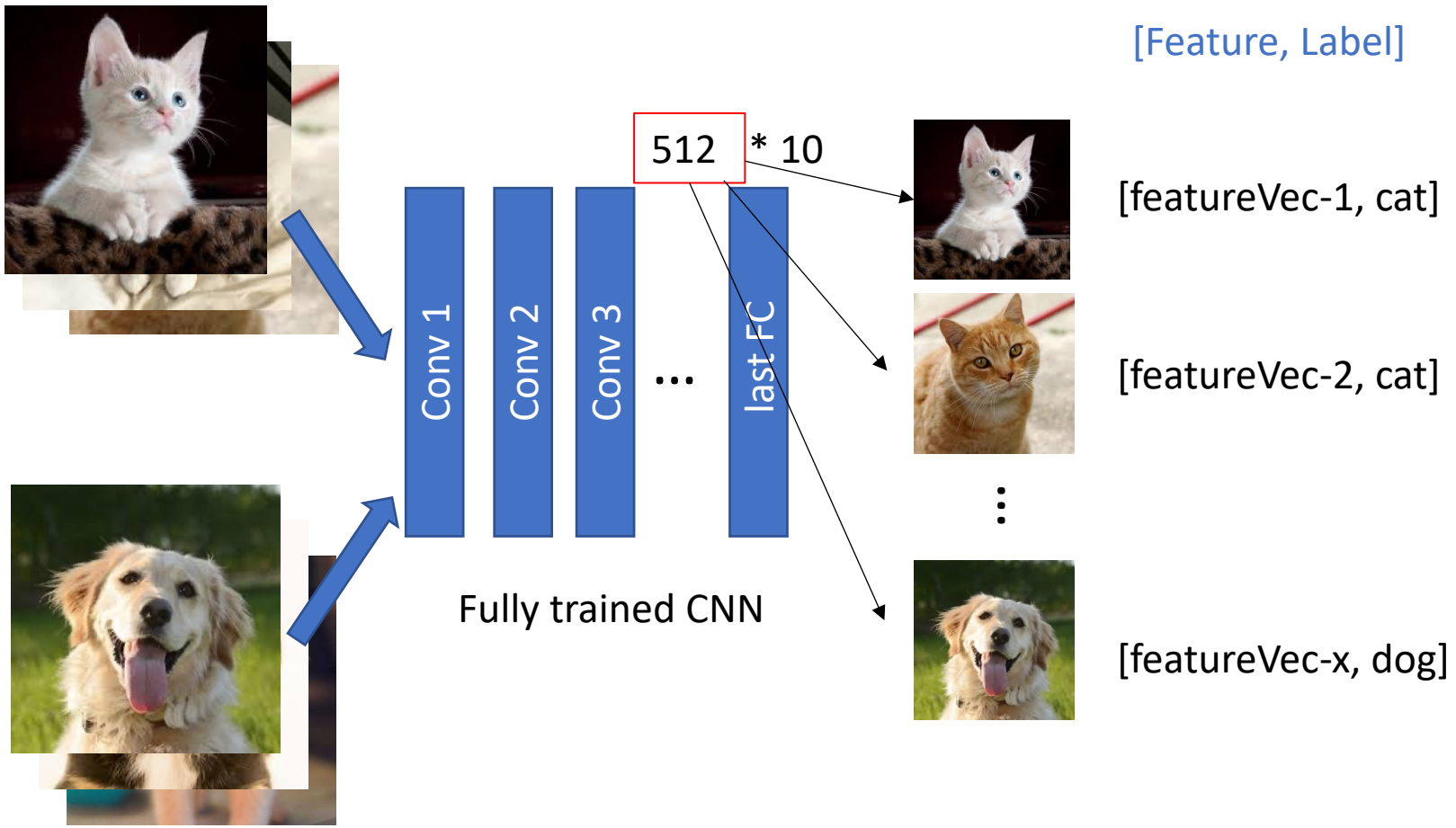
Class-specific Pruning

- Binary Classifiers
- Grouped Classifiers
  - Class grouping



Class-specific Pruning

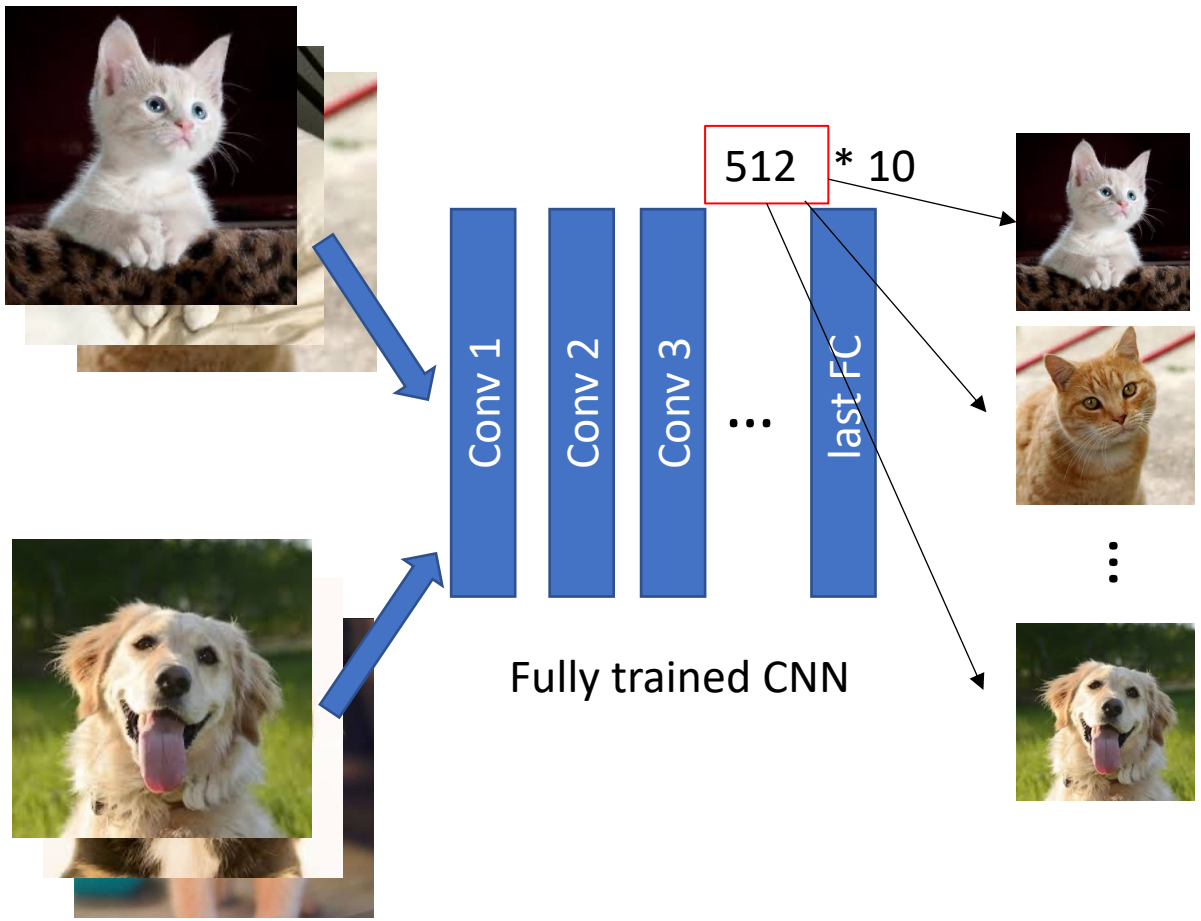
- Binary Classifiers
- Grouped Classifiers
  - Class grouping

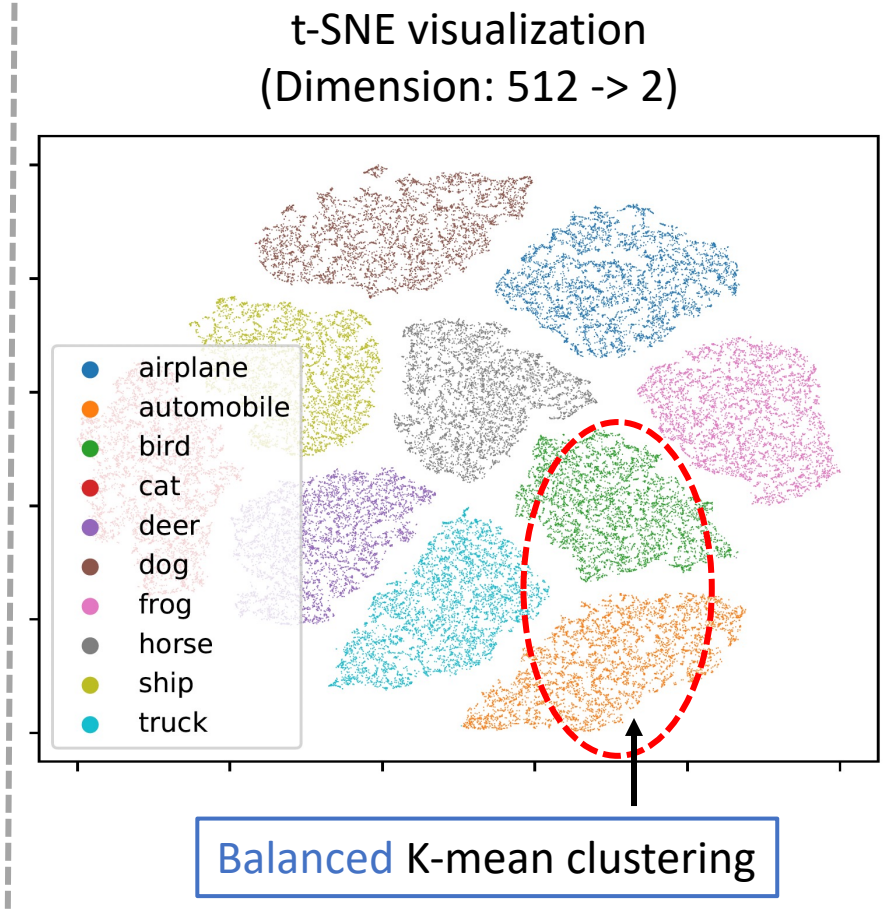


t-SNE visualization
(Dimension: 512 -> 2)

Class-specific Pruning

- Binary Classifiers
- Grouped Classifiers
  - Class grouping



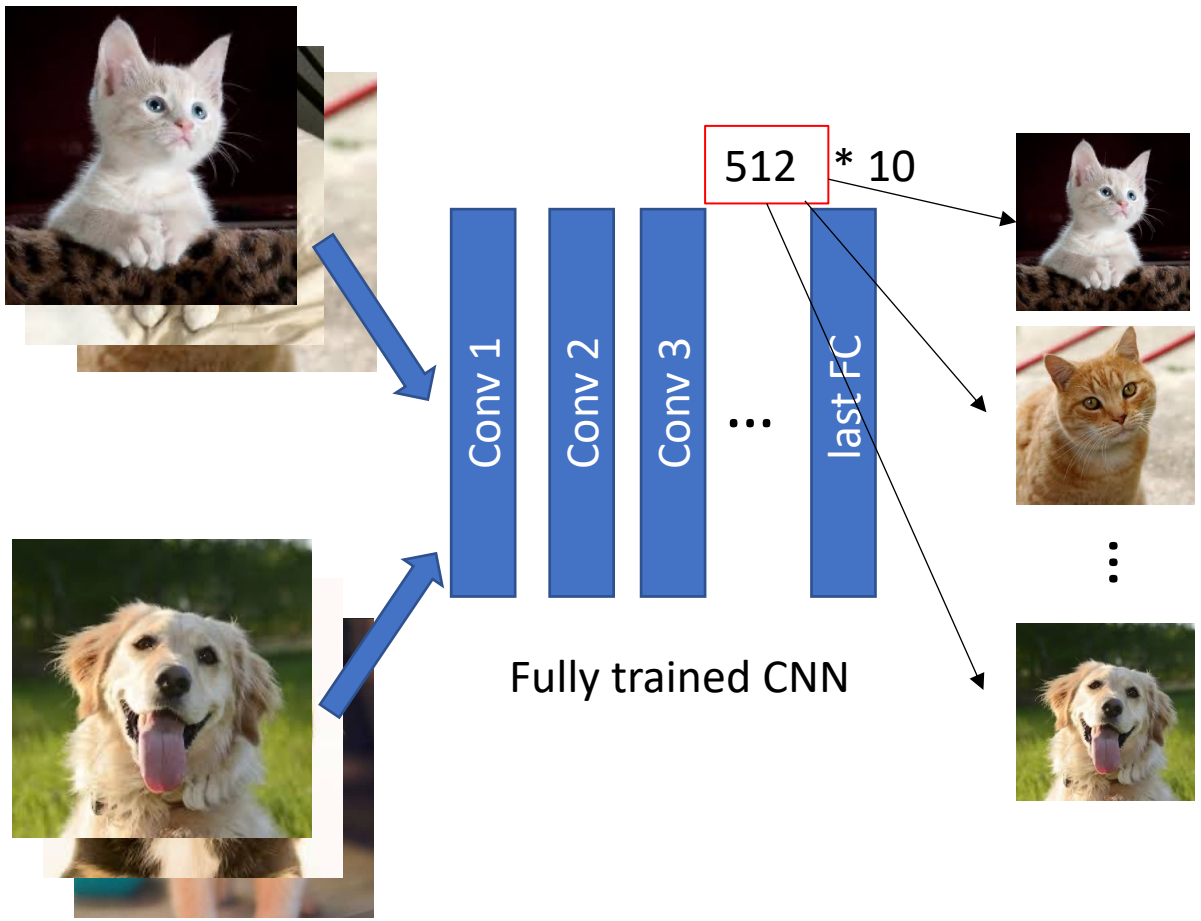


Feature representation

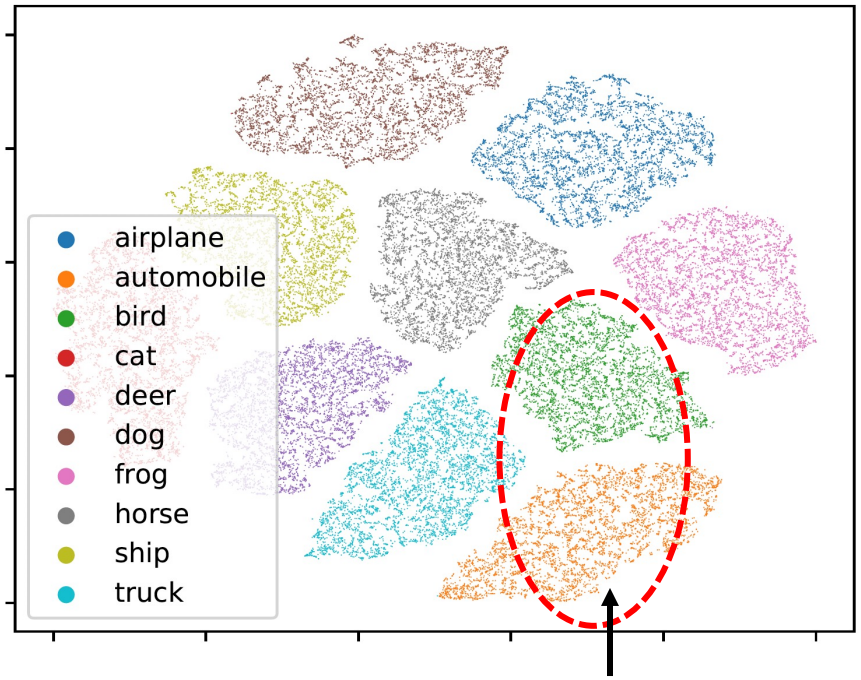
Class grouping

Class-specific Pruning

- Binary Classifiers
- Grouped Classifiers
  - Class grouping



t-SNE visualization
(Dimension: 512 -> 2)



Balanced K-mean clustering

Evenly split task → Similar subnet size

Feature representation

Class grouping

Talk Outline

- Motivation
- Design
  - Class-specific Pruning
  - **Retraining**
  - Combine results back to the original N-way Predictions
- Evaluation
- Extensions

Retraining

- Each binary/grouped classifier is retrained independently.
  (*no communication at all*)

Retraining

- Each binary/grouped classifier is retrained independently.
  (*no communication at all*)

- Balance positive/negative training samples in each training iteration

Retraining

- Each binary/grouped classifier is retrained independently.
  (*no communication at all*)

- Balance positive/negative training samples in each training iteration

- Loss function:
  - Binary classifiers: binary cross-entropy loss
  - Grouped classifiers: multi-way cross-entropy loss

Talk Outline

- Motivation
- Design
  - Class-specific Pruning
  - Retraining
  - Combine results back to the original N-way Predictions
- Evaluation
- Extensions

sensAI workflow

Fully-trained /half-baked CNN

Class-specific pruning

Dog classifier

retrain

Cat classifier

dog?

0.2

cat?

0.6

SoftMax

Decision: Cat

Active neurons

Inactive/pruned neurons

Pruning

Retraining

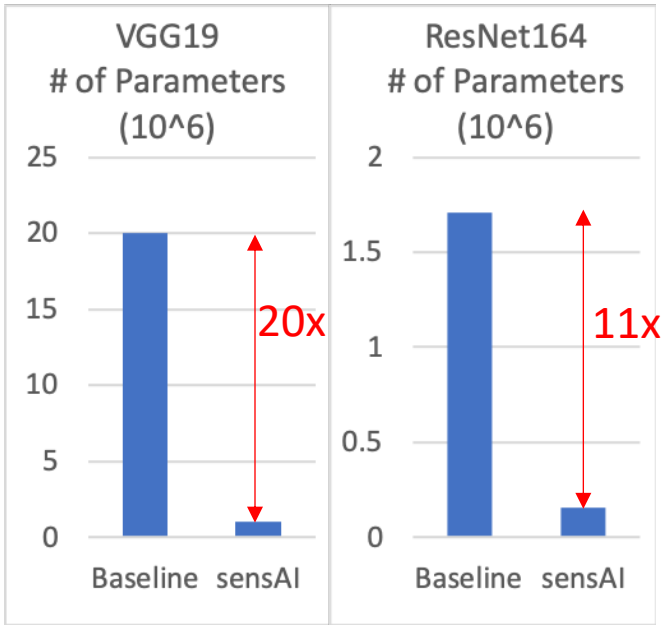
Inference

Talk Outline

- Motivation
- Design
  - Class-specific Pruning
  - Retraining
  - Combine results back to the original N-way Predictions
- **Evaluation**
- Extensions

sensAI v.s. 1-GPU baseline (CIFAR-10)

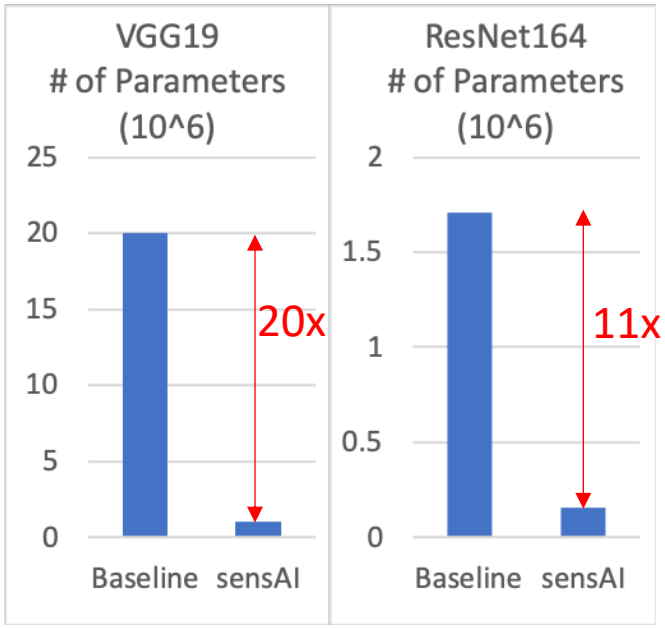
One shot pruning *No accuracy loss*



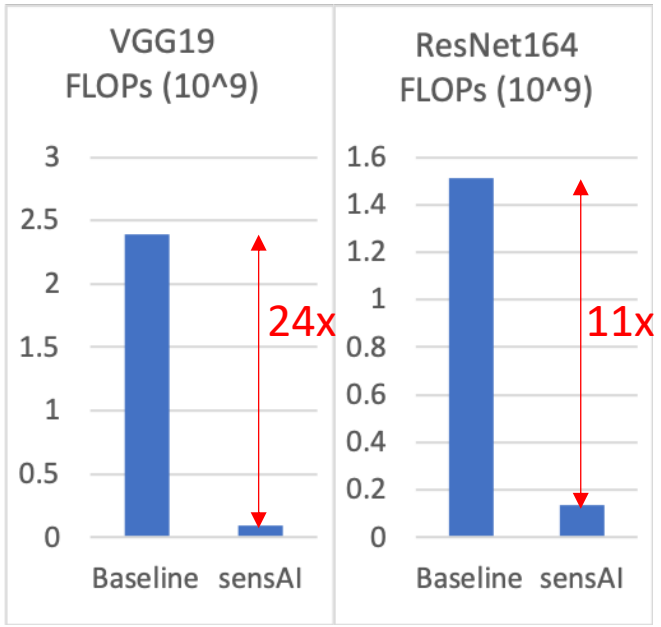
Model Size Reduction

sensAI v.s. 1-GPU baseline (CIFAR-10)

One shot pruning *No accuracy loss*



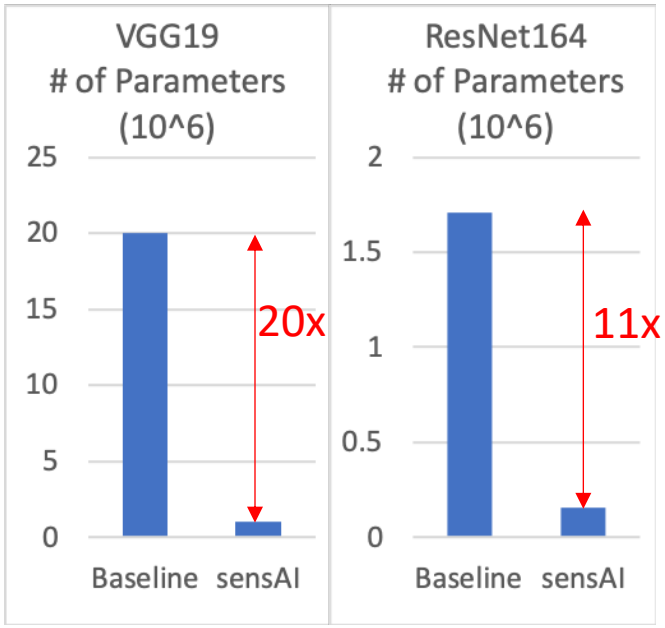
Model Size Reduction



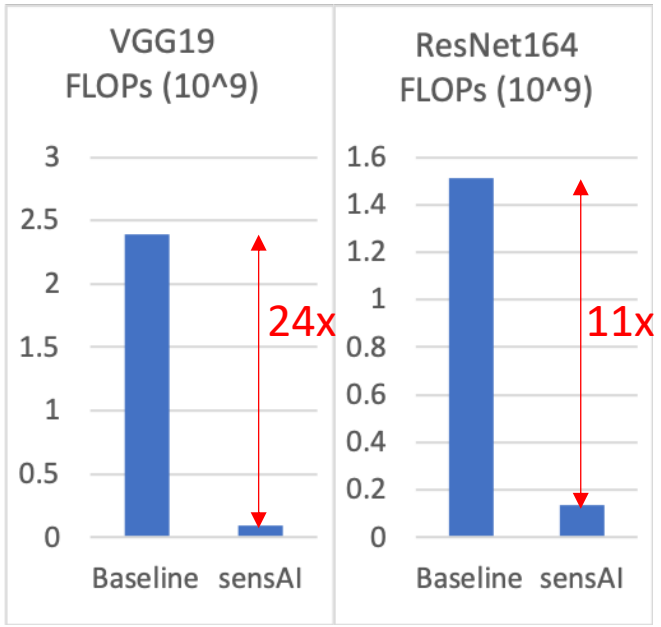
FLOPs Reduction

sensAI v.s. 1-GPU baseline (CIFAR-10)

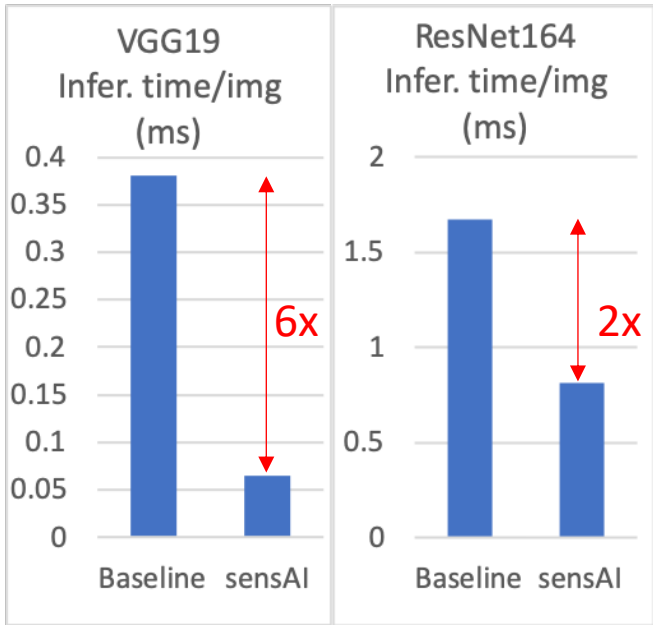
One shot pruning *No accuracy loss*



Model Size Reduction



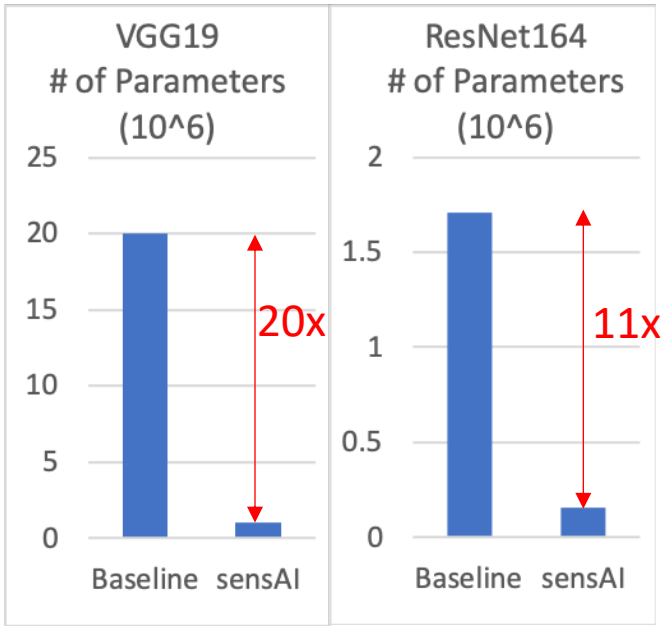
FLOPs Reduction



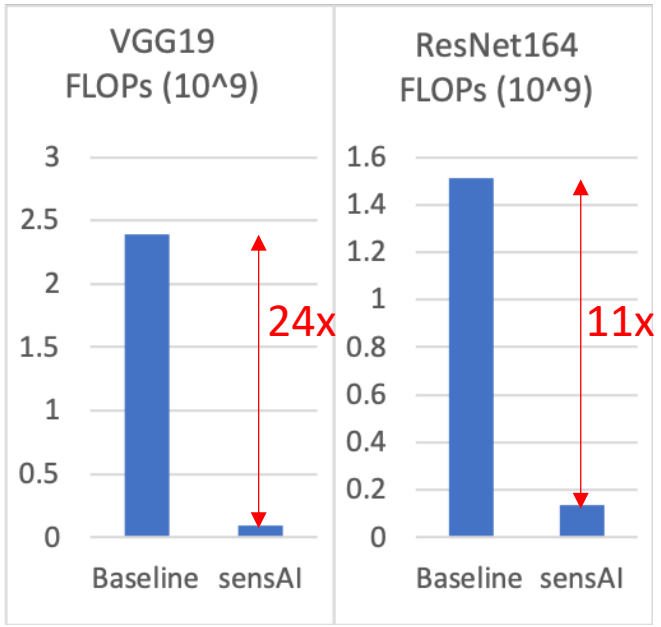
Inference Time Reduction

sensAI v.s. 1-GPU baseline (CIFAR-10)

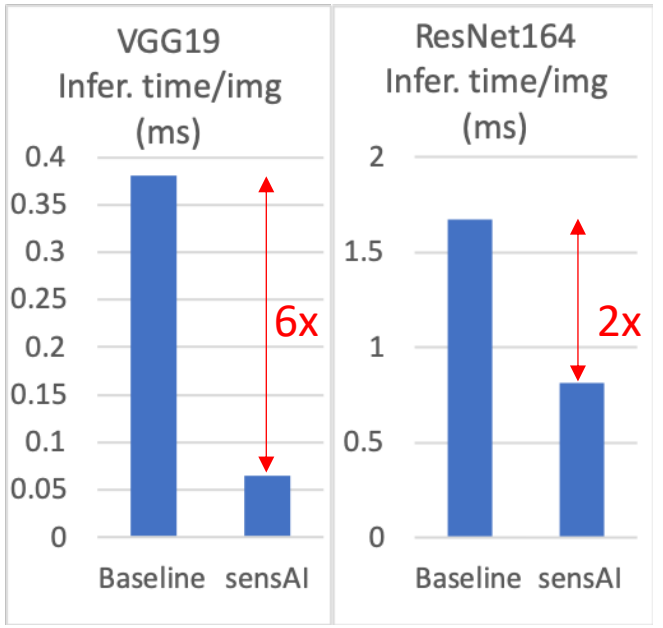
One shot pruning *No accuracy loss*



Model Size Reduction



FLOPs Reduction

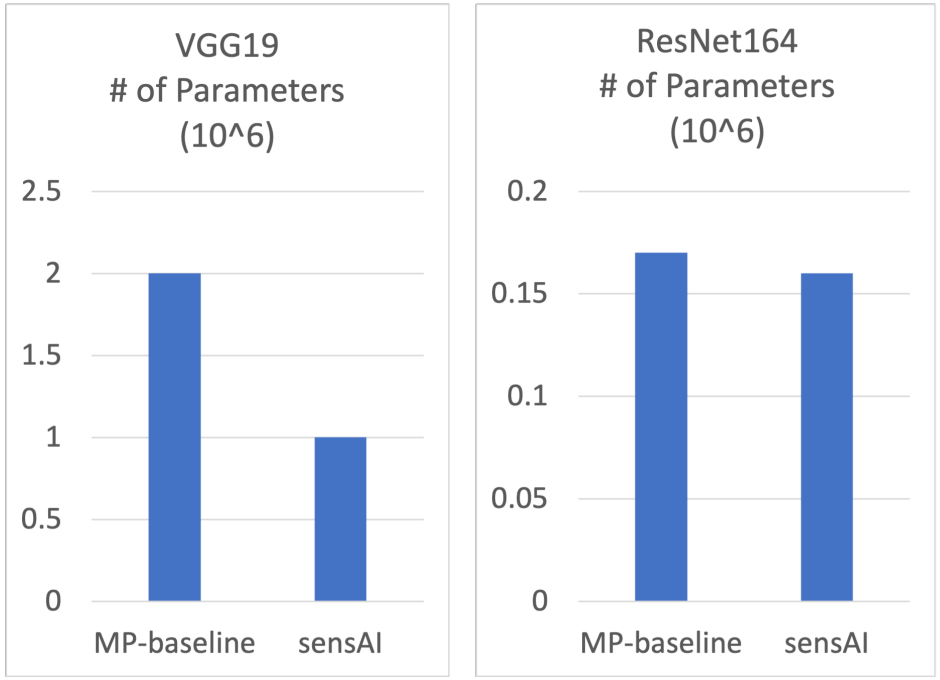


Inference Time Reduction

Why can we prune so much?
We simplify the task.

sensAI v.s. Model-Parallel baseline (CIFAR-10)

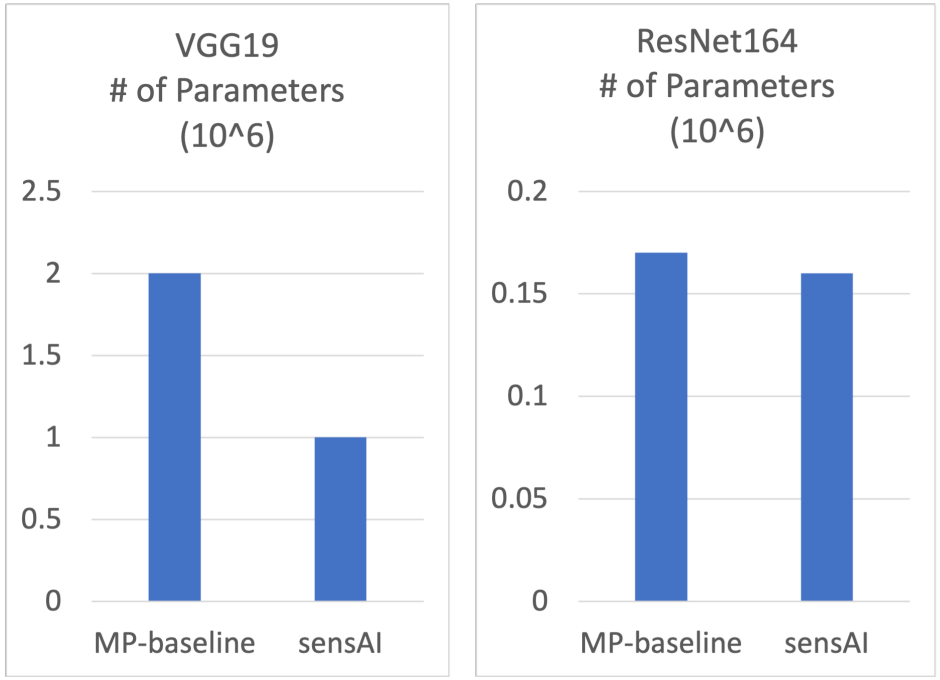
10-GPU *One shot pruning* *No accuracy loss*



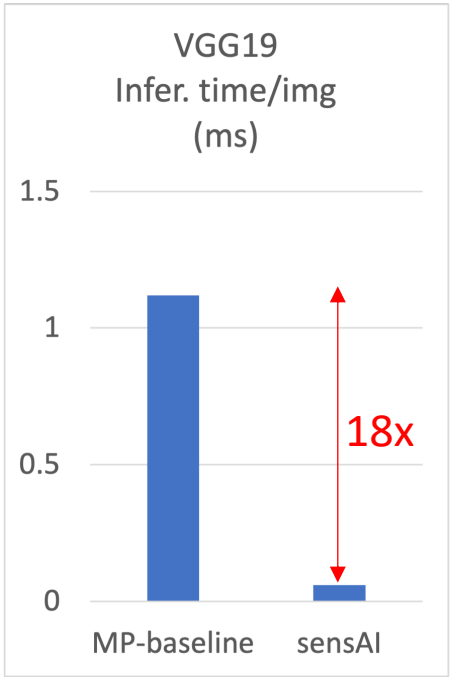
Similar per-GPU model size

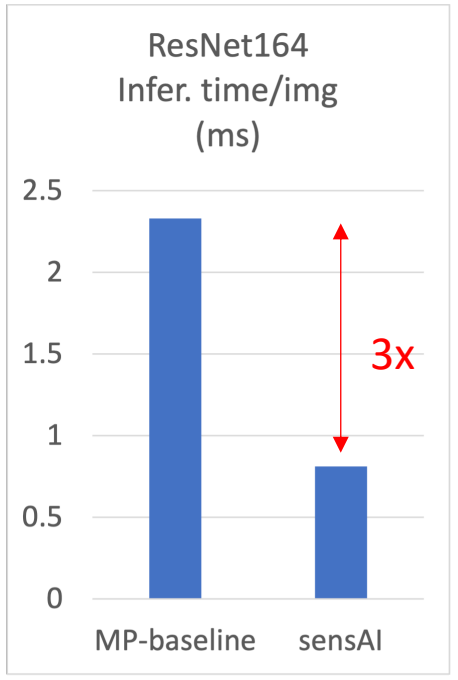
sensAI v.s. Model-Parallel baseline (CIFAR-10)

10-GPU *One shot pruning* *No accuracy loss*



Similar per-GPU model size

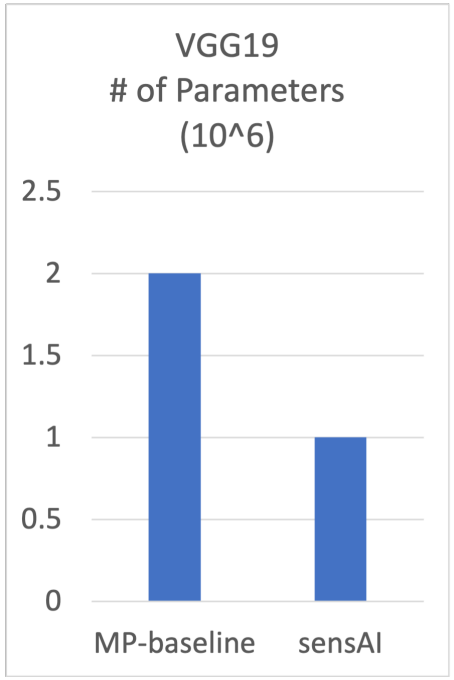


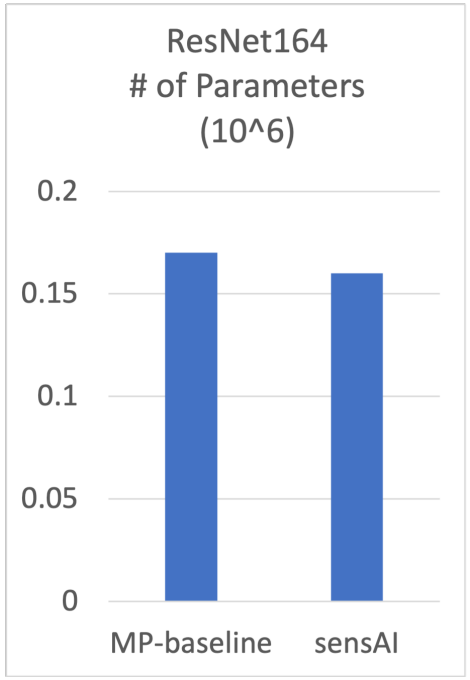


Inference Time Reduction

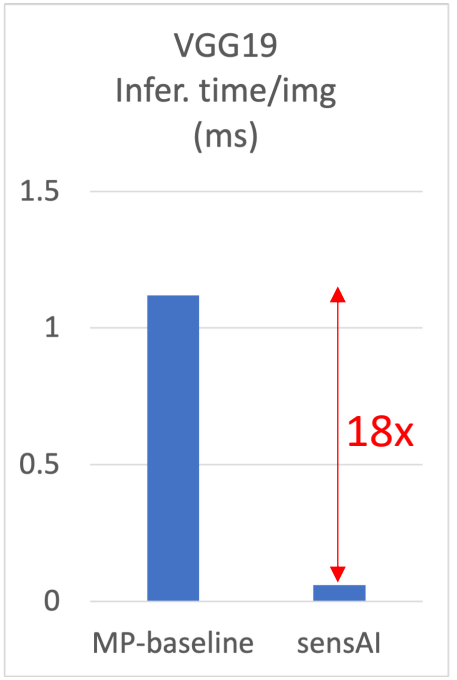
sensAI v.s. Model-Parallel baseline (CIFAR-10)

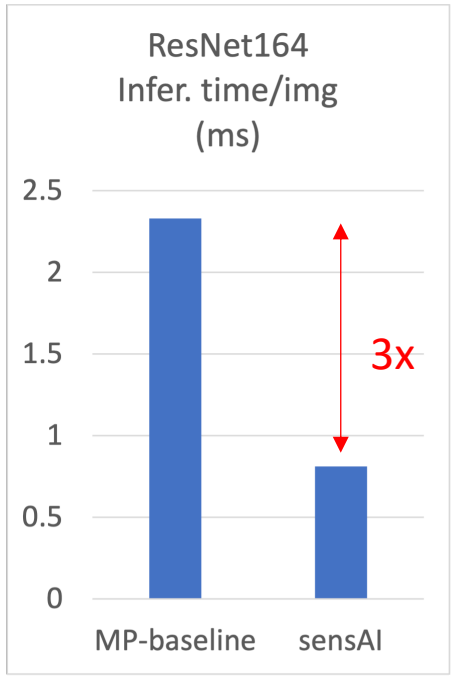
10-GPU *One shot pruning* *No accuracy loss*





Similar per-GPU model size

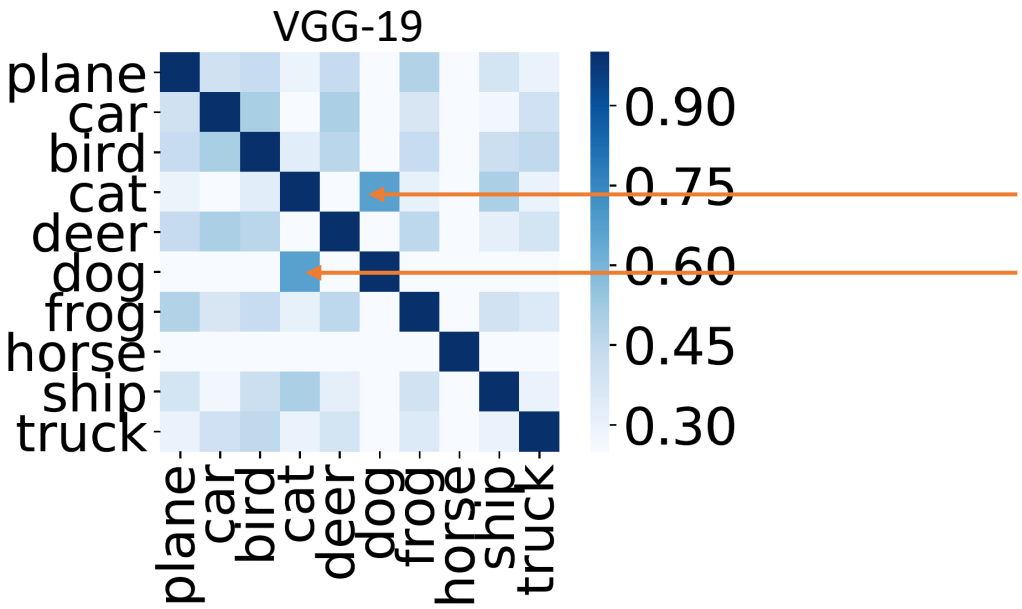




Inference Time Reduction

Why big difference in inference time per image?
sensAI removes all possible communication and synchronization barriers.

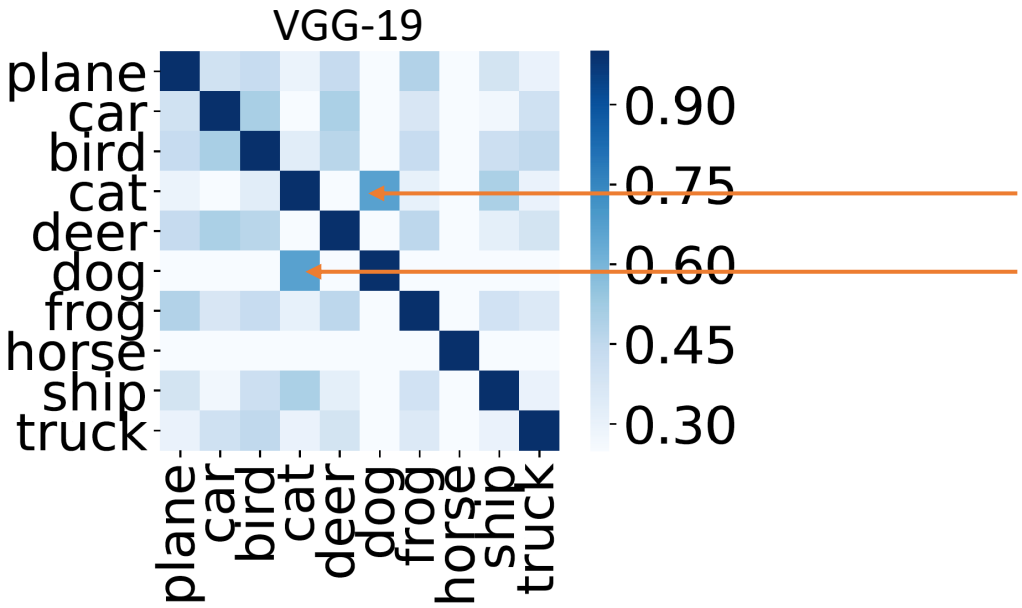
sensAI Binary Model Analysis (CIFAR-10)



Cat and Dog binary classifiers share most channels among all pairs.

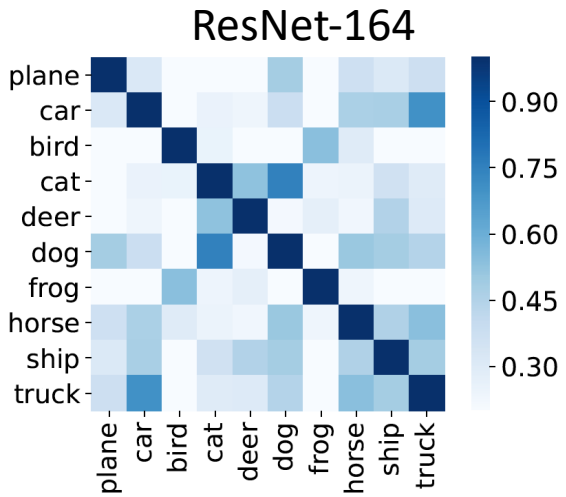
Intersection of Unions (IoU) of the channel indices

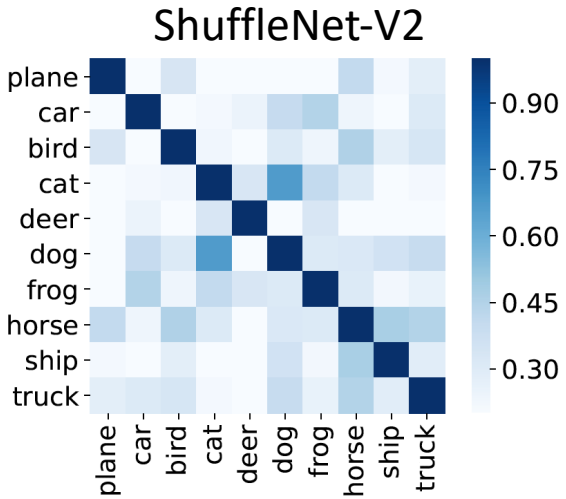
sensAI Binary Model Analysis (CIFAR-10)

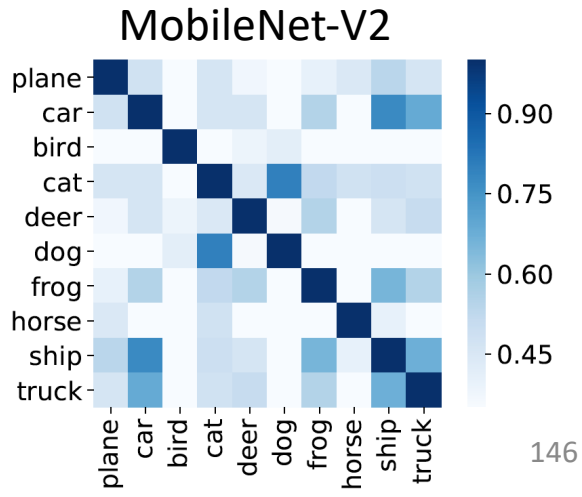


Cat and Dog binary classifiers share most channels among all pairs.

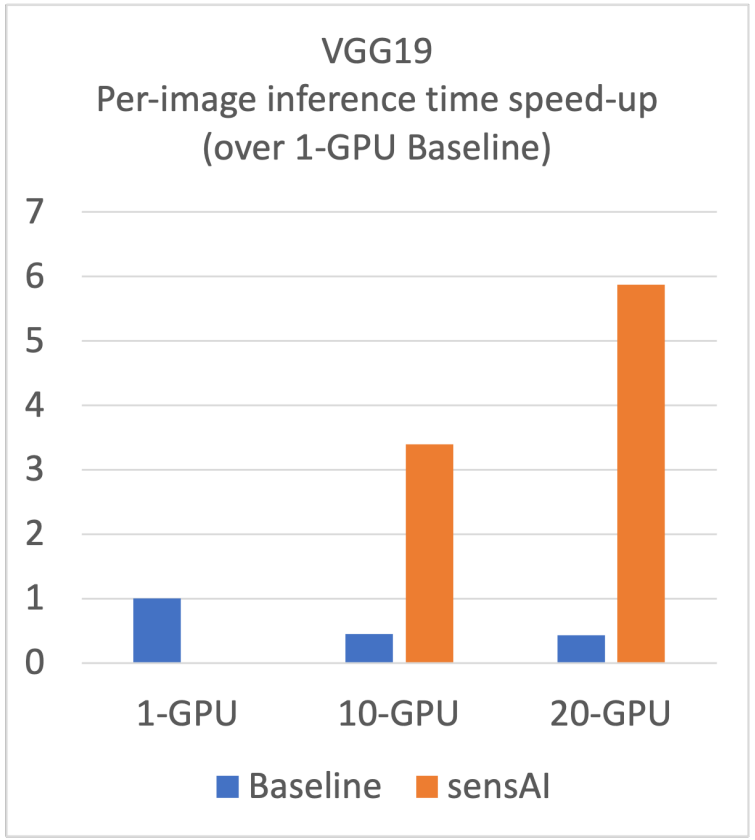
Intersection of Unions (IoU) of the channel indices





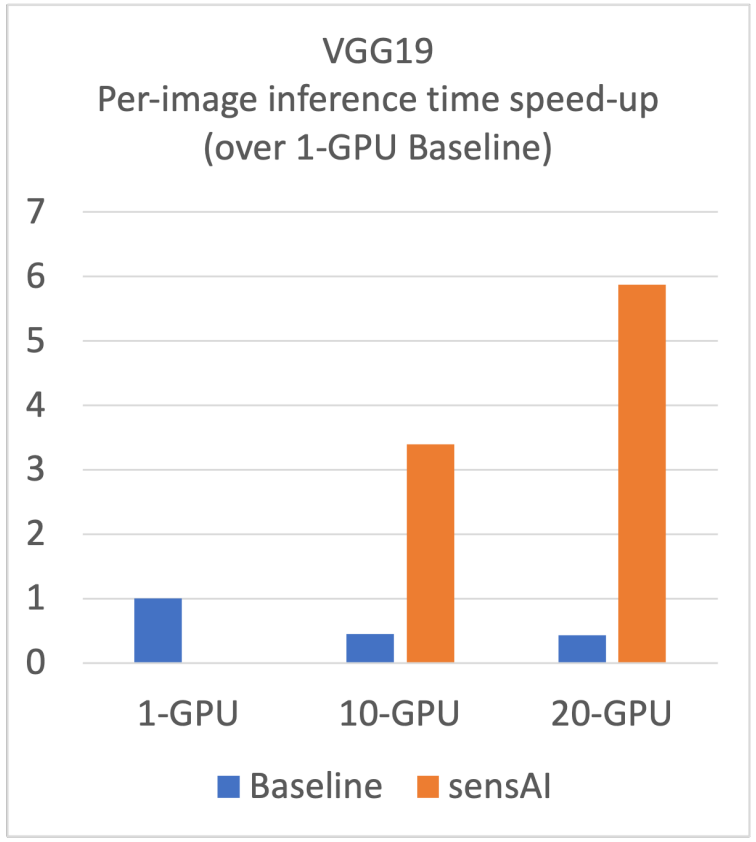


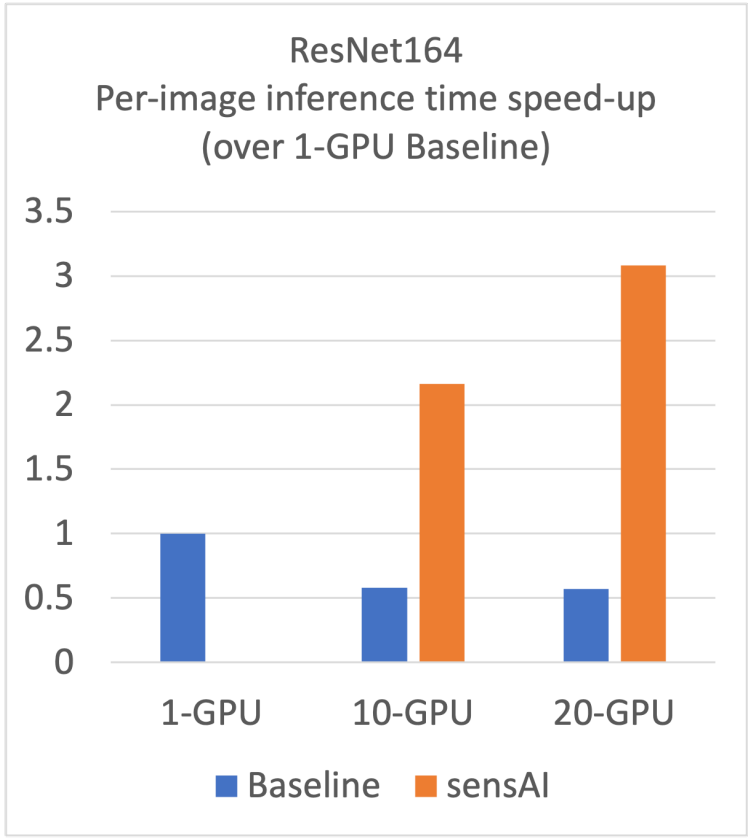
ImageNet-1K Results



sensAI speed-up results over 1-GPU baseline and 10/20-GPU MP-baseline

ImageNet-1K Results

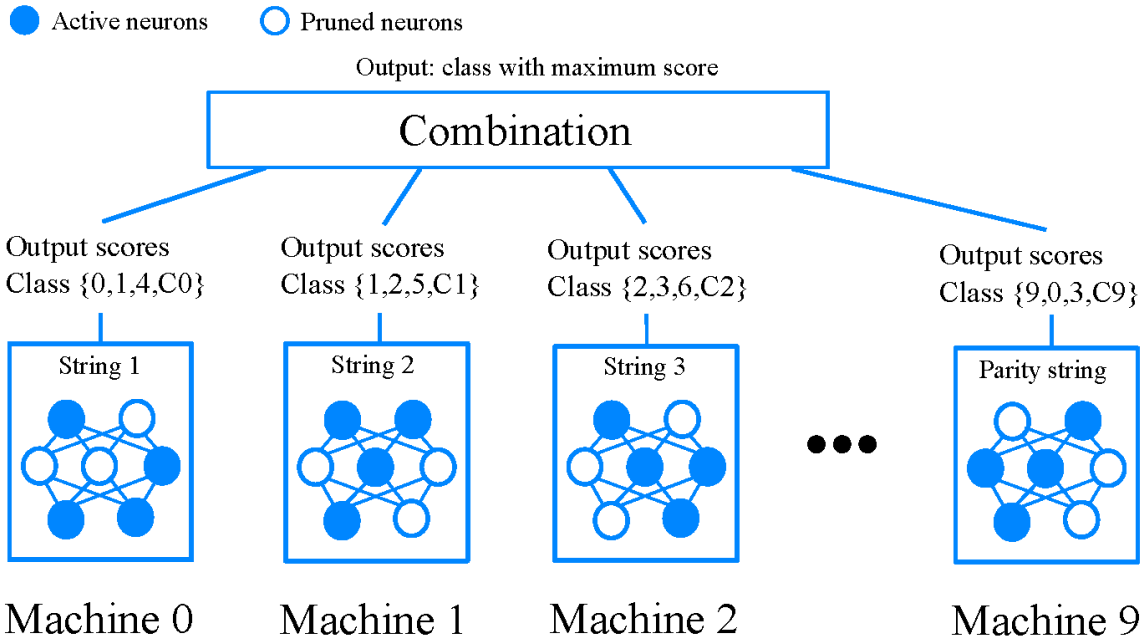




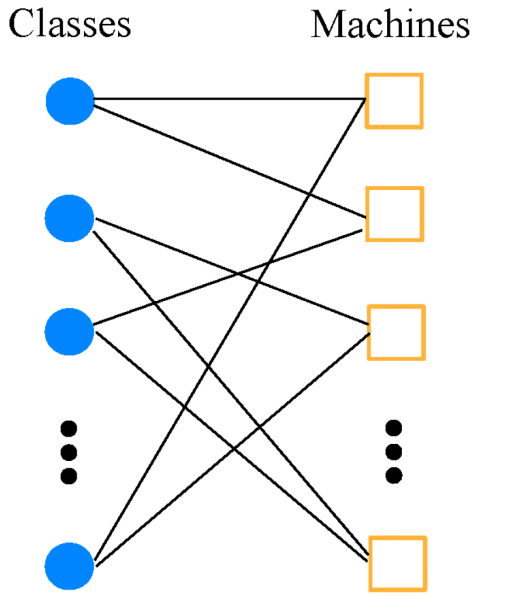
sensAI speed-up results over 1-GPU baseline and 10/20-GPU MP-baseline

sensAI Extensions

Robust Class Parallelism with Cyclic Coding*



Overlapping class coverage
on different machine/GPU



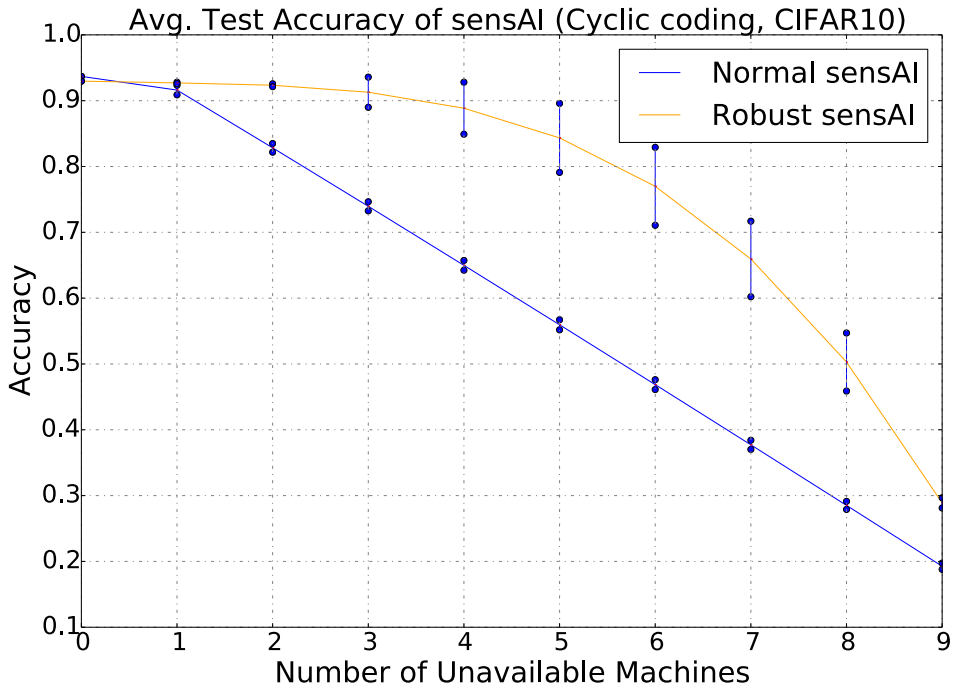
The class-machine assignment is a bipartite graph.

- One example:
- Cyclic class assignment
  - (0,1,3) -> Machine 1
  - (1,2,4) -> Machine 2
  - (2,3,5) -> Machine 3
  - (9,0,2) -> Machine 10

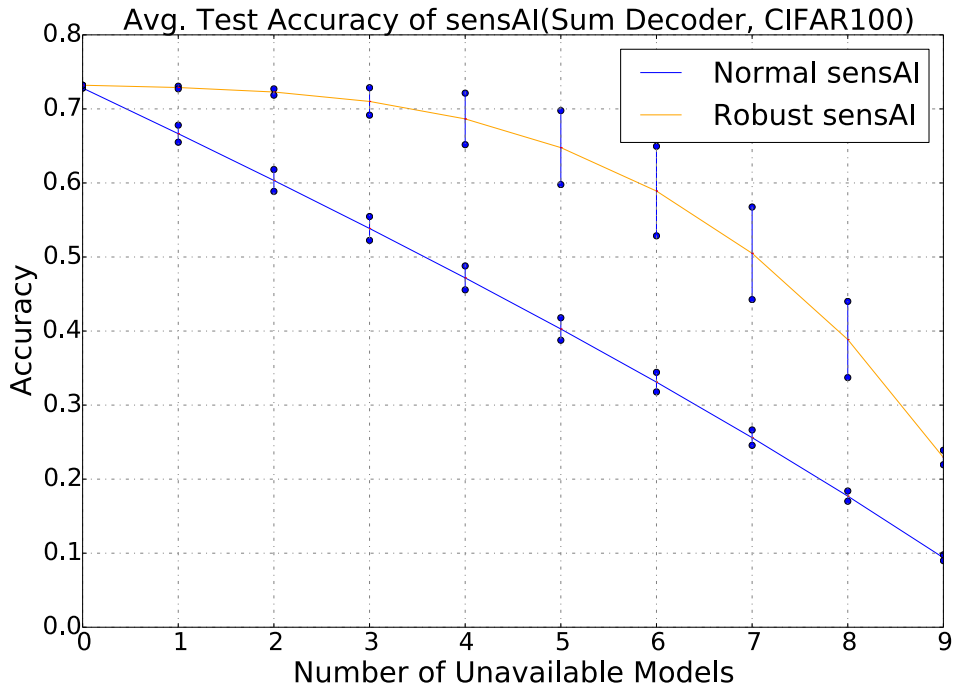
Cyclic coding
(redundancy ratio 3 → major vote)

*Robust Class Parallelism - Error Resilient Parallel Inference with Low Communication Cost, Asilomar 2020

CIFAR-10/100 Results*



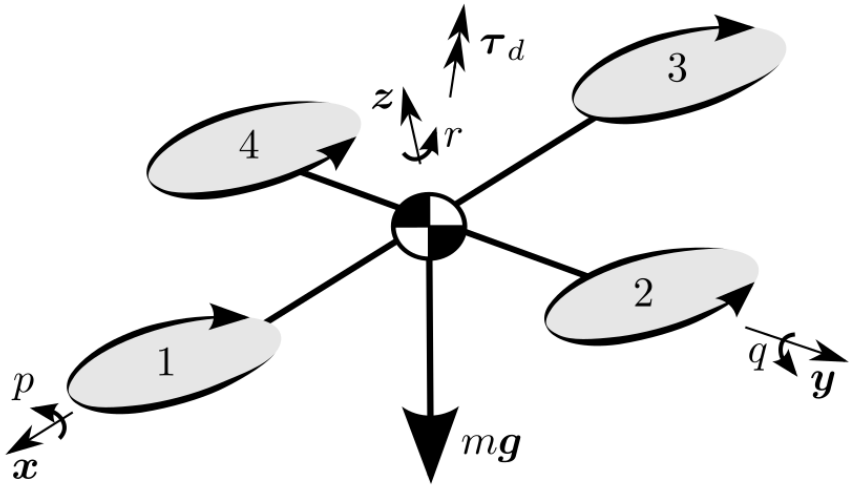
Binary classifiers
ResNet-110, CIFAR 10



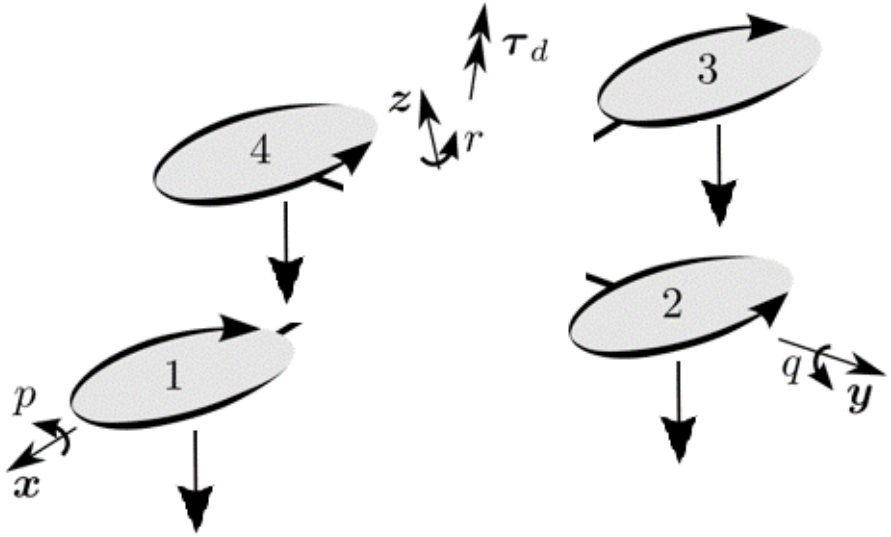
10-class grouped classifiers
ResNet-110, CIFAR 100

*Robust Class Parallelism - Error Resilient Parallel Inference with Low Communication Cost, Asilomar 2020

Control Systems in Drone



Centralized Control Systems in Quadcopter



sensAI-RL*

*Composing MPC with LQR and Neural Networks for Efficient and Stable Control, arXiv:2112.07238

RNN and Transformer

- RNN
  - Hsin-Li Chu (hsinli.chu@berkeley.edu), Siming Liu, Hung-Chun Chen
- Transformer
  - Gurkarn Goindi (gurkarn.goindi@berkeley.edu), Matt Zhou, Abel Yagubyan
- Vision Transformer
  - Zeyang Bao (zeyang_bao@berkeley.edu), Russell Wang, Zuang Yu

sensAI Summary

- Real-time model serving on live data is important.
- Neither data parallelism nor model parallelism can reduce model serving latency on single input piece.
- sensAI's class parallelism is a generic approach to reduce model serving latency on live data, with up to 18x time reduction on CIFAR-10 and 6x speed-up on ImageNet-1K.
- sensAI approach is quite general and should be able to apply to other domains like NLP, RL.
- sensAI is an open-source project, available at https://github.com/GuanhuaWang/sensAI

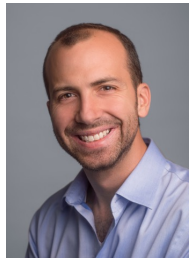
Team members

sensAI

riselab UC Berkeley

- Advisors:


Ion Stoica

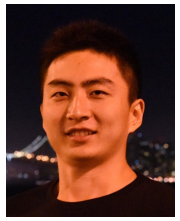

Joseph Gonzalez

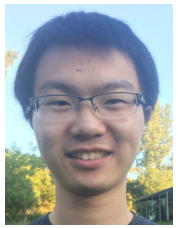

Kannan Ramchandran


Alexandre Bayen


Trevor Darrell

- PhDs:

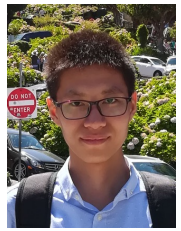

Guanhua Wang


Zhuang Liu


Yaoqing Yang


Jichan Chung

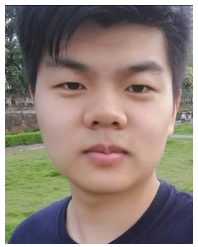

Fangyu Wu

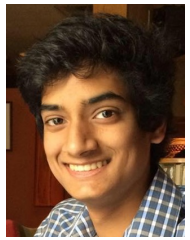

Siyuan Zhuang


Vipul Gupta

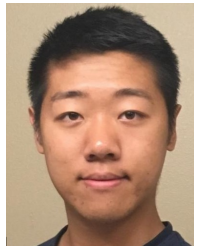
- Masters & Undergrads:

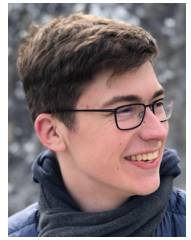

Brandon Hsieh

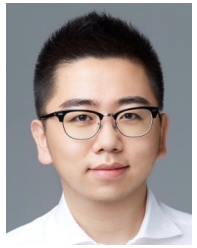

Kenan Jiang


Adarsh Karnati


Yingxin Kang


Kehan Wang


Hank O'Brien

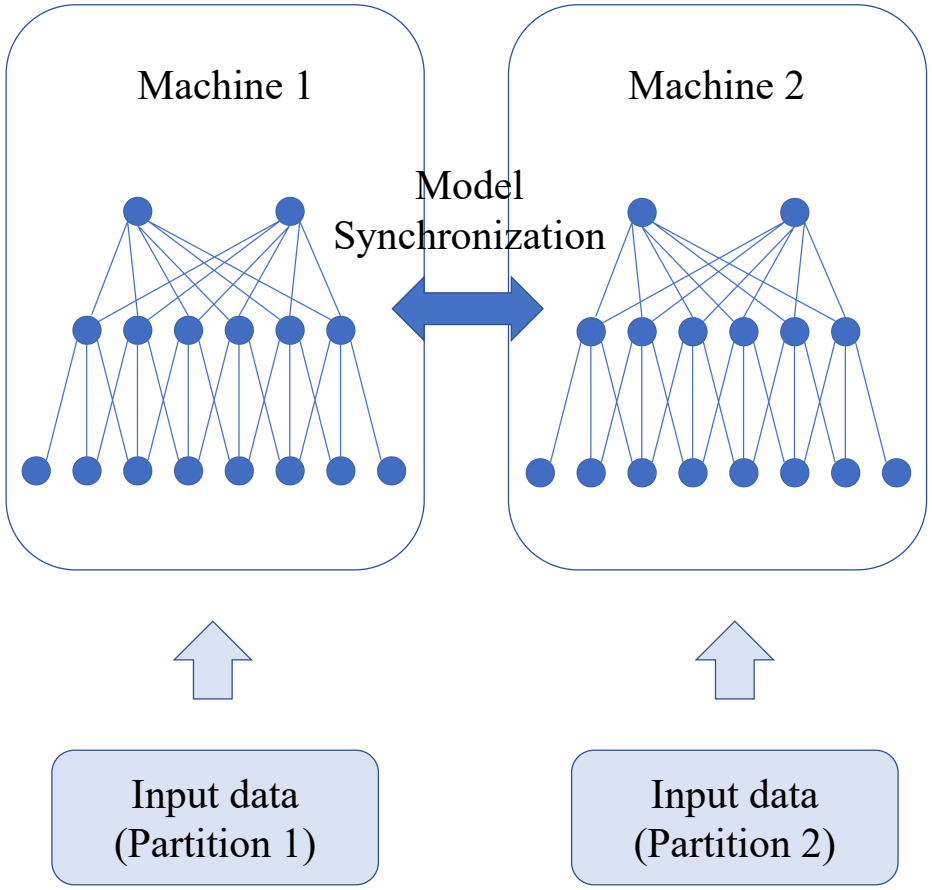

Zihao Fan

Wavelet

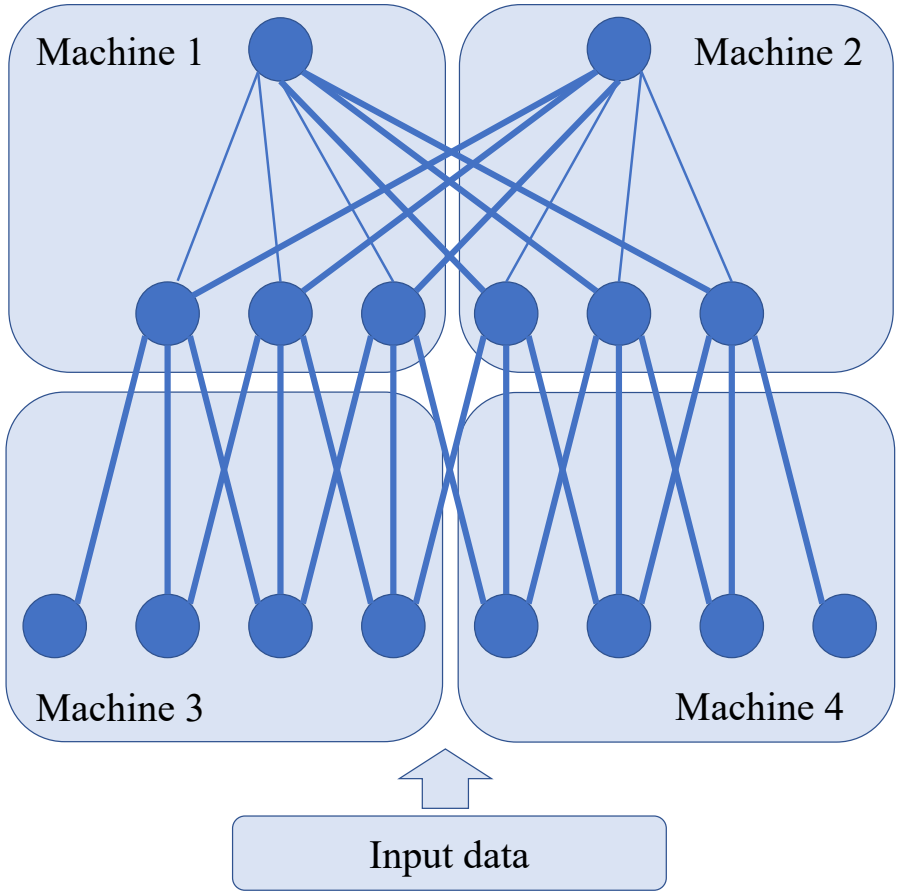
Efficient DNN Training with Tick-Tock Scheduling

MLSys 2021

Speed-up DNN training: Data and Model Parallelism

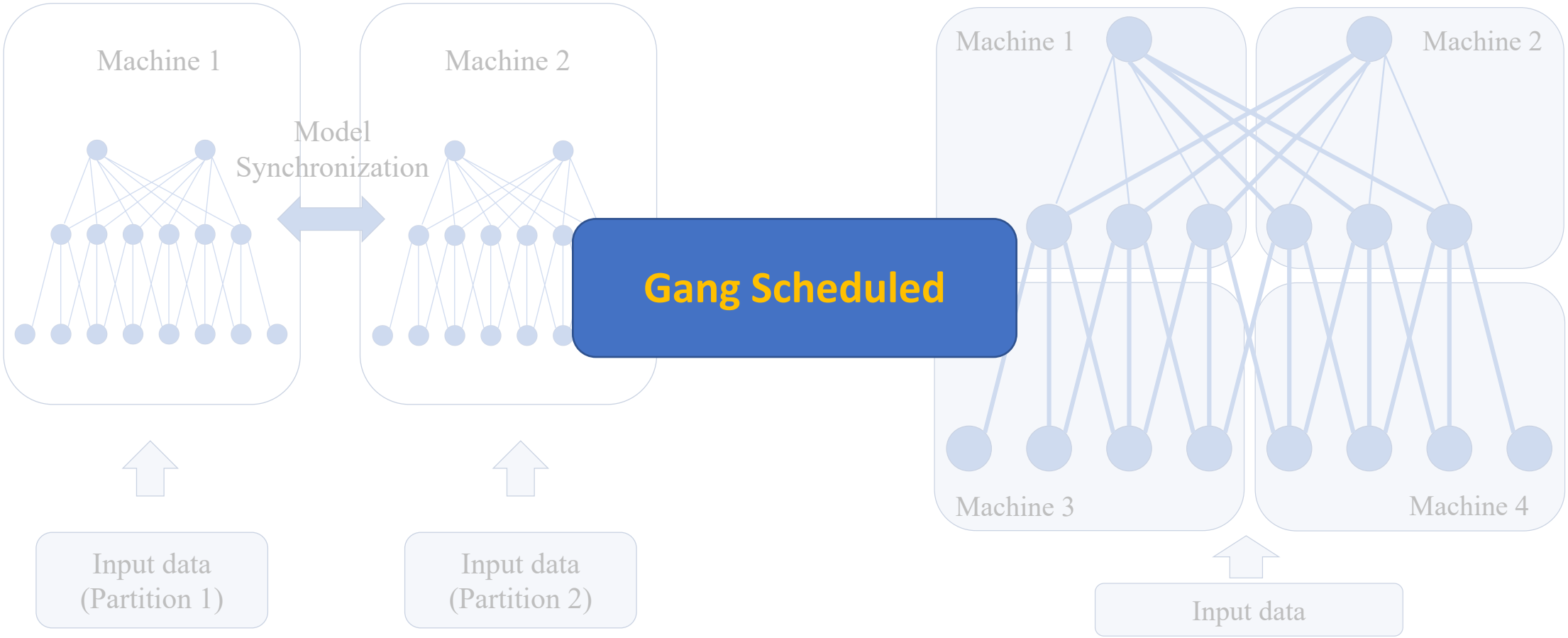


Data Parallelism



Model Parallelism

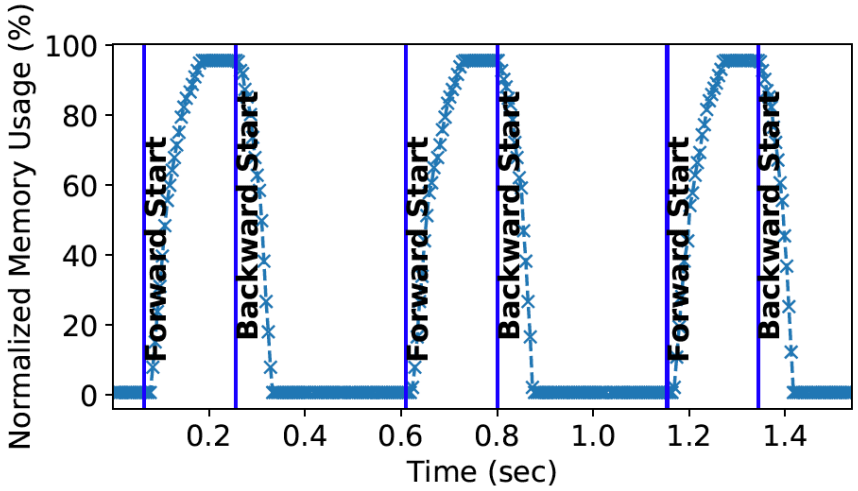
Speed-up DNN training: Data and Model Parallelism

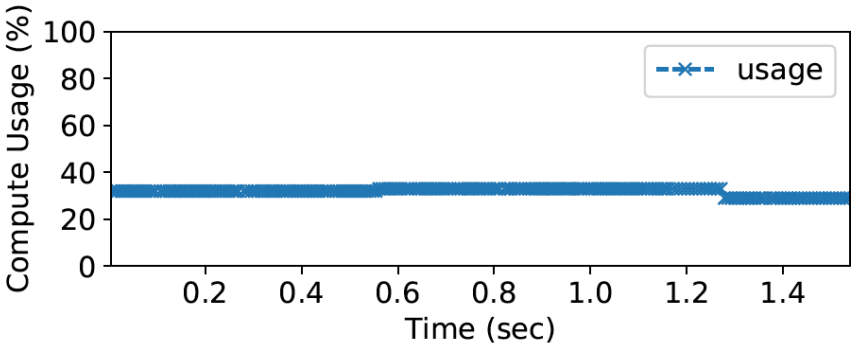


Data Parallelism

Model Parallelism

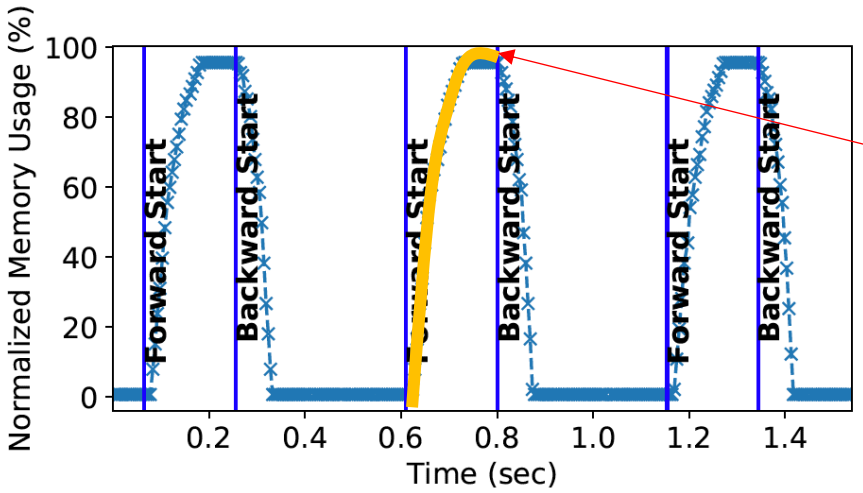
Resource utilization of gang-scheduled tasks



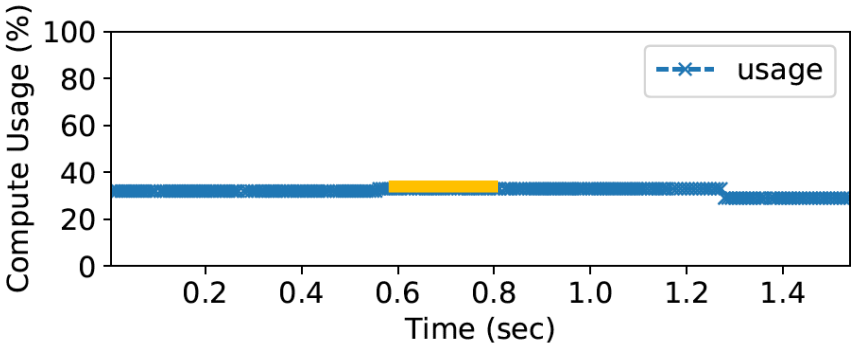


Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Resource utilization of gang-scheduled tasks

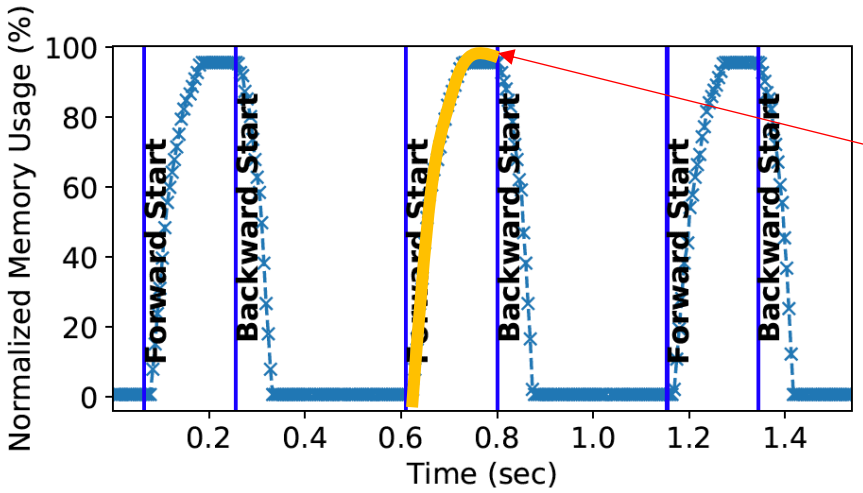


Computation is memory-bounded.

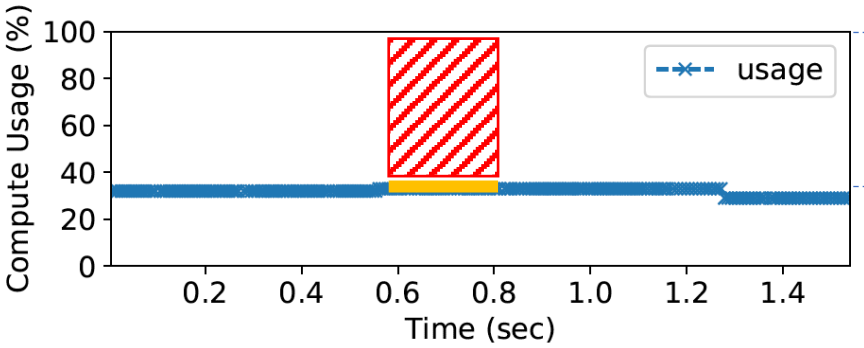


Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Resource utilization of gang-scheduled tasks



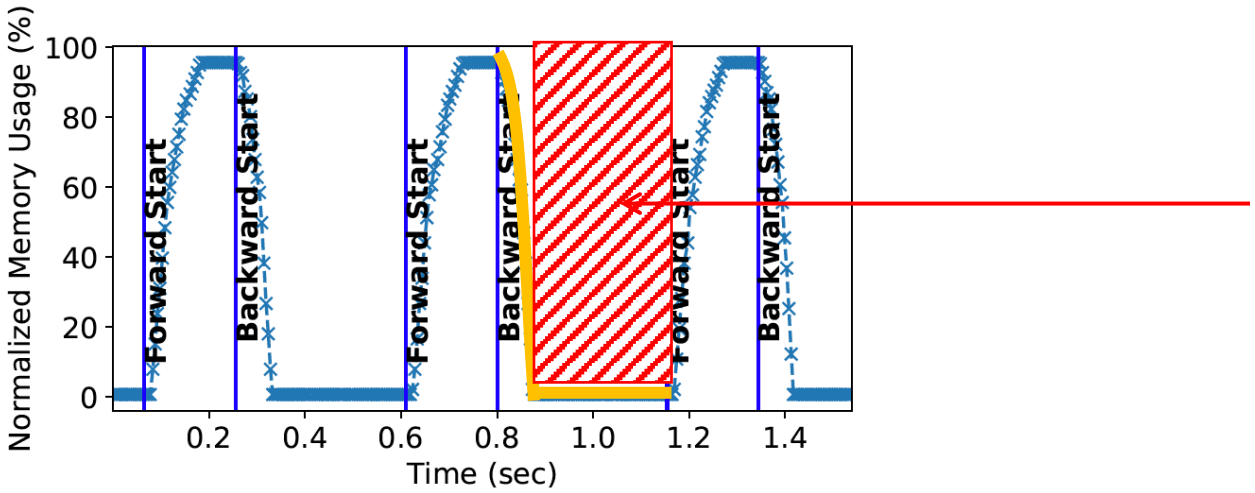
Computation is memory-bounded.



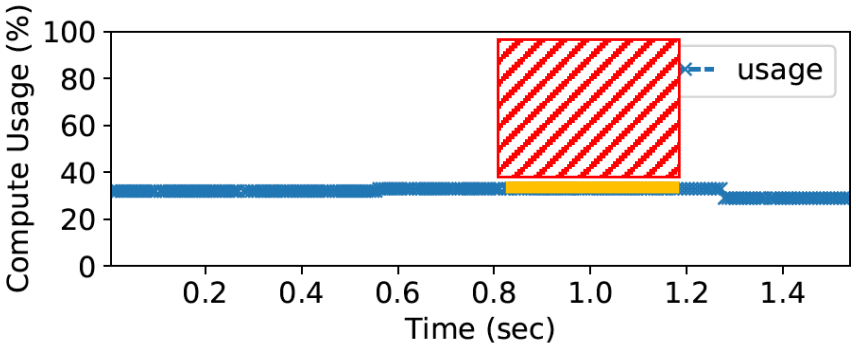
~60% compute cores are under-utilized.

Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Resource utilization of gang-scheduled tasks

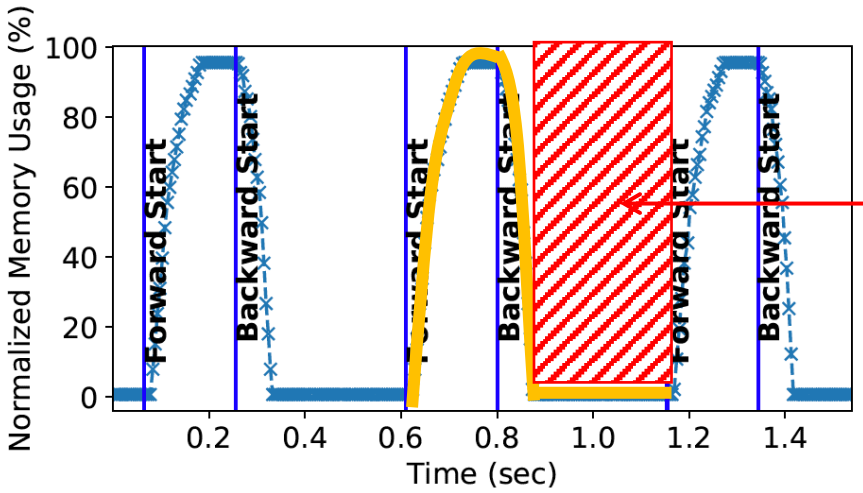


GPU memory is also under-utilized.



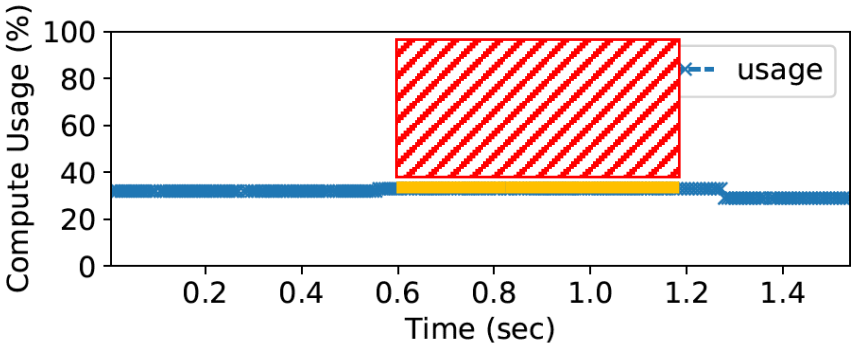
Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Resource utilization of gang-scheduled tasks



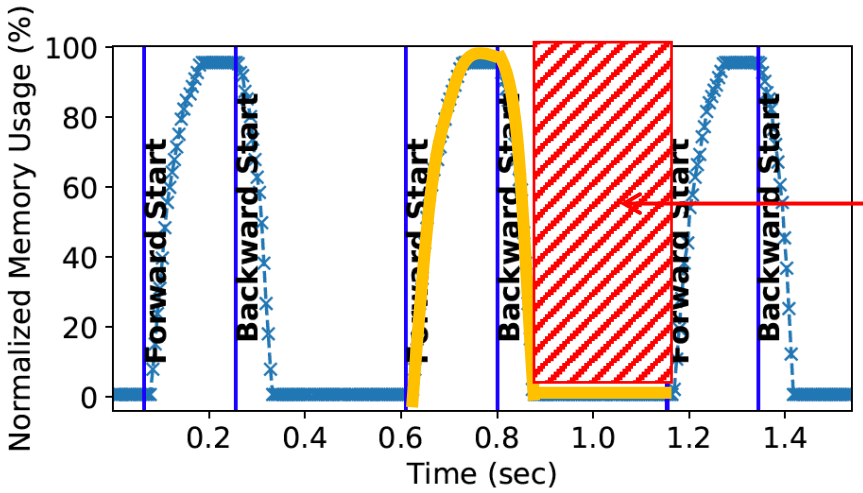
GPU memory is also under-utilized.

Gang-scheduling guarantees all tasks reach memory peak and valley at the same time.



Data parallel training
(2V100, ImageNet-1K, ResNet-56)

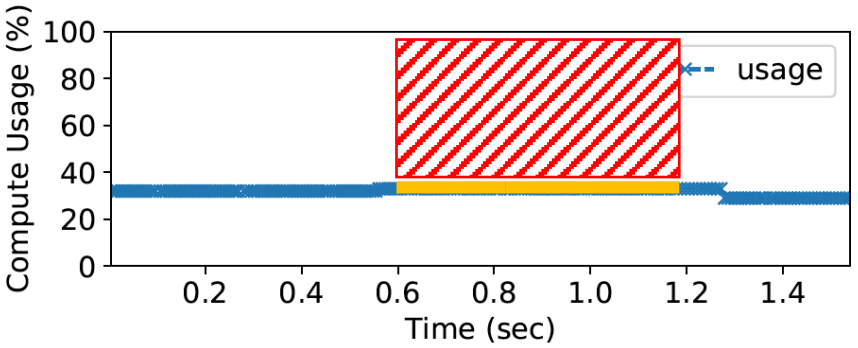
Resource utilization of gang-scheduled tasks



GPU memory is also under-utilized.

Gang-scheduling guarantees all tasks reach memory peak and valley at the same time.

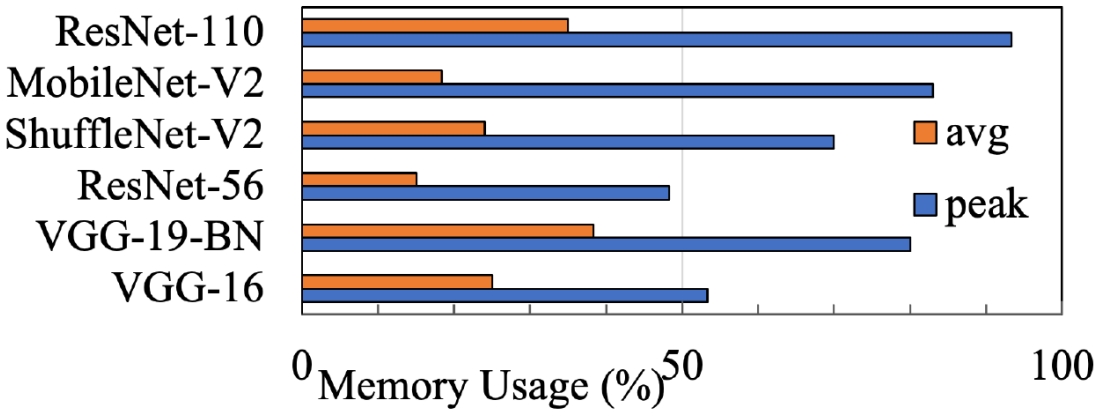
Leaving memory valley period wasted during backward propagation.



Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Resource utilization of gang-scheduled tasks

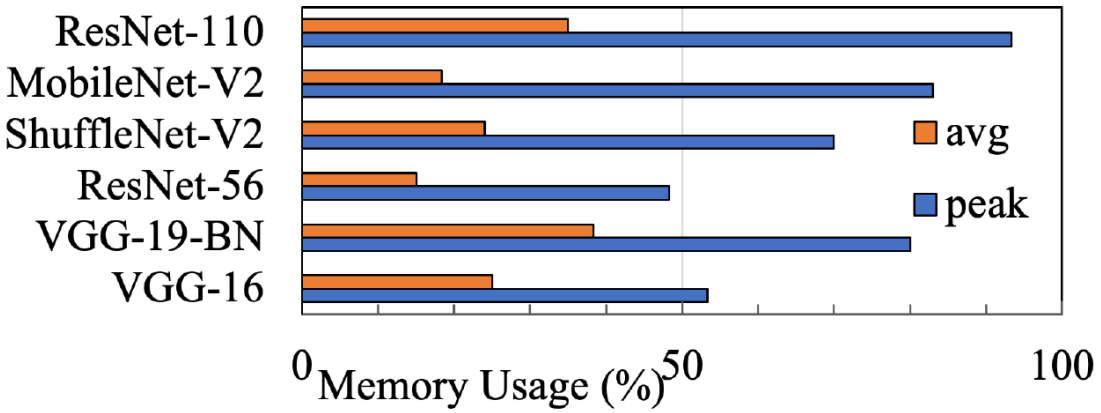
Data parallel training



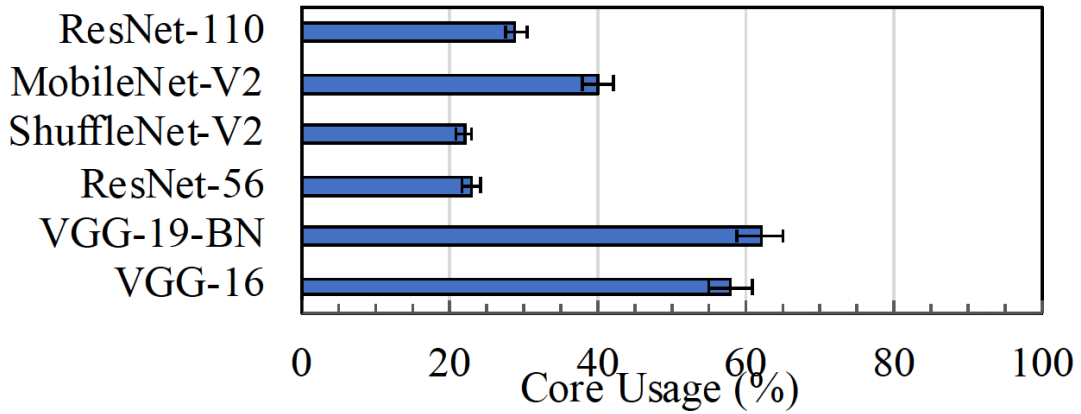
Average on-device memory usage (~30%)

Resource utilization of gang-scheduled tasks

Data parallel training

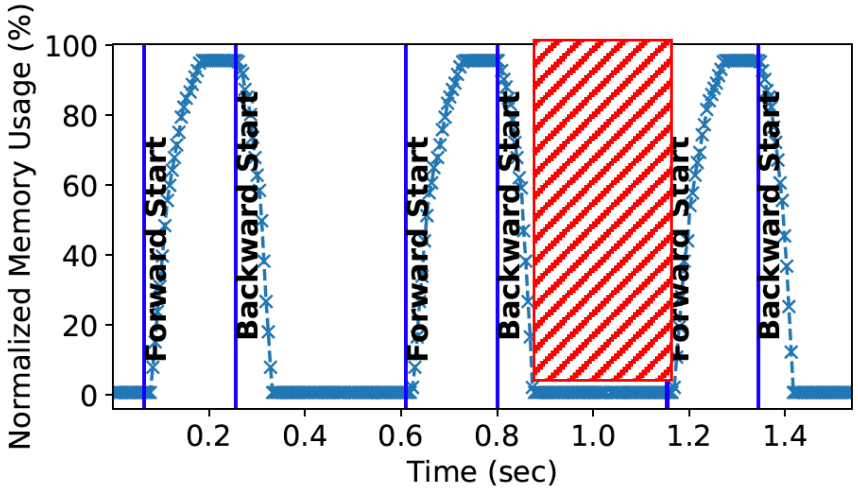


Average on-device memory usage (~30%)

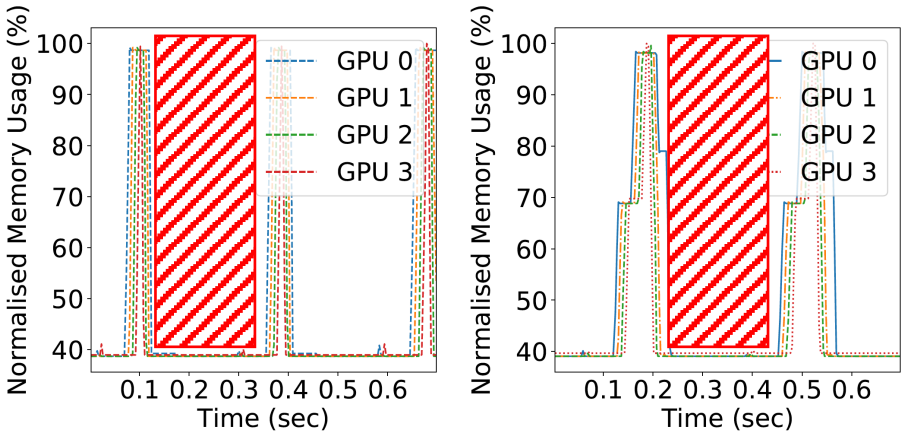


Average computation utilization (~40%)

Resource utilization of gang-scheduled tasks



Memory Valley



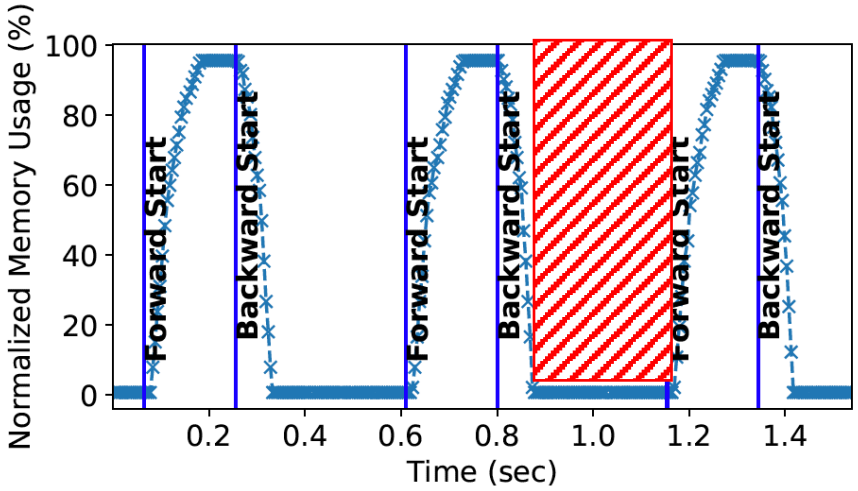
(a) no pipelining

(b) with pipelining

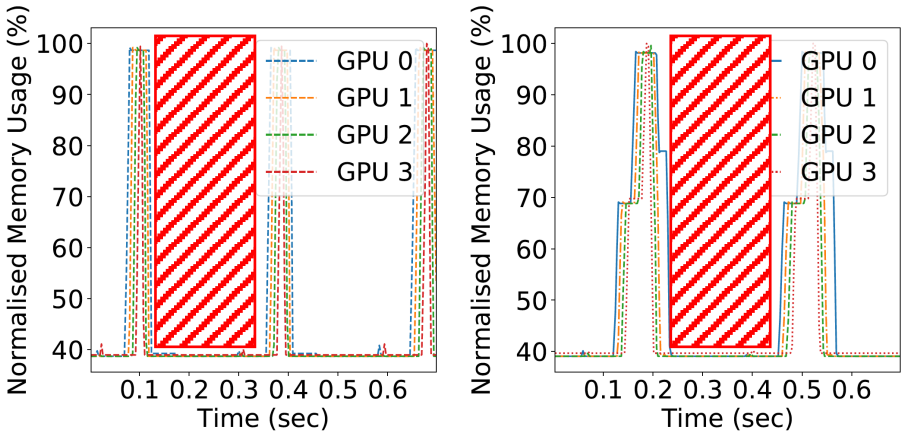
Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Model parallel training
(4V100, BERT, w/wo pipeline parallelism)

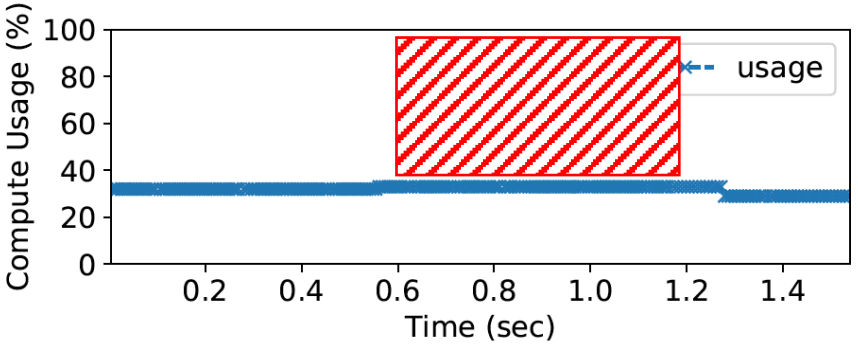
Resource utilization of gang-scheduled tasks



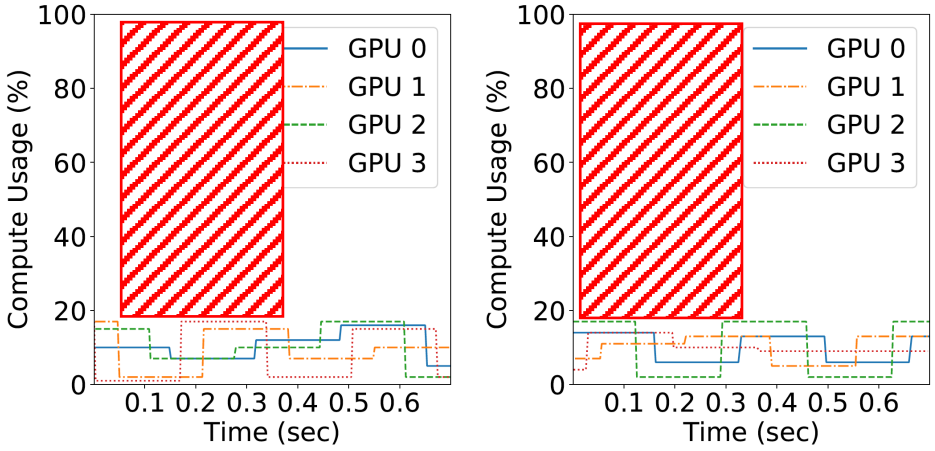
Memory Valley



(a) no pipelining (b) with pipelining



Memory-bounded Compute

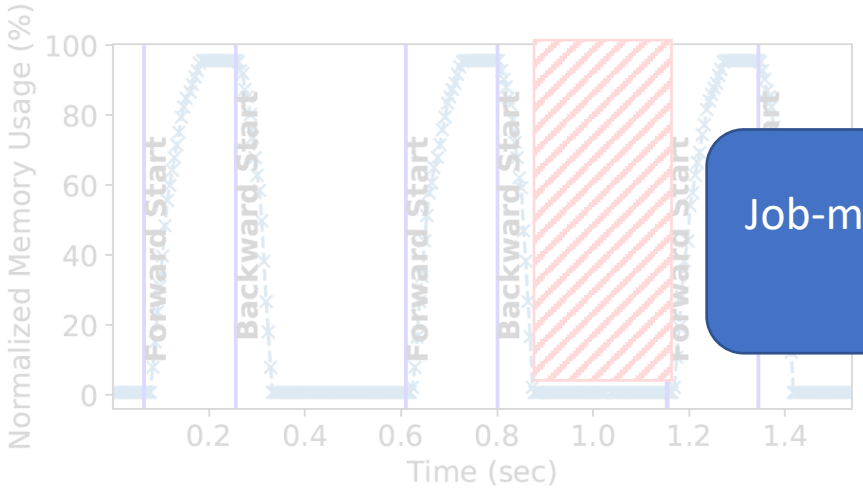


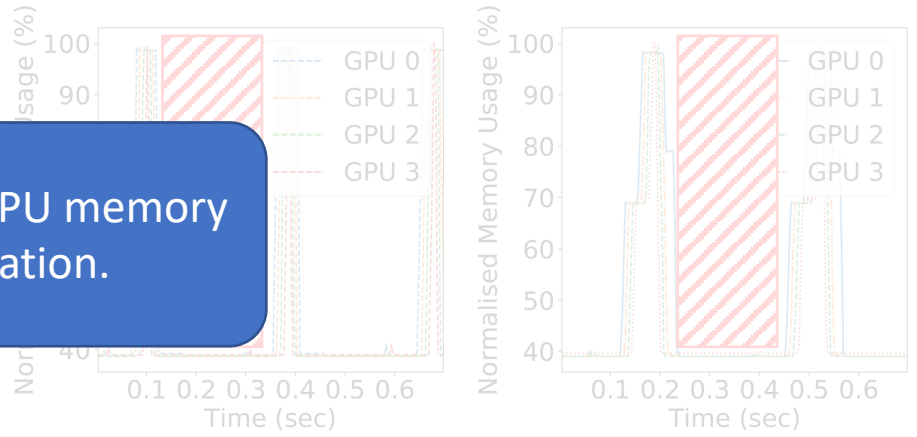
(a) no pipelining (b) with pipelining

Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Model parallel training
(4V100, BERT, w/wo pipeline parallelism)

Resource utilization of gang-scheduled tasks





(a) no pipelining (b) with pipelining

Job-multiplexing increases GPU memory and computation utilization.

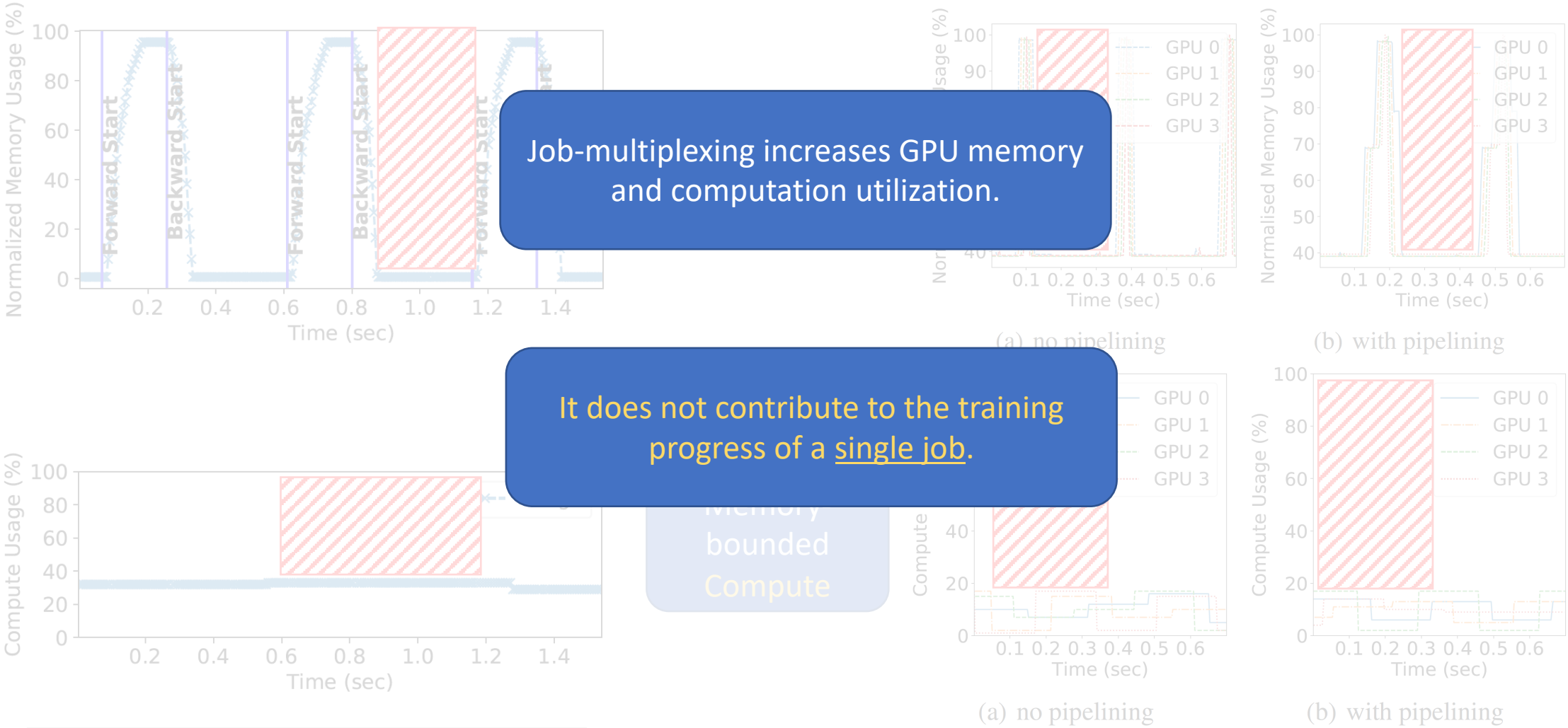
Memory-bounded Compute

(a) no pipelining (b) with pipelining

Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Model parallel training
(4V100, BERT, w/wo pipeline parallelism)

Resource utilization of gang-scheduled tasks



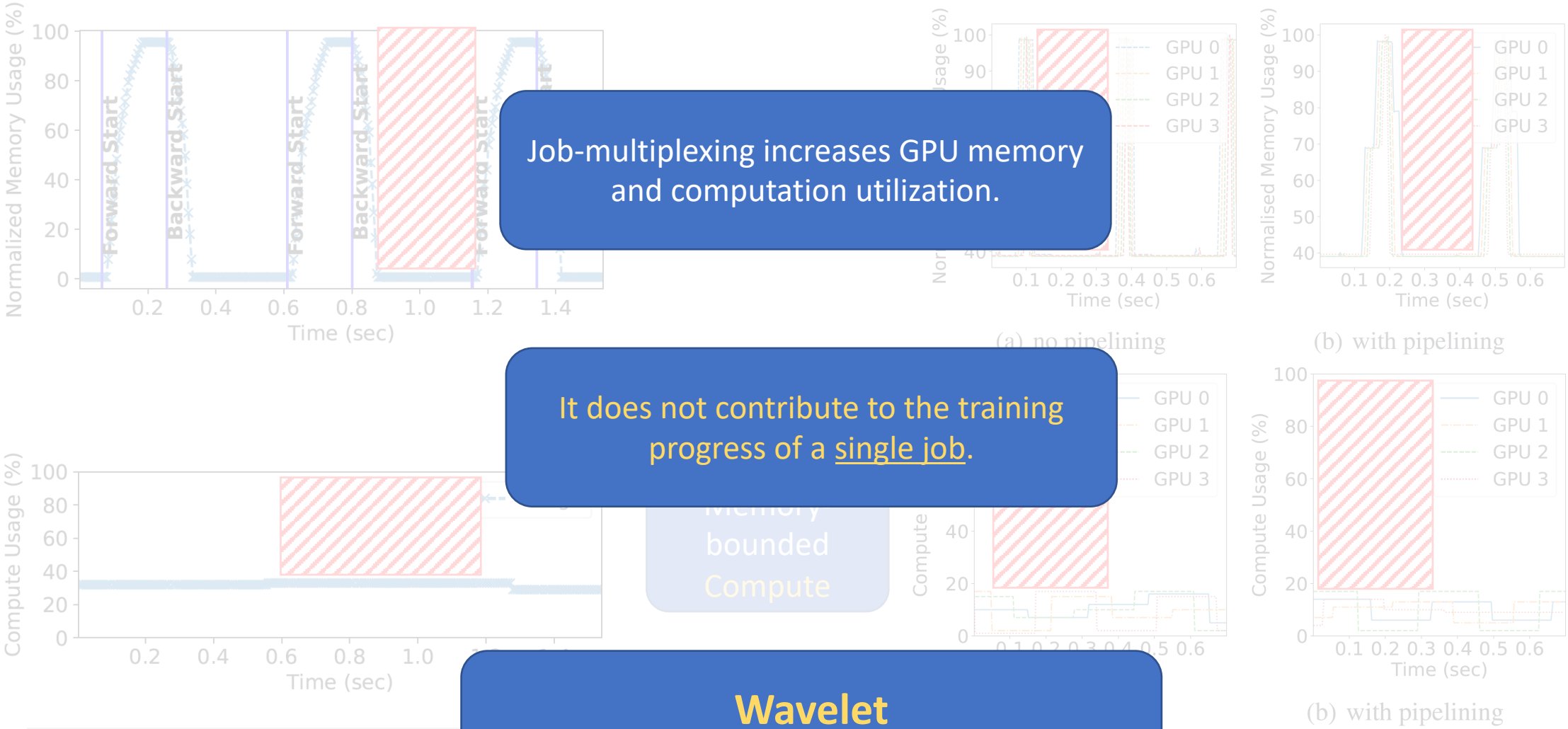
Job-multiplexing increases GPU memory and computation utilization.

It does not contribute to the training progress of a single job.

Data parallel training
(2V100, ImageNet-1K, ResNet-56)

Model parallel training
(4V100, BERT, w/wo pipeline parallelism)

Resource utilization of gang-scheduled tasks



Job-multiplexing increases GPU memory and computation utilization.

It does not contribute to the training progress of a single job.

Wavelet

improves system efficiency in single job case

Data parallel training
(2V100, ImageNet-1K, ResNe

l parallel training
w/wo pipeline parallelism)

Outline

- Motivation

- Design
  - Data Parallelism
  - Model Parallelism

- Evaluation

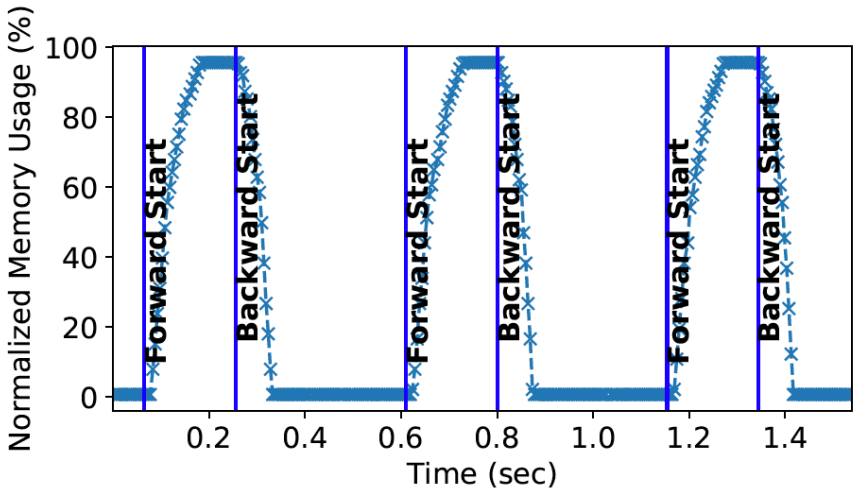
Outline

- Motivation

- Design
  - **Data Parallelism**
  - Model Parallelism

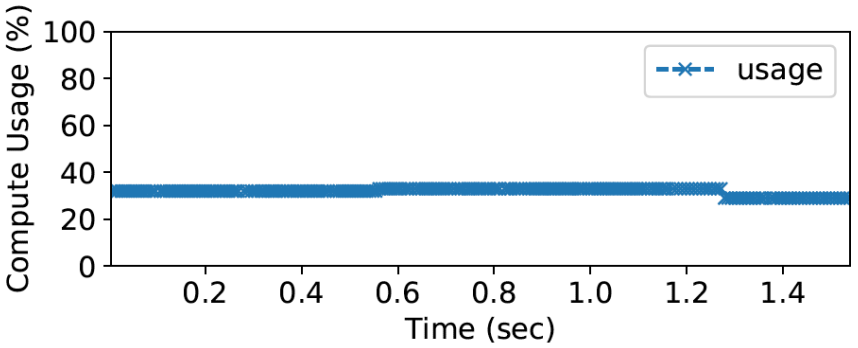
- Evaluation

Wavelet's Tick-Tock scheduling



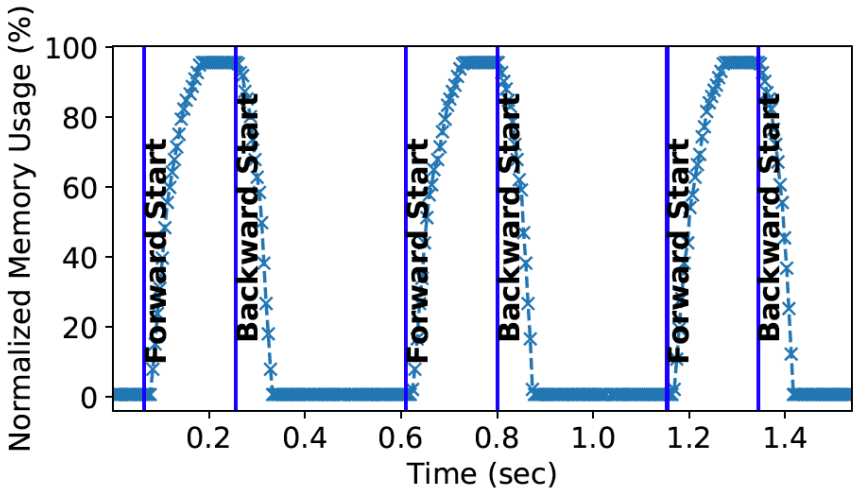
Tick-wave tasks (original)

Data parallel training
(2V100, ImageNet-1K, ResNet-56)



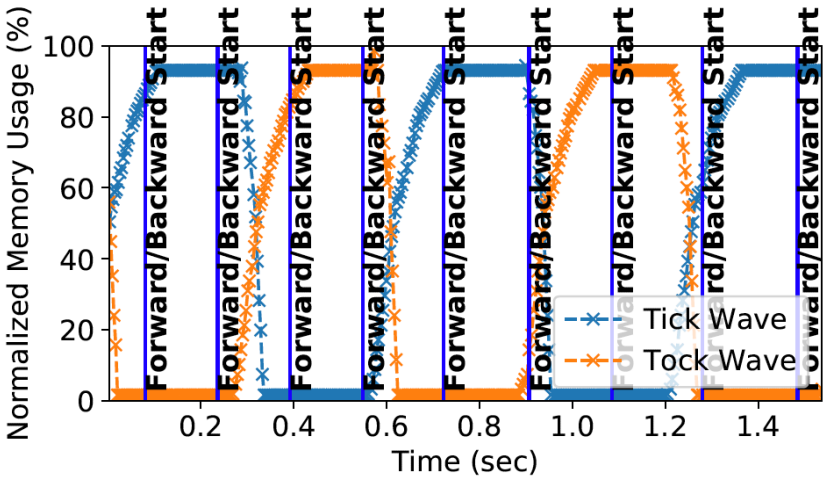
Gang-scheduling

Wavelet's Tick-Tock scheduling

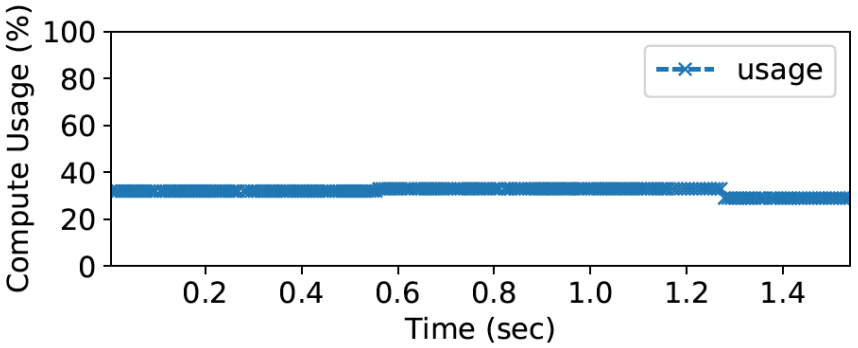


Tick-wave tasks (original)

Data parallel training
(2V100, ImageNet-1K, ResNet-56)



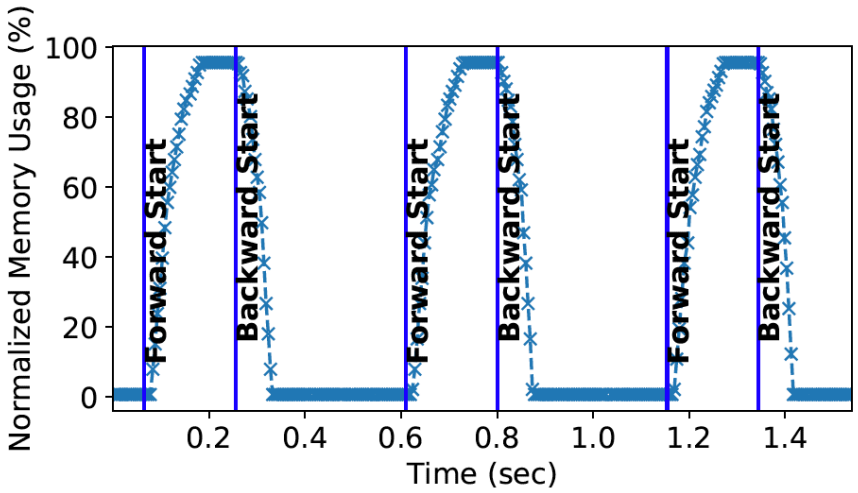
Tick-wave tasks (original)
Tock-wave tasks (injected)



Gang-scheduling

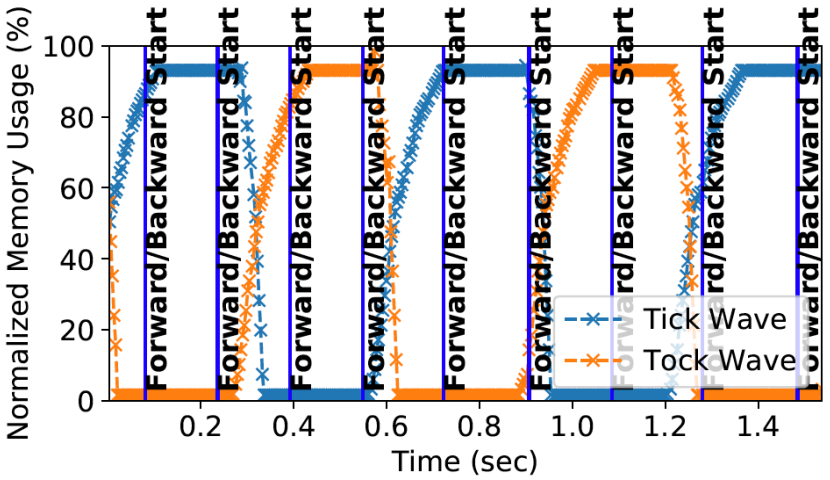
Tick-Tock scheduling

Wavelet's Tick-Tock scheduling

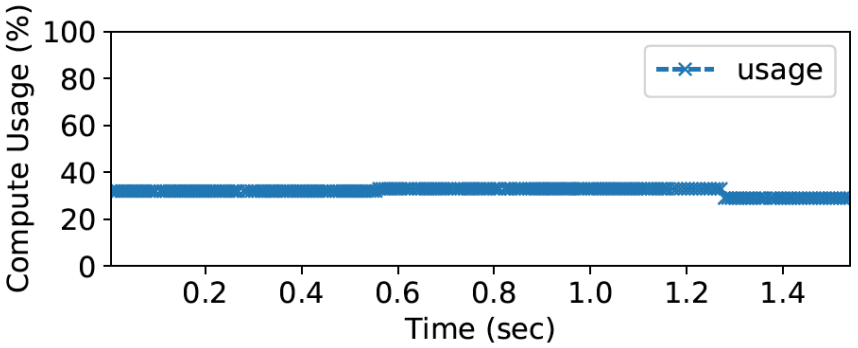


Tick-wave tasks (original)

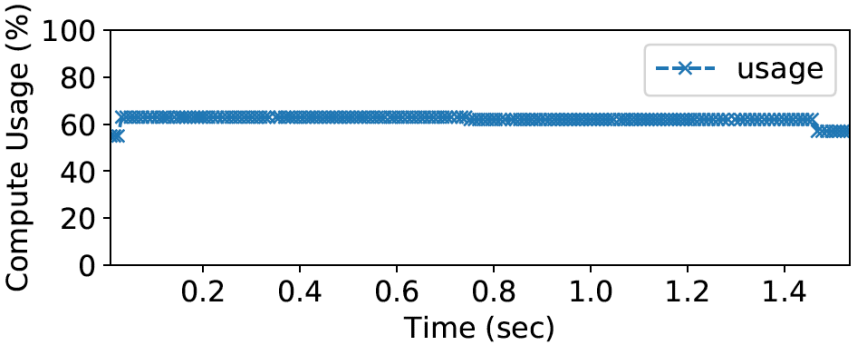
Data parallel training
(2V100, ImageNet-1K, ResNet-56)



Tick-wave tasks (original)
Tock-wave tasks (injected)

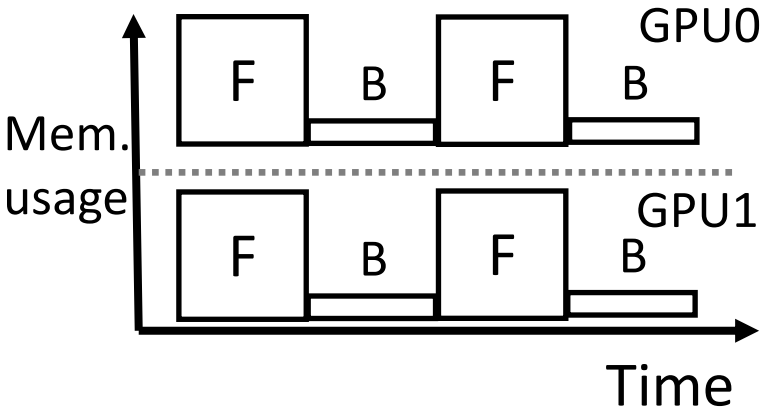


Gang scheduling



Tick-Tock scheduling

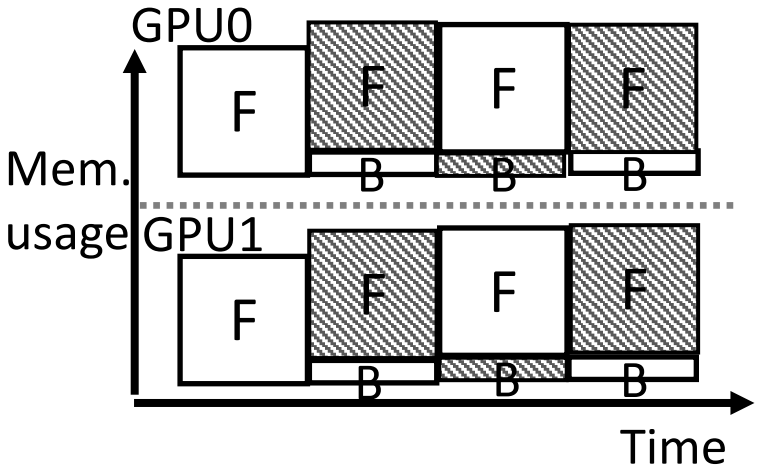
Tick-Tock scheduling in data parallel training



F/B= Forward/Backward

Tick-wave tasks

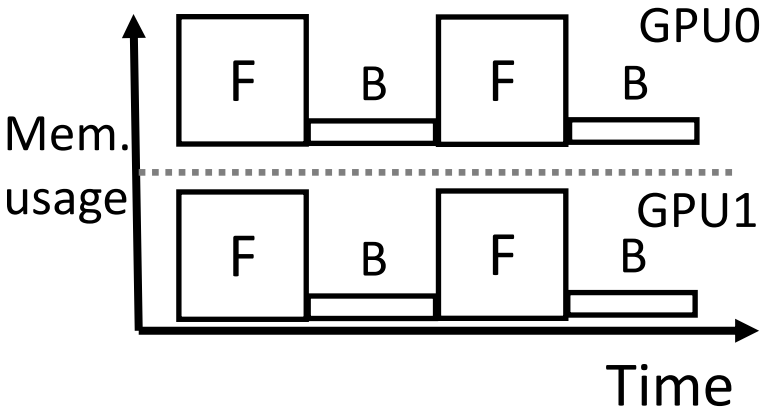
Tock-wave tasks



Gang scheduling

Tick-Tock scheduling

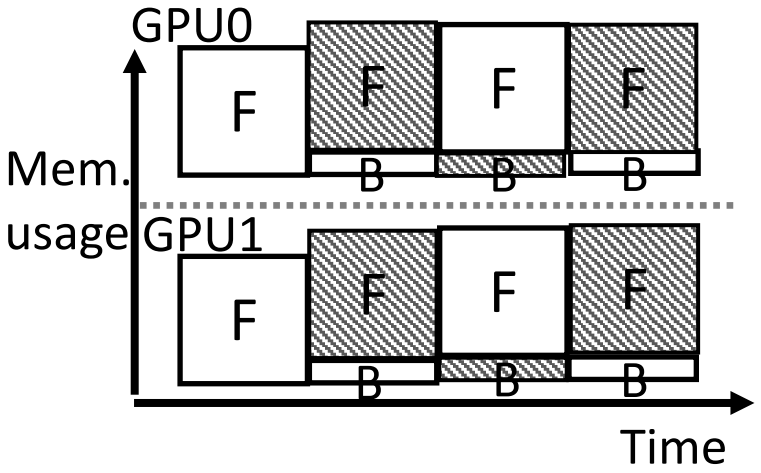
Tick-Tock scheduling in data parallel training

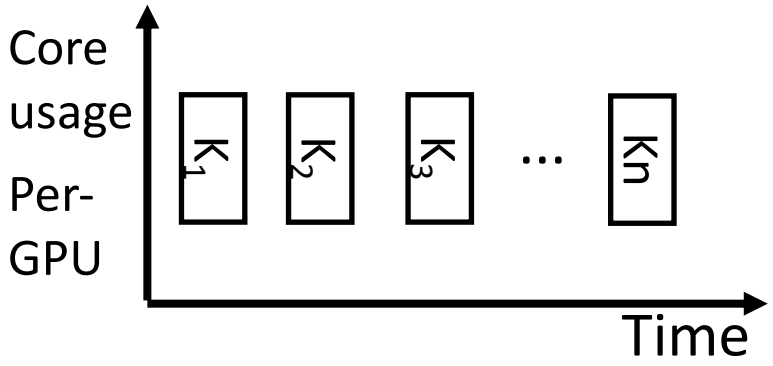


F/B= Forward/Backward

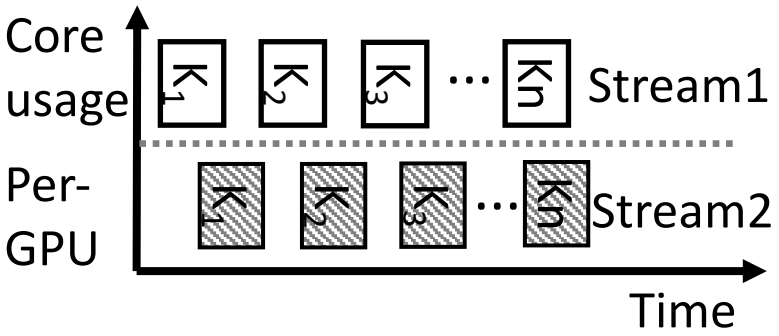
Tick-wave tasks

Tock-wave tasks





K = cuda_kernels



Gang scheduling

Tick-Tock scheduling

Model Synchronization between Tick-Tock Waves

- Inherent model synchronization within a wave (e.g. All-Reduce)

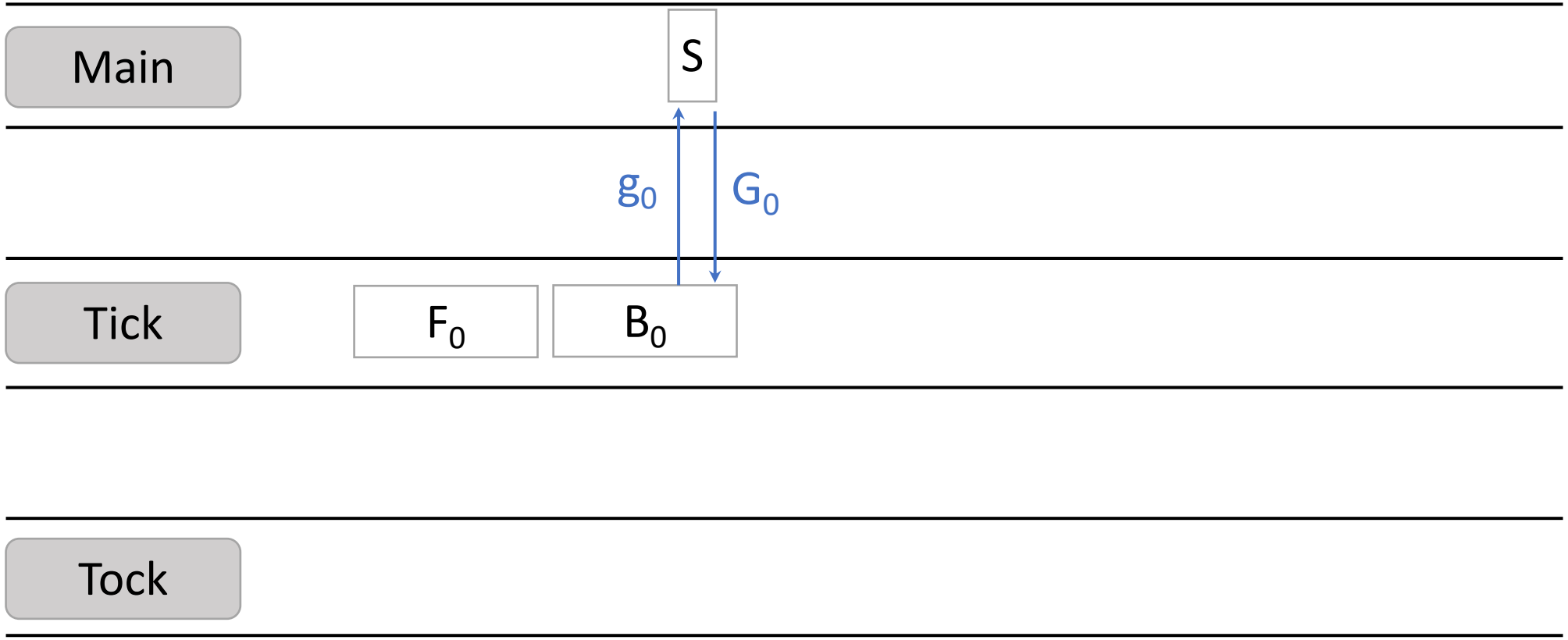
Model Synchronization between Tick-Tock Waves

- Inherent model synchronization within a wave (e.g. All-Reduce)

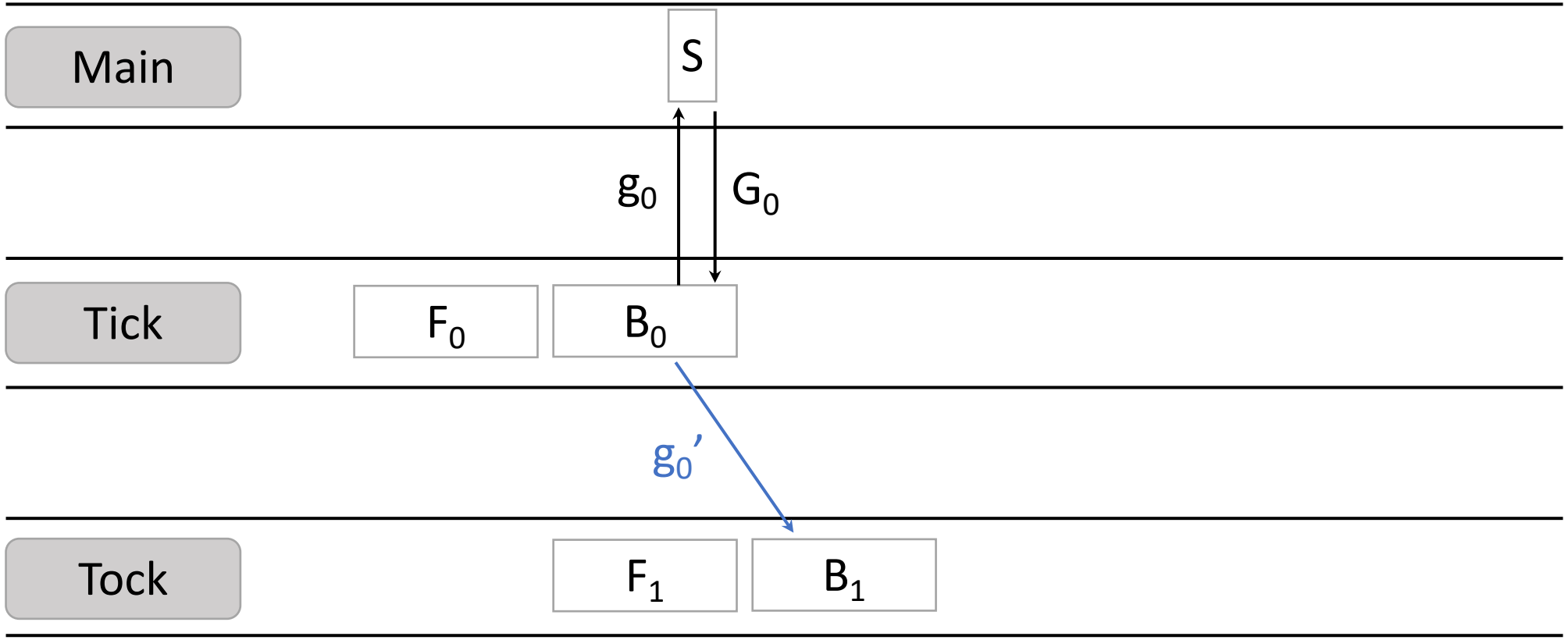
- How can we synchronize model between waves?

Model Synchronization between Tick-Tock Waves

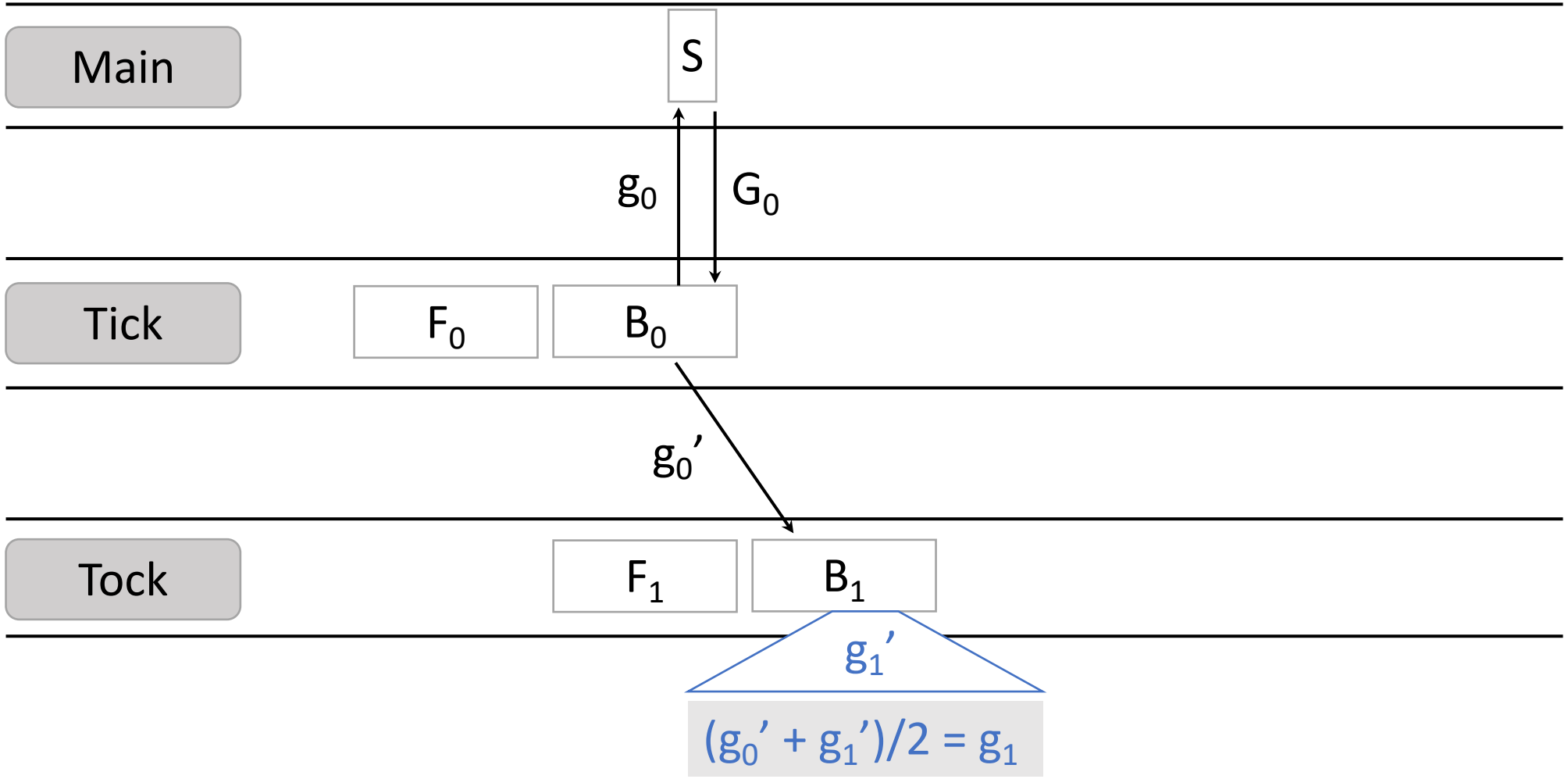
Model Synchronization between Tick-Tock Waves



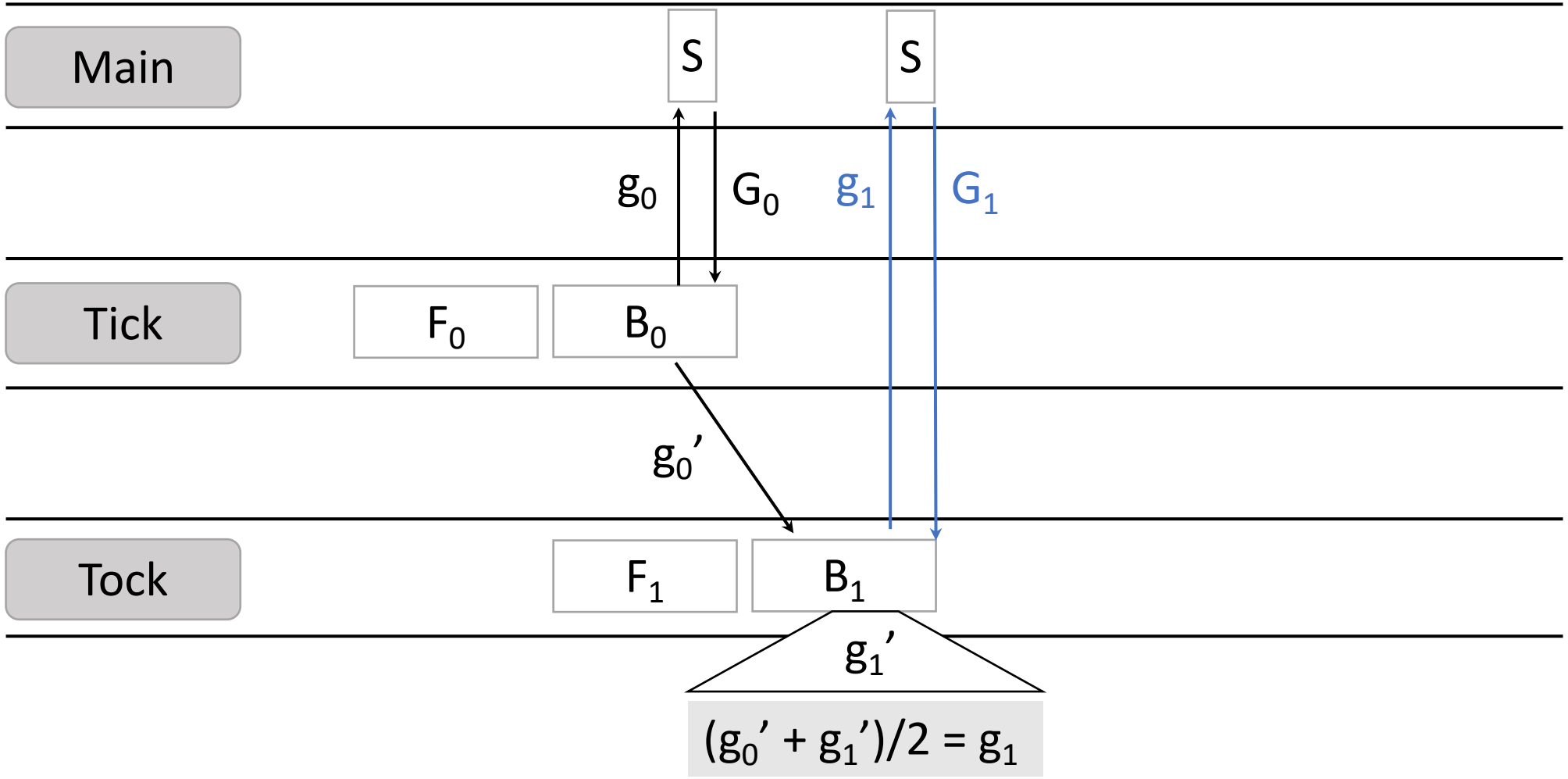
Model Synchronization between Tick-Tock Waves



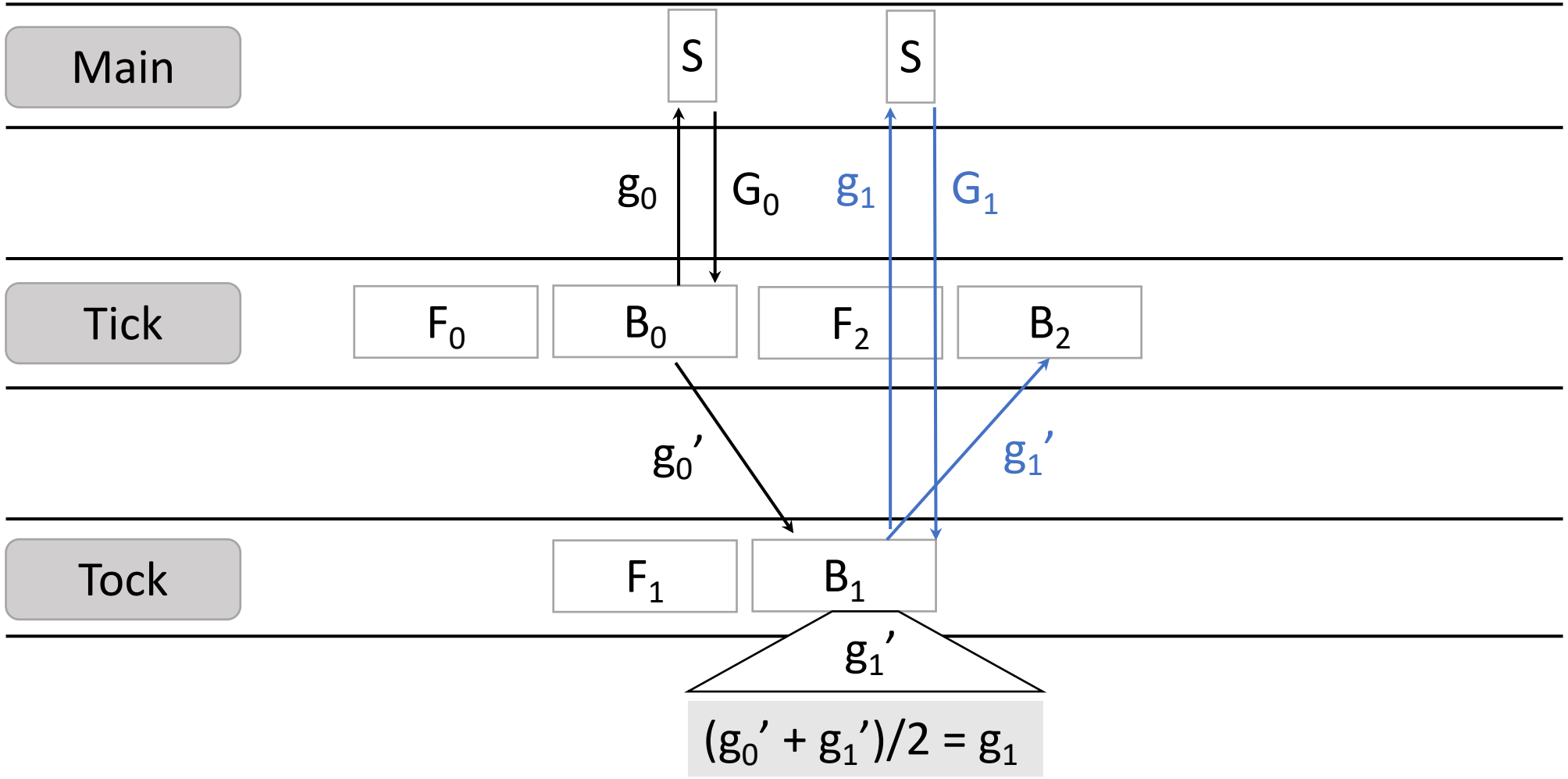
Model Synchronization between Tick-Tock Waves



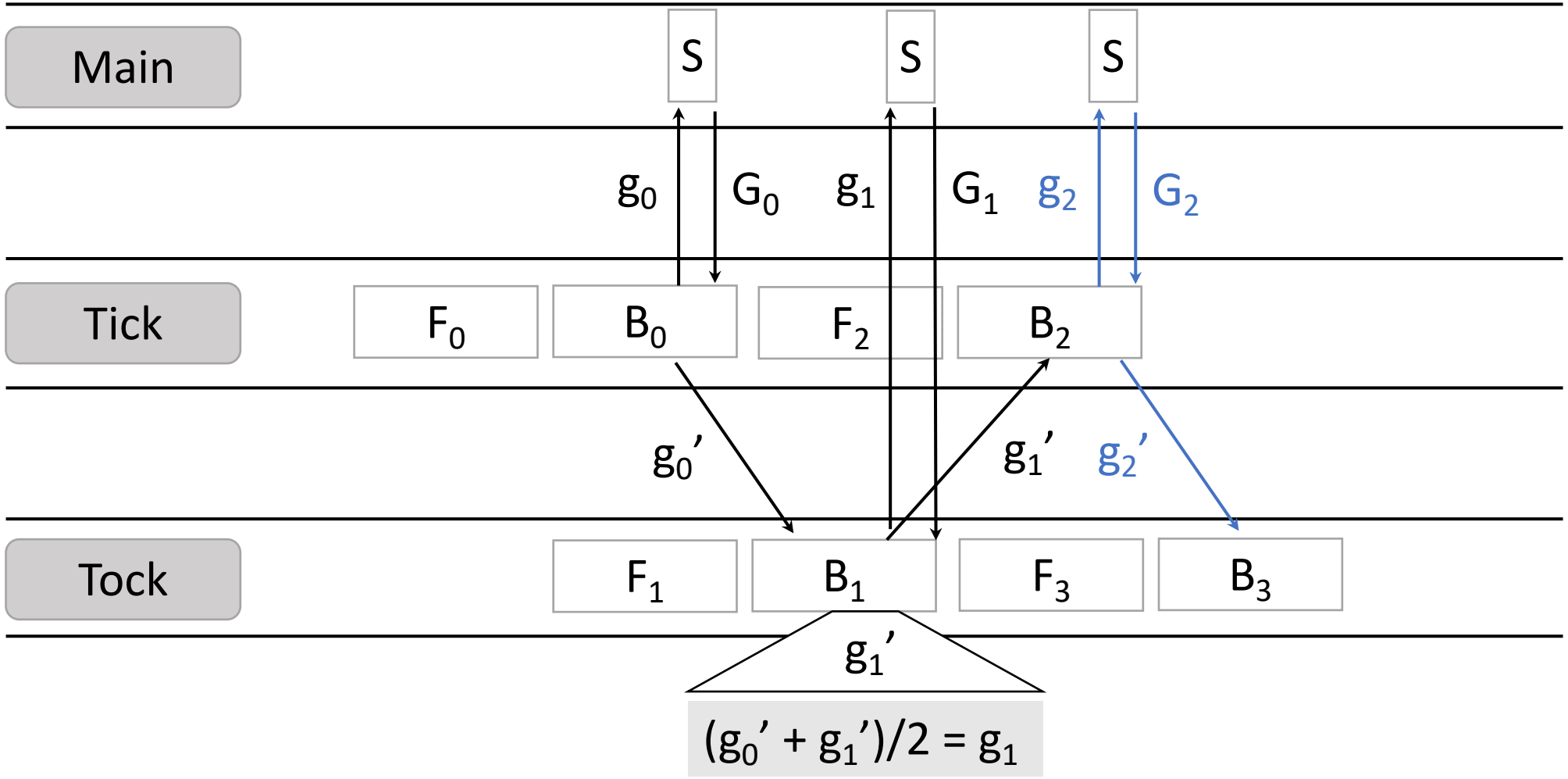
Model Synchronization between Tick-Tock Waves



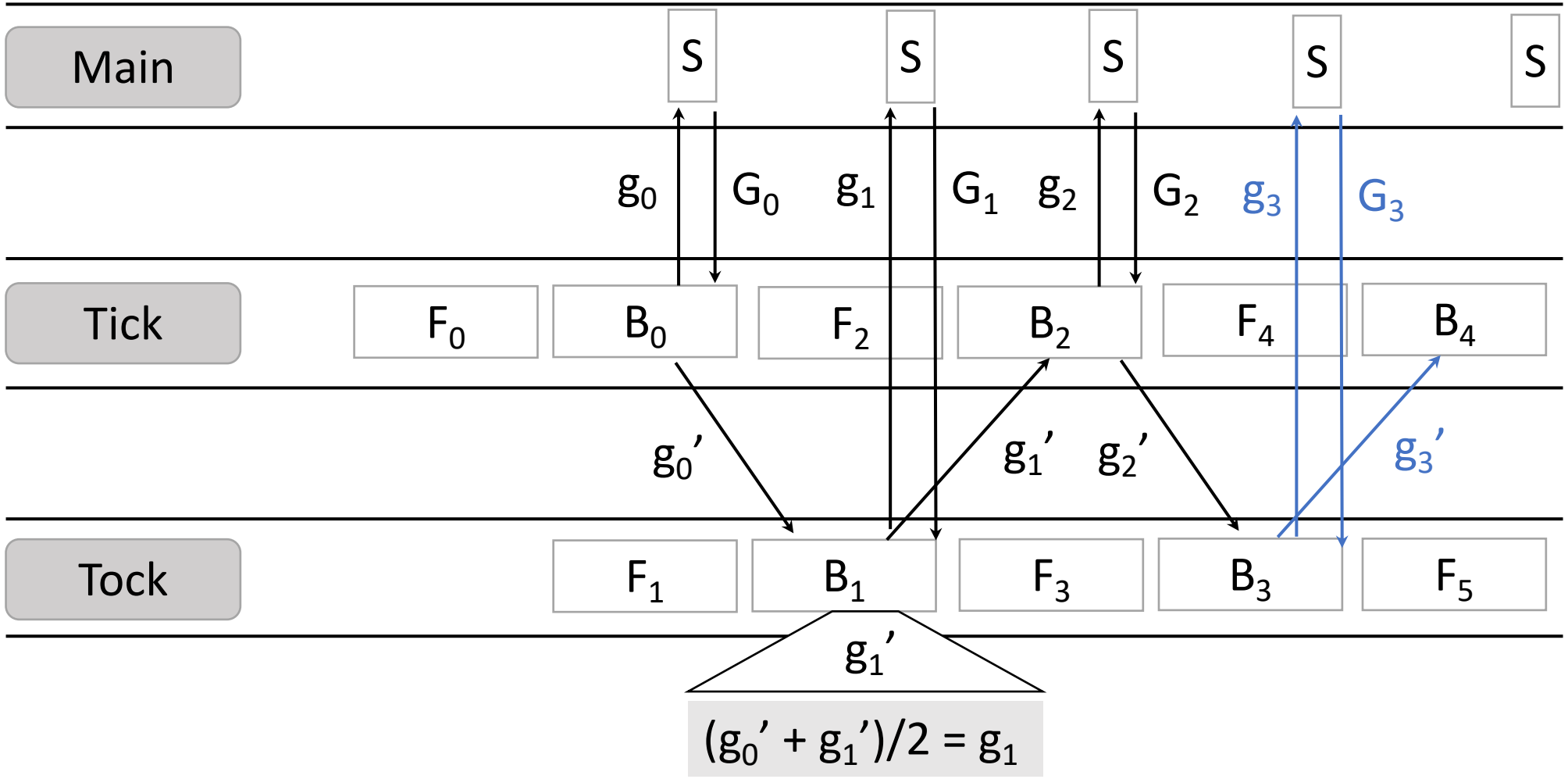
Model Synchronization between Tick-Tock Waves



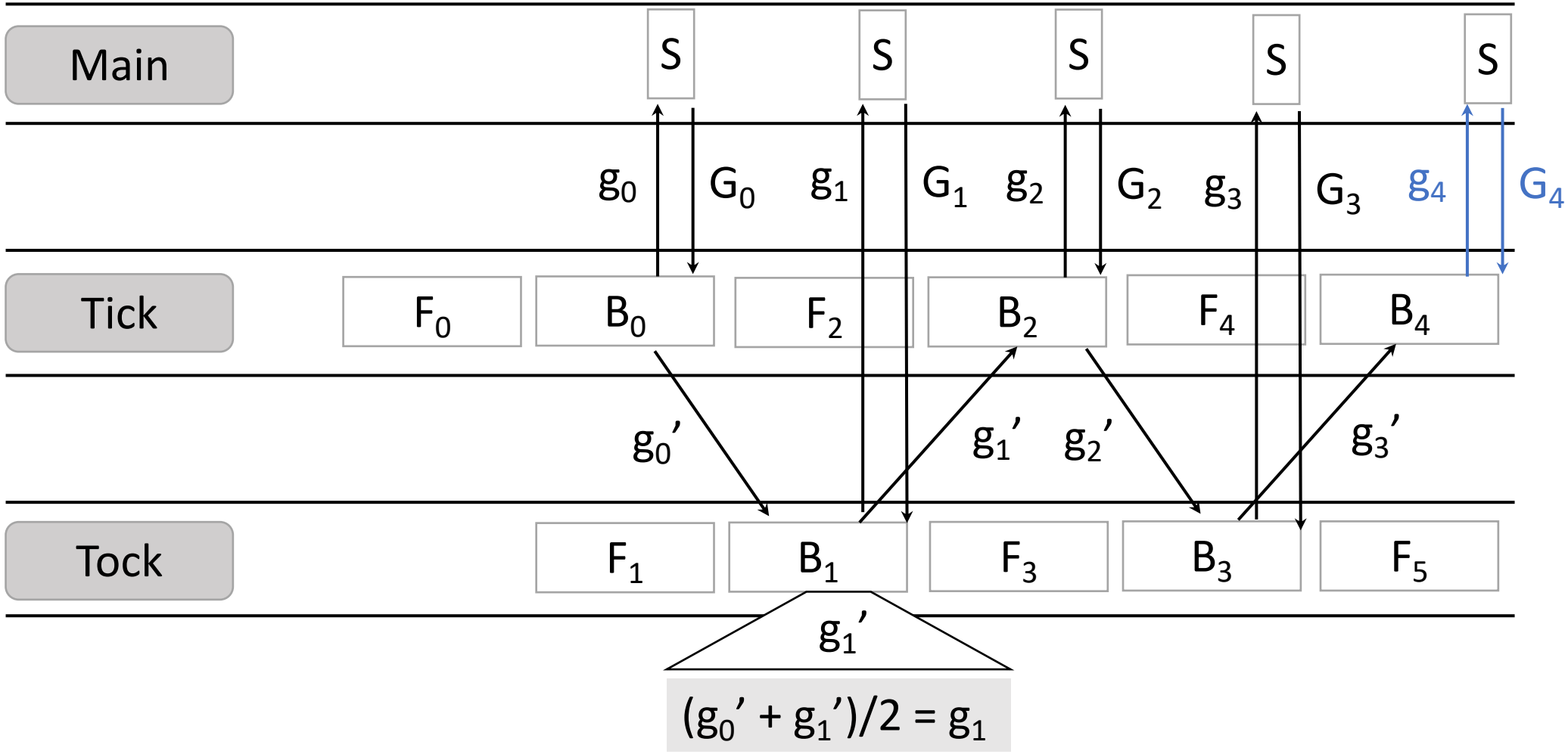
Model Synchronization between Tick-Tock Waves



Model Synchronization between Tick-Tock Waves

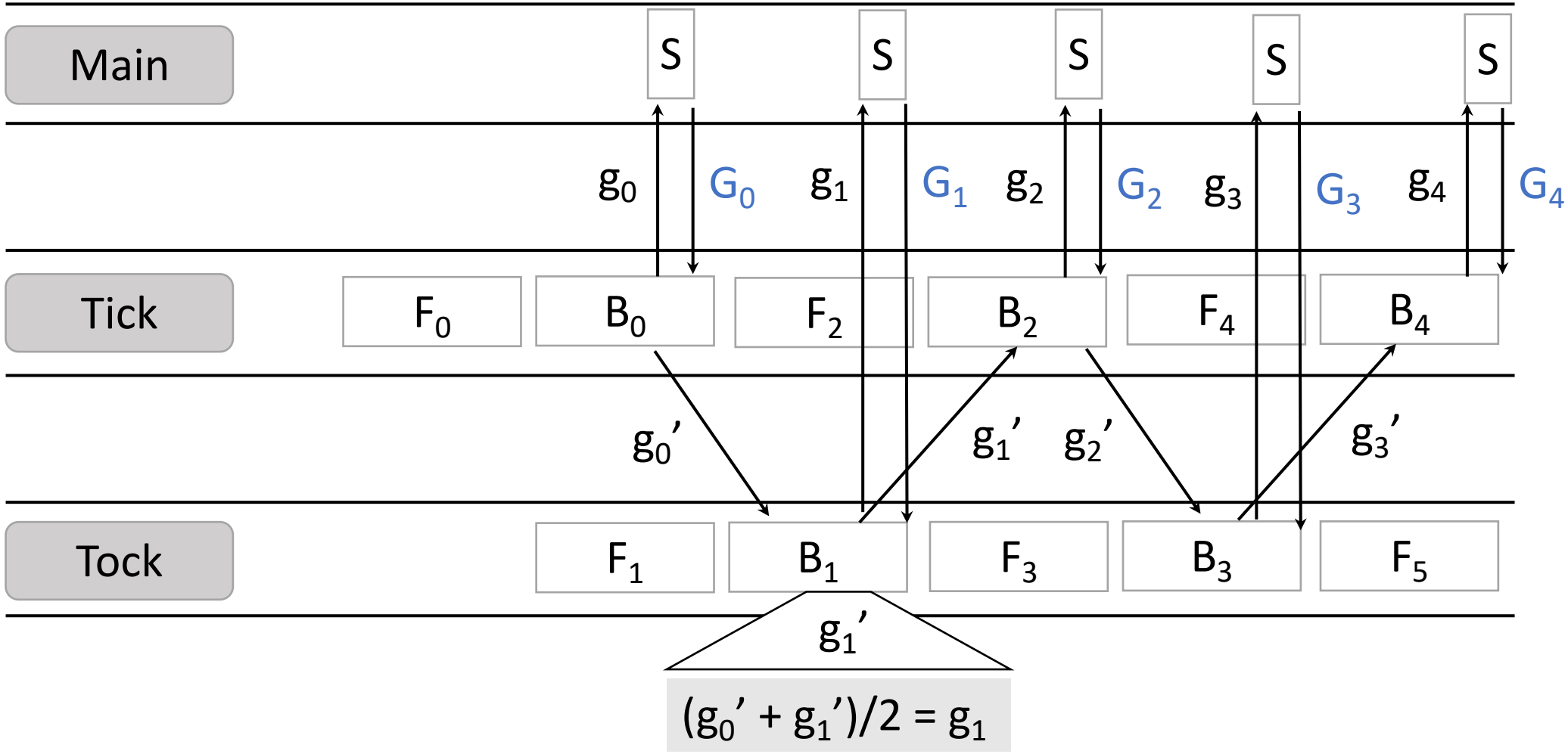


Model Synchronization between Tick-Tock Waves



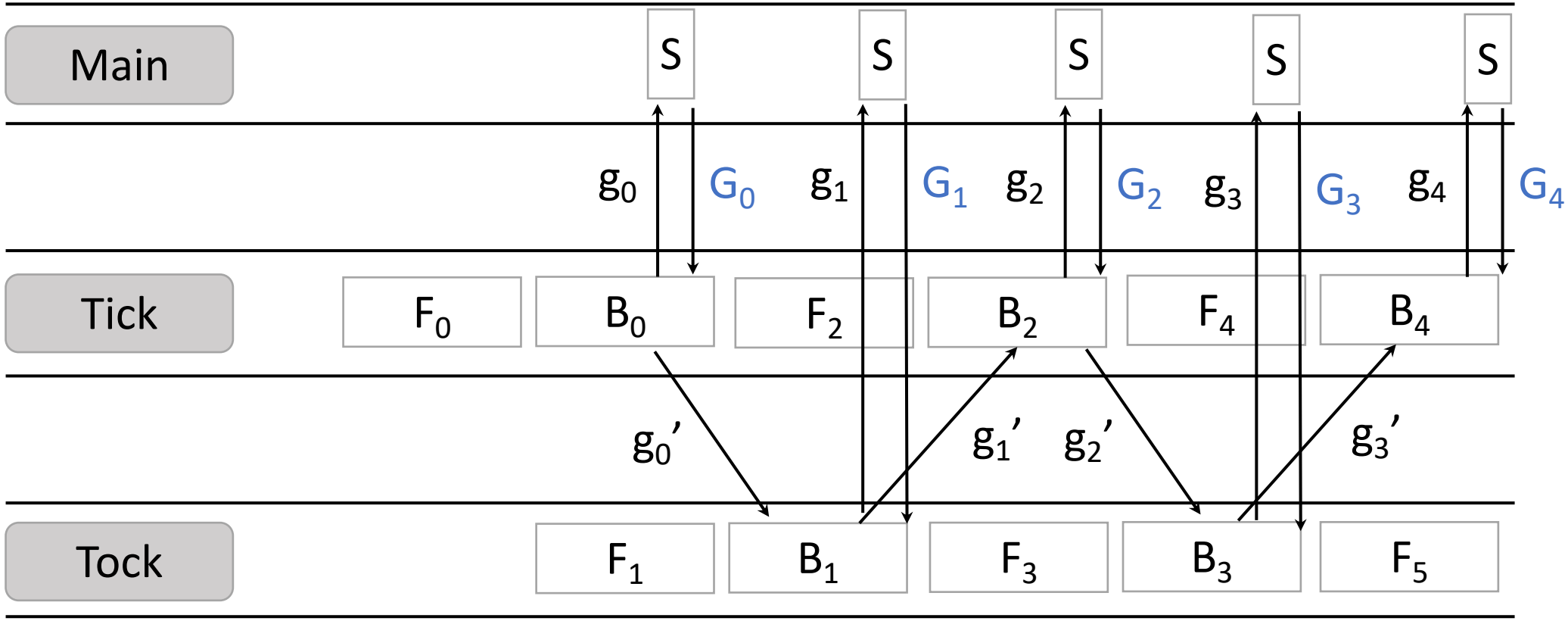
Model Synchronization between Tick-Tock Waves

$$G_t = \sum_{i=0}^{N} (g_{t-1}^{i}{}' + g_t^{i'})/2N$$



Model Synchronization between Tick-Tock Waves

$$G_t = \sum_{i=0}^{N} (g_{t-1}^{i}{}' + g_t^{i'})/2N$$



For N GPUs, Global Sync. over consequent 2*N batches → Guaranteed Convergence

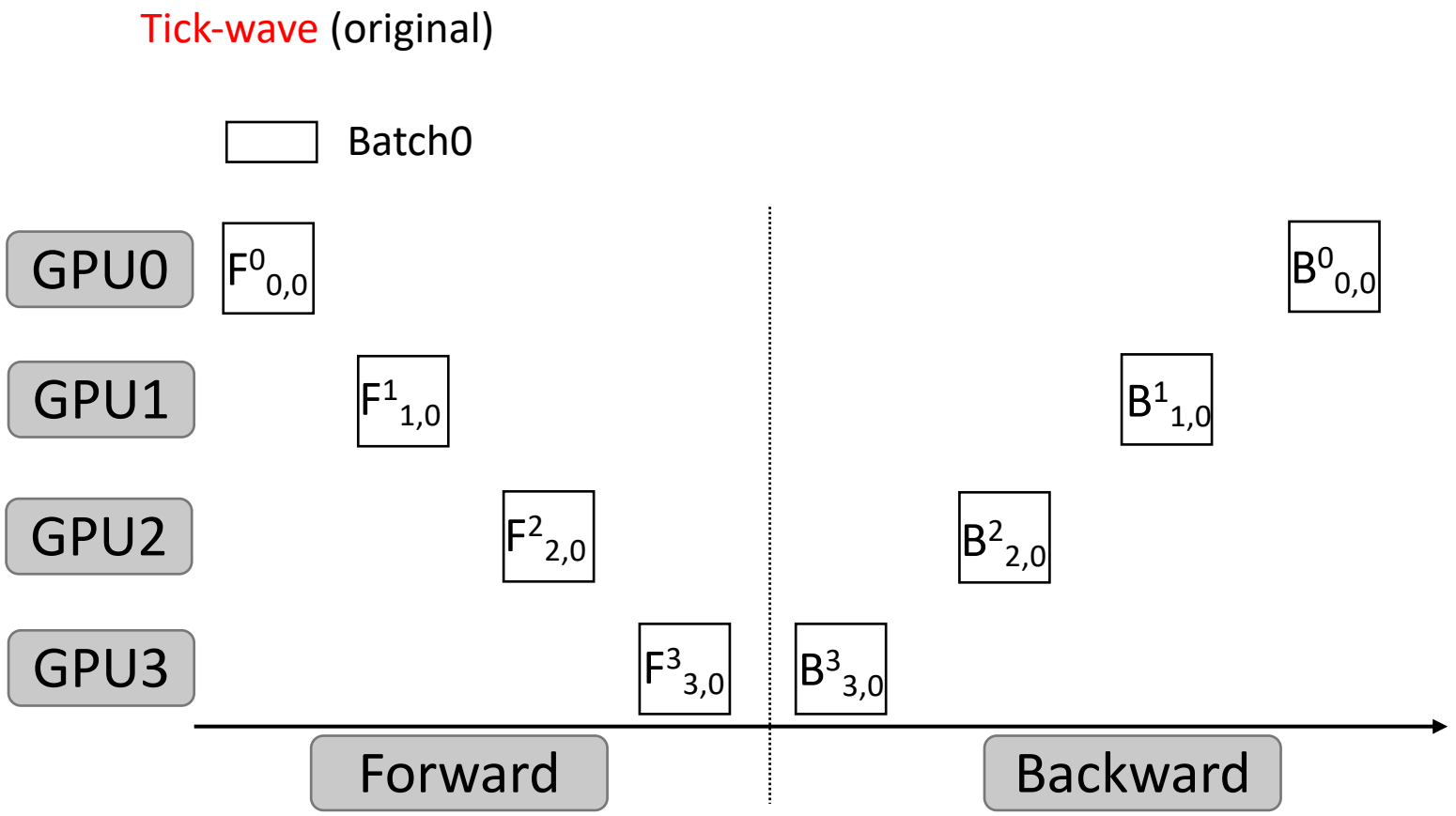
Outline

- Motivation

- Design
  - Data Parallelism
  - **Model Parallelism**

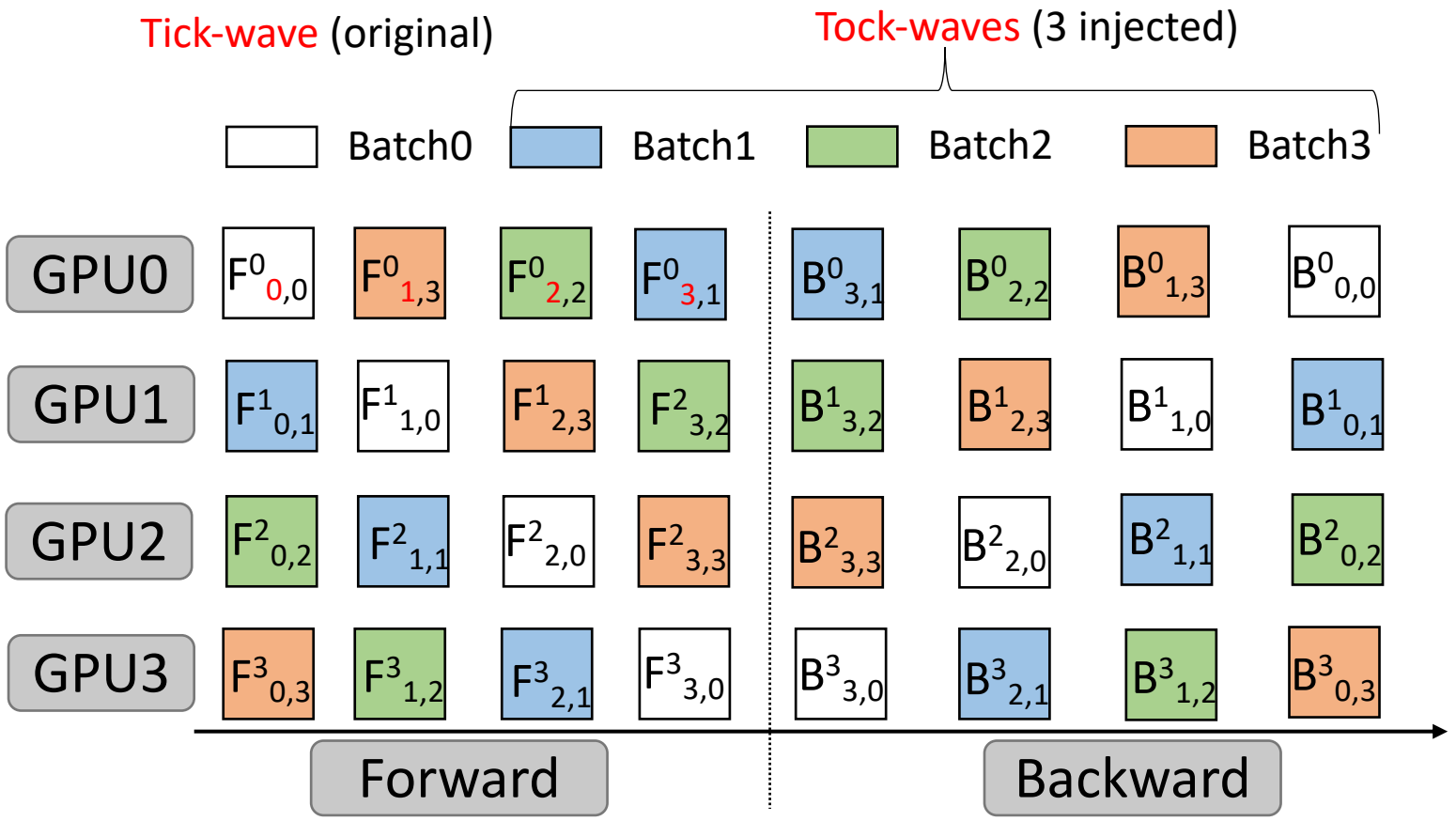
- Evaluation

Tick-Tock scheduling in model parallel training



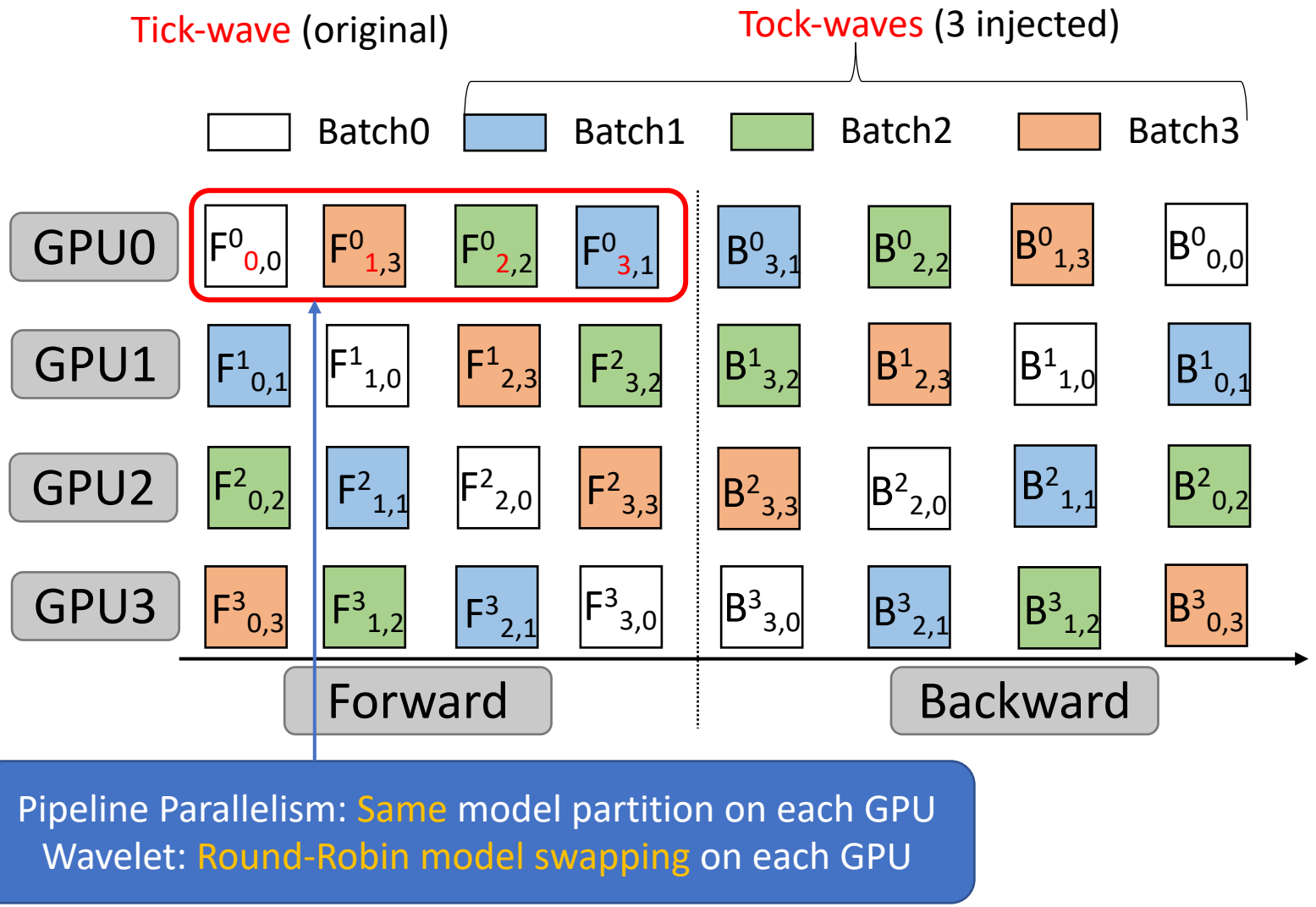
$F/B^x_{y,z}$ =
Forward/Backward at GPU x
with model partition y
on batch z

Tick-Tock scheduling in model parallel training



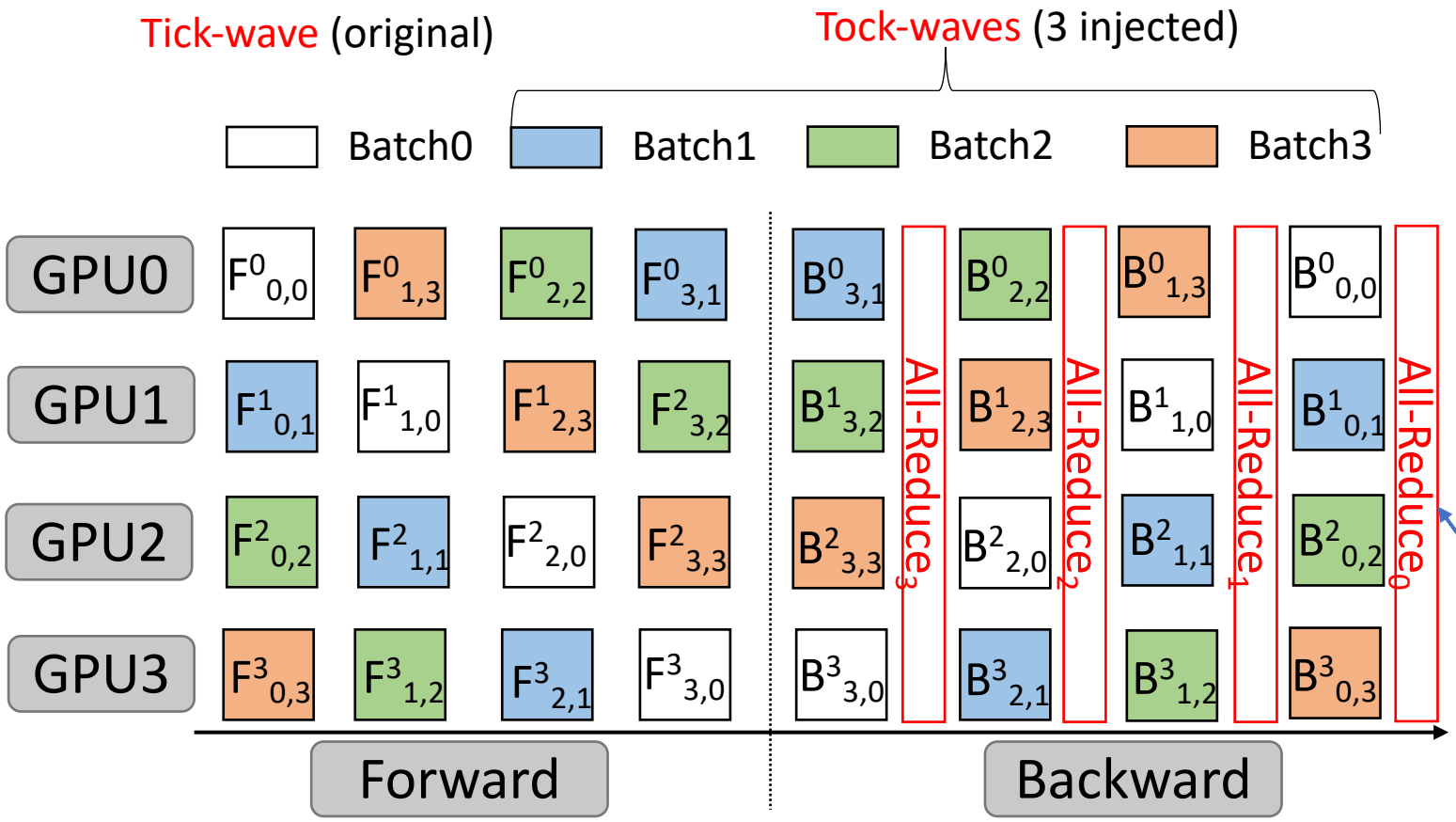
$F/B^x_{y,z}$ =
Forward/Backward at GPU x
with model partition y
on batch z

Tick-Tock scheduling in model parallel training



$F/B^x_{y,z}$ =
Forward/Backward at GPU x
with model partition y
on batch z

Tick-Tock scheduling in model parallel training



$F/B^x_{y,z}$ =
Forward/Backward at GPU x
with model partition y
on batch z

Pipeline Parallelism: Same model partition on each GPU
Wavelet: Round-Robin model swapping on each GPU

Pipeline Parallelism: staleness of weight updates
Wavelet: No staleness for each model partition

Outline

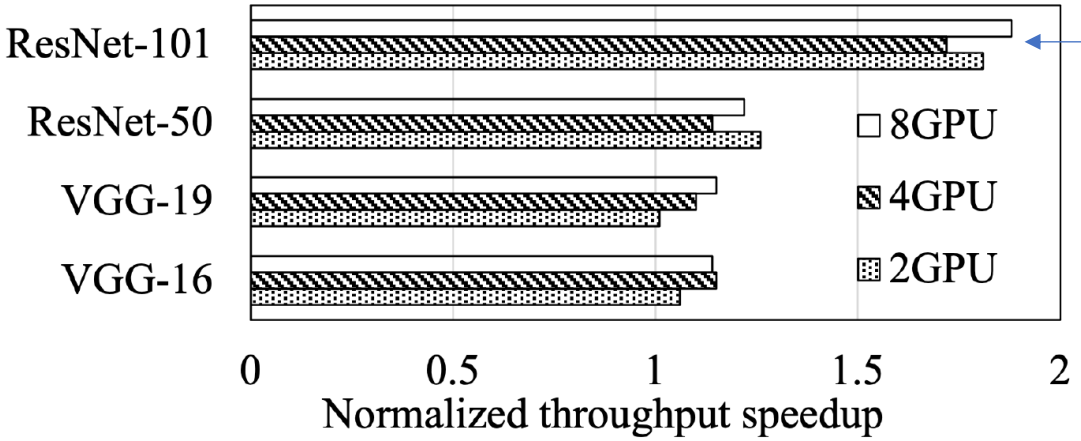
- Motivation

- Design
  - Data Parallelism
  - Model Parallelism

- **Evaluation**
  - **Data Parallelism**
  - **Model Parallelism**

Evaluations on Data Parallel Training

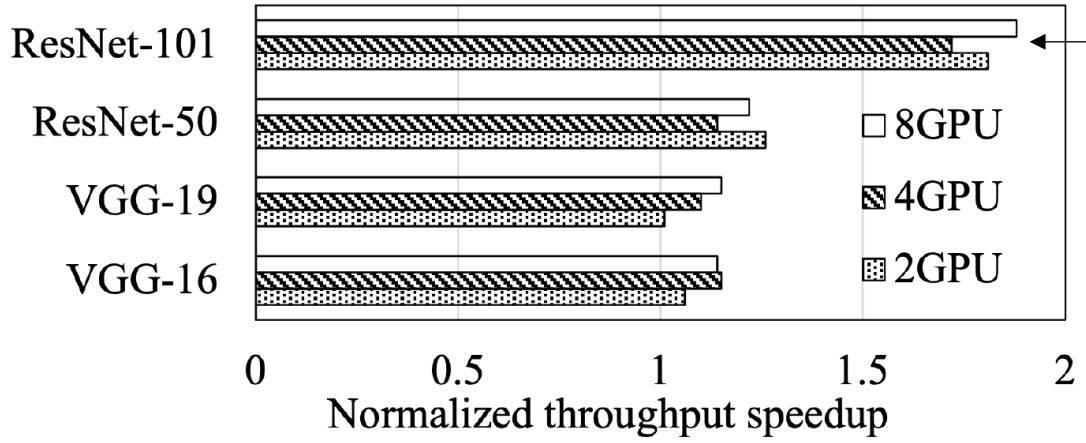
Single Machine (V100)



up to 1.88x speedup over DP baseline.

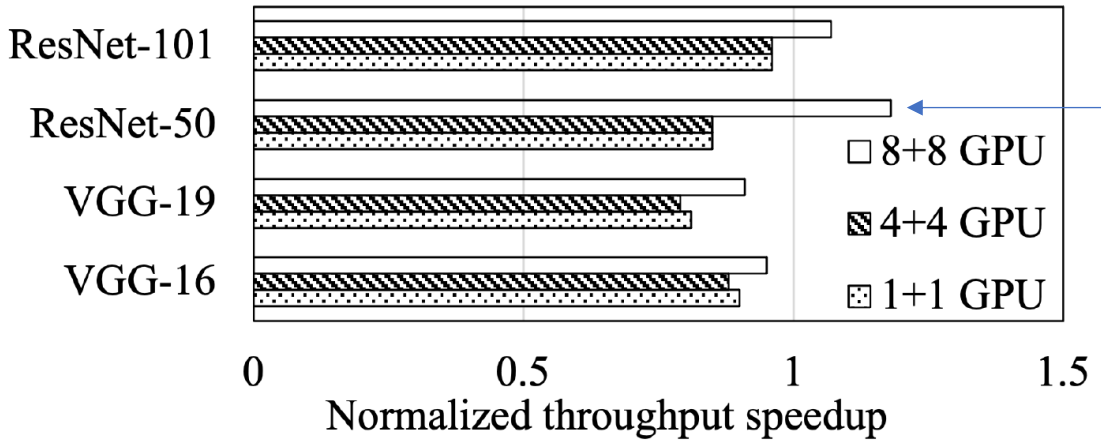
Evaluations on Data Parallel Training

Single Machine (V100)



up to 1.88x speedup over DP baseline.

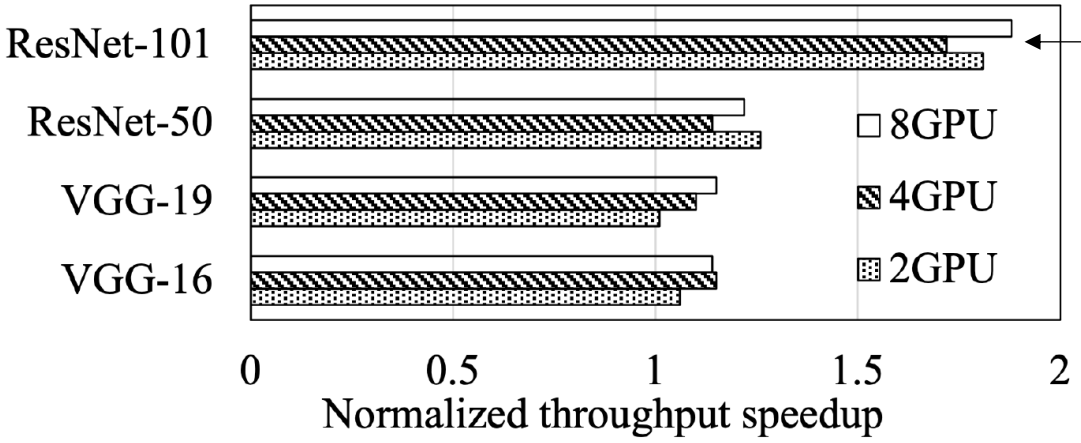
Cross Machines (V100)



up to 1.18x speedup over DP baseline.

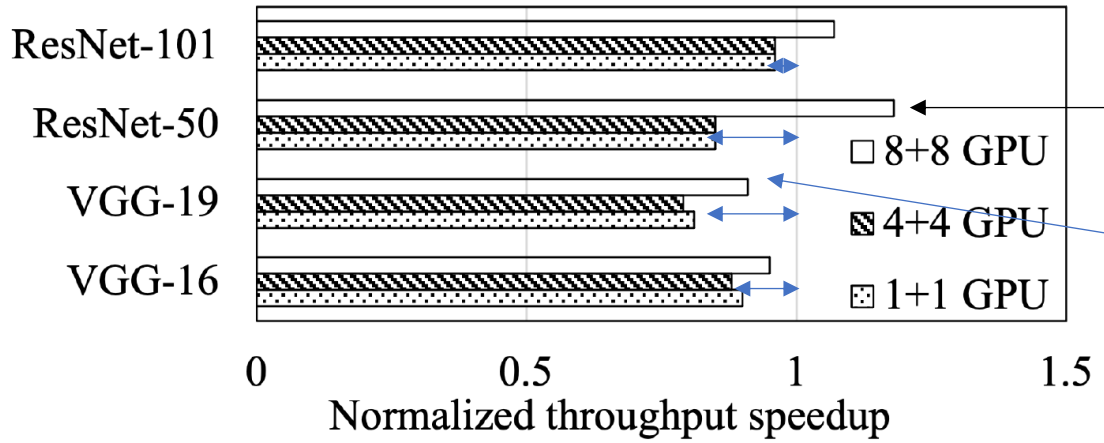
Evaluations on Data Parallel Training

Single Machine (V100)



up to 1.88x speedup over DP baseline.

Cross Machines (V100)



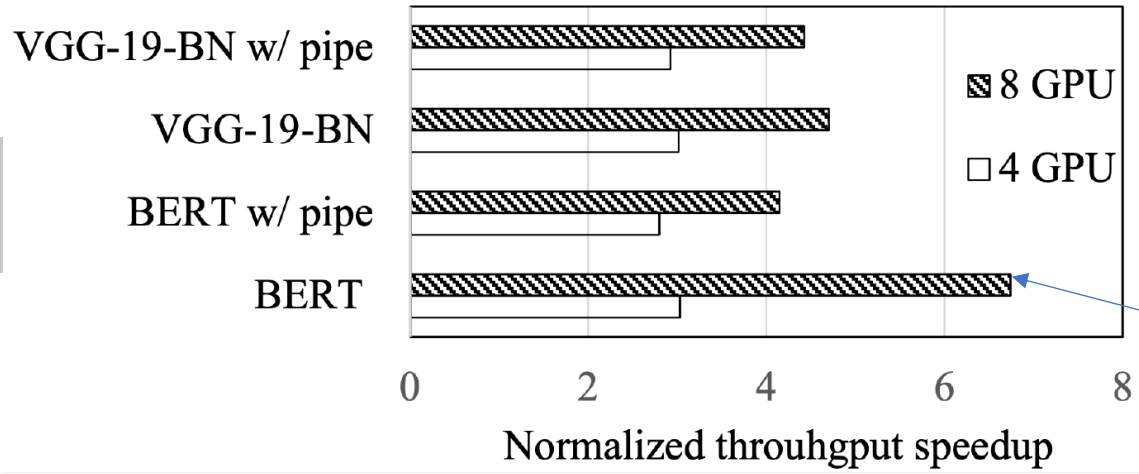
up to 1.18x speedup over DP baseline.

Additional tock-wave model sync.

cross-machine low-bandwidth network is the bottleneck.

Evaluations on Model Parallel Training

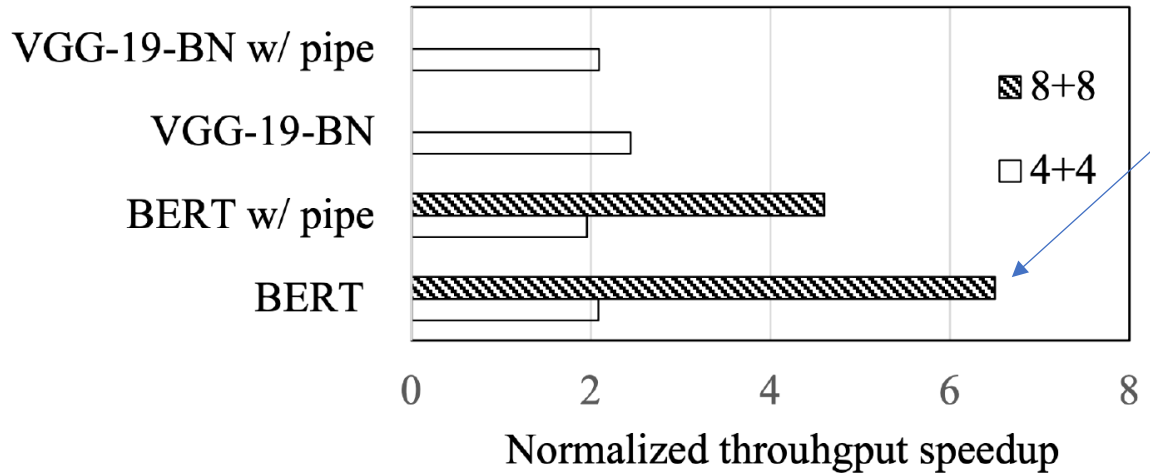
Single Machine (V100)



up to ~6x speedup over MP baseline

(w/wo pipeline parallelism)

Cross Machines (V100)



Wavelet Summary

- A wide range of distributed DNN training jobs are memory bounded.
- For single job training, gang-scheduling leaves memory valley period underutilized.
- Wavelet interleaves peak memory usages among waves of training tasks via tick-tock scheduling.
- For a single job, Wavelet achieves up to 1.88x speedup in data parallel training, and up to 6.7x speedup in model parallel training.

Side note: Pathways cites Wavelet

Introducing Pathways: A next-generation AI architecture

Oct 28, 2021
5 min read

Too often, machine learning systems overspecialize at individual tasks, when they could excel at many. That's why we're building Pathways—a new AI architecture that will handle many tasks at once, learn new tasks quickly and reflect a better understanding of the world.

Jeff Dean
Google Senior Fellow and SVP, Google Research

Share

Jeff Dean (@🏡)
@JeffDean

This paper is about the Pathways system that is designed to support the broader Pathways vision of creating large scale, multi-task, multi-modal models w/flexible support for both large dense models as well as a variety of sparse architectures. Stay tuned for uses of the system!

> **Michael Isard** @m_isard · Mar 29
> First publication about the Pathways system for ML that I have been working on with many colleagues at Google for a few years. To appear at MLSys22! arxiv.org/abs/2203.12533 @JeffDean @achowdhery @bsaeta @danhurt

10:46 PM · Mar 29, 2022 · Twitter for Android

Side note: Pathways cites Wavelet

Introducing Pathways: A next-generation AI architecture

Oct 28, 2021
5 min read

Too often, machine learning systems overspecialize at individual tasks, when they could excel at many. That's why we're building Pathways—a new AI architecture that will handle many tasks at once, learn new tasks quickly and reflect a better understanding of the world.

Jeff Dean
Google Senior Fellow and SVP, Google Research

Share

Jeff Dean (@🏠)
@JeffDean

This paper is about the Pathways system that is designed to support the broader Pathways vision of creating large scale, multi-task, multi-modal models w/flexible support for both large dense models as well as a variety of sparse architectures. Stay tuned for uses of the system!

> **Michael Isard** @m_isard · Mar 29
> First publication about the Pathways system for ML that I have been working on with many colleagues at Google for a few years. To appear at MLSys22! arxiv.org/abs/2203.12533 @JeffDean @achowdhery @bsaeta @danhurt

10:46 PM · Mar 29, 2022 · Twitter for Android

PATHWAYS: ASYNCHRONOUS DISTRIBUTED DATAFLOW FOR ML

Paul Barham [1] **Aakanksha Chowdhery** [1] **Jeff Dean** [1] **Sanjay Ghemawat** [1] **Steven Hand** [1] **Dan Hurt** [1]
Michael Isard [1] **Hyeontaek Lim** [1] **Ruoming Pang** [1] **Sudip Roy** [1] **Brennan Saeta** [1] **Parker Schuh** [1]
Ryan Sepassi [1] **Laurent El Shafey** [1] **Chandramohan A. Thekkath** [1] **Yonghui Wu** [1]

Guanhua Wang, Kehan Wang, Kenan Jiang, Xiangjun Li, and Ion Stoica. Wavelet: Efficient DNN training with Tick-Tock scheduling. In *Proceedings of Machine Learning and Systems*, 2021.

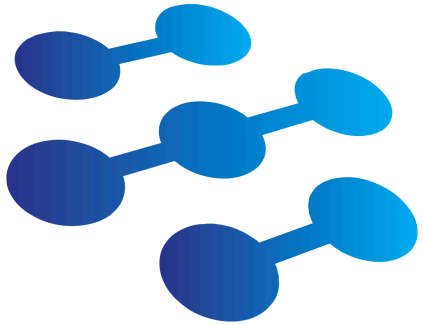
Research Summary

BLINK

Packing spanning tree is
better than forming rings.

Research Summary

BLINK



sensAI

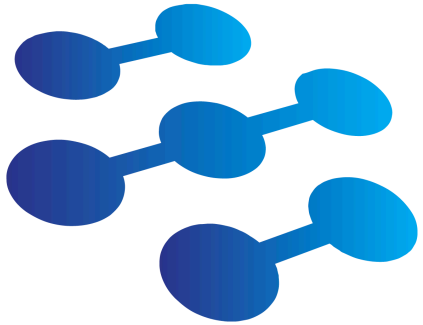
Packing spanning tree is better than forming rings.

Communication can be eliminated with class parallelism.

Research Summary

BLINK

Packing spanning tree is better than forming rings.



sensAI

Communication can be eliminated with class parallelism.

Wavelet

Interleaving peak memory usage increase system efficiency.

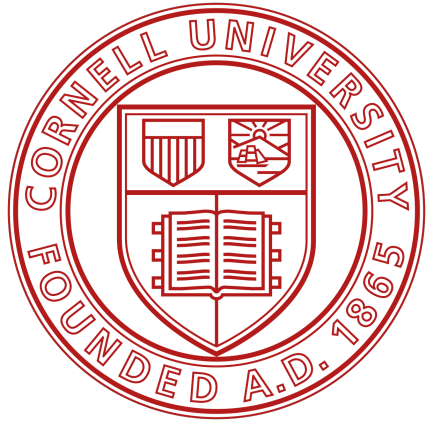
Recent Talks in both Academia and Industry

Stanford
MLSys Seminar

Tim Kraska
(Prof. Michael
Stonebraker listened)

Eric Xing/Tianqi Chen

Michael Freedman/
Wyatt Lloyd

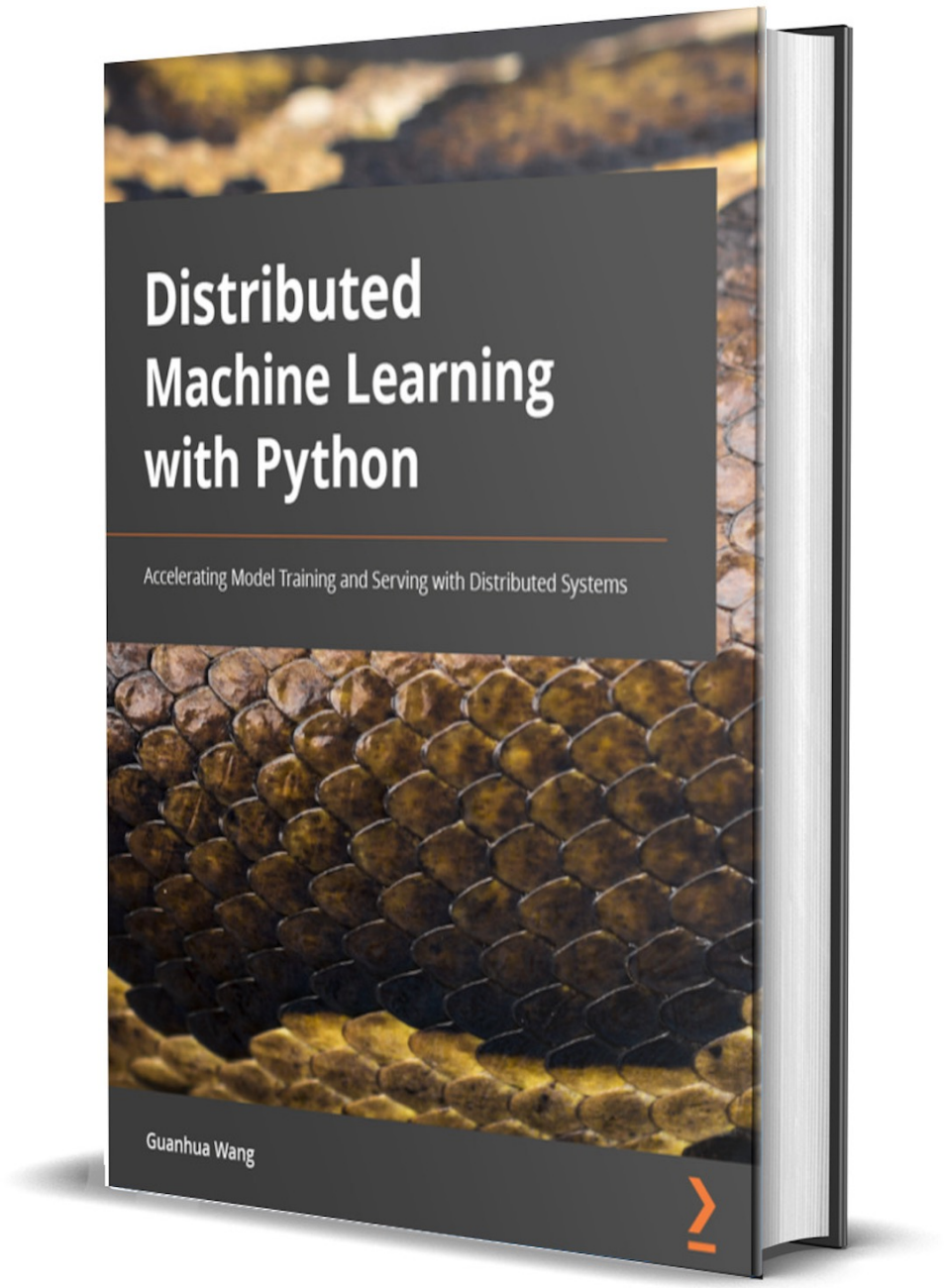


Immanuel Trummer

PyTorch Distributed

DeepSpeed

Autonomous Driving



My book will be published in May 2022.

Pre-order on Amazon, Barnes&Noble, etc.

Thanks for your listening