CS 181AI
Lecture 22

Distributed Training Pt. 2

Arthi Padmanabhan

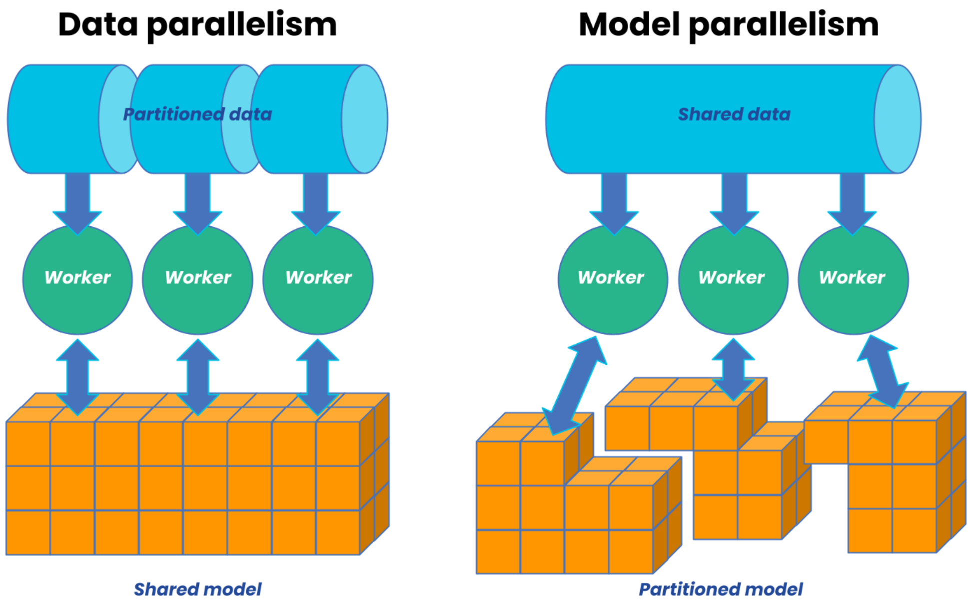
Apr 10, 2023

Logistics

- All assignments done!
- Remaining: 1 round of paper reading, project presentation, project report
- Outline of rest of semester (!!)
  - Today: Distributed Training part 2
  - 4/12: Profiling + debugging
  - 4/17: Efforts to lower model resource usage (+ different types of models)
  - 4/19: Working session
  - 4/24: Scale of models/resources in industry + look at full stack of ML pipelines
  - 4/26: Project presentations

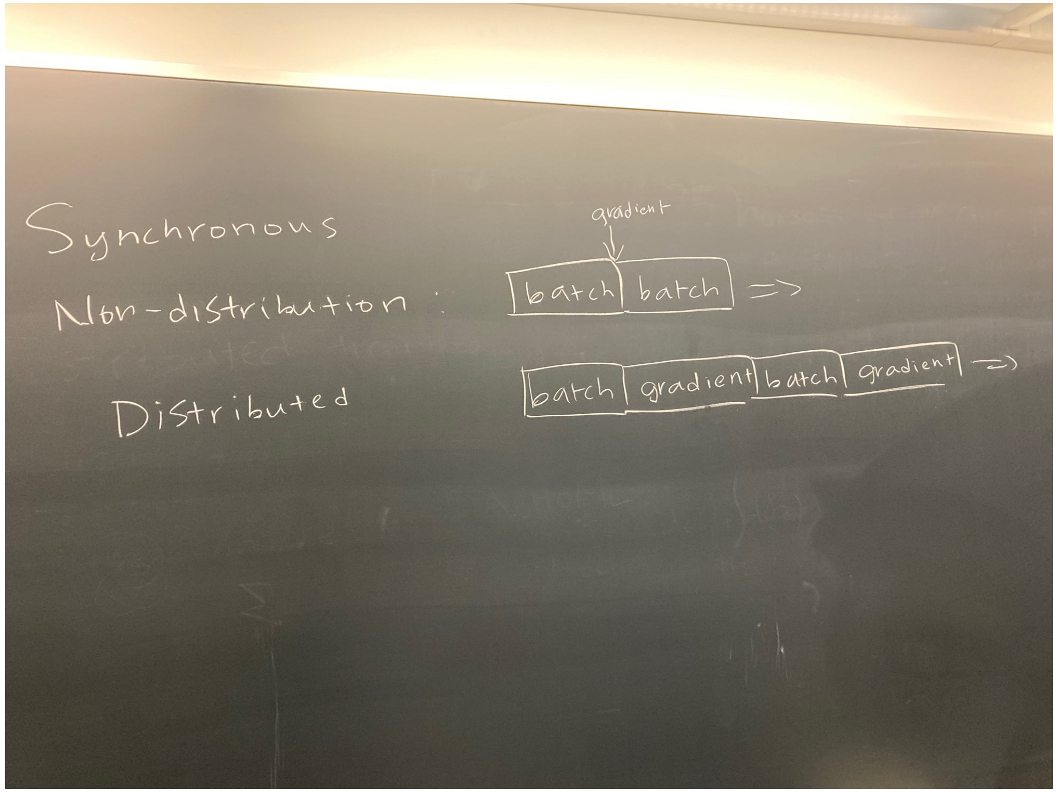
Last Time

- Data vs Model Parallelism
- Synchronous Distributed Training
- Strategies for Gradient Aggregation



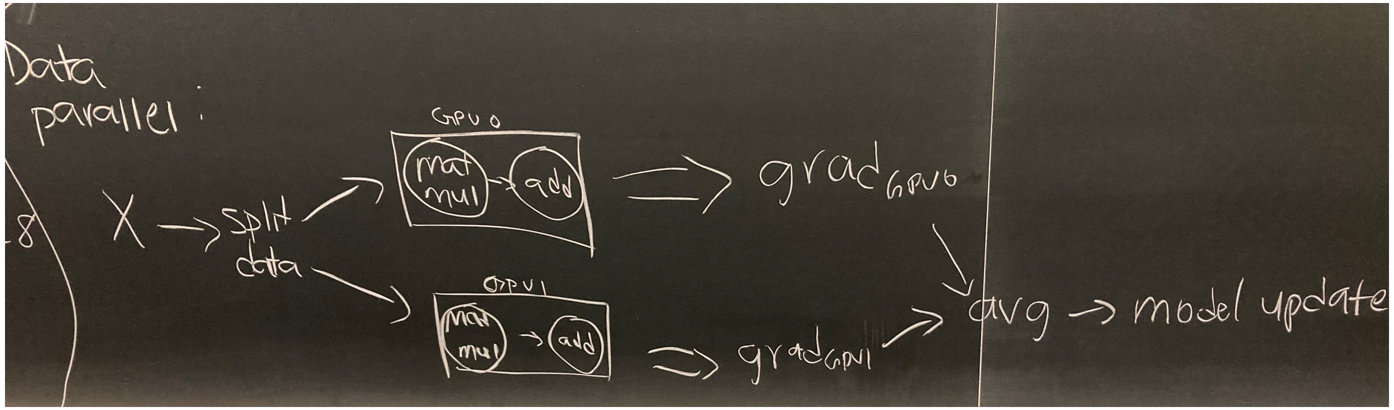
Last Time: Synchronous Distributed Training

- Gradients are computed and model is updated after each batch
- Calculation must happen very efficiently



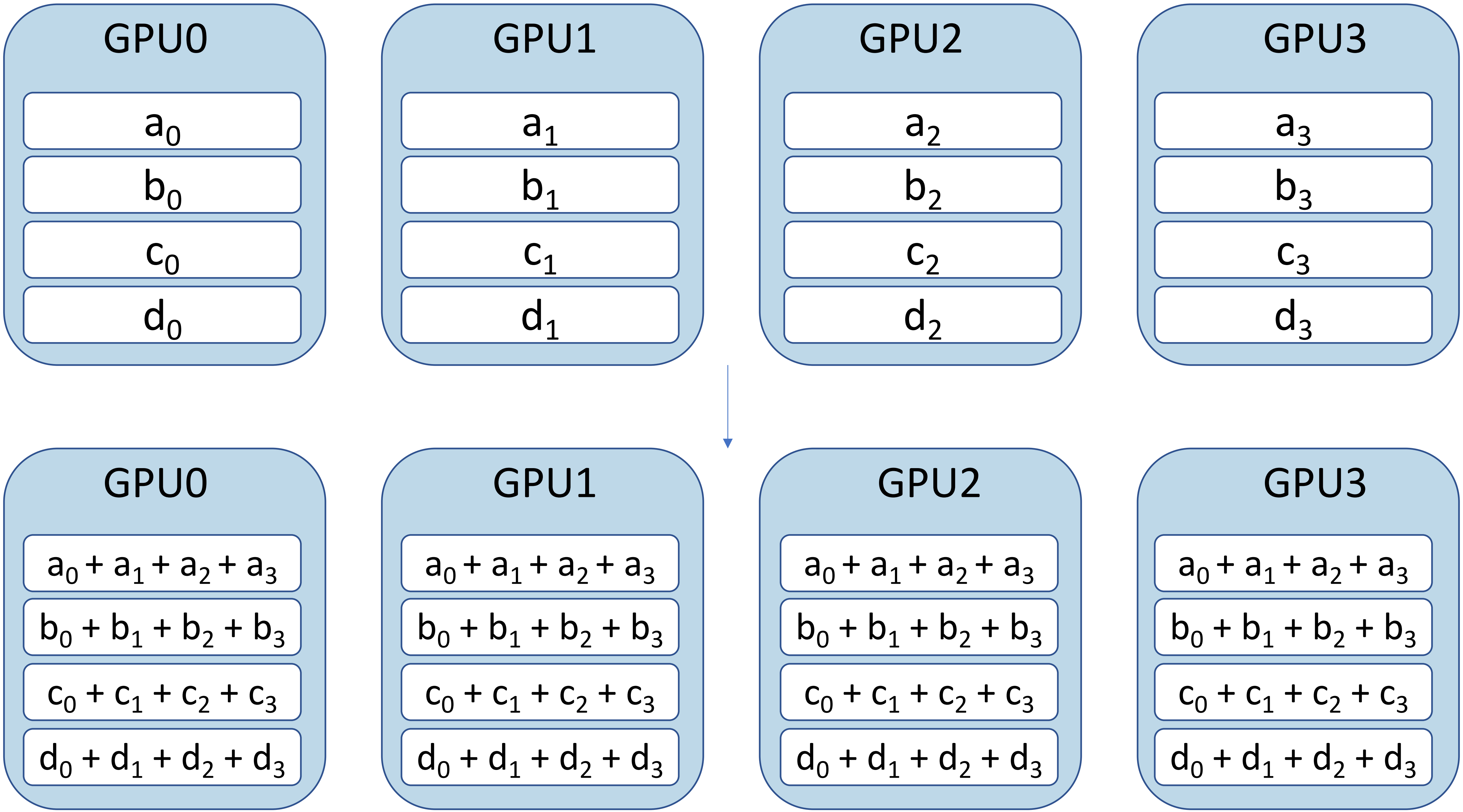
Last Time: Gradient Calculation

- At the end of every batch, gradients for the whole model are calculated
- How does this work if each GPU sees part of the batch?
  - Each GPU computes its own loss and gradients. Gradients are averaged and result is used to update the model



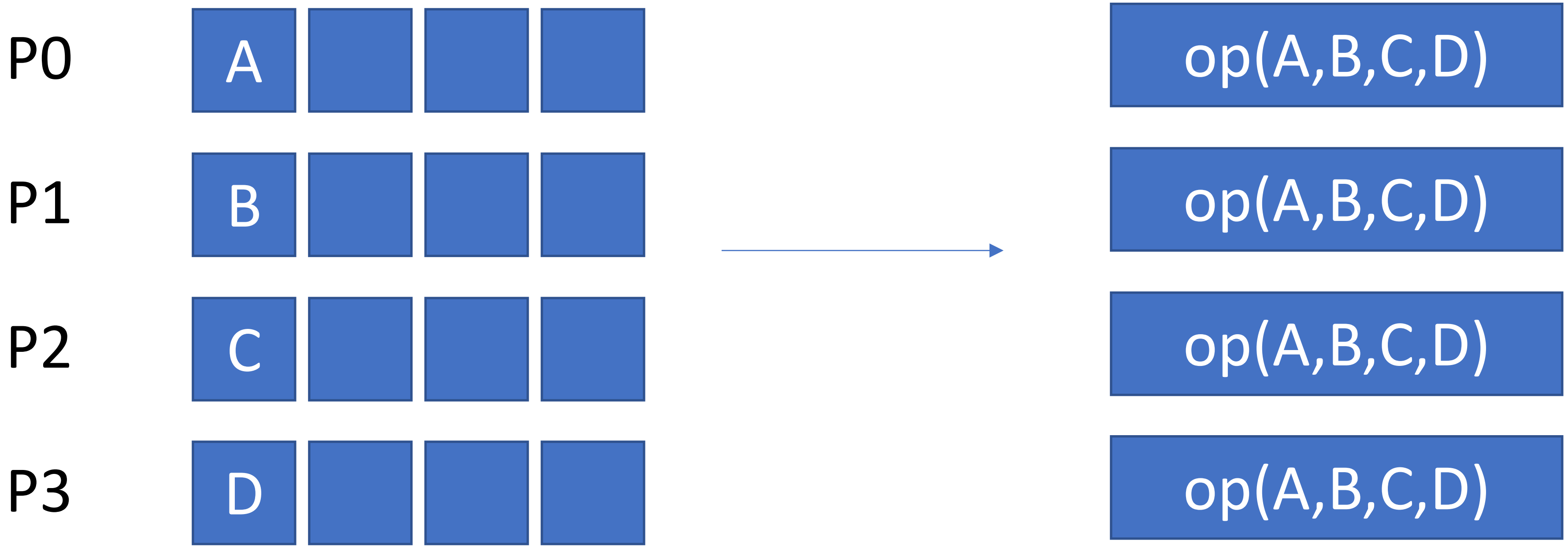
Today

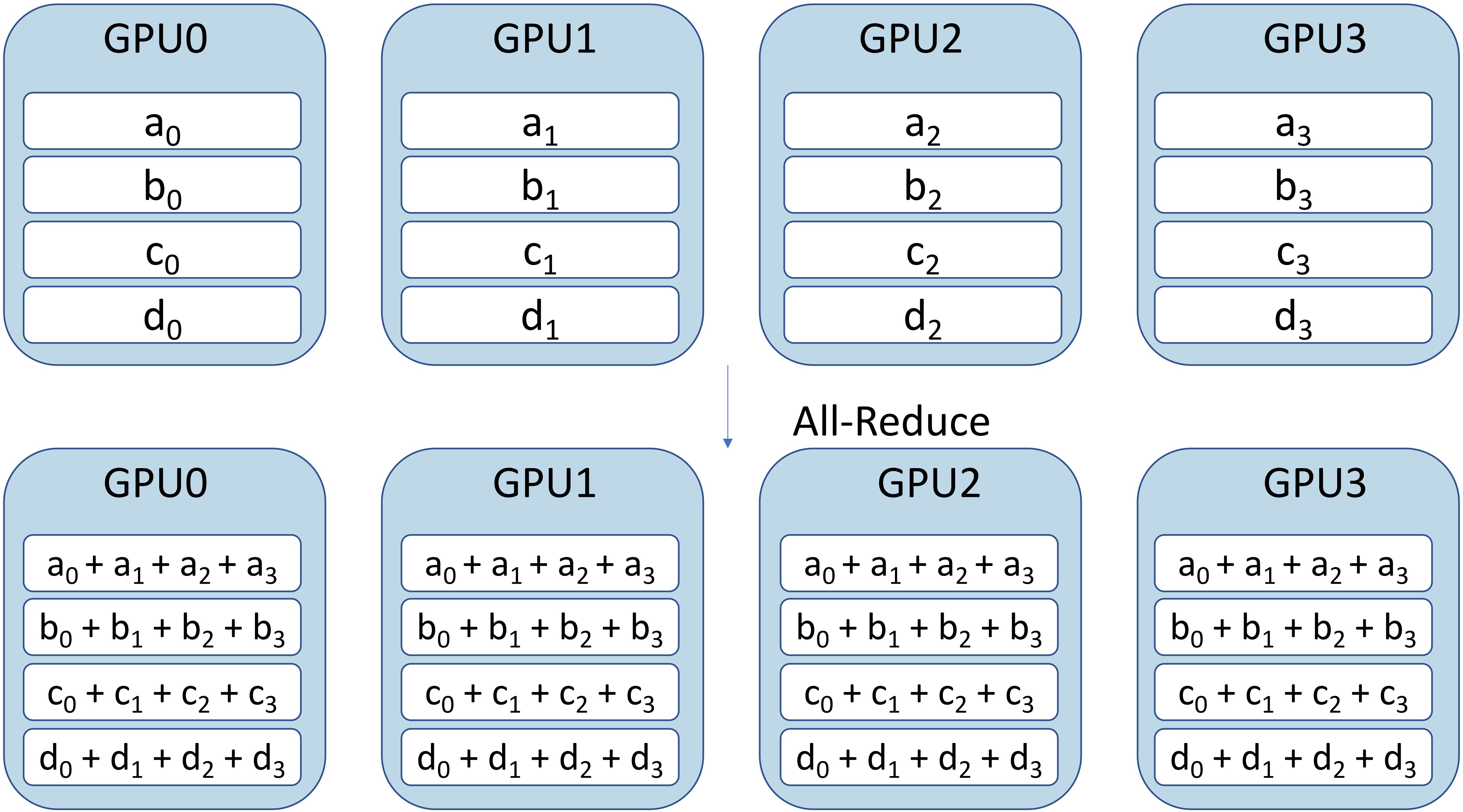
- Synchronous Gradient Calculation Strategies
- Asynchronous Distributed Training
- Consistency Models


GPU0
a_0
b_0
c_0
d_0
GPU1
a_1
b_1
c_1
d_1
GPU2
a_2
b_2
c_2
d_2
GPU3
a_3
b_3
c_3
d_3
GPU0
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$
GPU1
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$
GPU2
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$
GPU3
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$

All-Reduce

- “Operation that reduces a set of arrays on distributed workers to a single array that is then redistributed back to each worker”




GPU0
a_0
b_0
c_0
d_0
GPU1
a_1
b_1
c_1
d_1
GPU2
a_2
b_2
c_2
d_2
GPU3
a_3
b_3
c_3
d_3
All-Reduce
GPU0
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$
GPU1
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$
GPU2
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$
GPU3
$a_0 + a_1 + a_2 + a_3$
$b_0 + b_1 + b_2 + b_3$
$c_0 + c_1 + c_2 + c_3$
$d_0 + d_1 + d_2 + d_3$

Gradient Aggregation Metrics

- Metrics
  - Bandwidth (number of messages)
  - Fault tolerance
  - Speed of convergence
  - Elasticity
  - Ease of use

Ring All-Reduce

- Most used in industry
- Down-side: scales linearly with number of GPUs
- Alternative: GCP Reduction Server

Synchronous Gradient Update Summary

- Keeps the models weights in sync after each batch
- Hardest part is getting gradients exchanged between GPUs after each batch
- Synchronous distributed training can be run across multiple GPUs on the same machine or even across multiple machines with several GPUs each

Asynchronous Gradient Update

- Keep one server for all gradients -> parameter server model
- Each worker computes gradients and makes updates asynchronously (no waiting for each other)

Parameter Server

Asynchronous Gradient Update

- Keep one server for all gradients -> parameter server model
- Each worker computes gradients and makes updates asynchronously (no waiting for each other)
- What could go wrong?

Asynchronous Gradient Update

- Keep one server for all gradients -> parameter server model
- Each worker computes gradients and makes updates asynchronously (no waiting for each other)
- Workers could be taking a stale version of parameters and computing gradients on those!

Consistency Models

- Strong
- Weak
- Eventual
- Bounded

Next Time

- Profiling + Debugging your ML System

Acknowledgments

- Nikita Namjoshi, Google Cloud Developer Advocate