Good morning!!

CS 744: PYTORCH

Shivaram Venkataraman

Spring 2024

ADMINISTRIVIA

Assignment 2 out! Due Feb 23rd 10PM!

↳ Friday next week

Course Project

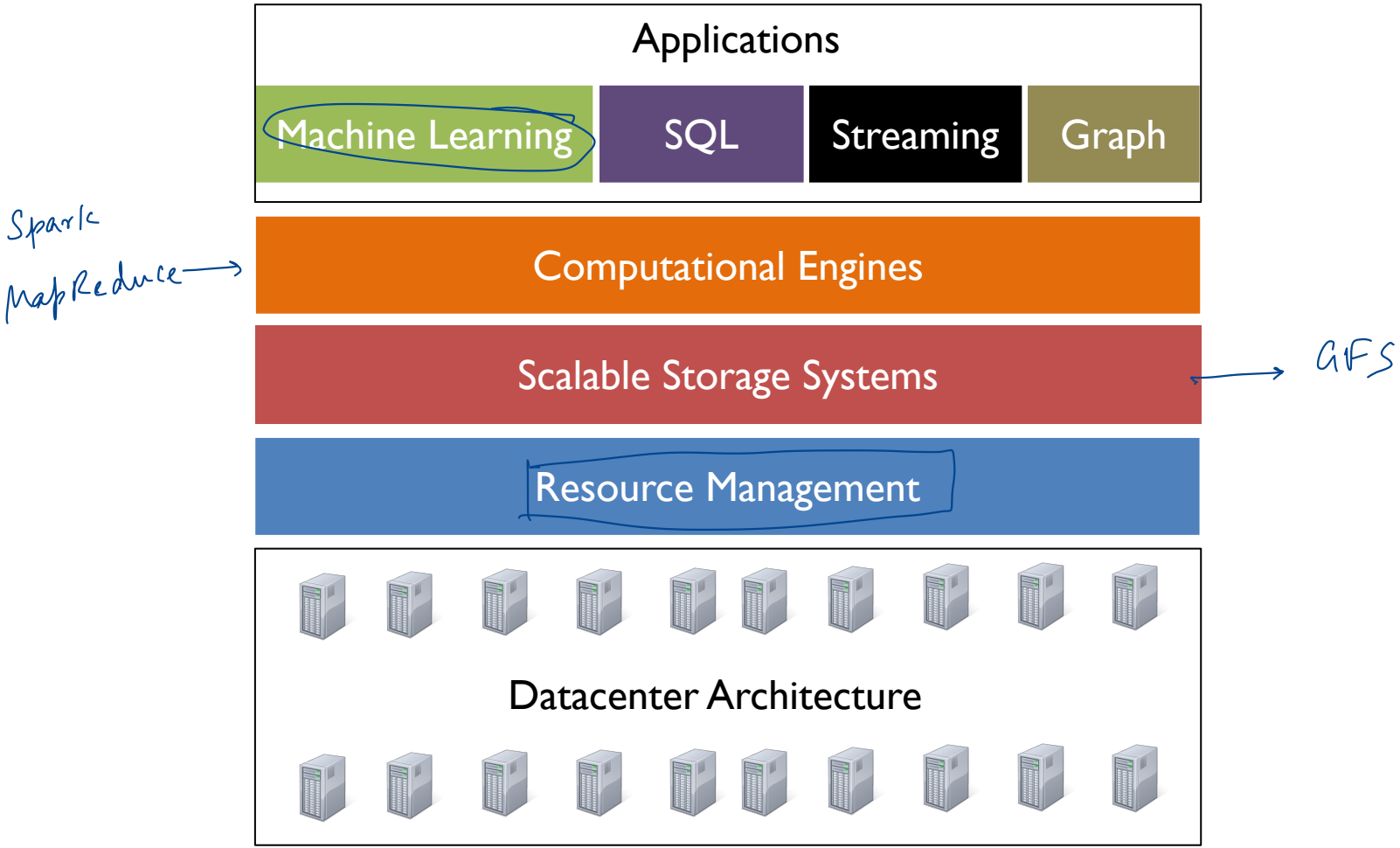
Topics list posted – Feb 21st

Propose / Bid on topics, submit group (1 sentence) – Feb 26th

Title confirmed – March 1

Project Proposal (2 pages) – March 8

- Introduction
- Related Work
- Timeline (with eval plan)


Applications
Machine Learning
SQL
Streaming
Graph
Spark
MapReduce
Computational Engines
Scalable Storage Systems
GFS
Resource Management
Datacenter Architecture

EMPIRICAL RISK MINIMIZATION

Supervised learning

loss function

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{N} f(w, z_i) + P(w)$$

Logistic Regression
Random Forests

Function

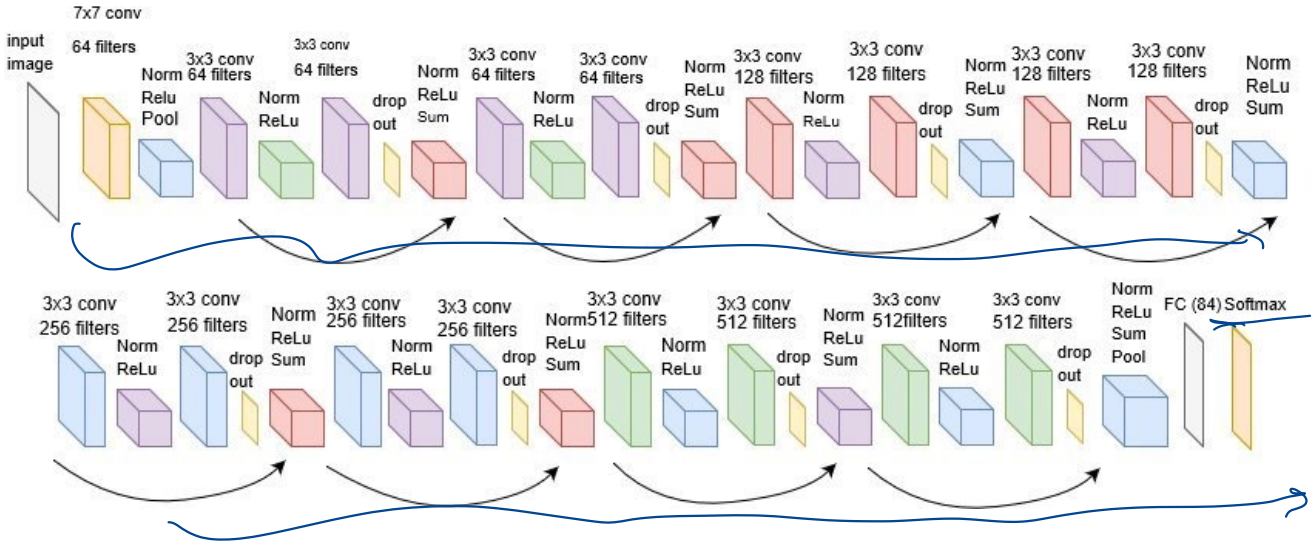
Regularization

Model

Data (Examples)

DEEP LEARNING

data items



prediction

forward pass

ResNet18

Convolution
ReLU
MaxPool
Fully Connected
SoftMax

STOCHASTIC GRADIENT DESCENT

→ Adam

Tensors
matrices

$$w^{(k+1)} = w^{(k)} - \alpha_k \nabla f(w^{(k)})$$

↳ step size

Initialize w → randomly

For many iterations: → Iterative repeating computation as model changes

    Loss = Forward pass

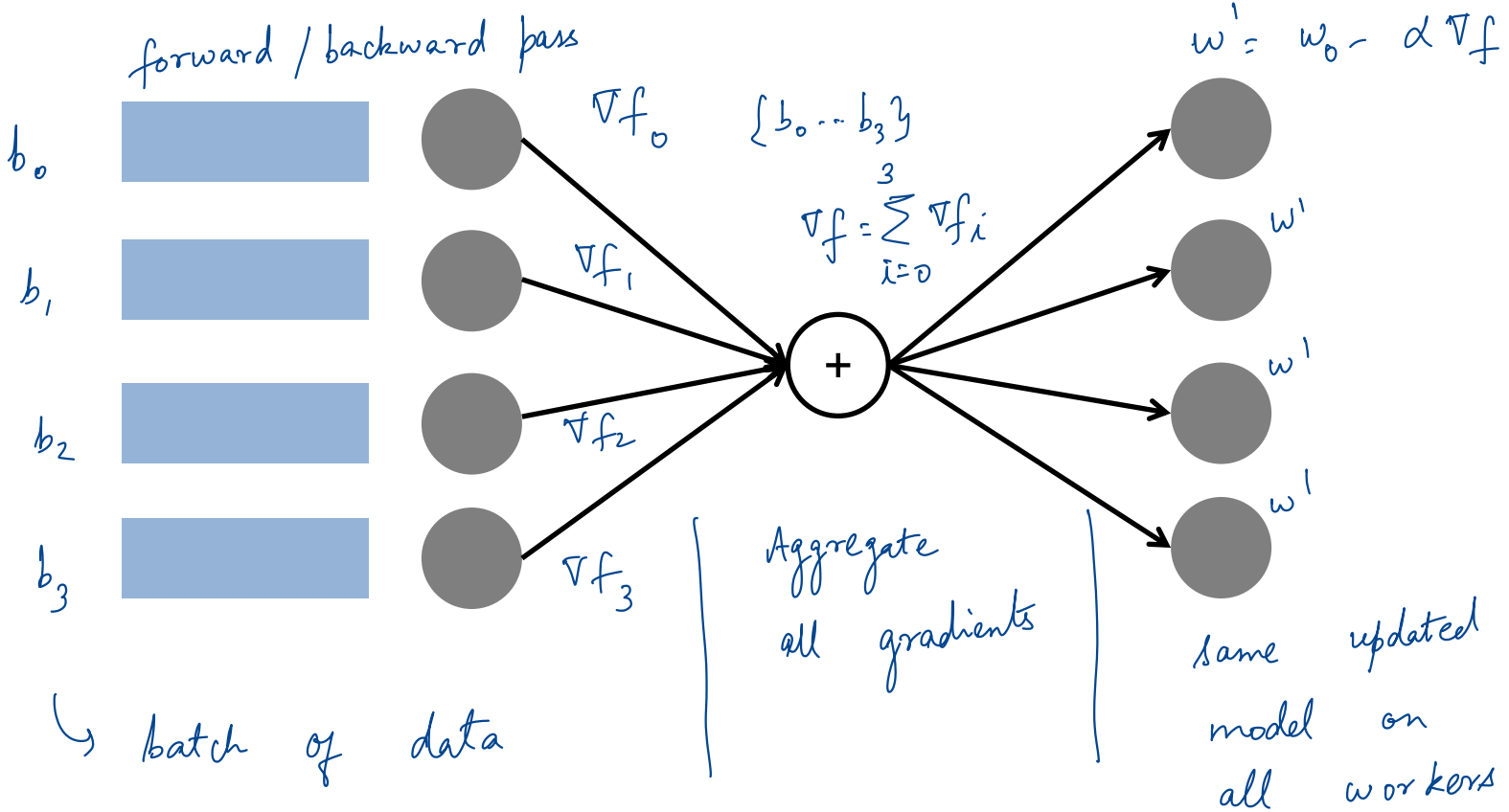
    Gradient = backward → automatic differentiation

    Update model

End

iteration on batch of 128 data 1M

DATA PARALLEL MODEL TRAINING



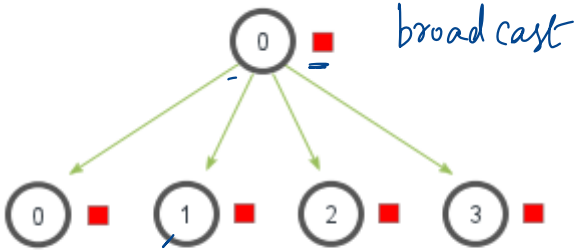
COLLECTIVE COMMUNICATION

MPI_Bcast

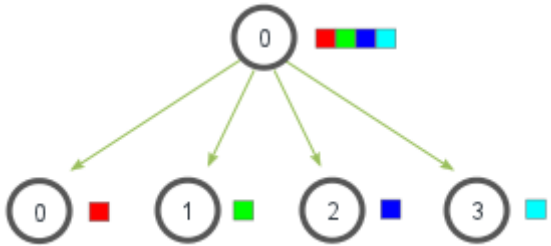
optimized

Broadcast, Scatter

MPI_Bcast

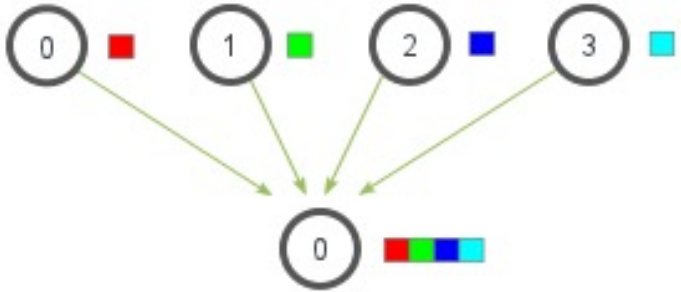


MPI_Scatter

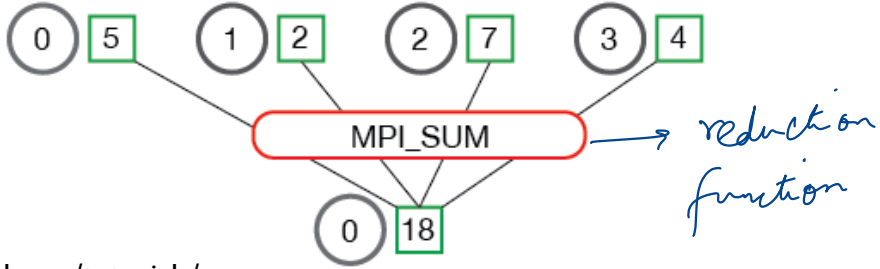


Gather, Reduce

MPI_Gather

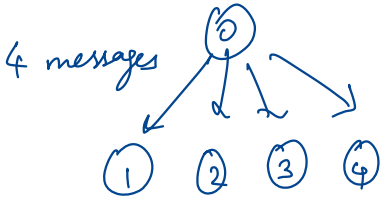


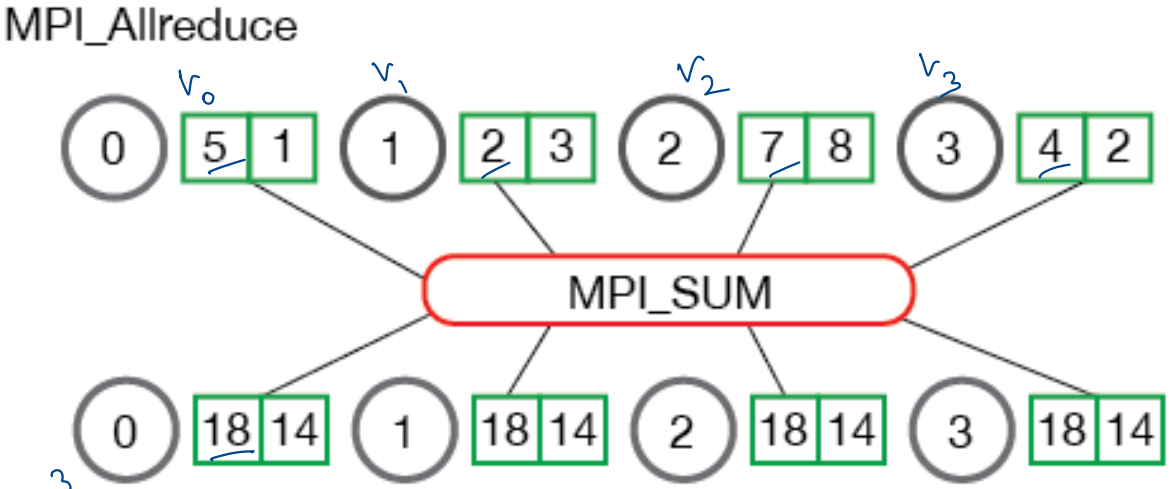
MPI_Reduce



From https://mpitutorial.com/tutorials/

ALL REDUCE USING A RING





$\equiv$ reduction + broadcast

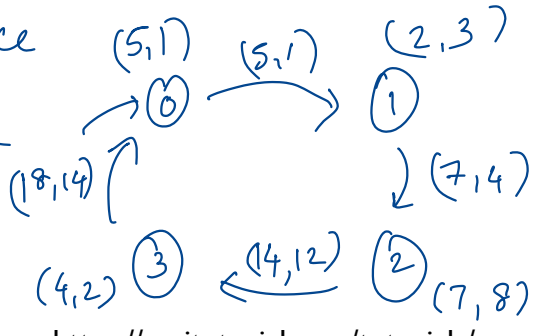
$\sum_{i=0}^{3} v_0$

Ring All Reduce

↳ Fault tolerance

→ Bandwidth usage

→ Latency



1 round of the ring, you get the final value at Process 0.

2nd round broadcast

From https://mpitutorial.com/tutorials/

DISTRIBUTED DATA PARALLEL API

```
9   # setup model and optimizer
10  net = nn.Linear(10, 10)
11  net = par.DistributedDataParallel(net)
12  opt = optim.SGD(net.parameters(), lr=0.01)
13
14  # run forward pass
15  inp = torch.randn(20, 10)
16  exp = torch.randn(20, 10)
17  out = net(inp)
18
19  # run backward pass
20  nn.MSELoss()(out, exp).backward()
21
22  # update parameters
23  opt.step()
```

magic

Minimize code changes go from single machine to Distributed training

$T = \alpha T_{latency} + \beta T_{bandwidth}$

GRADIENT BUCKETING

1 bucket → miss out on overlap

1M buckets → overhead!!

Why do we need gradient bucketing?

gradient = [3.2, 4.5 ....] 1M params

bucket$_0$ bucket$_1$

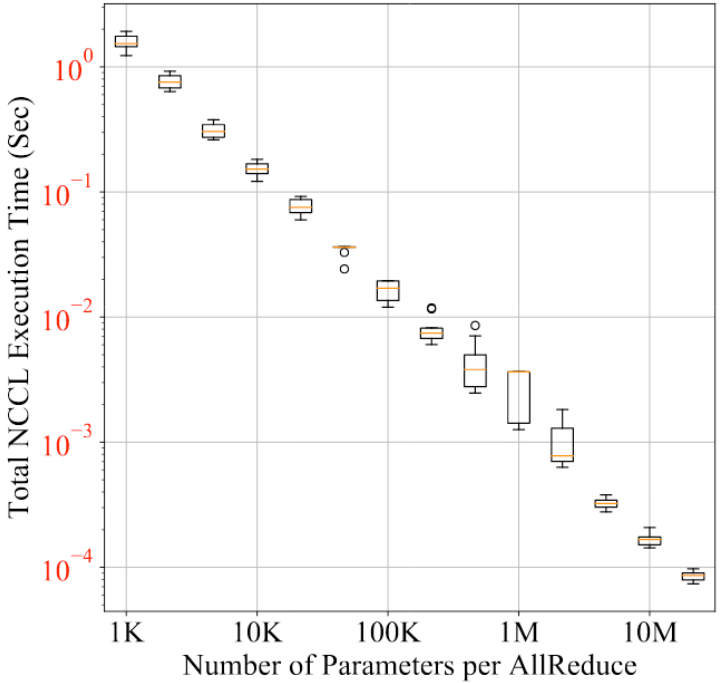
bucket

all reduce on gradient

↳ all reduce (bucket$_0$) allreduce (bucket$_1$) ....

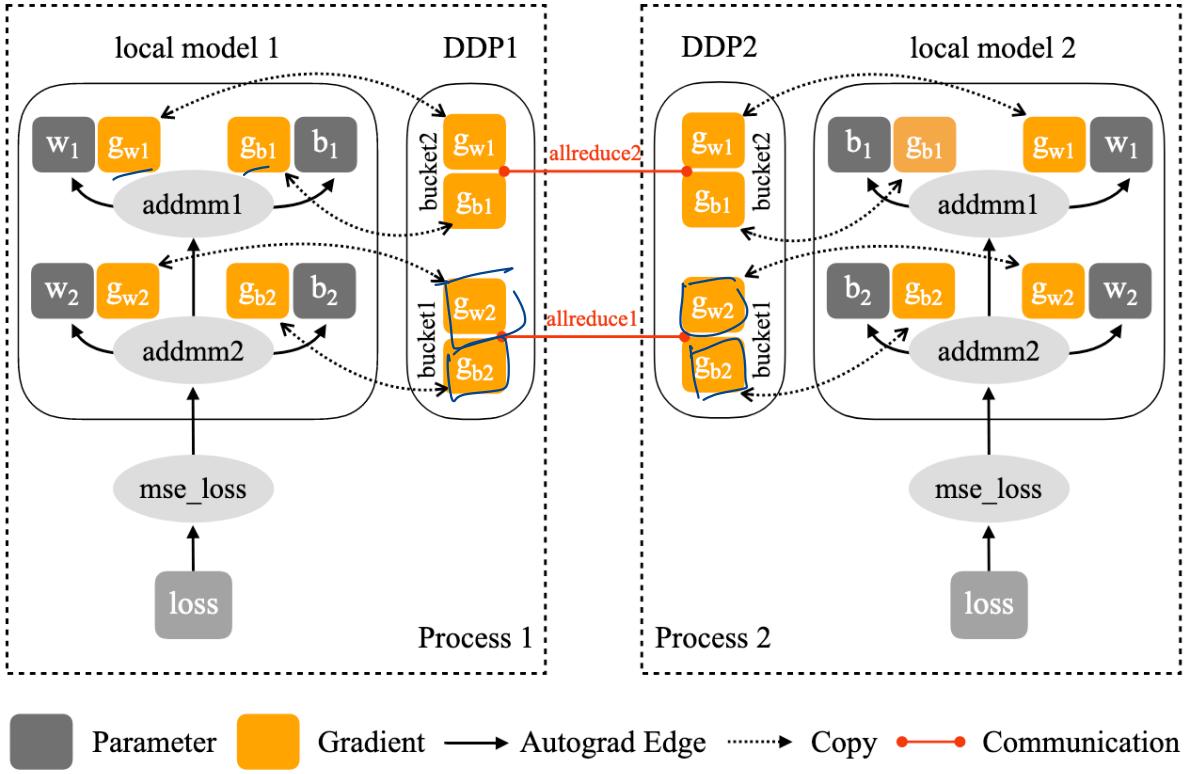
every bucket is independent

↳ if bucket is ready → trigger all reduce



GRADIENT BUCKETING + ALL REDUCE

which gradients in which bucket



fwd / backward

- gradients for final layers are available earliest

---

overlap ▽ calculation for first bucket with allreduce of second bucket

GRADIENT ACCUMULATION

large batch size where

```
1  ddp = DistributedDataParallel(net)
2  with ddp.no_sync():
3    for inp, exp in zip(inputs, expected_outputs):
4      # no synchronization, accumulate grads
5      loss_fn(ddp(inp), exp).backward()
6  # synchronize grads
7  loss_fn(ddp(another_inp), another_exp).backward()
8  opt.step()
```

no allred operations in background

IMPLEMENTATION

Bucket_cap_mb → 25 MB is a good default

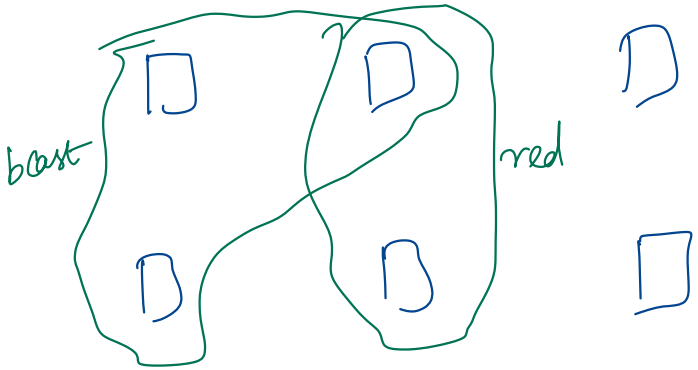
Parameter-to-bucket mapping → walk backwards using model.parameters()

Round-robin ProcessGroups

↳ Multiple link types

→ NVLink

→ PCIe



SUMMARY

Pytorch: Framework for deep learning

DistributedDataParallel API

Gradient bucketing, AllReduce

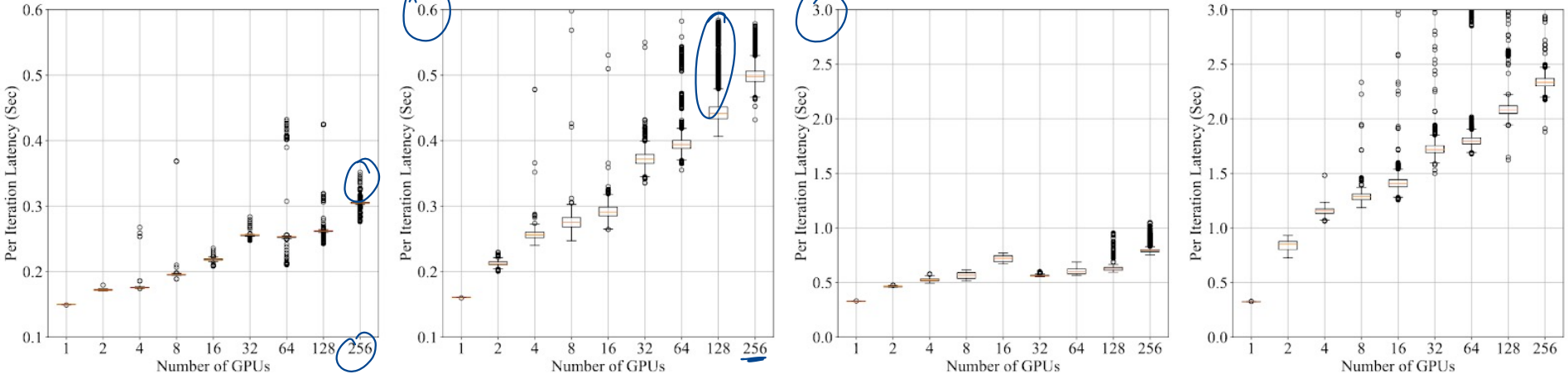
Overlap computation and communication

DISCUSSION

https://forms.gle/aUFy5fsN8KMS4L1i6

More GPUs → more time.
→ linear vs. exponential ?

Variance for Gloo is much higher at 128/256 GPUs



(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

Figure 9: Scalability

BERT has higher latency → larger model than Resnet-50

What could be some challenges in implementing similar optimizations for AllReduce in Apache Spark?

↳ overlap all Reduce with grad compute

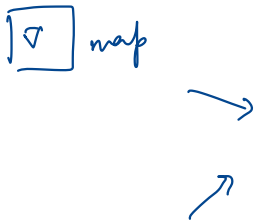
- Fault Tolerance difference
  ↳ MPI-style vs. Spark

- Barrier between map and reduce stages prevents overlap

- Slice the ~~data~~ ~~is~~ gradient doesn't fit very well with Spark API

AllReduce Spark

Reduction tree + broadcast



What could be some challenges in implementing similar optimizations for AllReduce in Apache Spark?

Spark

↳ Tasks → Not all tasks are active at the same time

assuming all are active at some time!

NEXT STEPS

Next class: PipeDream

Assignment 2 is out!

BREAKDOWN

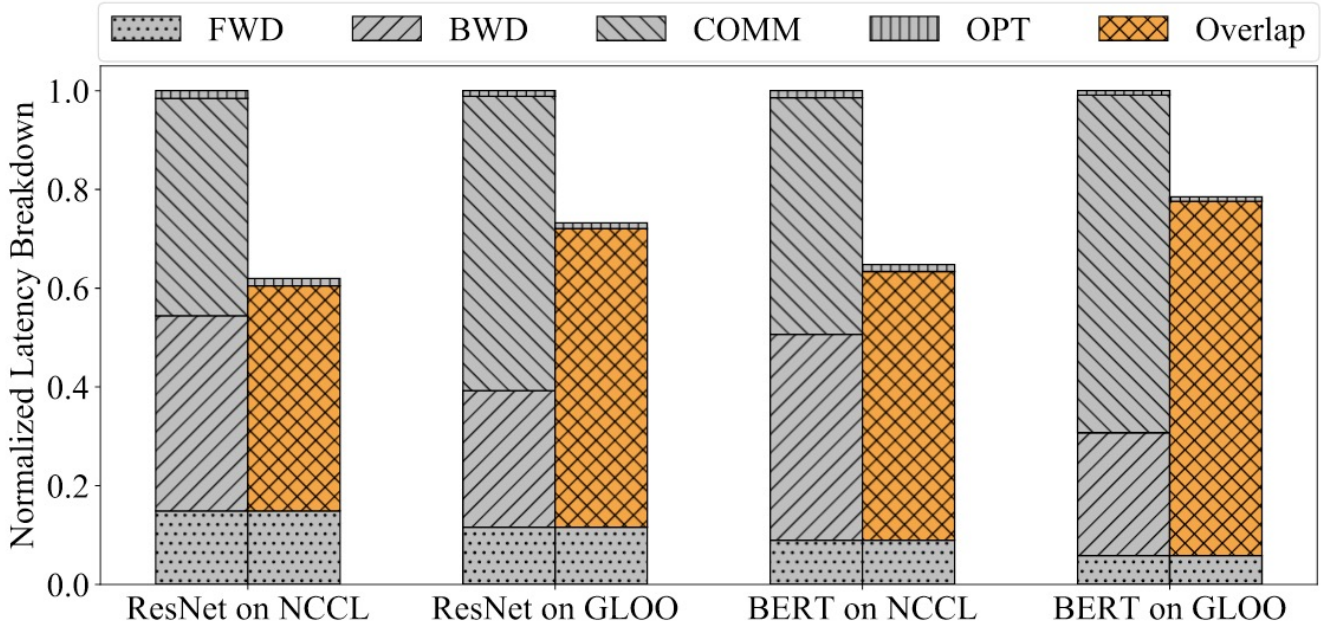


Figure 6: Per Iteration Latency Breakdown