Hello!

CS 744: PYTORCH

Shivaram Venkataraman

Fall 2022

ADMINISTRIVIA

Assignment 2 out! Due Oct 12th 11 AM! → Maybe start soon?

Paper review deadline – 12:59pm on Tue/Thu → Survey in one or two weeks

Bid on topics, submit group (1 sentences) – Oct 11

Title confirmed – Oct 15

Project Proposal (2 pages) – Oct 25

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

WRITING AN INTRODUCTION

Apache Spark

→ 1 or 2 page preface to your paper

interactive data analytics

1-2 paras: what is the problem you are solving

why is it important (need citations) → Many industries need this

1-2 paras: How other people solve and why they fall short

MapReduce jobs and these are slow

1-2 paras: How do you plan on solving it and why your approach is better

1 para: Anticipated results or what experiments you will use

Project proposal: plan | Final report: approach you took

RELATED WORK, EVAL PLAN

Group related work into 2 or 3 buckets (1-2 para per bucket)

Explain what the papers / projects do

Why are they different / insufficient

8-10 related papers or projects

Eval Plan → what resources will help you succeed?

Describe what datasets, hardware you will use

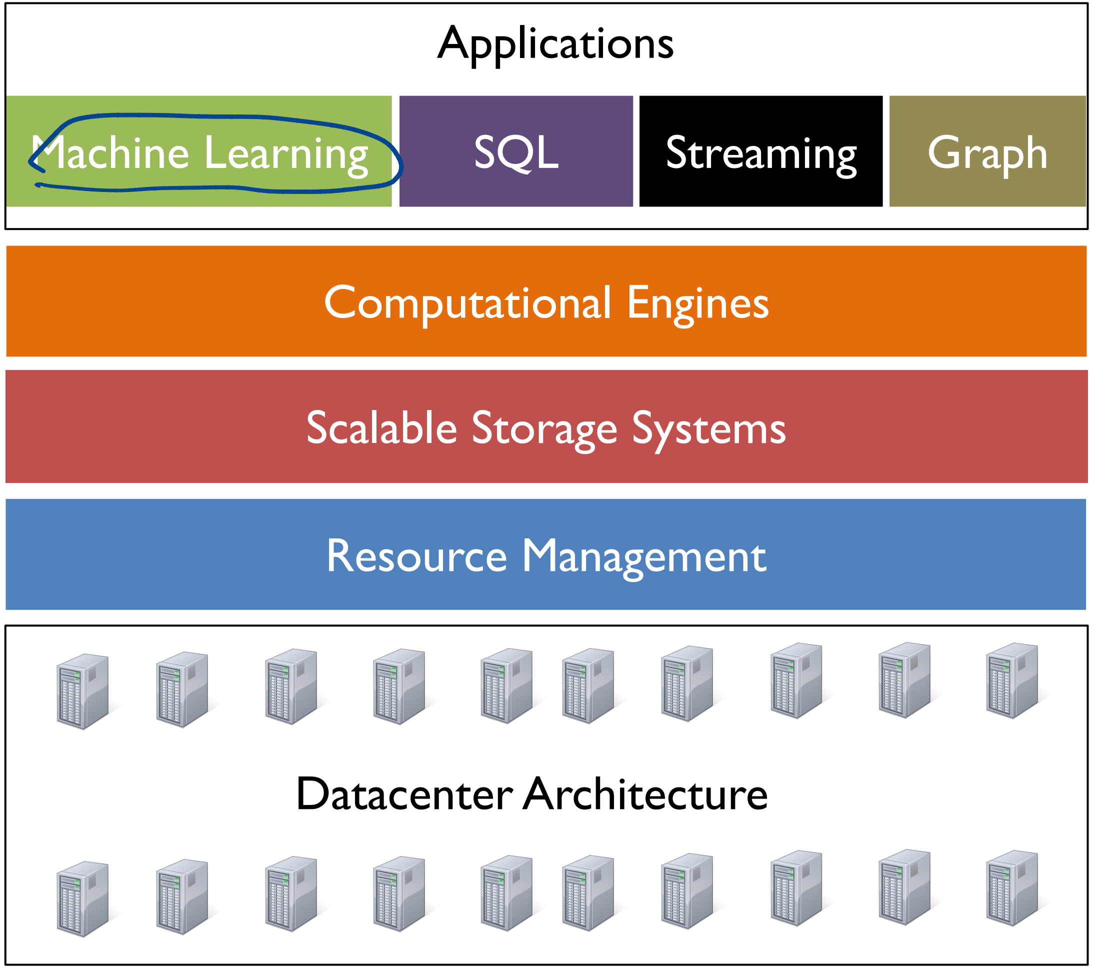
Available: Cloudlab, Google Cloud (~$150), Jetson TX2 etc.

MapReduce
Spark

GFS

Mesos | DRF

Hardware



EMPIRICAL RISK MINIMIZATION

Supervised learning

Training data, labels

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

Regularization

Function

Model

Data (Examples)

Linear Regression

Logistic "

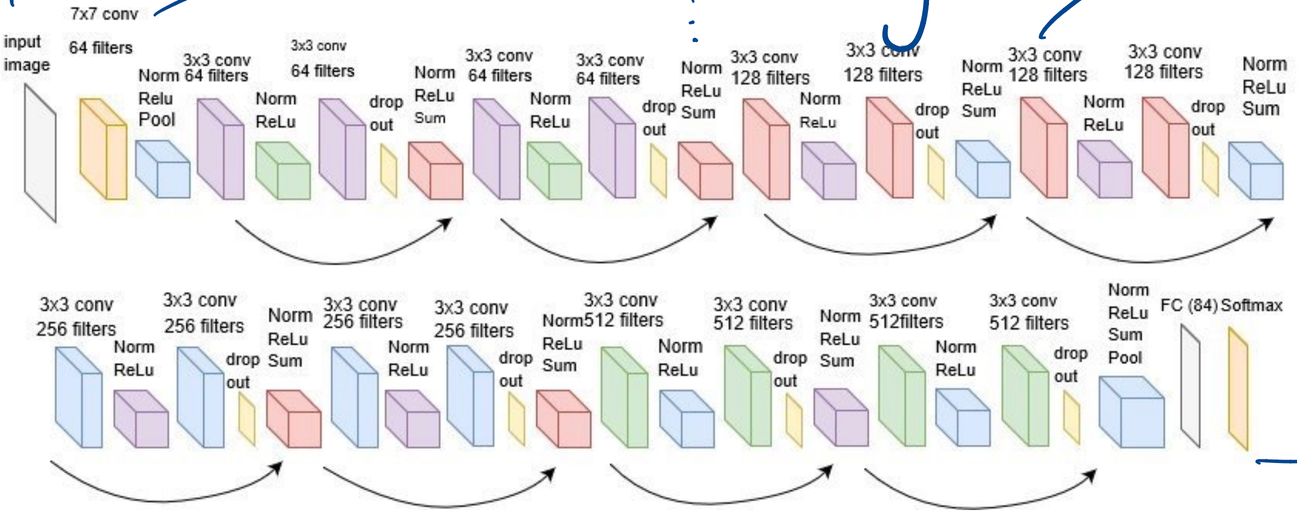
DEEP LEARNING

$t_1 = l_1(\text{input})$
$t_2 = l_2(t_1)$
$\vdots$

forward pass

layers

weights
parameters



ResNet18

Convolution
ReLU
MaxPool
Fully Connected
SoftMax

classify images or other tasks

STOCHASTIC GRADIENT DESCENT

SGD

model at iter k+1

model at iter k

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

Initialize w ← model initialization

For many iterations: → Batch of data

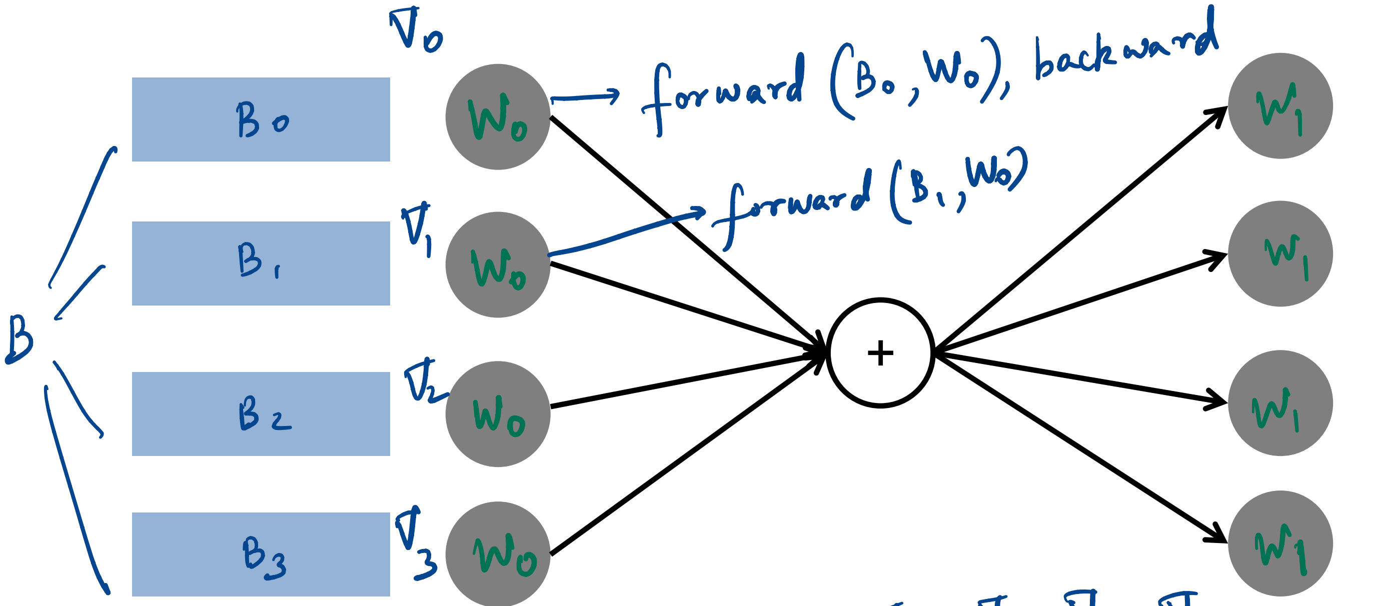
Loss = Forward pass → input and current model
return "loss"
↳ how good are your predictions

Gradient = backward → Chain rule
differentiate sequence of functions

Update model

End

DATA PARALLEL MODEL TRAINING



$\nabla = \nabla_0 + \nabla_1 + \nabla_2 + \nabla_3$

same as grad with B

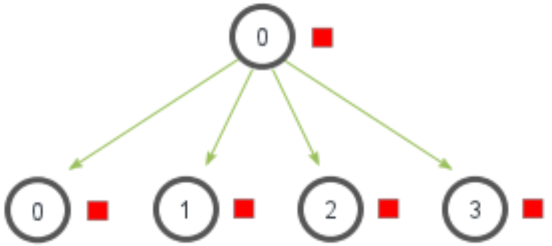
Mimic or reproduce behavior of single machine training!

COLLECTIVE COMMUNICATION

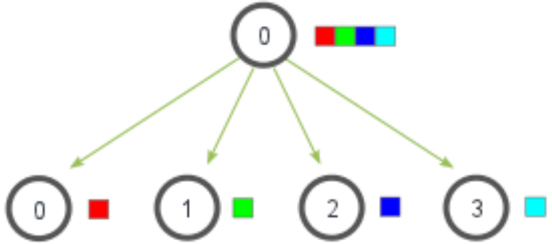
→ 1991 first MPI standard

Broadcast, Scatter

MPI_Bcast → broadcast

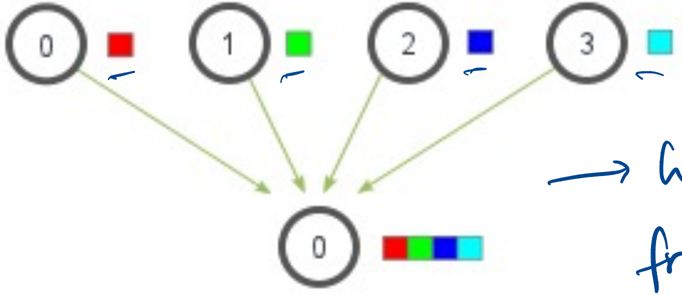


MPI_Scatter



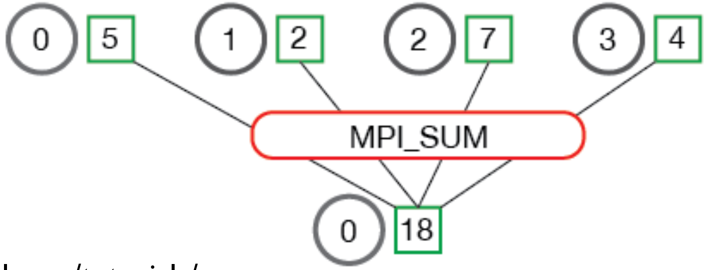
Gather, Reduce

MPI_Gather

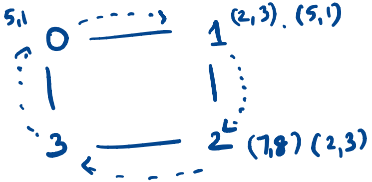


→ Gathering data from all machines

MPI_Reduce



From https://mpitutorial.com/tutorials/

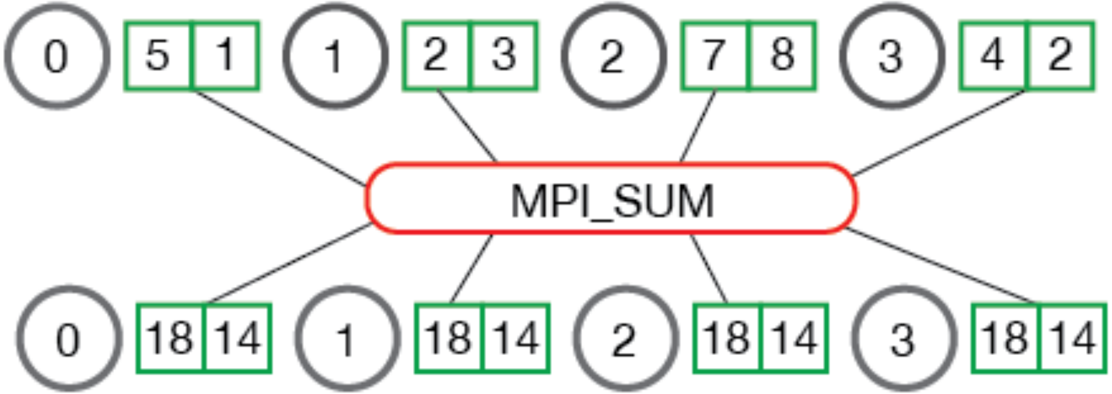


ring

ALL REDUCE USING A RING

→ data transmitted per worker

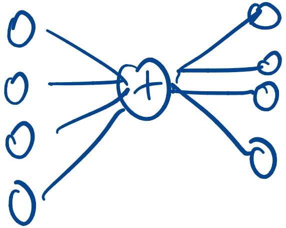
MPI_Allreduce → number of connections



Reduce = Aggregate data from all workers

Agg data + Broadcast agg value to all workers

reduce + broadcast



bottleneck from one machine

- All nodes broadcast to other nodes. Aggregate locally
-

From https://mpitutorial.com/tutorials/

DISTRIBUTED DATA PARALLEL API

```python
# setup model and optimizer
net = nn.Linear(10, 10)
net = par.DistributedDataParallel(net)
opt = optim.SGD(net.parameters(), lr=0.01)

# run forward pass
inp = torch.randn(20, 10)
exp = torch.randn(20, 10)
out = net(inp)

# run backward pass
nn.MSELoss()(out, exp).backward()

# update parameters
opt.step()
```

gradient agg happens under the hood!

fwd, backward happens in ||ł

GRADIENT BUCKETING

Why do we need gradient bucketing?

↳ Bucketing gradient

[______ , ______ , ______ ,]

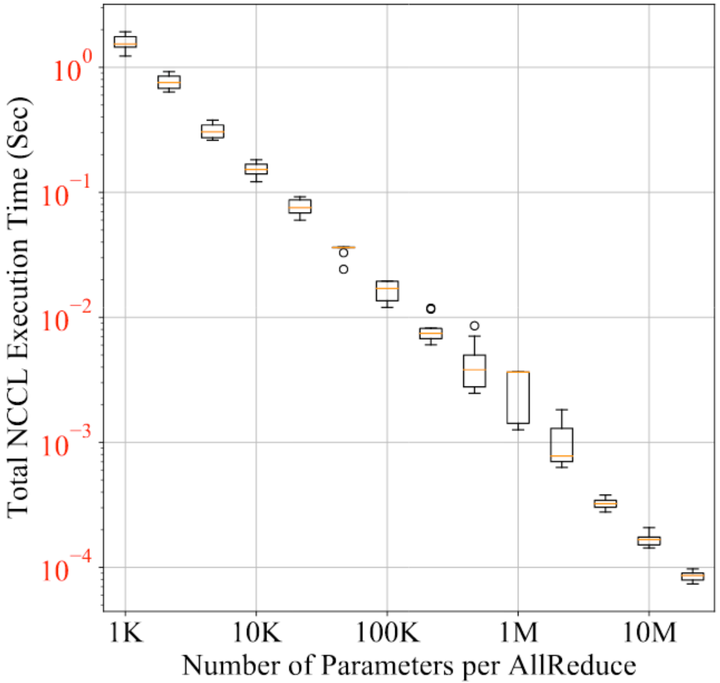
bucket

gradient

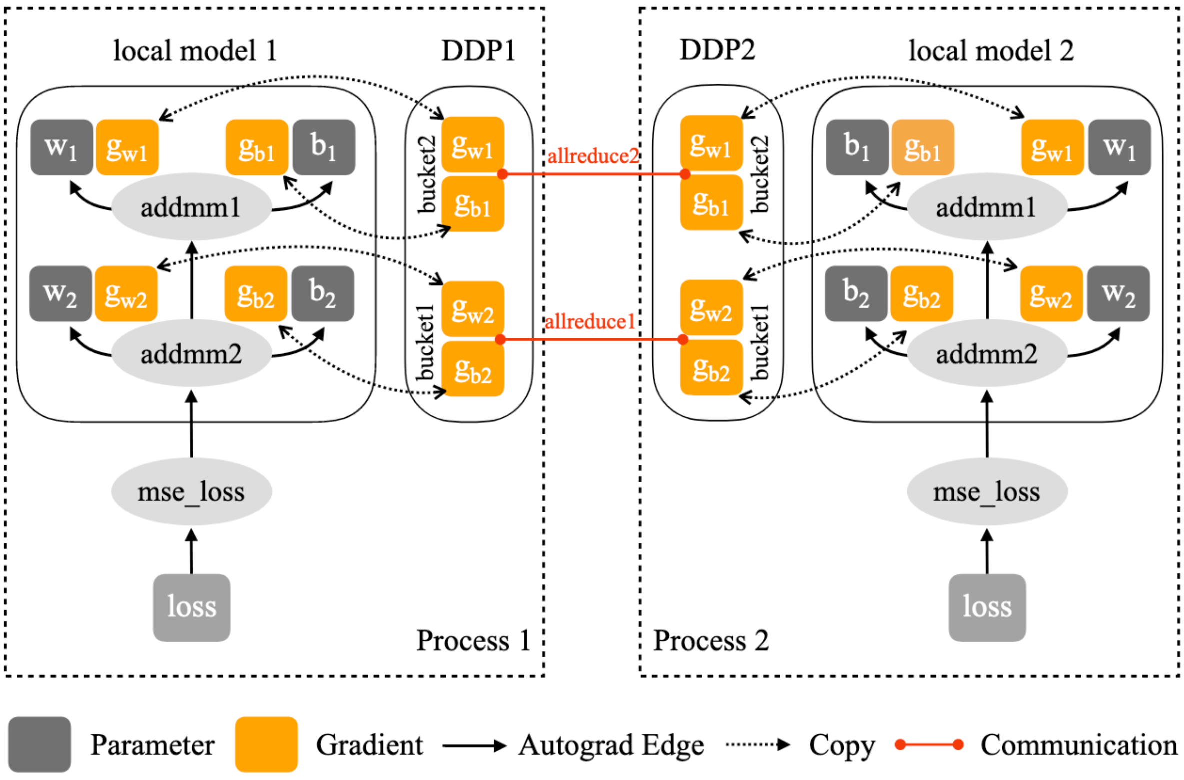
aggregate every layer [Small Tensor]

Bucketing

aggregate whole gradient



GRADIENT BUCKETING + ALL REDUCE



forward
backward → last layer gradient is available first !

You can pipeline backward & gradient aggregation

$T_{com} = \alpha T_{lat} + \beta T_{BW}$

GRADIENT ACCUMULATION

```python
ddp = DistributedDataParallel(net)
with ddp.no_sync():
  for inp, exp in zip(inputs, expected_outputs):
    # no synchronization, accumulate grads
    loss_fn(ddp(inp), exp).backward()
# synchronize grads
loss_fn(ddp(another_inp), another_exp).backward()
opt.step()
```

IMPLEMENTATION

Bucket_cap_mb → 25 MB is a good default

Parameter-to-bucket mapping → model walk backwards and assign params to buckets

Round-robin ProcessGroups → Multiple network types

Ethernet

NVLink

SUMMARY

Pytorch: Framework for deep learning

DistributedDataParallel API

Gradient bucketing, AllReduce

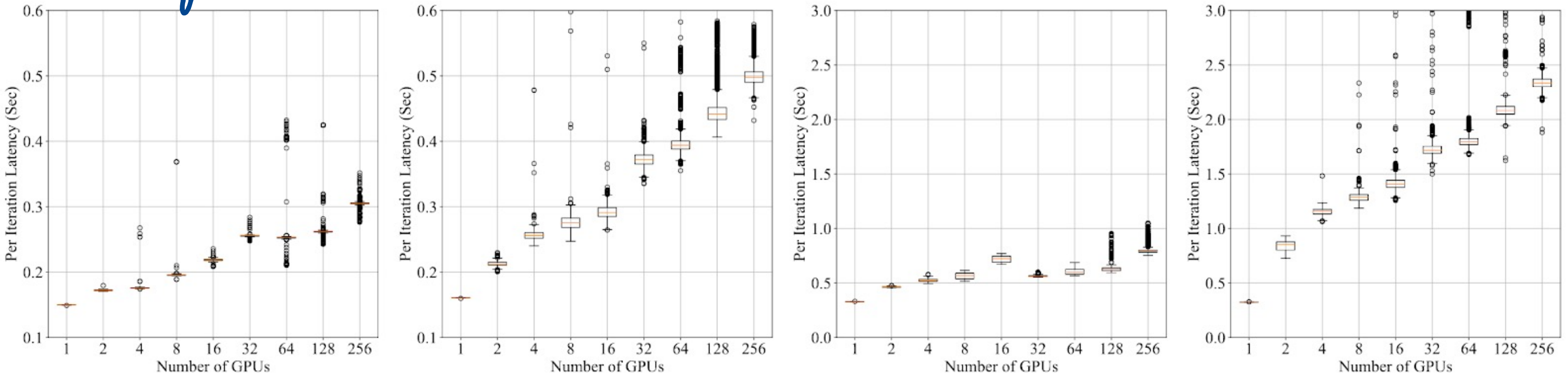
Overlap computation and communication

DISCUSSION

https://forms.gle/jivzEEo5oz8tugYH9

- NCCL is faster than Gloo → NCCL is optimized for GPUs

- Latency increases with num GPUs



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

Figure 9: Scalability

NCCL is linear / sub-linear while Gloo has worse scaling

Tail latency is higher for larger num GPUs

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

NEXT STEPS

Next class: PipeDream

Assignment 2 is out!

Project Proposal – Check Piazza!

BREAKDOWN

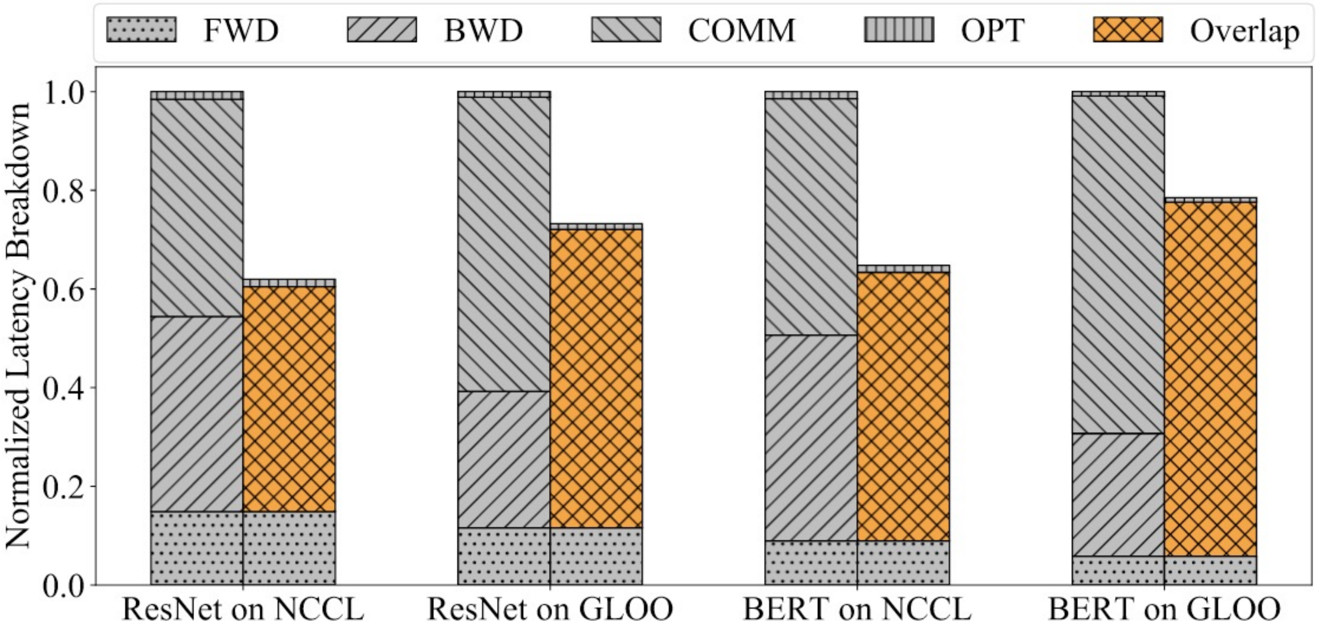


Figure 6: Per Iteration Latency Breakdown