Communication Primitives for Distributed Training Algorithms

ECE 826

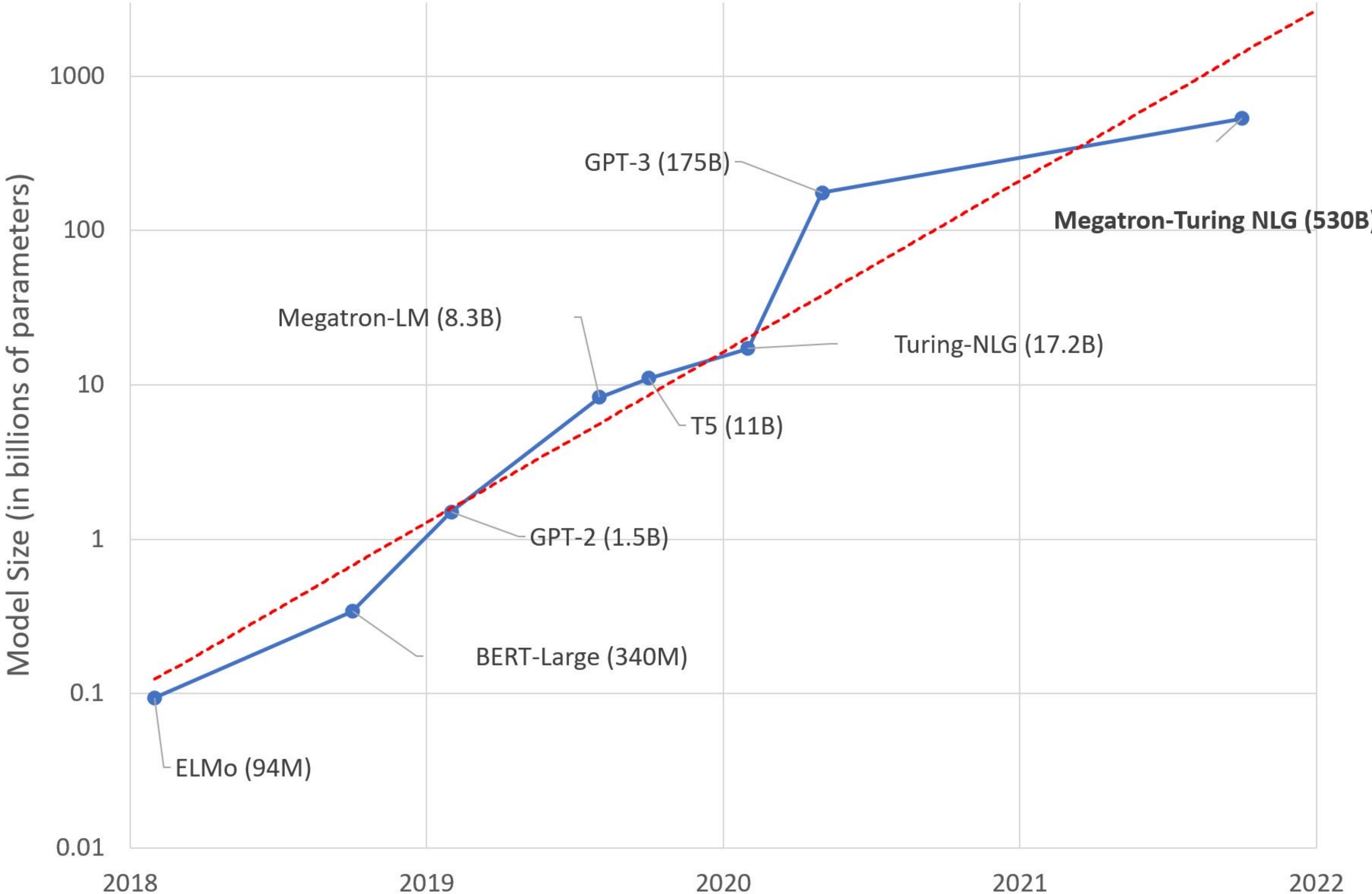
Hongyi Wang
Machine Learning Department
Carnegie Mellon University
April 14th

Overview

- <u>Motivations of Distributed ML</u>

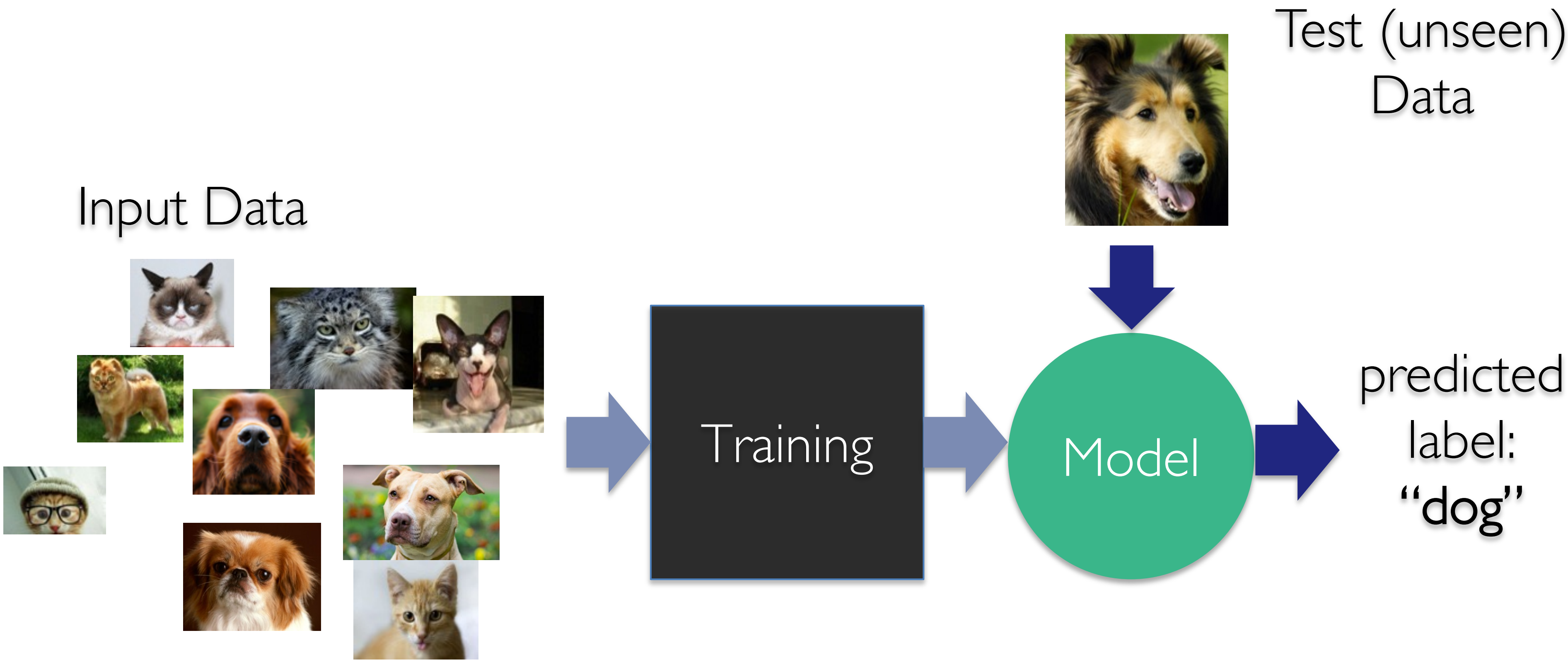
- Parallelism & Comm. Topologies
  - Data parallelism (PS, all-reduce)
  - Model parallelism
  - Pipeline parallelism
  - Hybrid parallelism

- Communication Bottlenecks & Solutions



Model Size (in billions of parameters)
1000
100
10
1
0.1
0.01
2018
2019
2020
2021
2022
GPT-3 (175B)
Megatron-Turing NLG (530B)
Megatron-LM (8.3B)
Turing-NLG (17.2B)
T5 (11B)
GPT-2 (1.5B)
BERT-Large (340M)
ELMo (94M)

A Simplified ML Pipeline



However,
model can contain 1.75 trillion params [1].
+ data set can be of 400 million of data points [2]

[1] *Wu Dao 2.0*
[2] *OpenAI CLIP*

Training a ResNet-50 on ImageNet takes 2.5 days on a single GPU
[EC2 P3.2xlarge (Tesla V100) + PyTorch]

Distributed Model Training

Overview

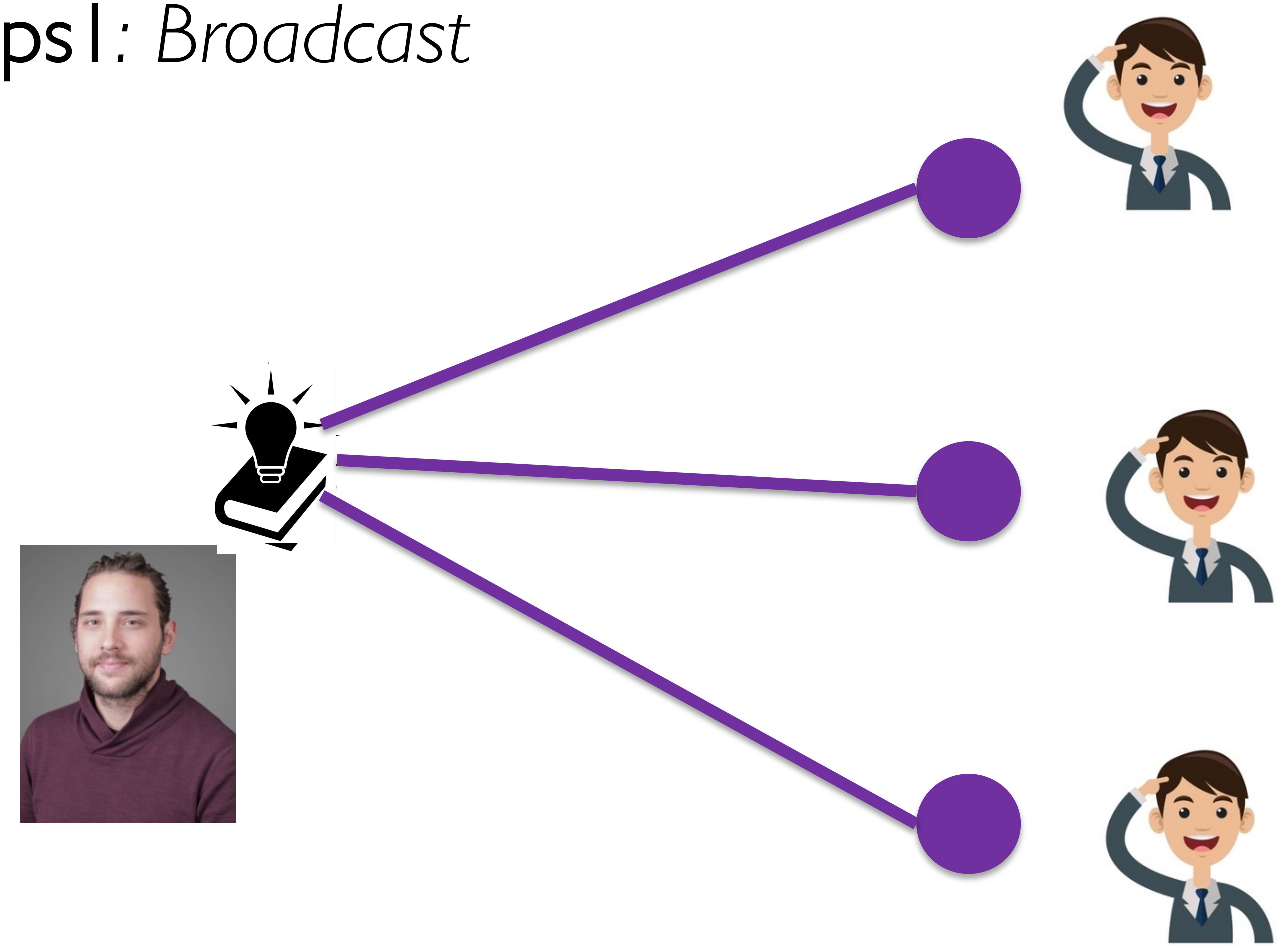
- Motivations of Distributed ML

- Parallelism & Comm. Topologies
  - Data parallelism (PS, all-reduce)
  - Model parallelism
  - Pipeline parallelism
  - Hybrid parallelism

- Communication Bottlenecks & Solutions

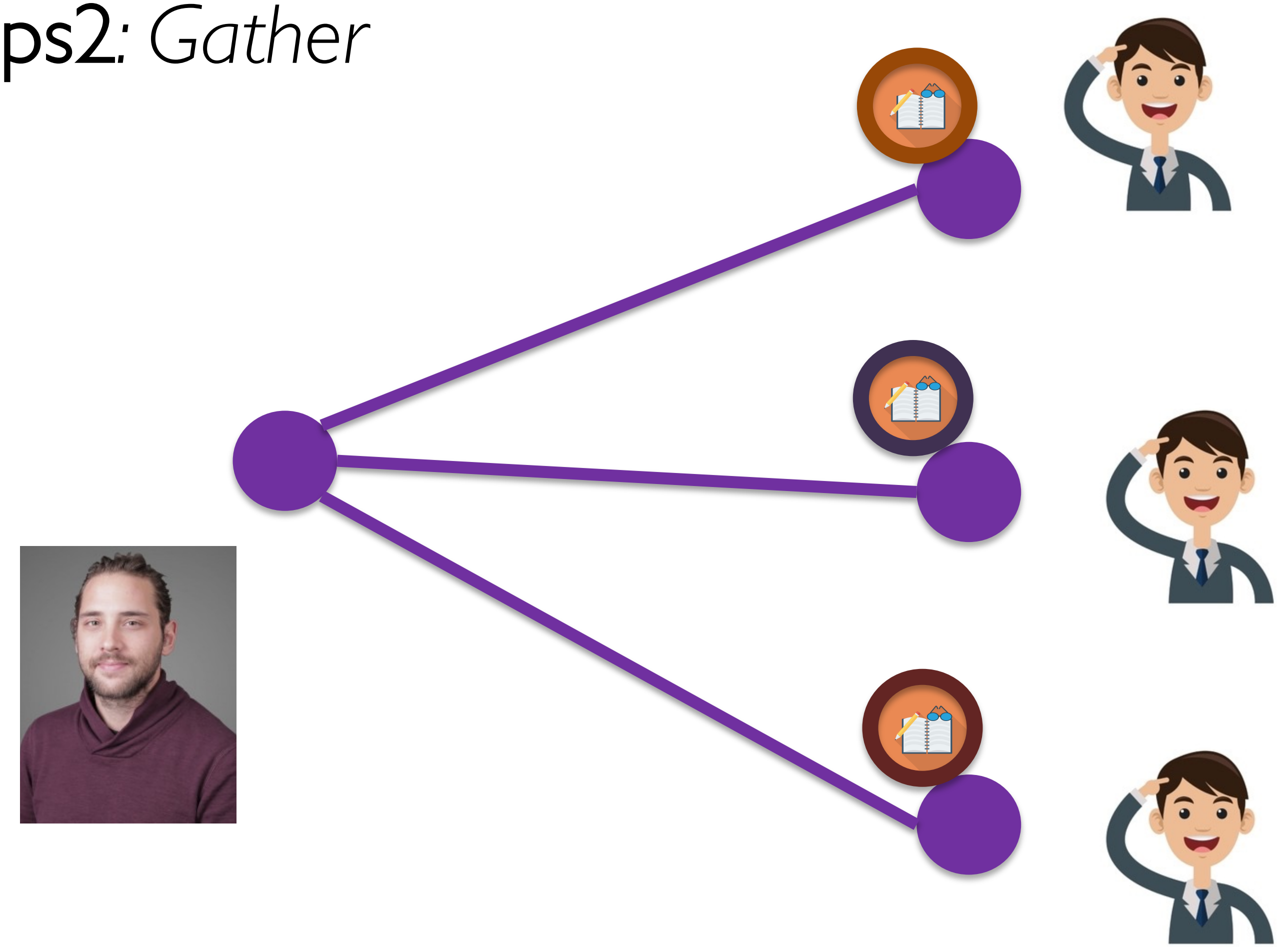
Intro to Comm. Operations

Ops1: *Broadcast*



Intro to Comm. Operations

Ops2: *Gather*



Data-parallel mini-batch SGD

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}; \mathbf{z}_i) \qquad \mathcal{X} = \{z_1, \cdots, z_n\}$$

For iteration t :

- Sample a mini-batch: $\mathcal{B}_t \subset \mathcal{X}$

- Compute gradient: $\nabla \frac{1}{|\mathcal{B}_t|} \sum_{j=1}^{|\mathcal{B}_t|} \ell(w_t; z_j)$

$$= \frac{1}{|\mathcal{B}_t|} \boxed{\sum_{j=1}^{|\mathcal{B}_t|} \nabla \ell(w_t; z_j)}$$

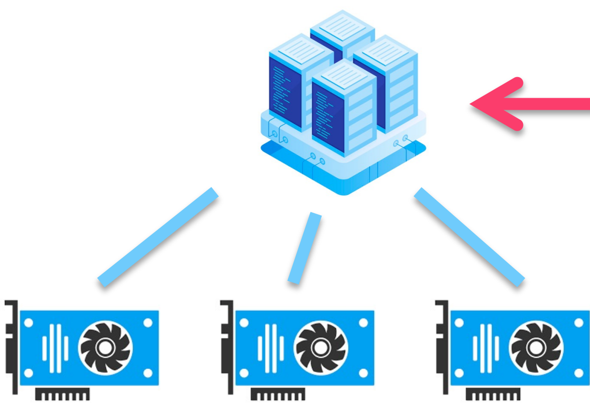
$\nabla \ell(w_t; z_1)$

$\nabla \ell(w_t; z_2)$

$\nabla \ell(w_t; z_3)$

Data-parallel mini-batch SGD

parameter server (PS)

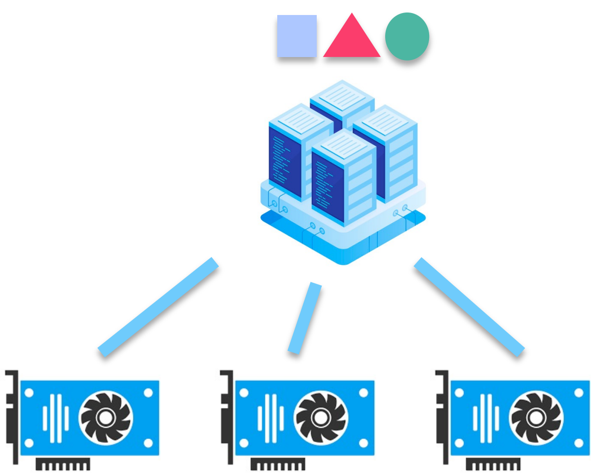


PS owns data, shares with GPUs

GPUs compute gradients

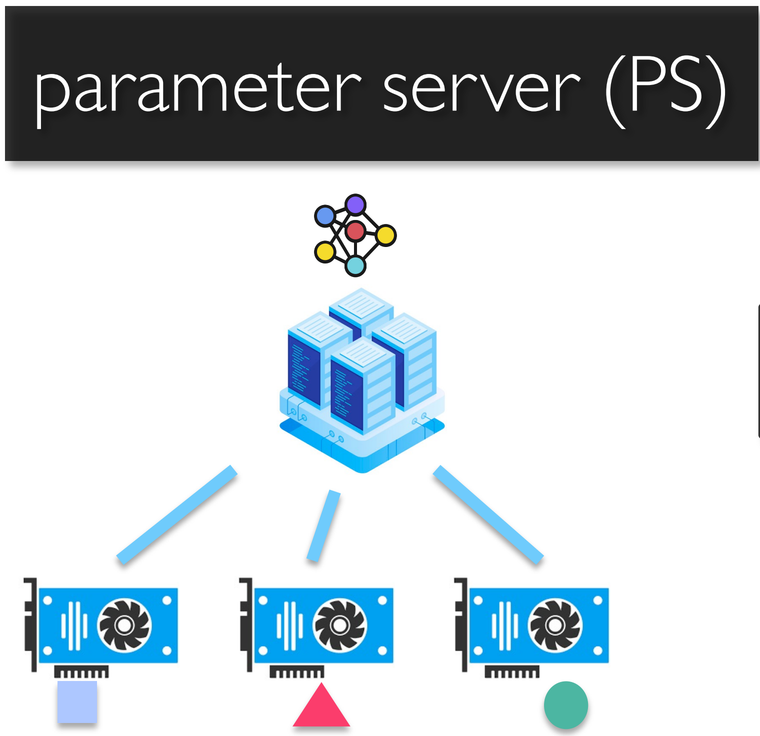
Data-parallel mini-batch SGD

parameter server (PS)



Data Batches Assignment

Data-parallel mini-batch SGD

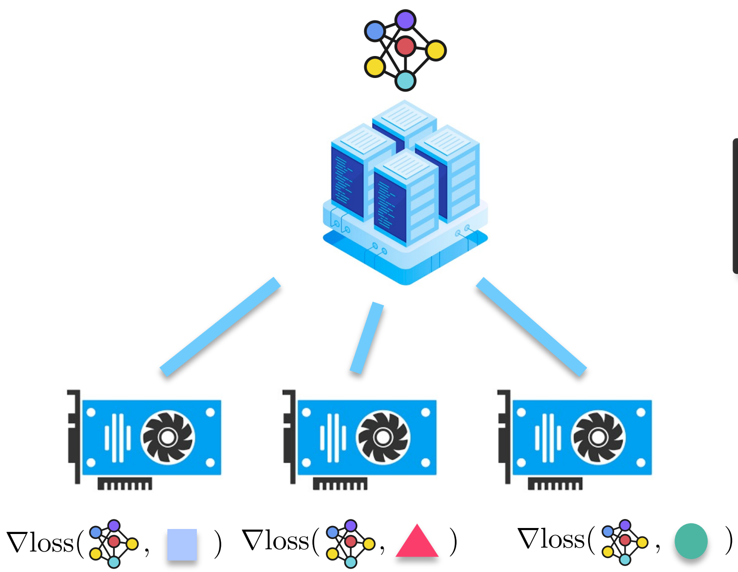


Model Broadcast

Data-parallel mini-batch SGD

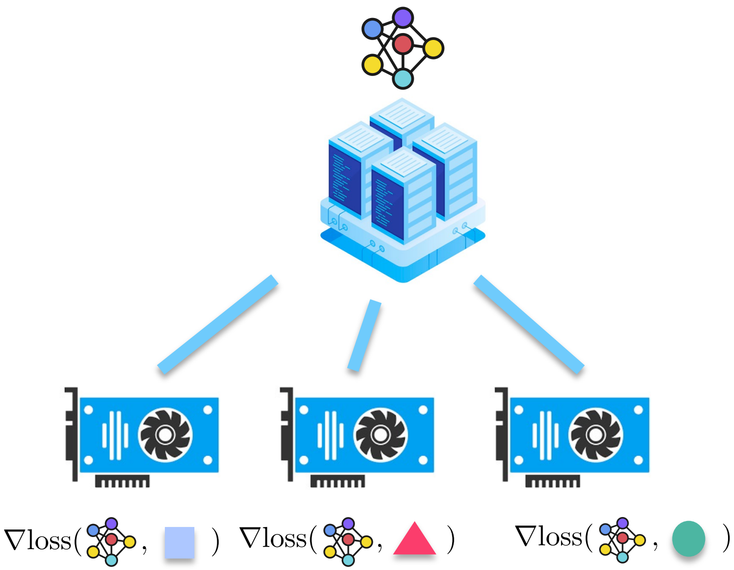
parameter server (PS)

Gradient Gathering



Comm. Cost of PS

parameter server (PS)



1. Model weights and gradients are d-dimensional vectors.

2. There are p GPUs in the cluster for computing.

Both broadcast and gather require to communicate $\mathcal{O}(pd)$ information

PS requires to incur TWO communication operations

A Generalized Comm. Cost Model

$\alpha - \beta$ *Cost model*

Comm. cost = $\beta \times$ Bandwidth cost

Total amount of info communicated.

pd

$+ \ \alpha \times$ Latency cost

The entire number of communication rounds.

2

Sarvotham, et al. Connection-level analysis and modeling of network traffic. SIGCOMM workshop on Internet Measurement 2001.

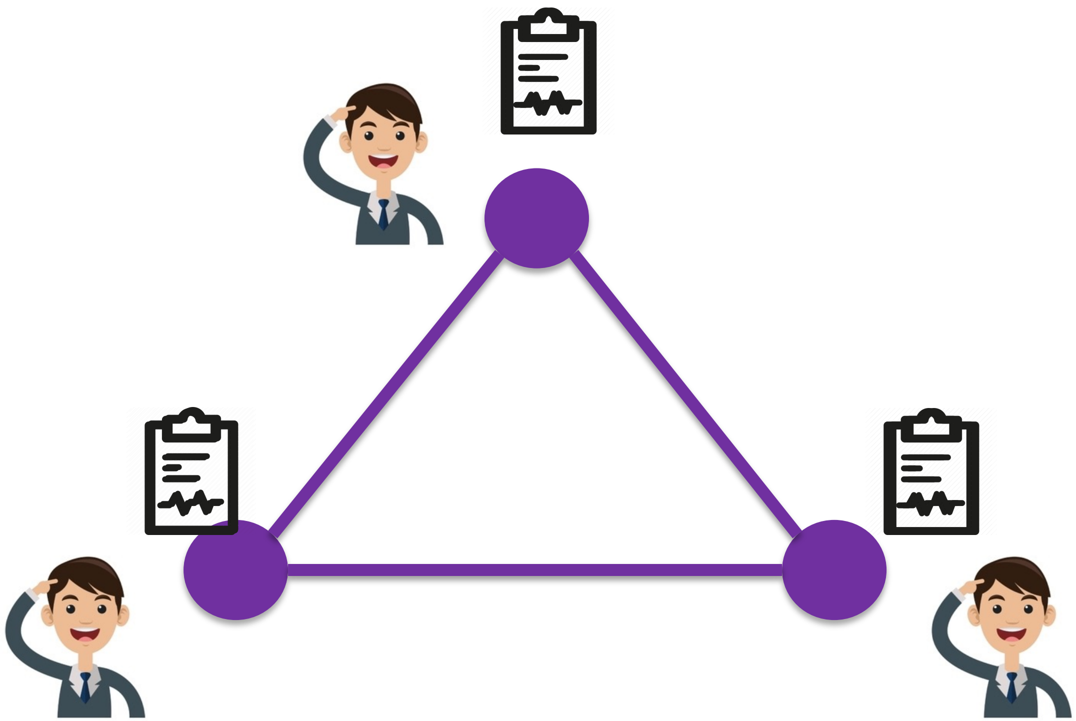
A Generalized Comm. Cost Model

$\alpha - \beta$ *Cost model*

Comm. Topology	Bandwidth	Latency
Param Server	$\beta \times 2pd$	$\alpha \times 2$

Intro to Comm. Operations

Ops3: *All-reduce*



Data-parallel mini-batch SGD

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}; \mathbf{z}_i)$$

For iteration t :

- Sample a mini-batch: $\mathcal{B}_t \subset \mathcal{X}$

- Compute gradient: $\nabla \frac{1}{|\mathcal{B}_t|}\sum_{j=1}^{|\mathcal{B}_t|} \ell(w_t; z_j)$

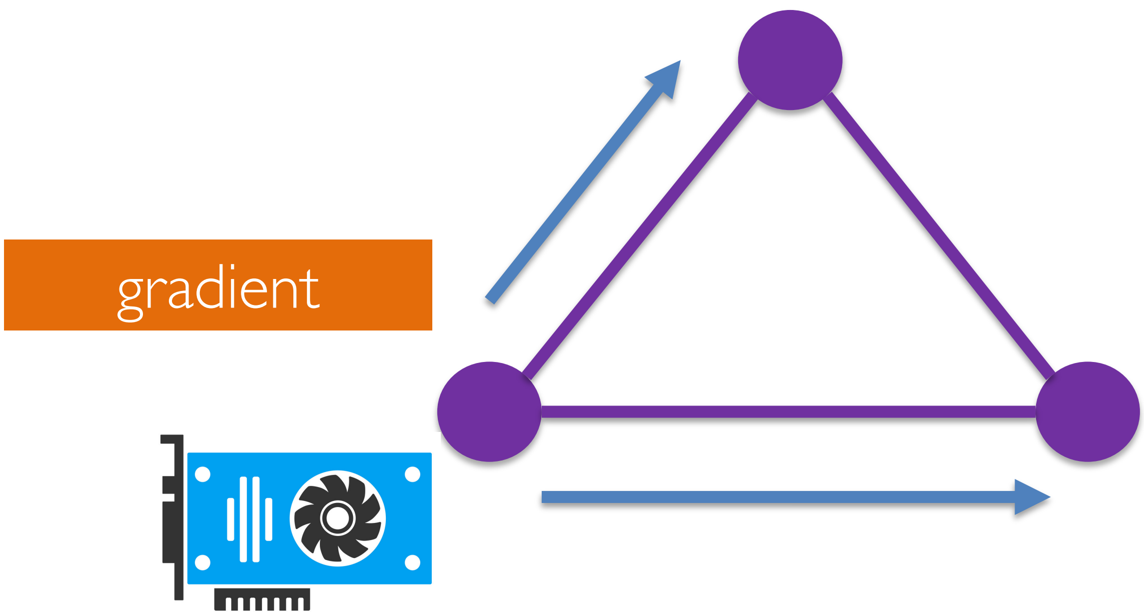
$$= \frac{1}{|\mathcal{B}_t|}\boxed{\sum_{j=1}^{|\mathcal{B}_t|} \nabla\ell(w_t; z_j)}$$

$\nabla\ell(w_t; z_1)$

$\nabla\ell(w_t; z_2)$

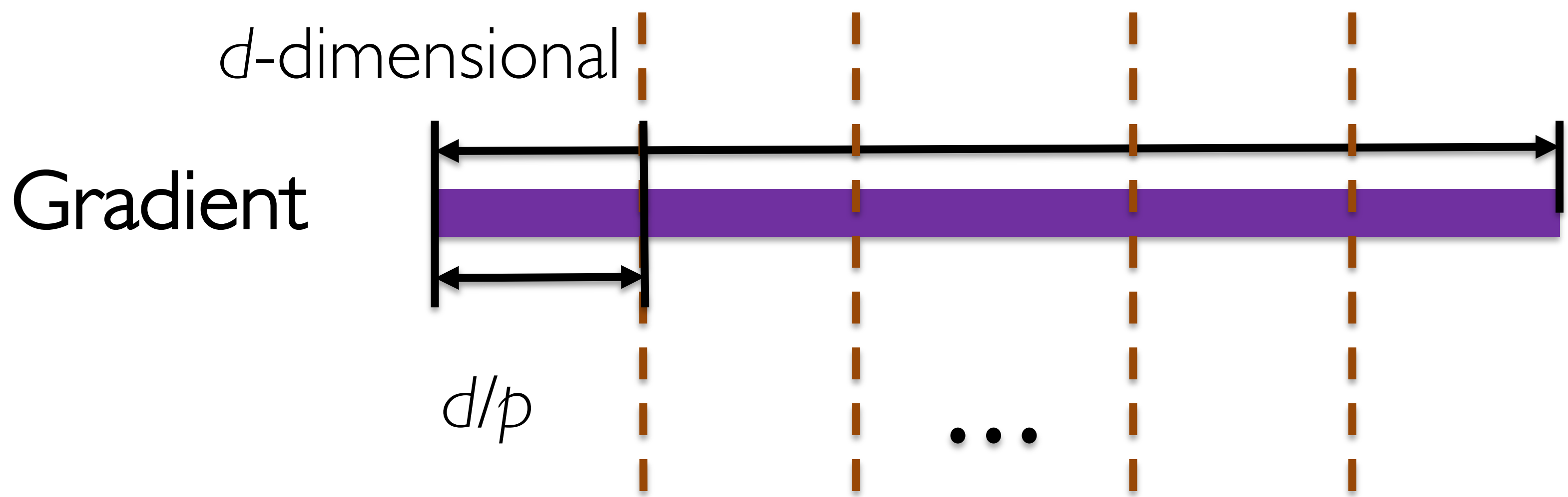
$\nabla\ell(w_t; z_3)$

All-reduce SGD



Data-parallel SGD

Ring-reduce SGD

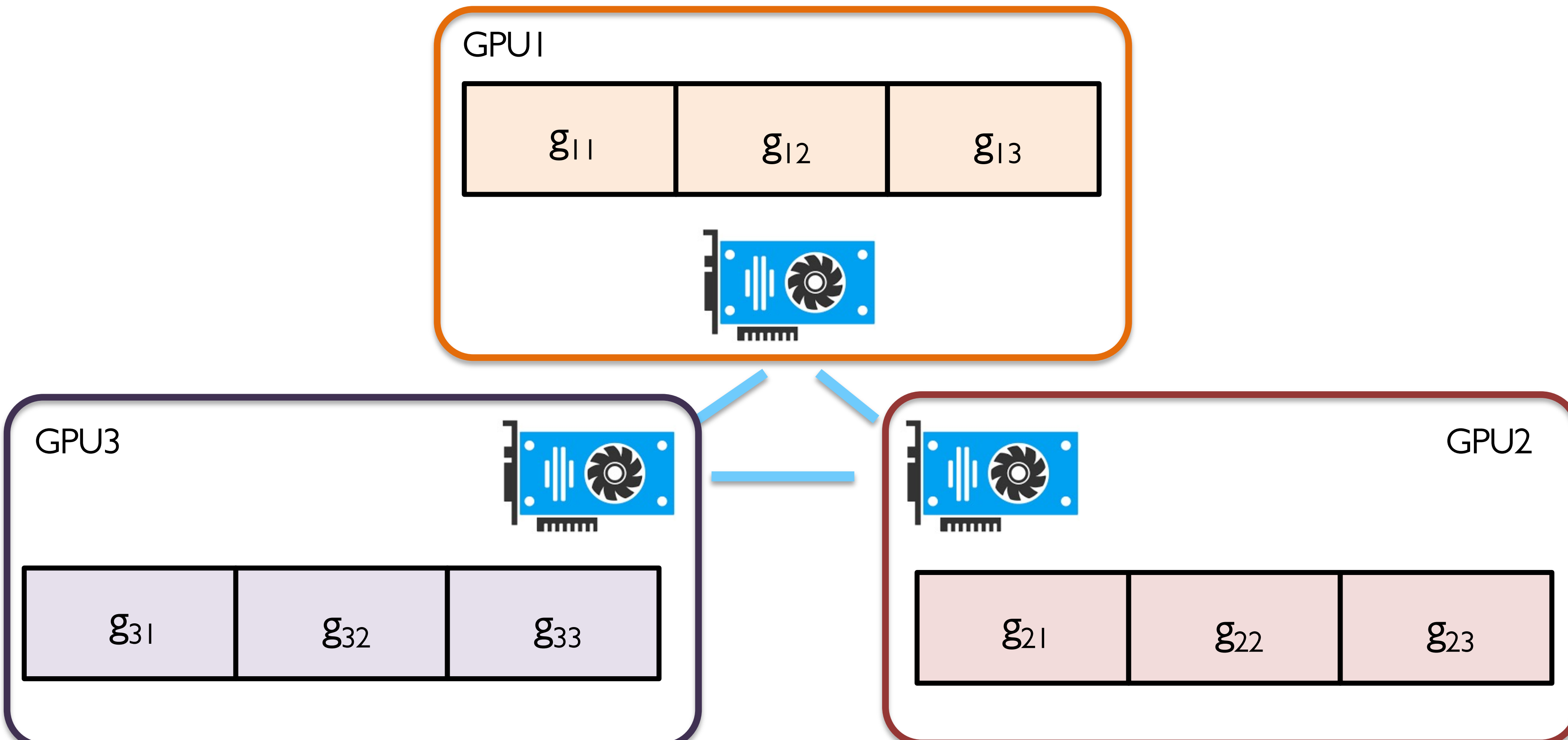


p GPUs in the cluster

Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

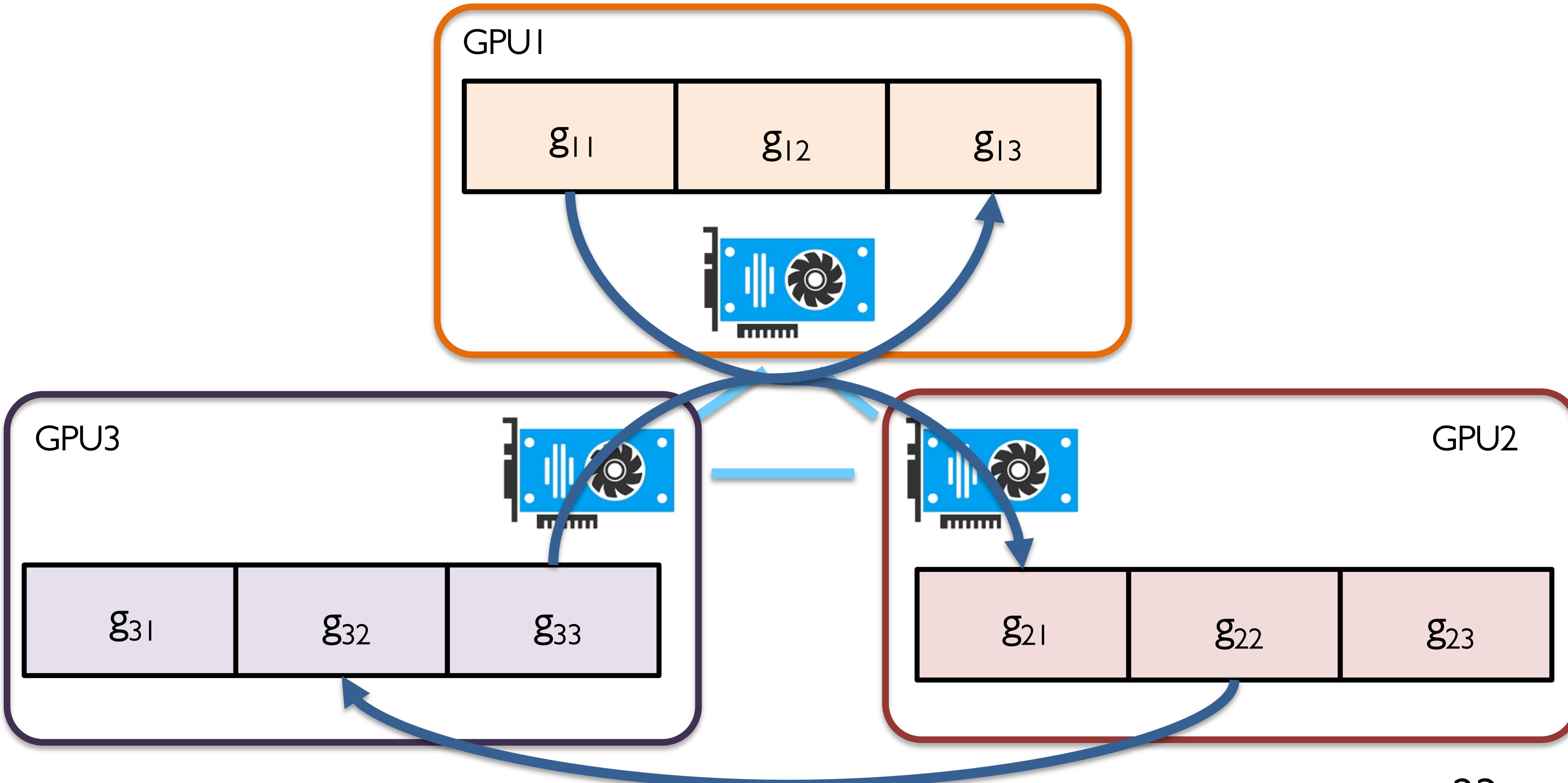
Initialization



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

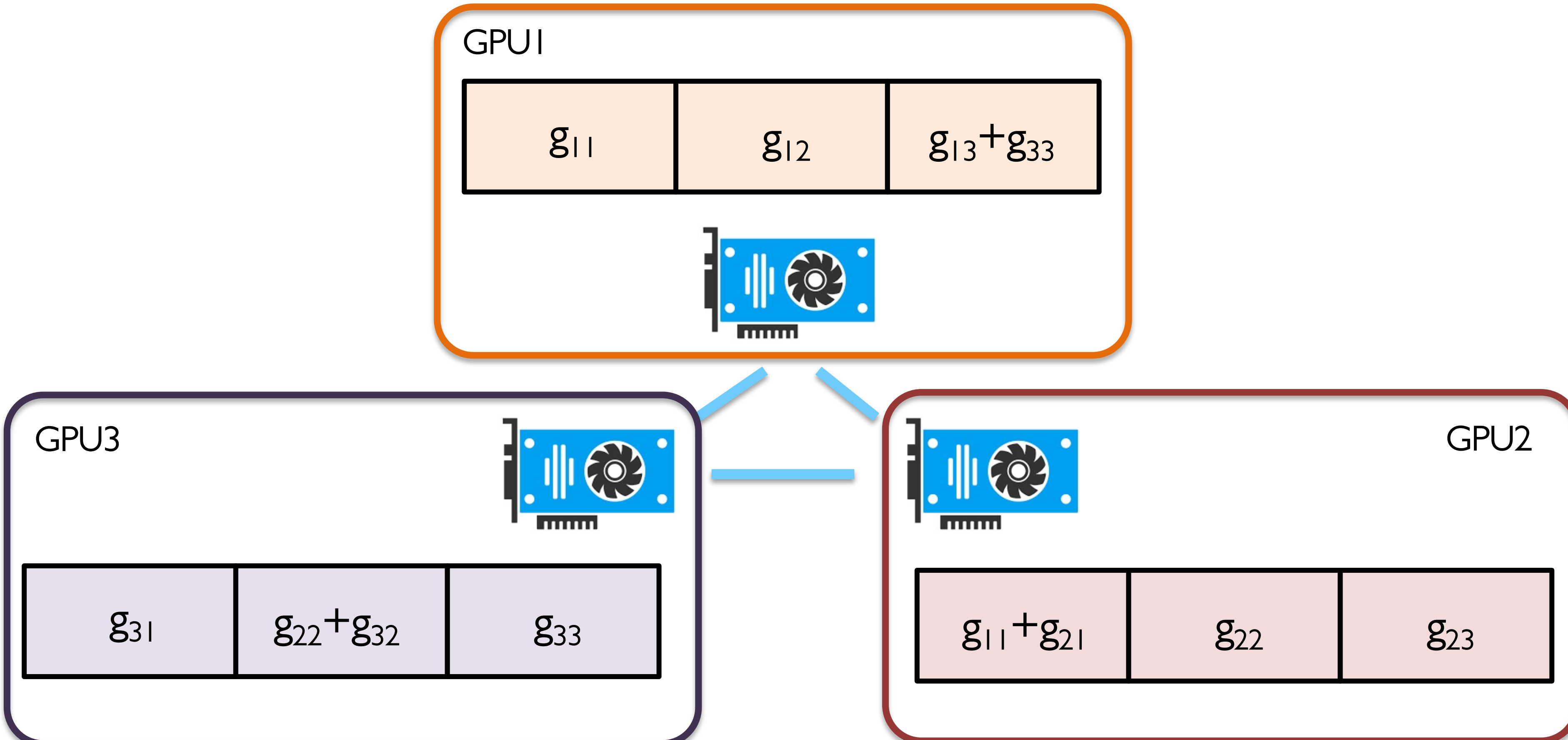
Round 1



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

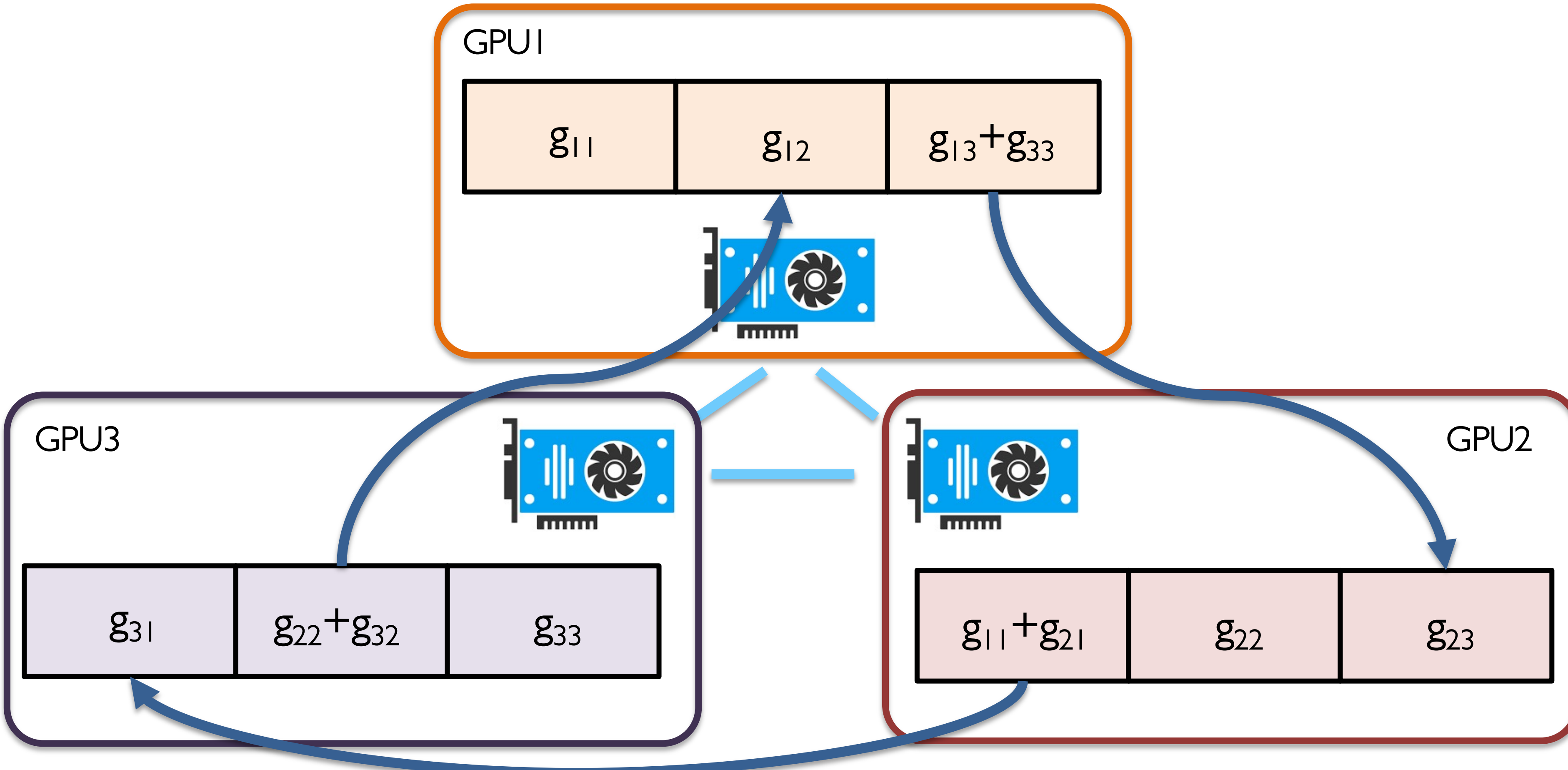
Round 1



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

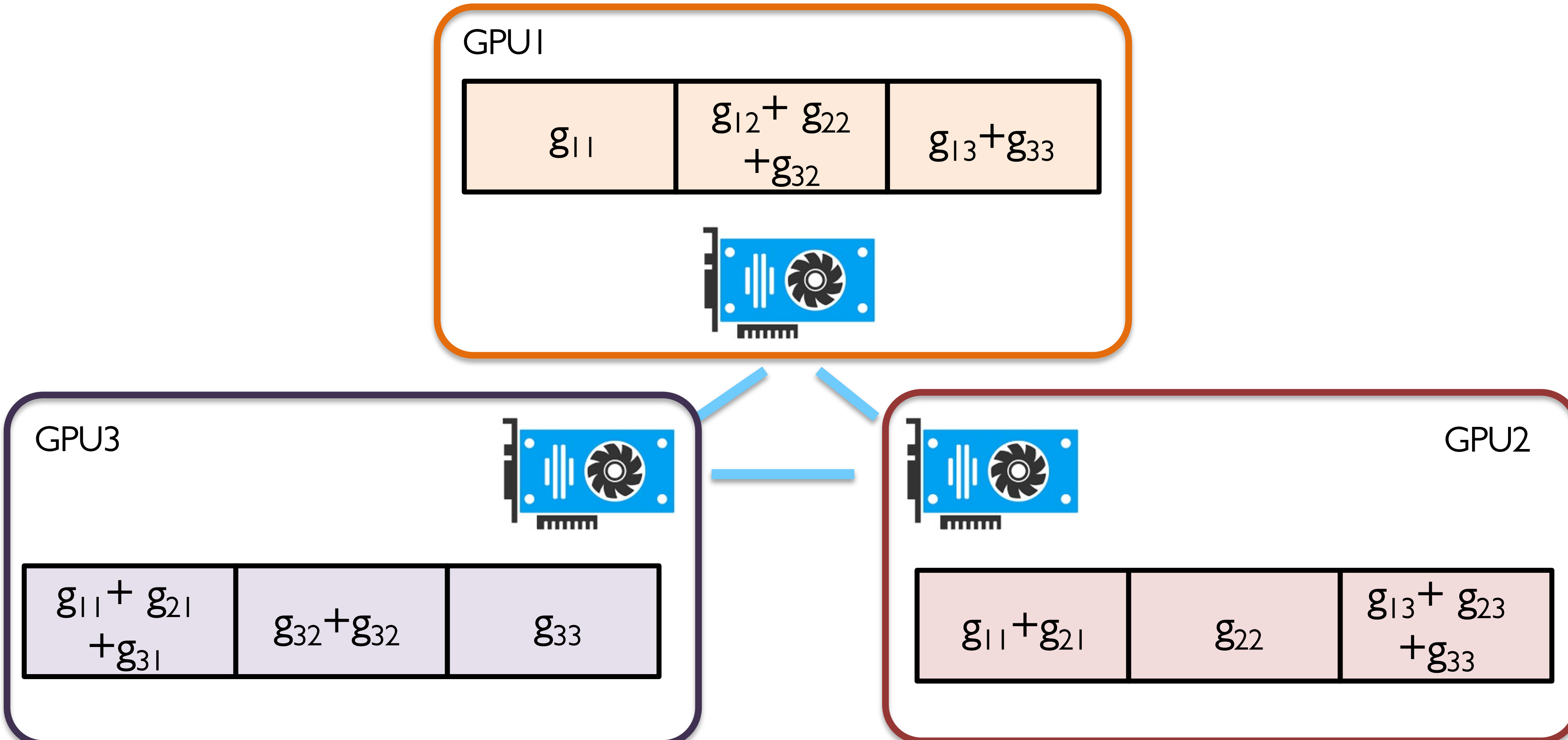
Round 2



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

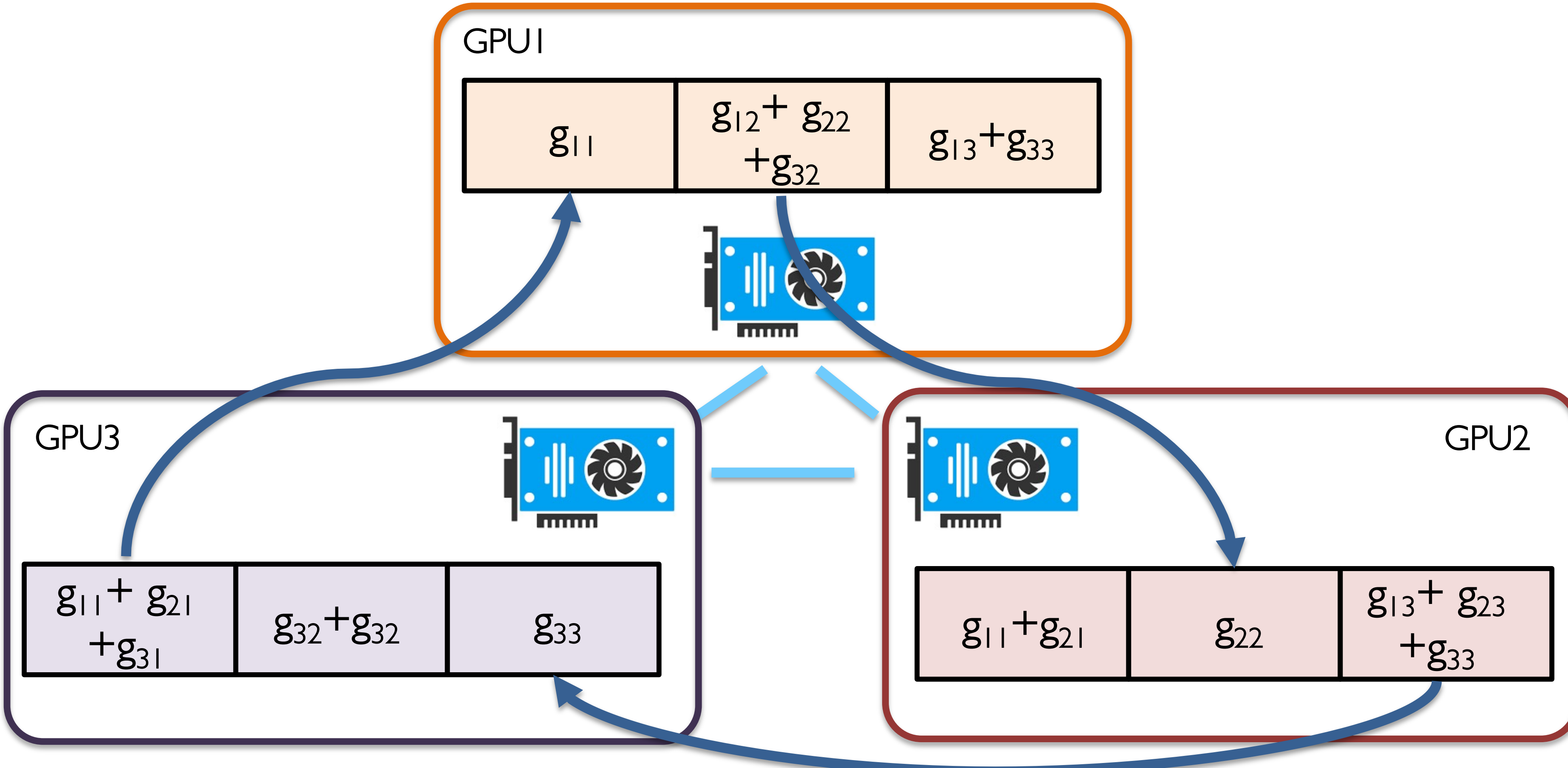
Round 2



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

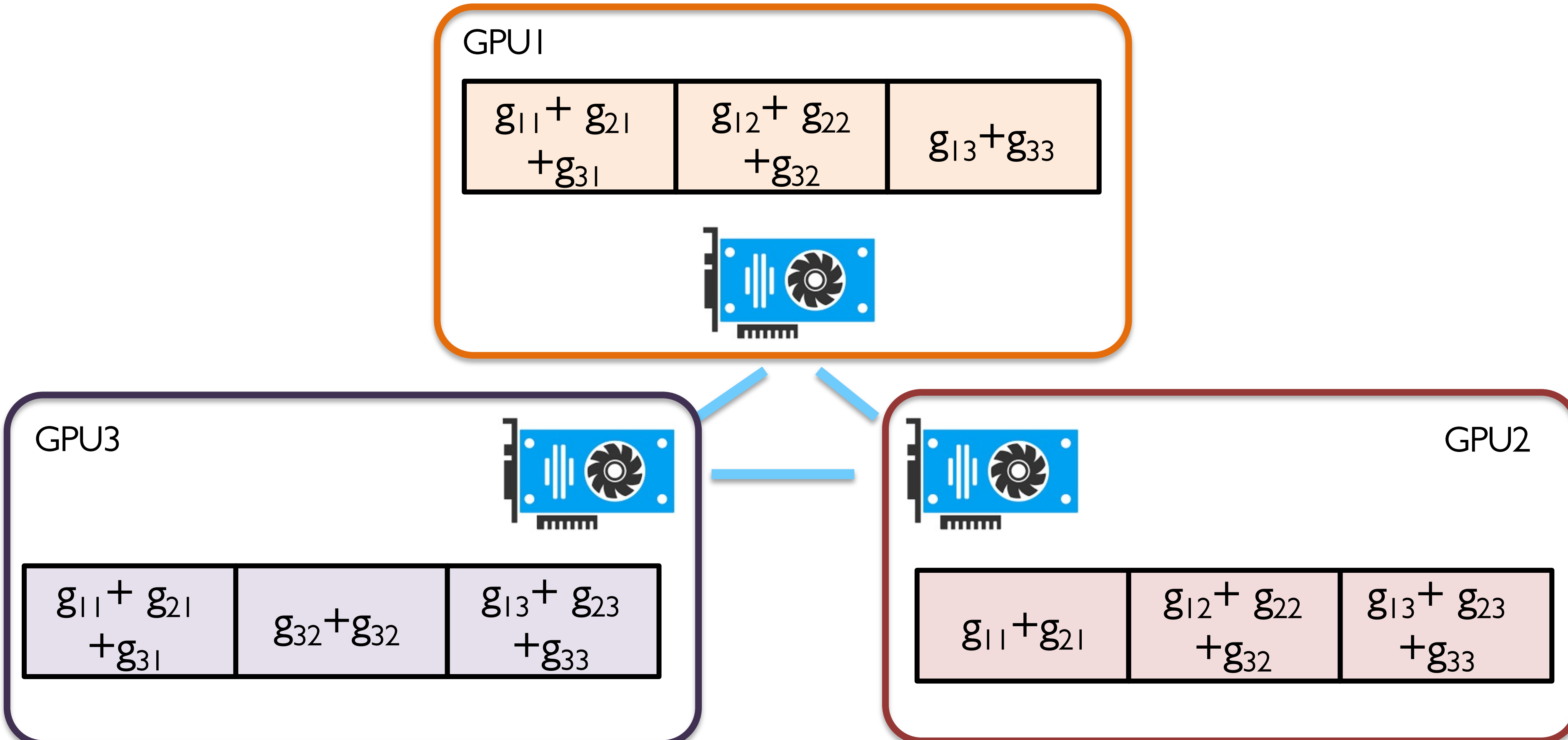
Round 3



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

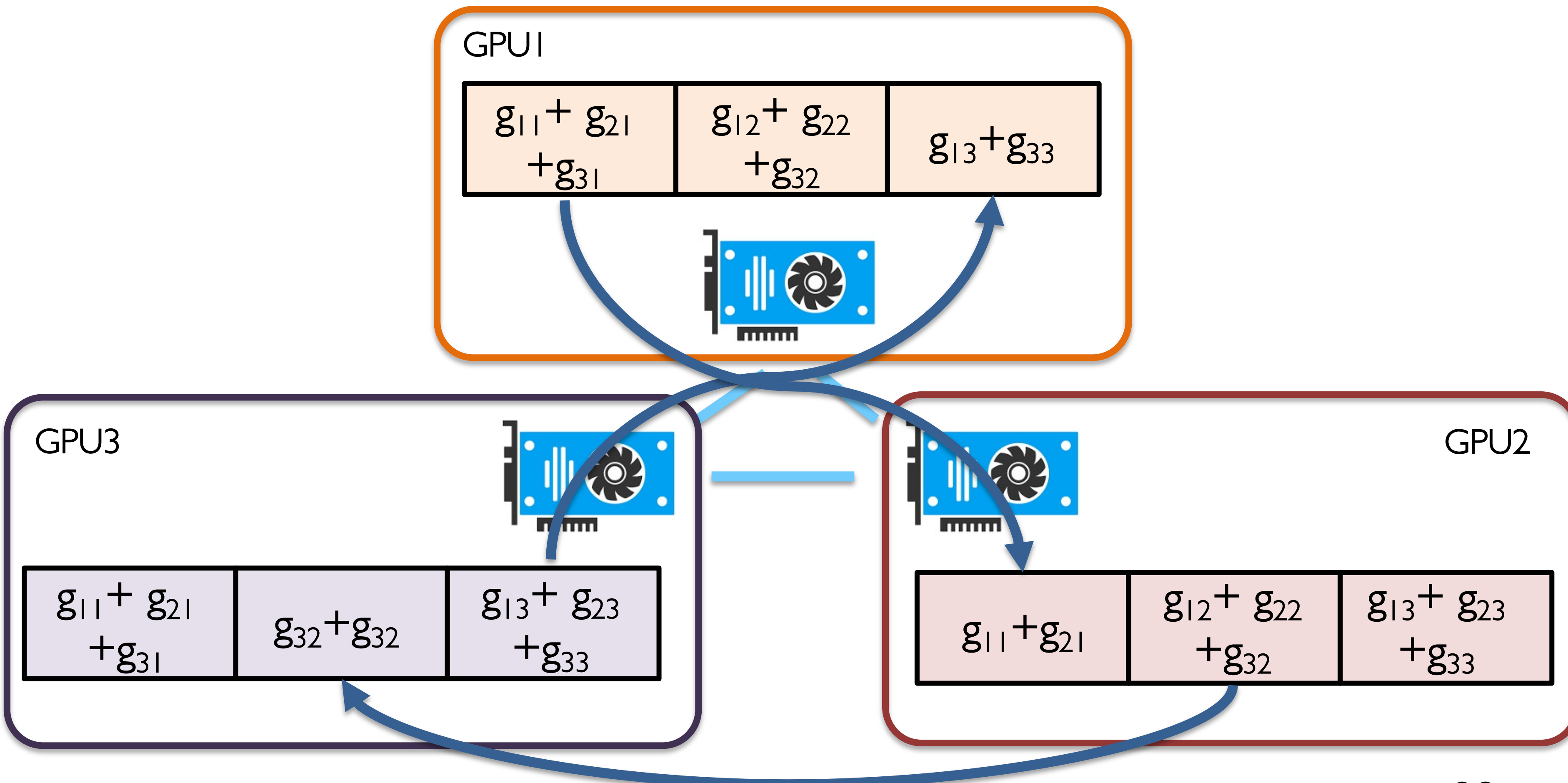
Round 3



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

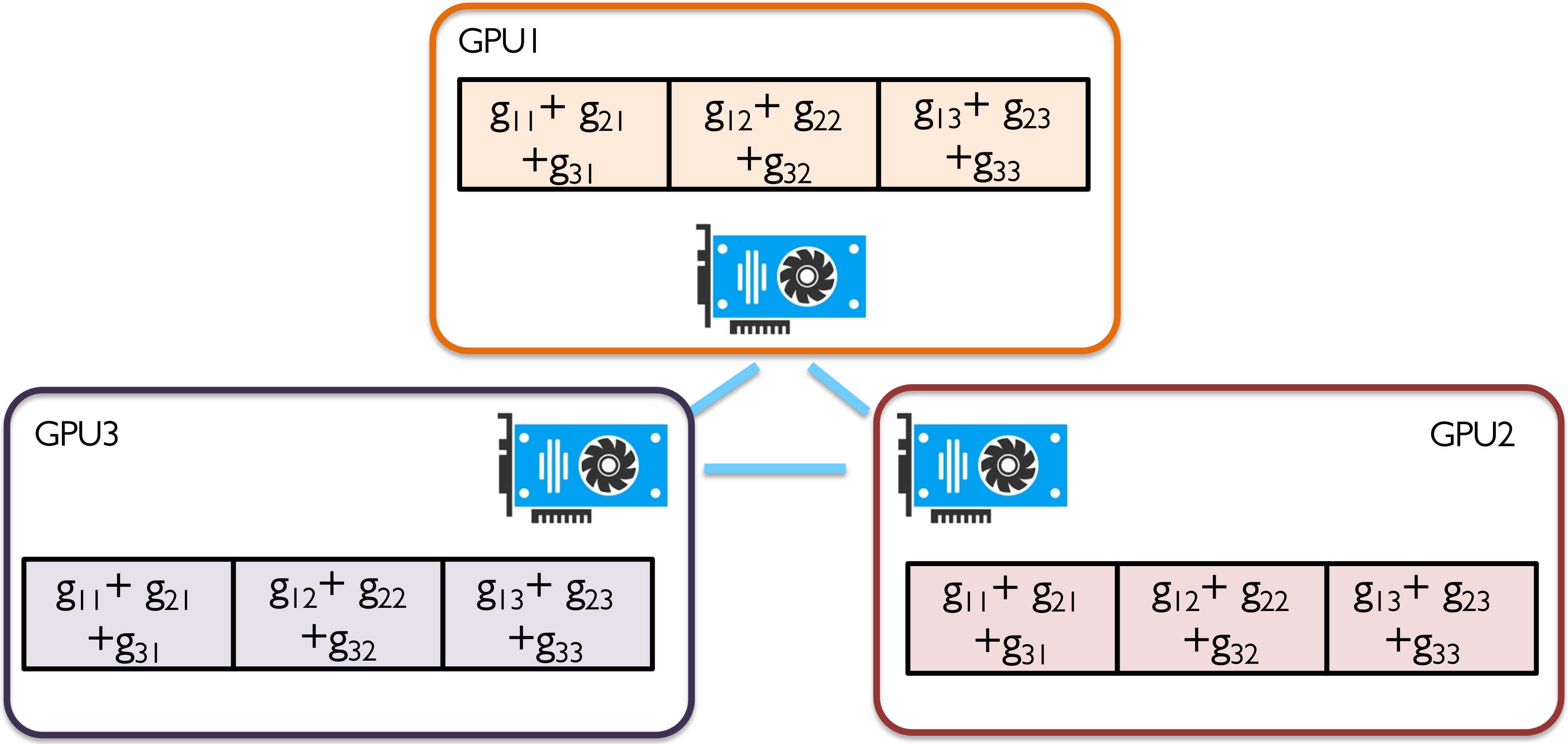
Round 4



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

Round 4



Sergeev, et al. Horovod: fast and easy distributed deep learning in TensorFlow

Ring-reduce SGD

$\alpha - \beta$ *Cost model*

Comm. cost = $\beta \times$ Bandwidth cost

Amount of info communicated per communication round

$$2\frac{d}{p}\cdot(p-1)$$

$+\ \alpha \times$ Latency cost

The entire number of communication rounds

$$2(p-1)$$

A Generalized Comm. Cost Model

$\alpha - \beta$ *Cost model*

Comm. Topology	Bandwidth	Latency
Param Server	$\beta \times 2pd$	$\alpha \times 2$
Ring-reduce	$\beta \times \frac{2(p-1)d}{p}$	$\alpha \times 2(p-1)$

Overview

- Motivations of Distributed ML
- Parallelism & Comm. Topologies
  - Data parallelism (PS, all-reduce)
  - <u>Model parallelism</u>
  - Pipeline parallelism
  - Hybrid parallelism
- Communication Bottlenecks & Solutions

An Issue of Data-parallelism

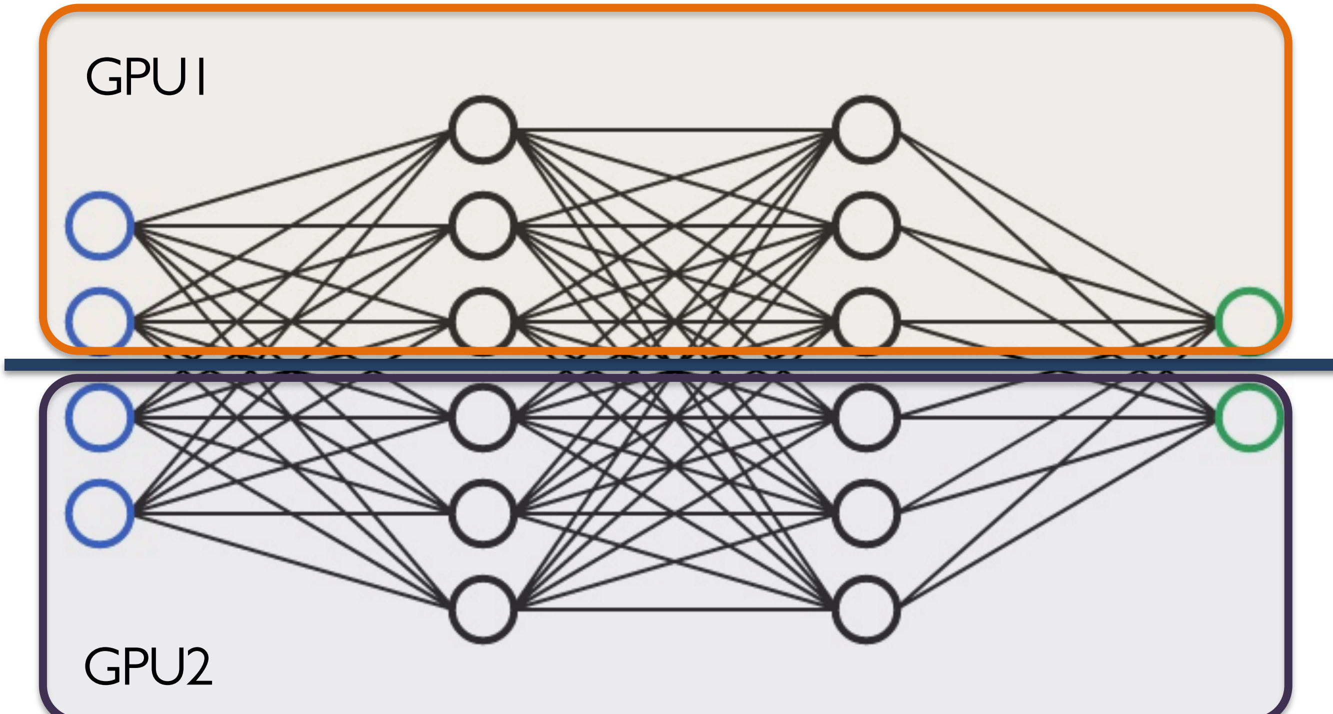
Each GPU must hold the entire copy of model parameters.

Solution:
Partition the model parameters among GPUs

- The best GPU on earth (Nvidia A100) has only 80GB of GPU memory.

Model Parallelism

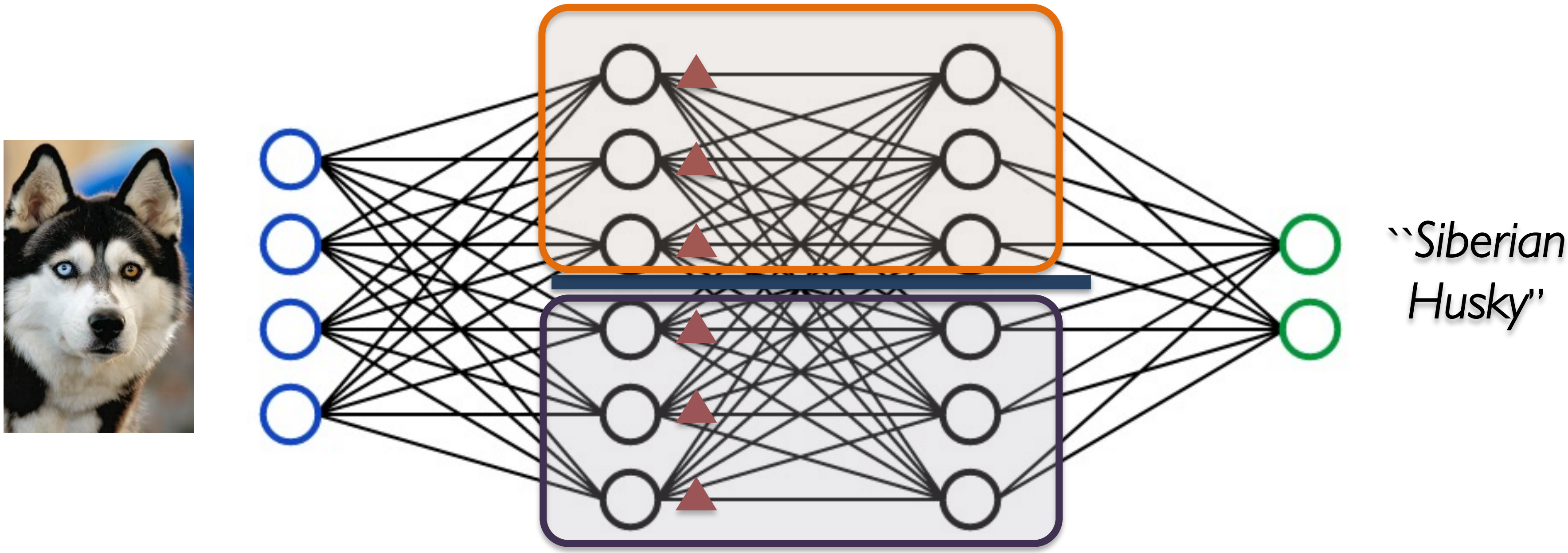
Also known as tensor-model parallelism

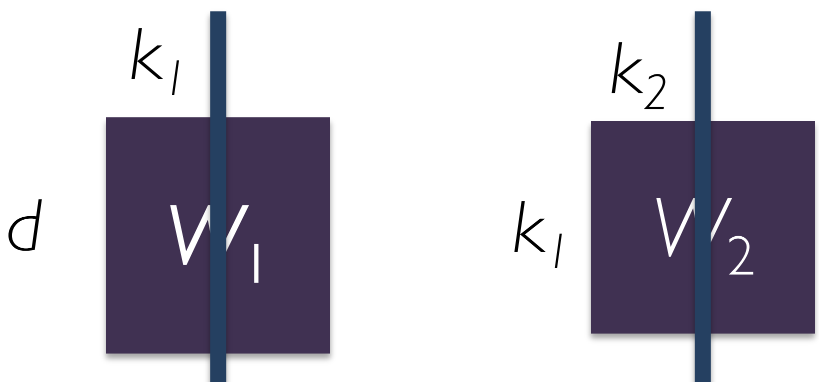


``Siberian Husky''

Model Parallelism

Also known as tensor-model parallelism





Model Parallelism

Also known as tensor-model parallelism

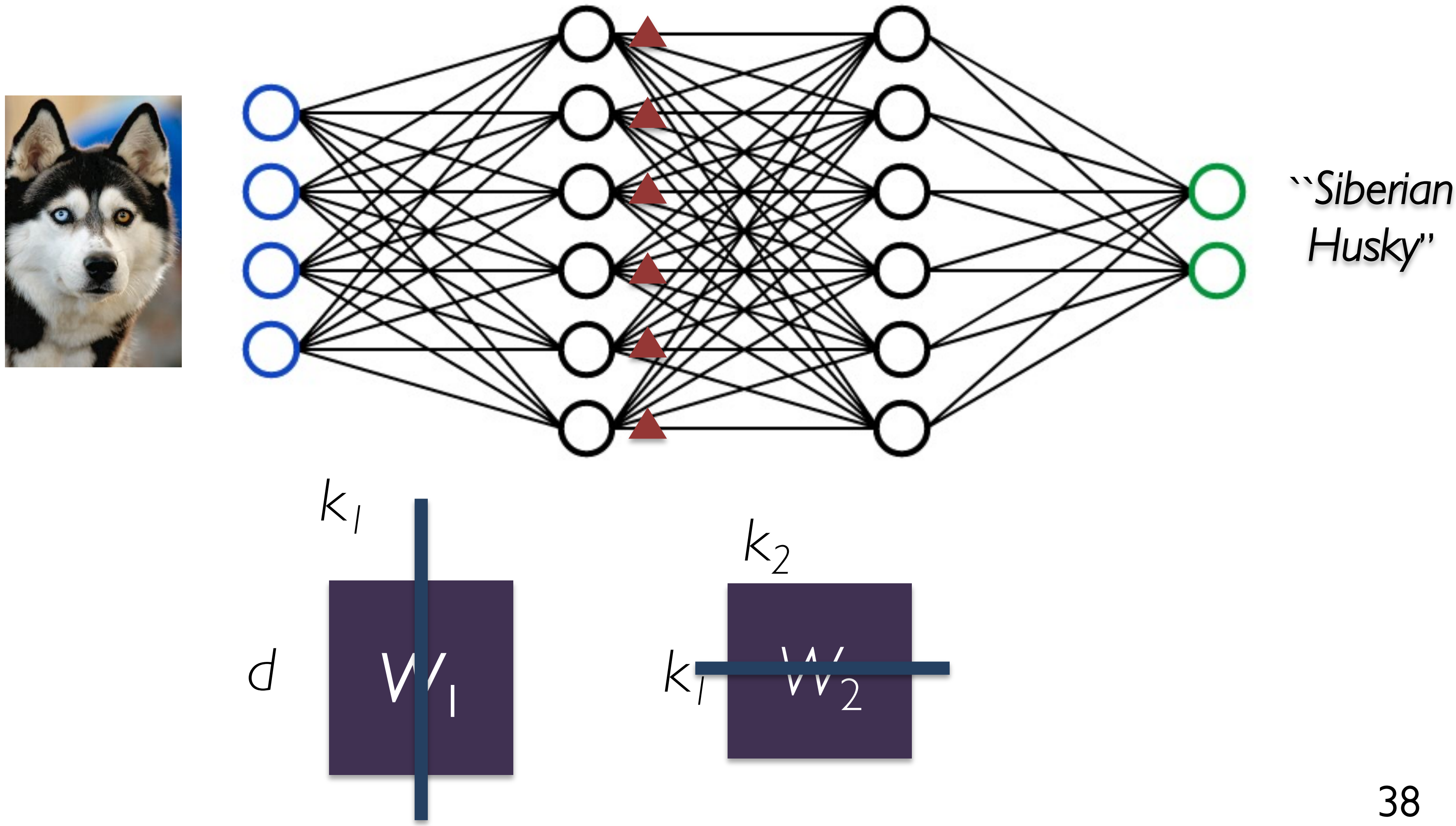
$$\left[\begin{array}{c|c} \blacktriangle & \bullet \end{array}\right]$$

$$\begin{bmatrix} a & b \end{bmatrix} \quad \left[\begin{array}{c|c} c & d \\ e & f \end{array}\right] \quad \left[\begin{array}{c|c} g & h \\ i & j \end{array}\right]$$

It requires to communicate per layer

Model Parallelism

Also known as tensor-model parallelism



Model Parallelism

Also known as tensor-model parallelism

$$\begin{bmatrix} a & b \end{bmatrix}$$

$$ac + be \qquad ad + bf$$

$$\begin{bmatrix} c & d \\ e & f \end{bmatrix}$$

$$\begin{bmatrix} acg + beg, ach + beh \end{bmatrix}$$

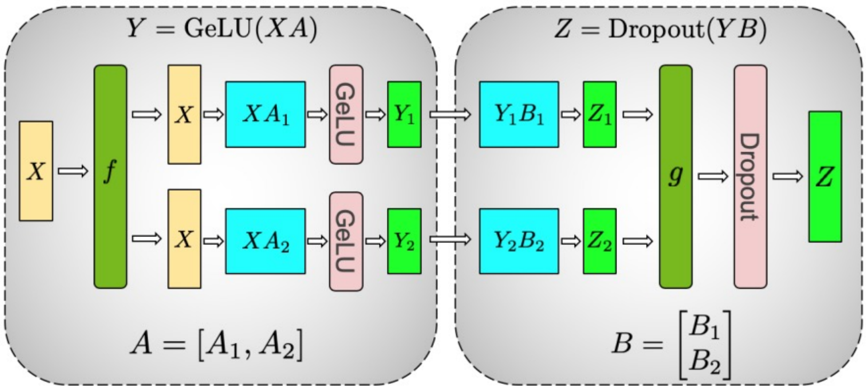
$$\begin{bmatrix} g & h \\ i & j \end{bmatrix}$$

$$\begin{bmatrix} adi + bfi, adj + bfj \end{bmatrix}$$

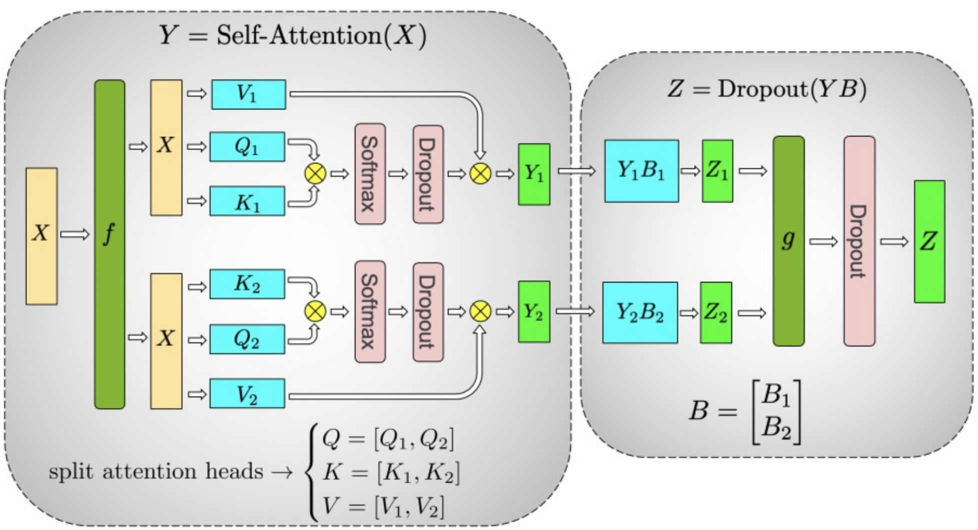
It requires communication per two layers

Model Parallelism

Also known as tensor-model parallelism



MLP

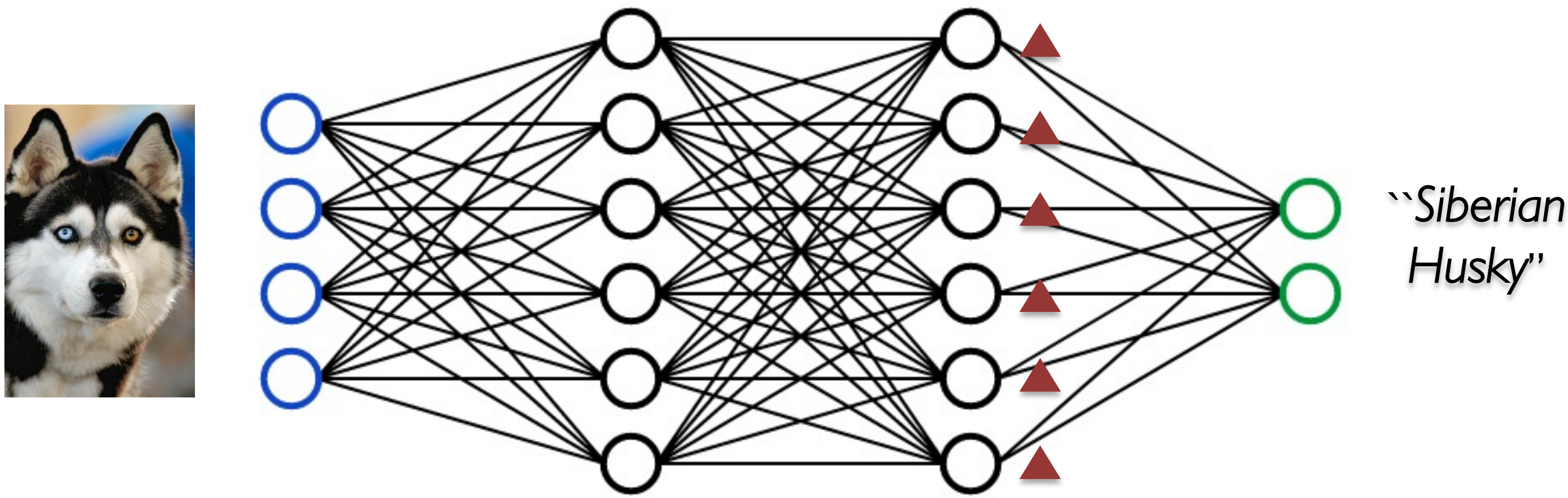


Self-Attention

Shoeybi, et al. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.

An Issue of Model Parallelism

Amount of comm. scales with # layers and batch size.

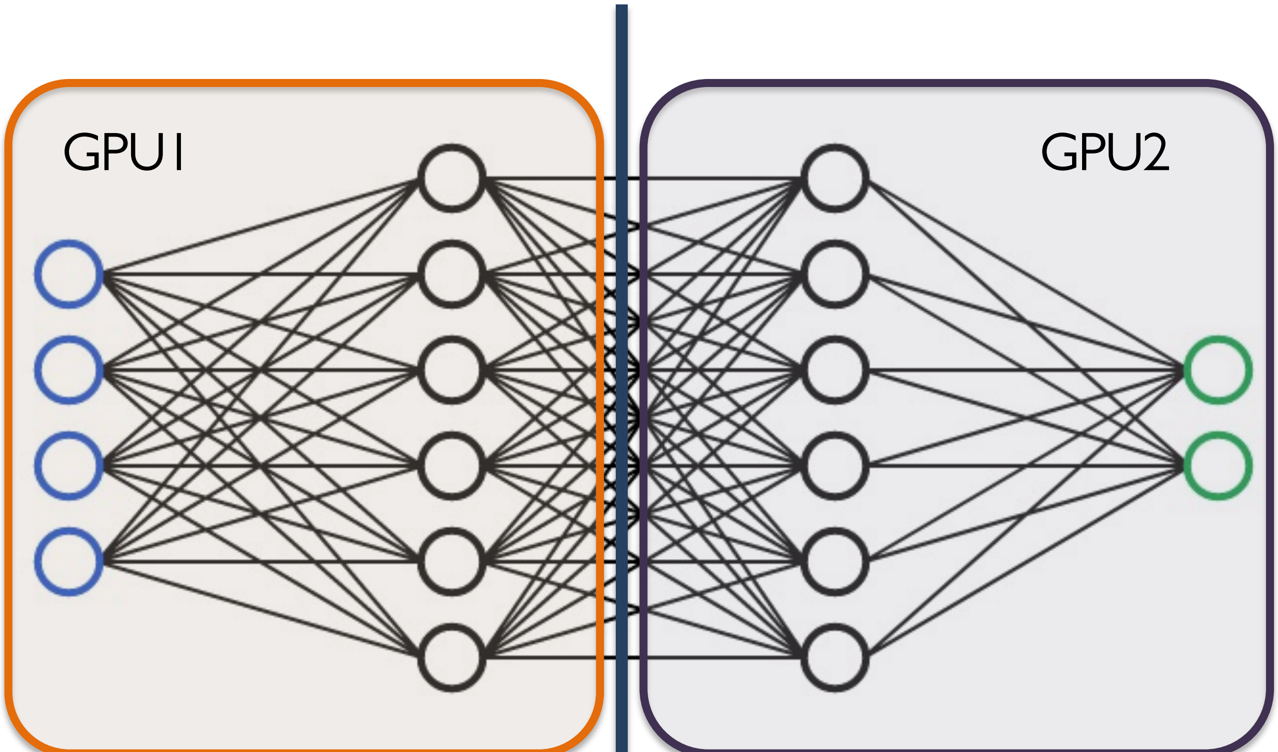


(batch size, hidden dim)

Overview

- Motivations of Distributed ML
- Parallelism & Comm. Topologies
  - Data parallelism (PS, all-reduce)
  - Model parallelism
  - Pipeline parallelism
  - Hybrid parallelism
- Communication Bottlenecks & Solutions

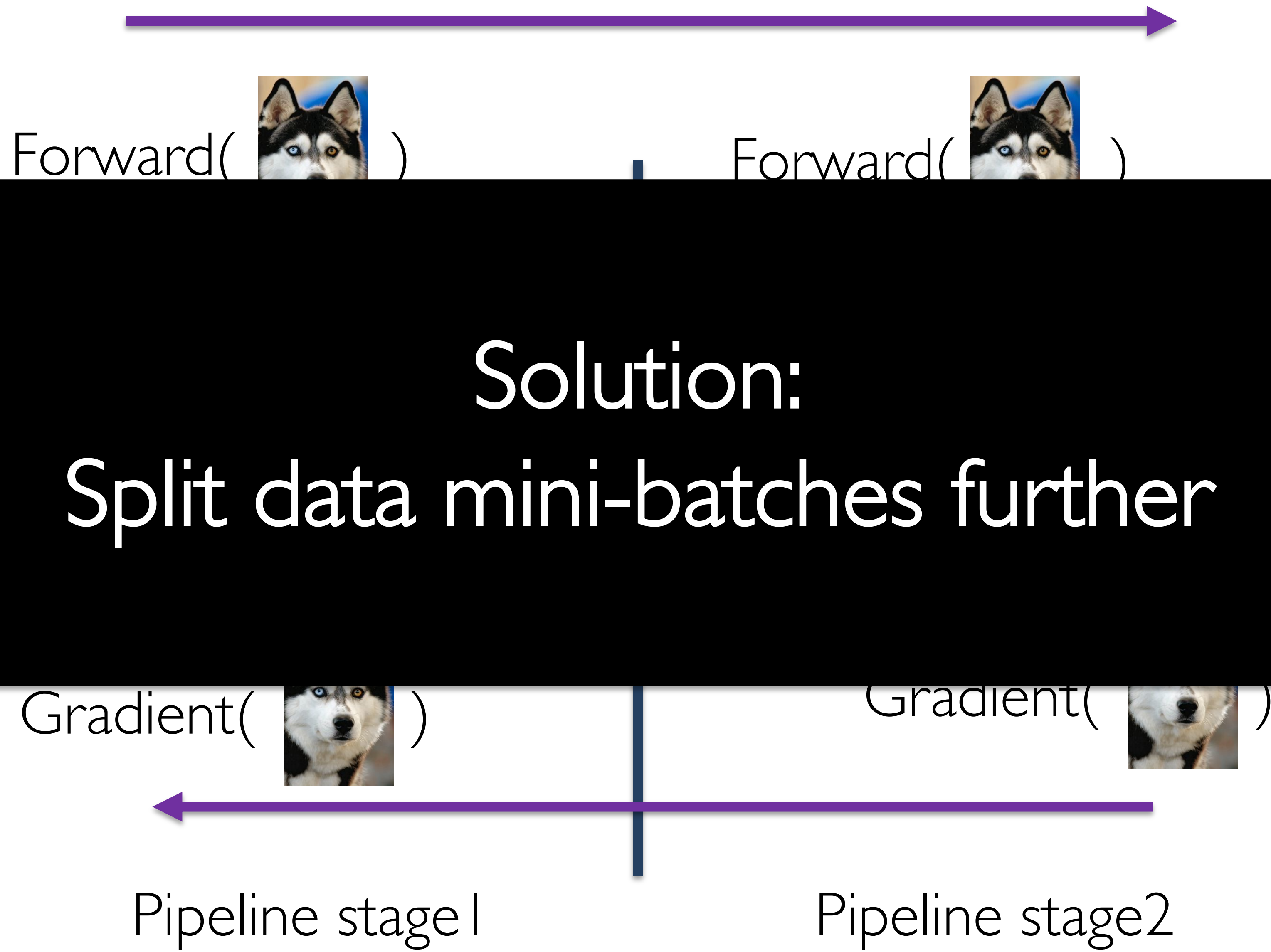
Pipeline Parallelism



``Siberian Husky''

Pipeline stage1 | Pipeline stage2

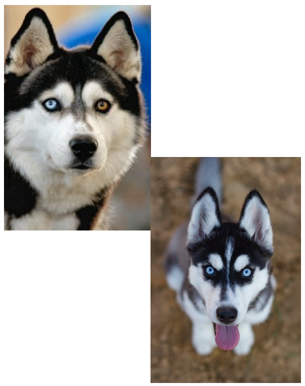
Pipeline Parallelism

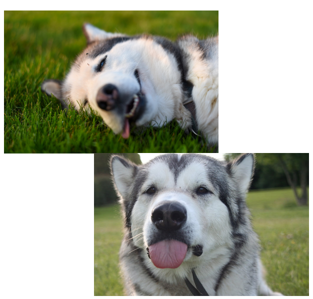


Solution:
Split data mini-batches further

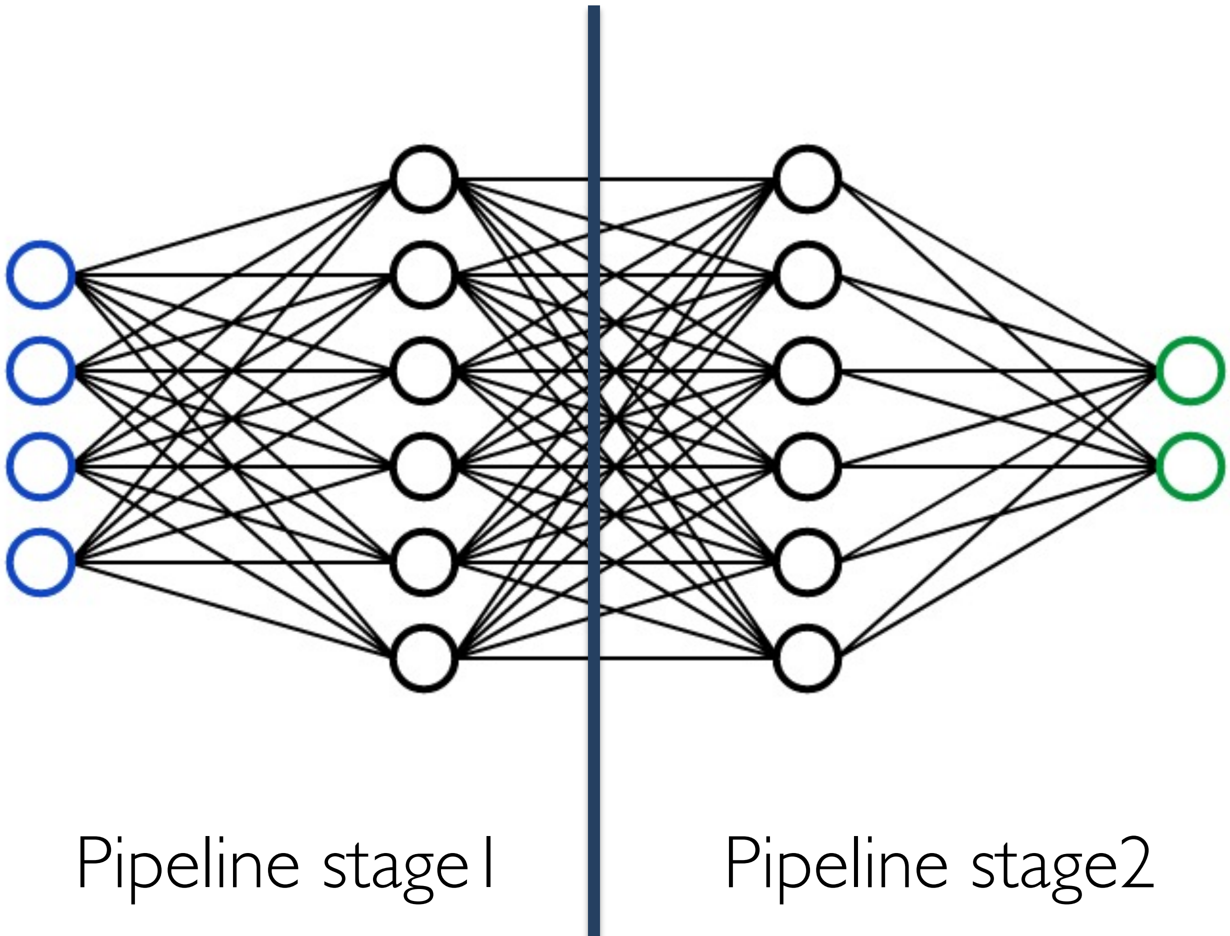
Pipeline Parallelism

Micro-batch1

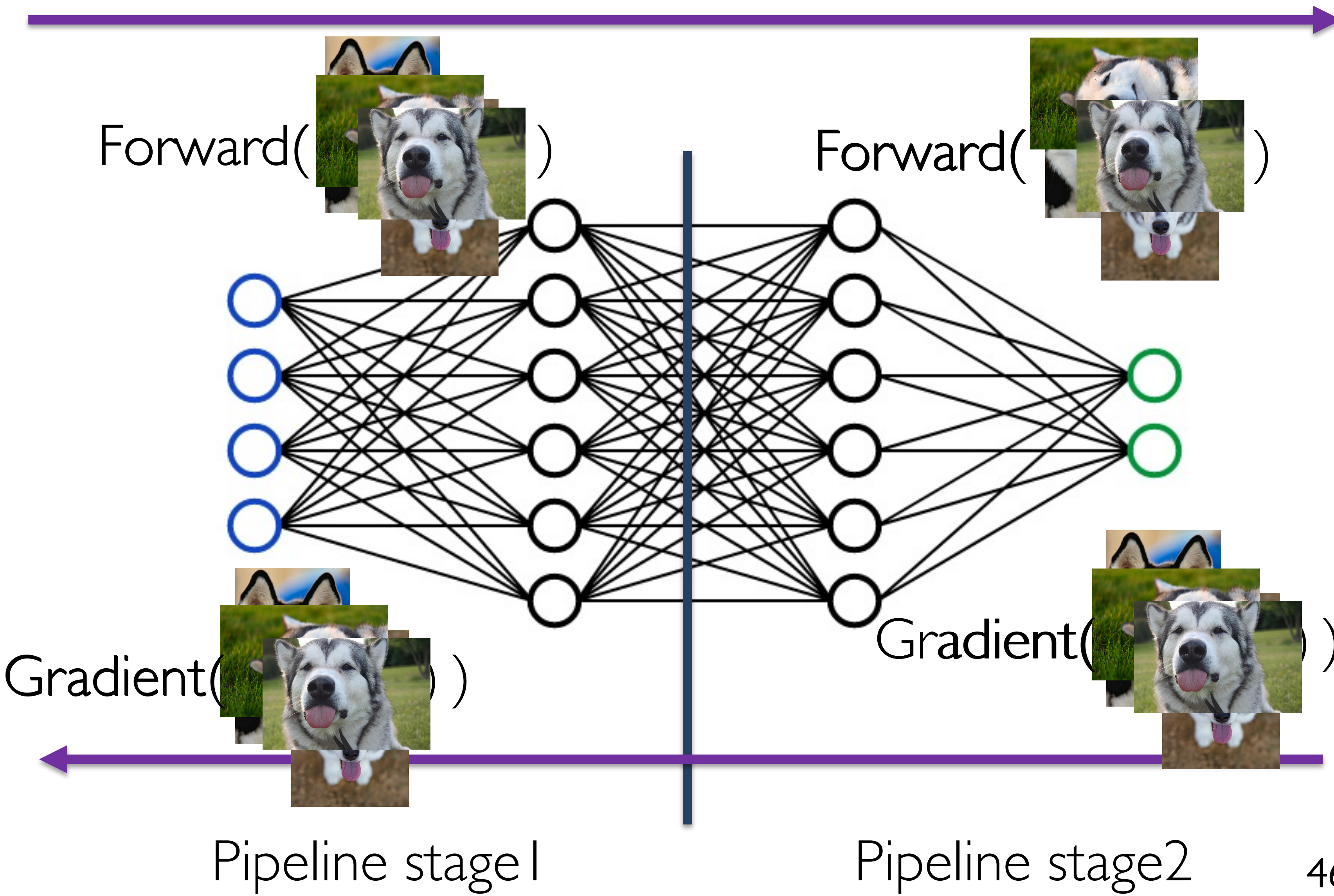




Micro-batch2

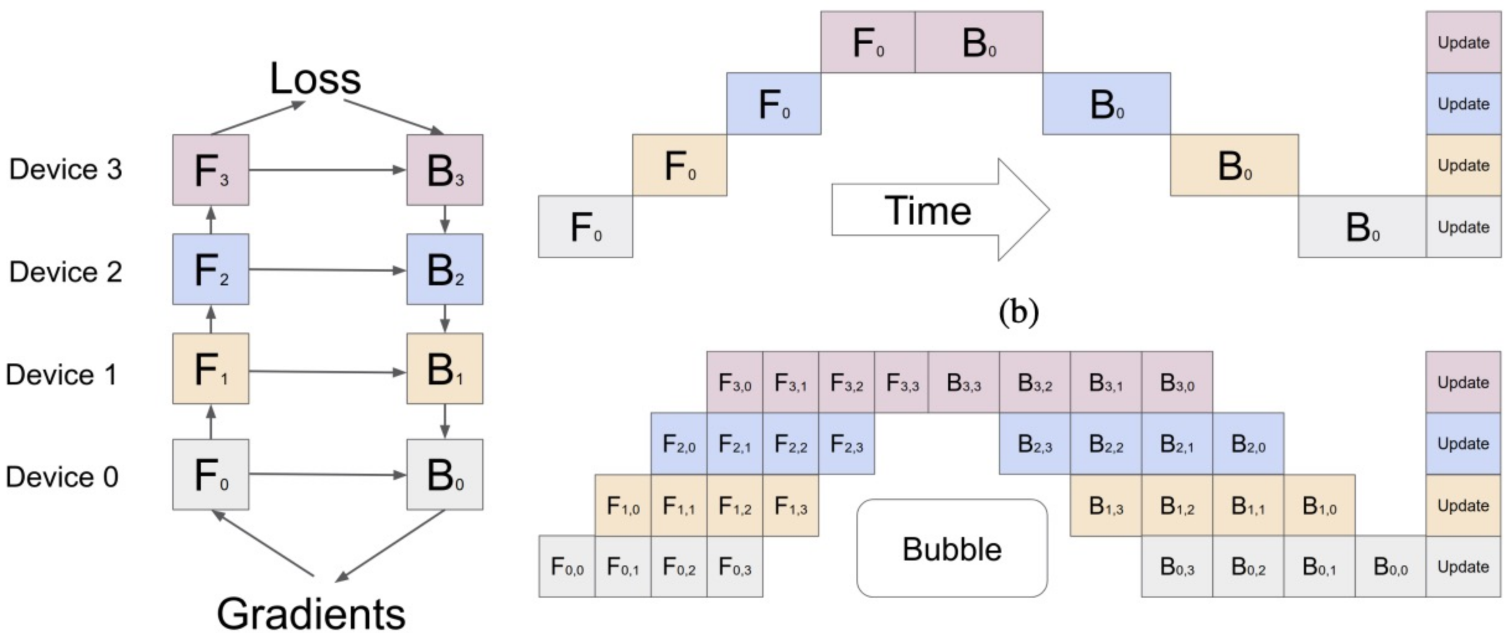


Pipeline Parallelism



Issue of Pipeline Parallelism

Pipeline bubble needs to be controlled carefully.

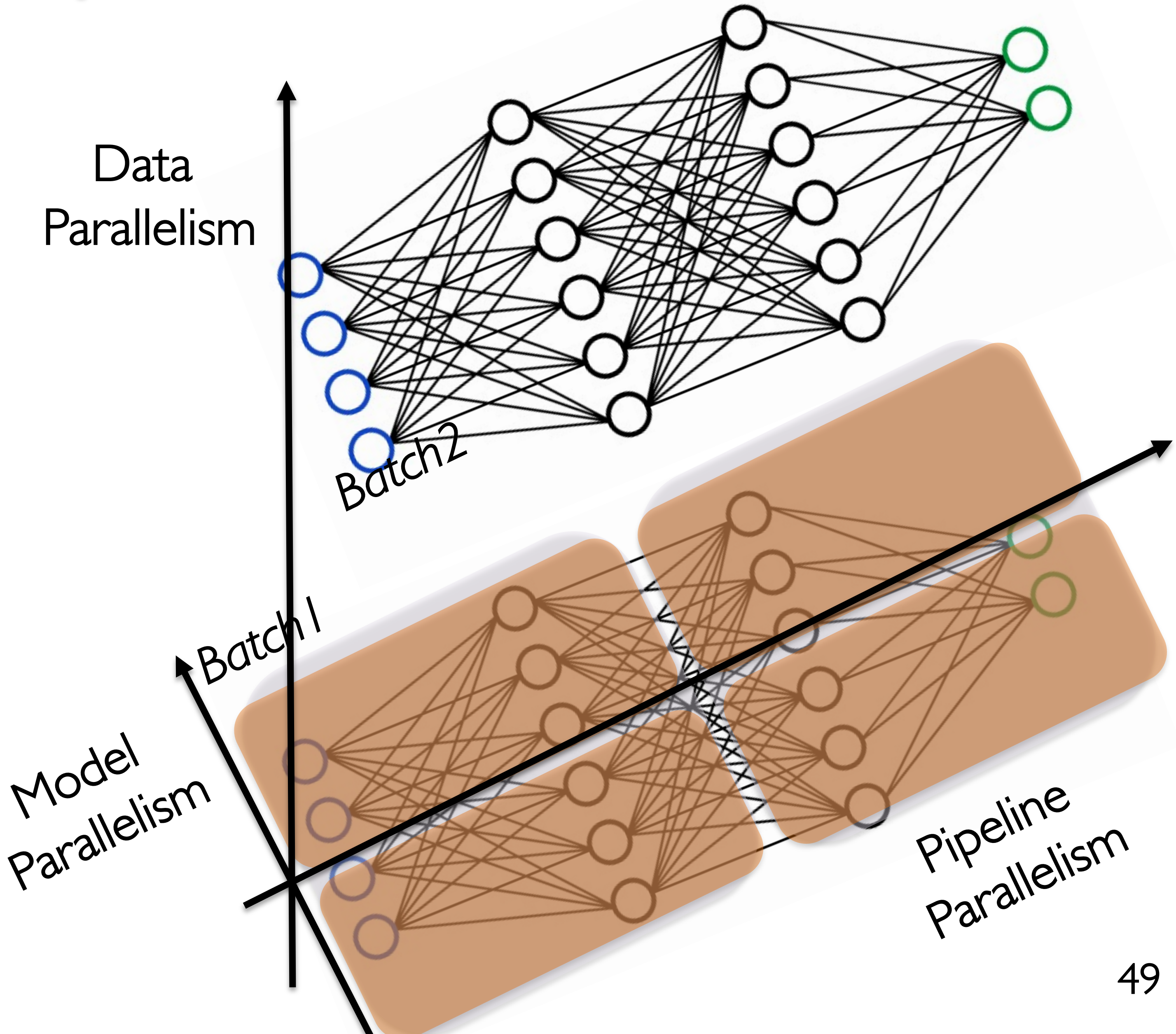


(b)

Overview

- Motivations of Distributed ML
- Parallelism & Comm. Topologies
  - ☐ Data parallelism (PS, all-reduce)
  - ☐ Model parallelism
  - ☐ Pipeline parallelism
  - ☐ Hybrid parallelism
- Communication Bottlenecks & Solutions

Hybrid/3D Parallelism



Overview

- Motivations of Distributed ML
- Parallelism & Comm. Topologies
  - Data parallelism (PS, all-reduce)
  - Model parallelism
  - Pipeline parallelism
  - Hybrid parallelism
- <u>Communication Bottlenecks & Solutions</u>

Data-parallel mini-batch SGD

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}; \mathbf{z}_i) \qquad \mathcal{X} = \{z_1, \cdots, z_n\}$$

For iteration t :

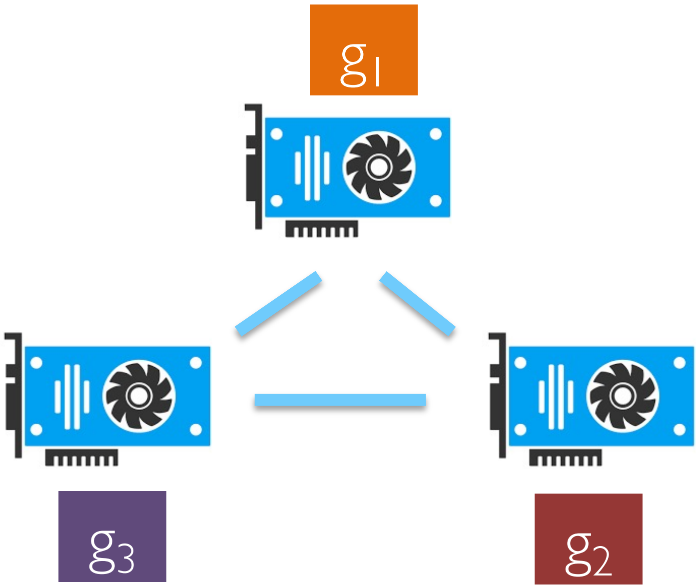
- Sample a mini-batch: $\mathcal{B}_t \subset \mathcal{X}$

- Compute gradient: $\nabla \frac{1}{|\mathcal{B}_t|} \sum_{j=1}^{|\mathcal{B}_t|} \ell(w_t; z_j)$

$$= \frac{1}{|\mathcal{B}_t|} \boxed{\sum_{j=1}^{|\mathcal{B}_t|} \nabla \ell(w_t; z_j)}$$

Data-parallel SGD

All-reduce SGD



Repeat until convergence

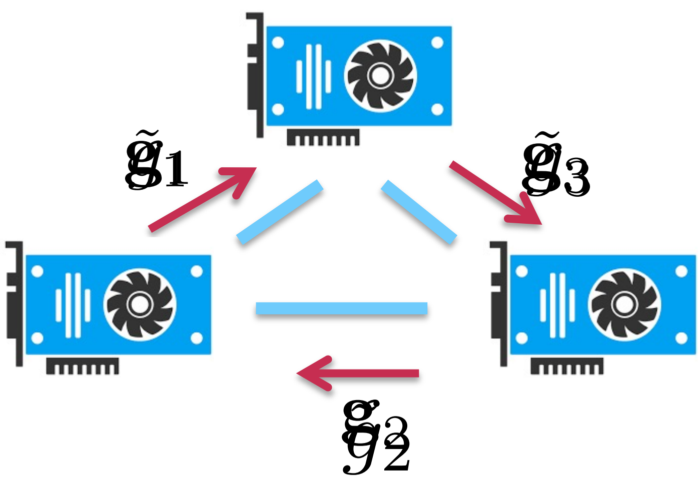
[Cotter, Shamir, Srebro, Sridharan, NIPS11]
[Dekel, Gilad-Bachrach, Shamir, Xiao, JMLR 2012]
[Friedlander and Schmidt, SIAM JSC 2012]

[Takác, Bijral, Richtárik, Srebro, ICML 2013]
[Li, Zhang, Chen, Smola, KDD 2014]
[Jain, Kakade, Kidambi, Netrapalli, Sidford, arxiv'16]
[De, Yadav, Jacobs, Goldstein, arxiv'16]

The ideal speedup should be proportional to #compute nodes

TL;DR:
Communication bottlenecks

Gradient compression



Communication / worker:

O(size of gradient) * 32 bits

Gradient Quantization

Quantize to precision:

O(size of gradient) * 2/4/8 bits

Gradient Sparsification

Sparsified Gradients (k-sparse):

O(k)

Top-*K* (Sparsification)

Examples:

$$\mathrm{top}_2\left(\begin{bmatrix} 2 \\ -3 \\ 1 \\ -4 \end{bmatrix}\right) \Rightarrow \begin{bmatrix} 0 \\ -3 \\ 0 \\ -4 \end{bmatrix}$$

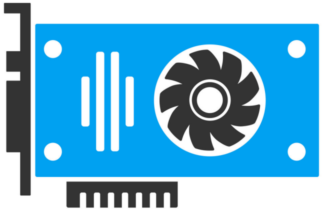
S. U. Stich et al. "Sparsified SGD with Memory", NeurIPS 2018.

*sign*SGD (Quantization)

$$\mathrm{signsgd}\left(\begin{bmatrix} -0.2 \\ 0.1 \\ 0.3 \\ -1.8 \end{bmatrix}\right) \Rightarrow \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

Bernstein et al. "signSGD: Compressed Optimisation for Non-Convex Problems", ICML 2018.
Bernstein et al. "signSGD with Majority Vote is Communication Efficient And Fault Tolerant", ICLR 2019.

ATOMO & *PowerSGD*
(Low-rank Factorization)



Factorization SVD((*U* G *V*))

Wang et al. *"ATOMO: Communication-efficient Learning via Atomic Sparsification", NeurIPS 18.*
Vogels et al. "PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization", NeurIPS19.

Next Week

- More about gradient compression.
- Convergence rate of gradient compressed distributed training.
- Federated learning.

Reading List

- Sergeev, A. and Del Balso, M., 2018. Horovod: fast and easy distributed deep learning in TensorFlow. arXiv preprint arXiv:1802.05799.

- Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J. and Catanzaro, B., 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053.

- Jia, Z., Zaharia, M. and Aiken, A., 2019. Beyond Data and Model Parallelism for Deep Neural Networks. Proceedings of Machine Learning and Systems, 1, pp.1-13.

- Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N.R., Ganger, G.R., Gibbons, P.B. and Zaharia, M., 2019, October. PipeDream: generalized pipeline parallelism for DNN training. In Proceedings of the 27th ACM Symposium on Operating Systems Principles (pp. 1-15).

- Gholami, A., Azad, A., Jin, P., Keutzer, K. and Buluc, A., 2018, July. Integrated model, batch, and domain parallelism in training neural networks. In Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures (pp. 77-86).

- Ben-Nun, T. and Hoefler, T., 2019. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. ACM Computing Surveys (CSUR), 52(4), pp.1-43.