

*Hello!*

# CS 744: PYTORCH

Shivaram Venkataraman

Spring 2025

# ADMINISTRIVIA

Assignment 2 out! Due Feb 13<sup>rd</sup> 10PM!

 next Thursday

Course Project Timeline:

Propose / Express interest on topics/skills

Submit group (1 sentence) – Feb 17<sup>th</sup>

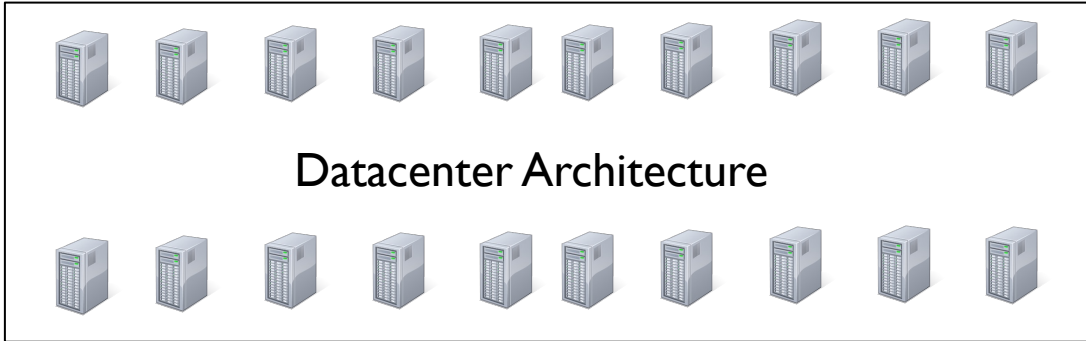
Title confirmed – Feb 21<sup>st</sup>

Project Proposal (2 pages) – March 4

Introduction

Related Work

Timeline (with eval plan)



*MapReduce*

*spark*

*GFS*

*Tectonic*



# EMPIRICAL RISK MINIMIZATION

*Training*

*↳ Supervised learning*

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

*weights*

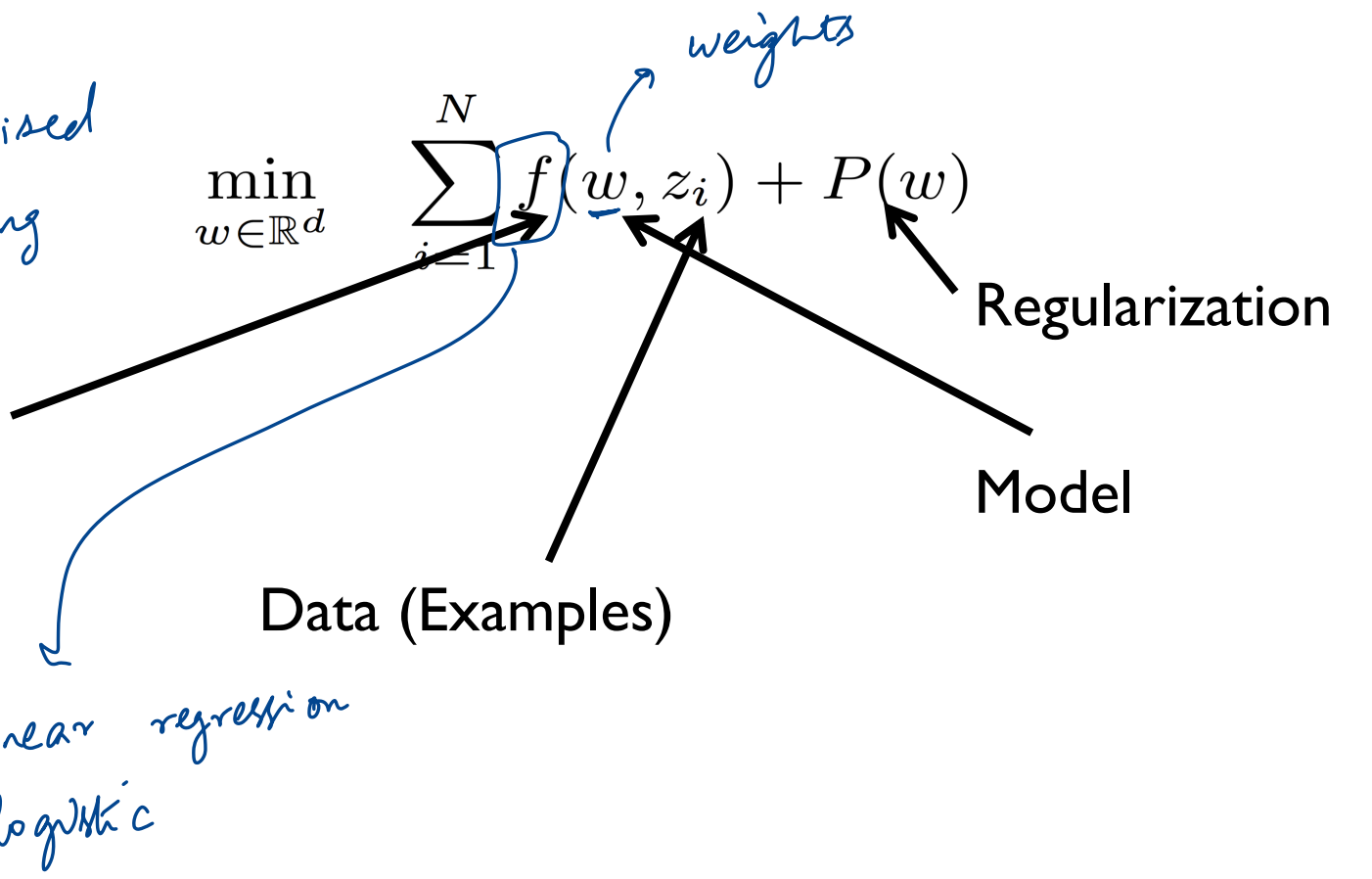
Regularization

Function

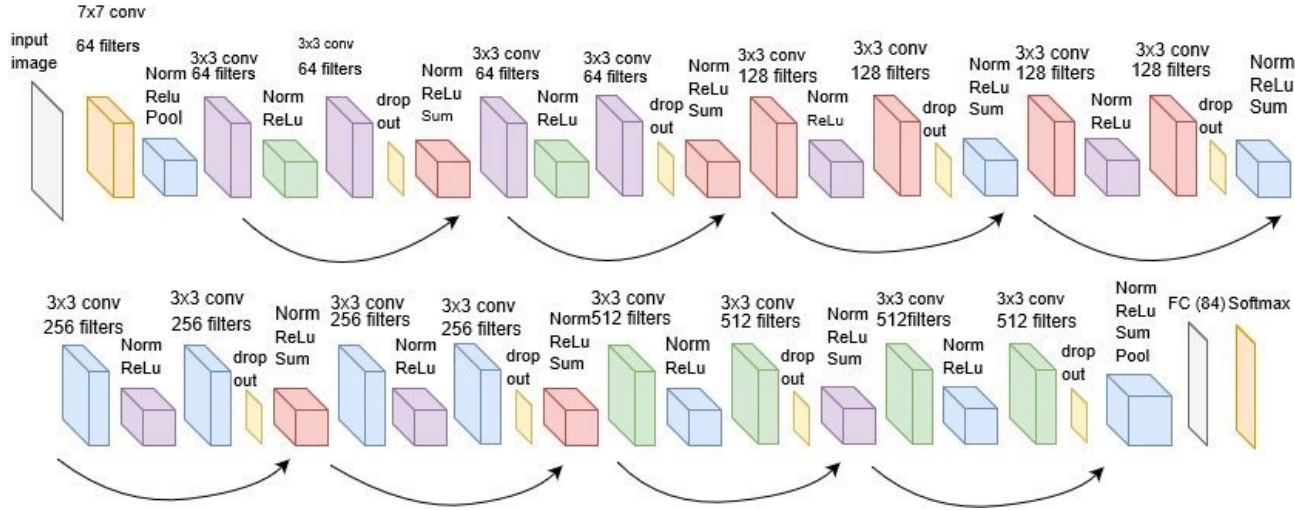
Model

Data (Examples)

*linear regression  
logistic*



# DEEP LEARNING



ResNet18

Convolution  
ReLU  
MaxPool  
Fully Connected  
SoftMax

*Attention*

# STOCHASTIC GRADIENT DESCENT → family

update rule

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

latest ←  
← Prev  
direction of change

sample input data

Initialize  $w$

For many iterations:

Loss = Forward pass → DNN

Gradient = backward

Update model

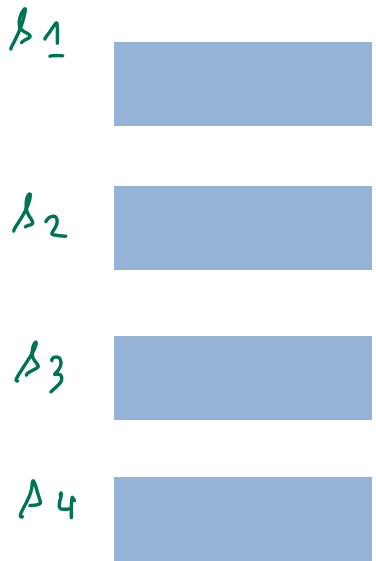
End

$f(w, \text{sample})$  how far away from labels  
automatic differentiation

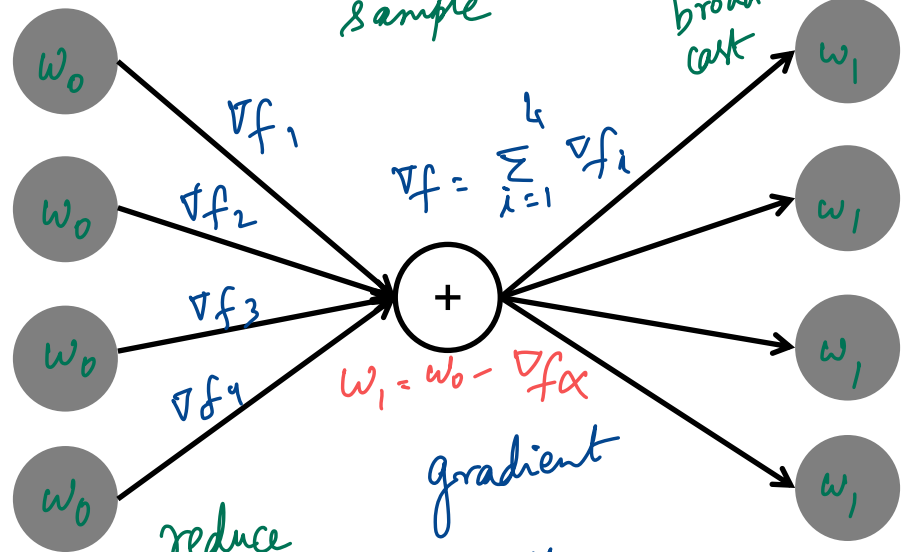
# DATA PARALLEL MODEL TRAINING

iter:  $\nearrow 1$   
 sample  $\rightarrow k$

(k) DP  $\nearrow$  match



forward pass, loss for its sample



next iteration

$\rightarrow$  retain the same computation if all data on single machine

$\hookrightarrow$  each machine samples data

reduce  $(\nabla f_1 \dots \nabla f_n) = \nabla f$

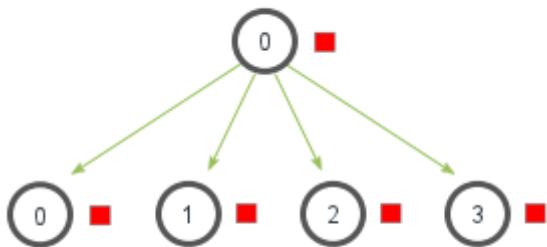
gradient across all samples

$\nabla f (s_1, s_2, s_3, s_4, \dots)$

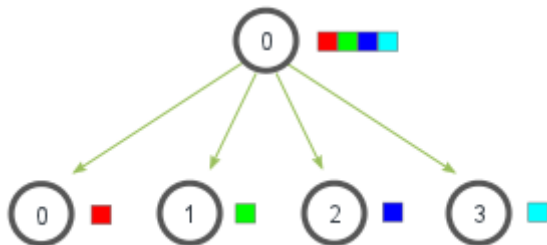
# COLLECTIVE COMMUNICATION

## Broadcast, Scatter

MPI\_Bcast



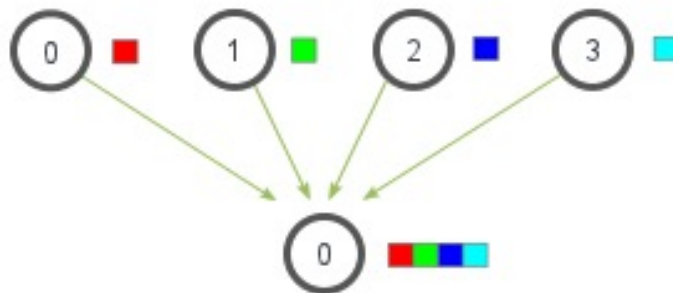
MPI\_Scatter



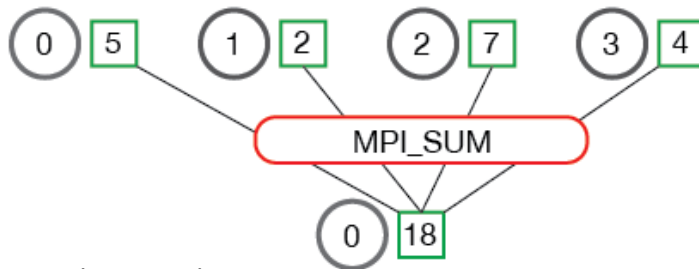
## Gather, Reduce

NCCL

MPI\_Gather



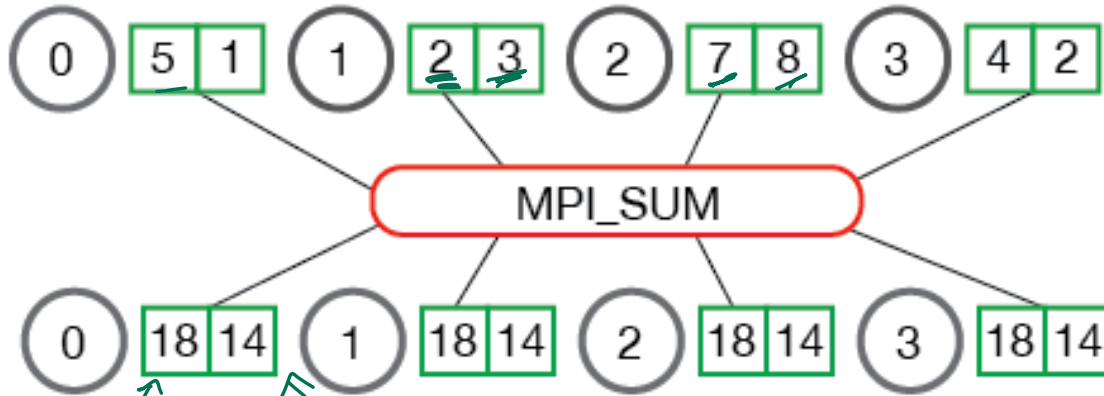
MPI\_Reduce



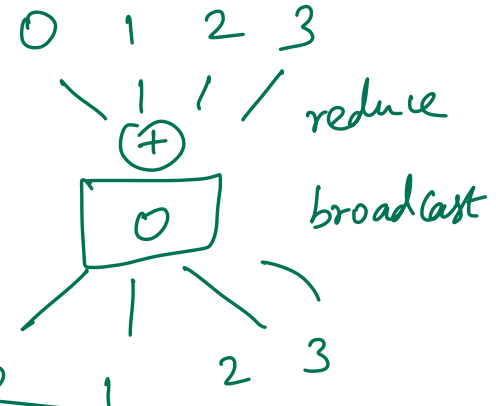
# ALL REDUCE USING A RING

BW  
latency

MPI\_Allreduce

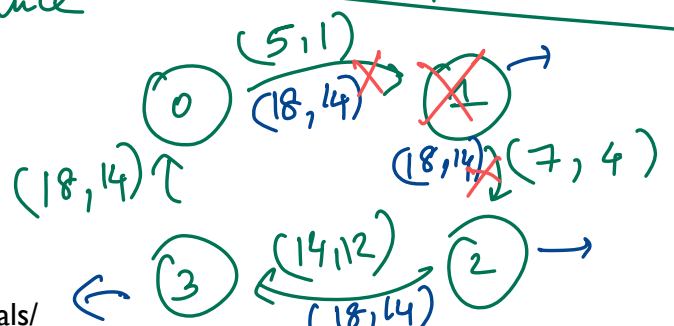


vector  $(x, y)$



$(x_0 + x_1 + \dots + x_3)$   $(y_0 + \dots + y_3)$  ring reduce

Fault Tolerance | 2 rounds then ring all reduce



# 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()
```

# GRADIENT BUCKETING

Why do we need gradient bucketing?

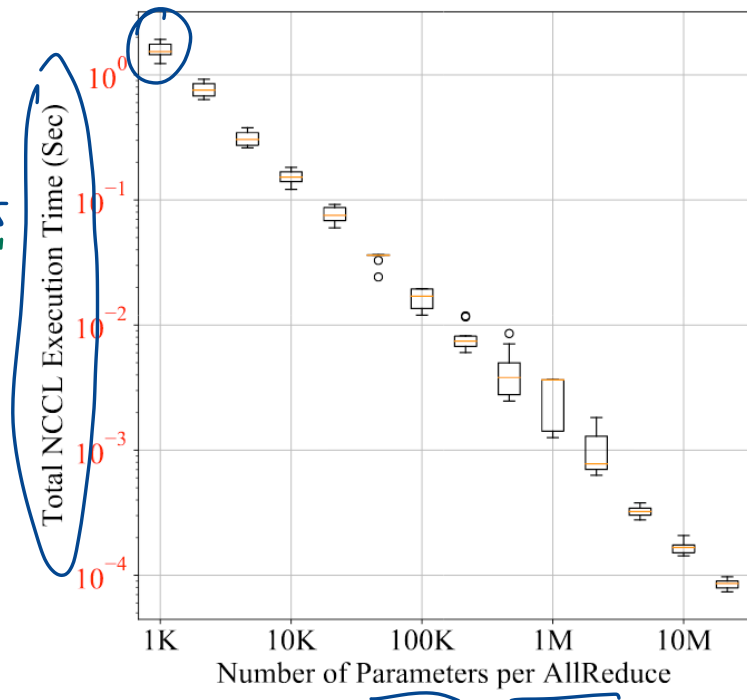
gradient = [ bucket<sub>0</sub> | bucket<sub>1</sub> | ... ]

millions of elements

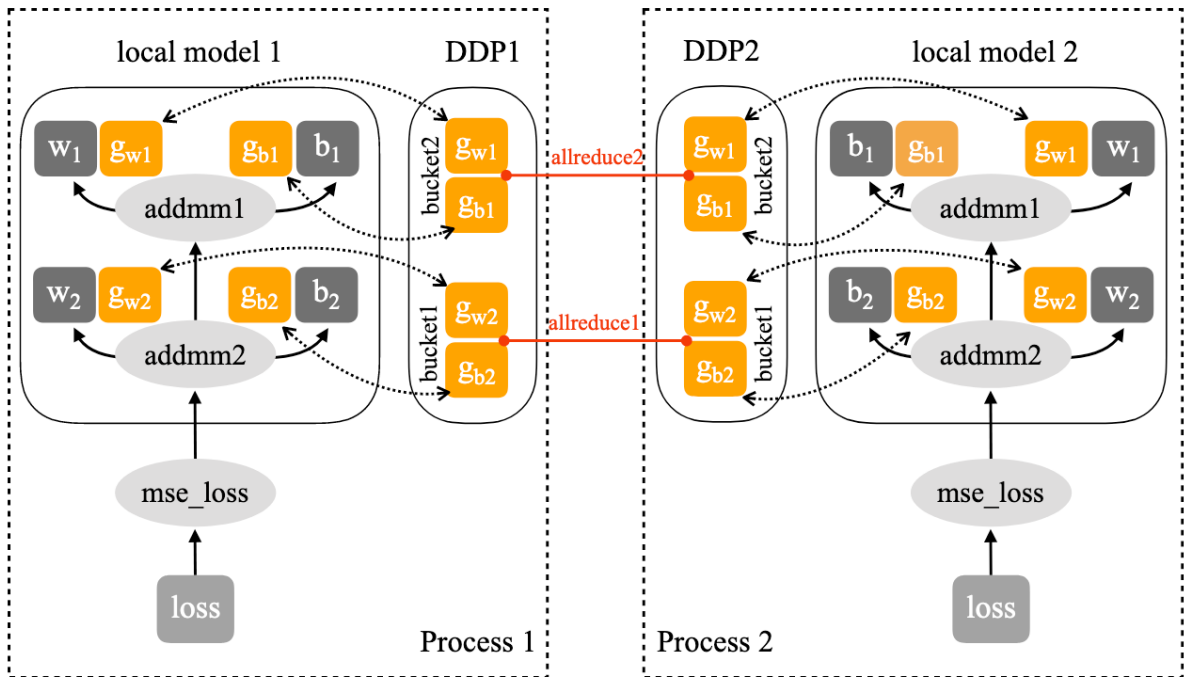
allreduce (bucket<sub>0</sub>)

... allreduce (bucket<sub>1</sub>)

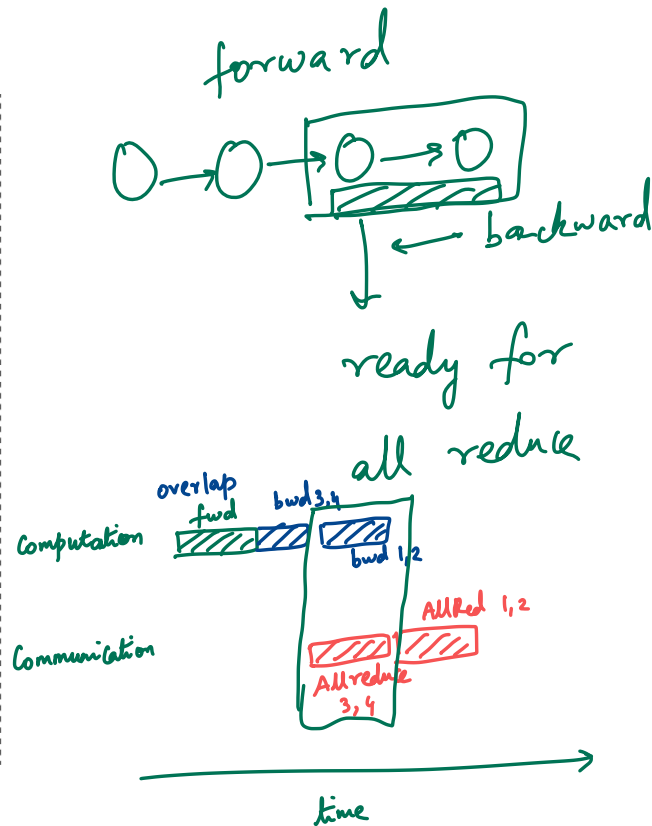
→ small bucket → overheads high



# GRADIENT BUCKETING + ALL REDUCE



Parameter
  Gradient
  Autograd Edge
  Copy
  Communication



# GRADIENT ACCUMULATION

---

```
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()
```

↪ large sample  
size

memory

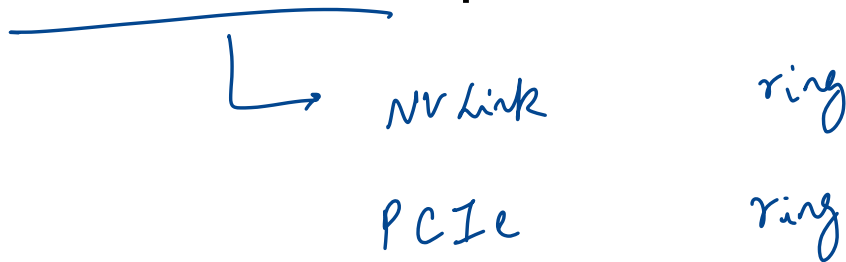
constraints

# IMPLEMENTATION

Bucket\_cap\_mb  $\longrightarrow$  25 mb

Parameter-to-bucket mapping

Round-robin ProcessGroups



# SUMMARY

Pytorch: Framework for deep learning

DistributedDataParallel API

Gradient bucketing, AllReduce

Overlap computation and communication

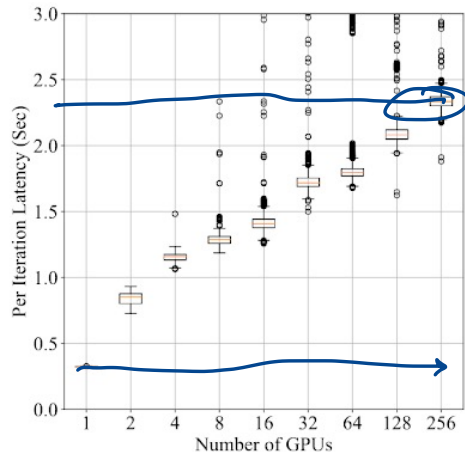
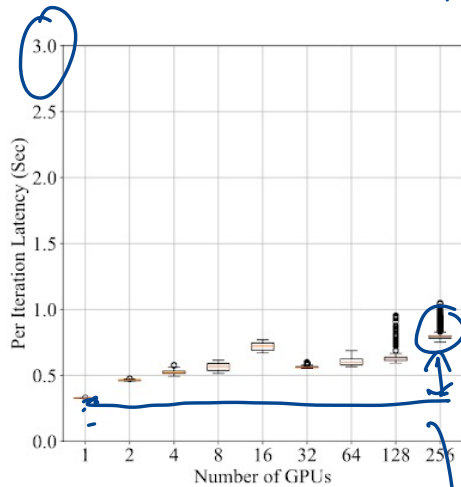
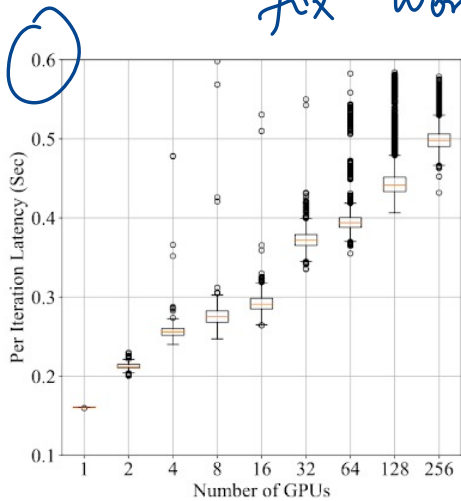
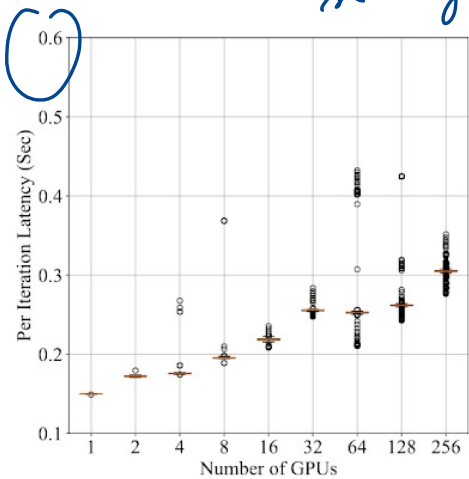


# DISCUSSION

<https://forms.gle/LcQFrbfcUQ2cBx6dA>

weak  
strong

increase processors → also increase work  
fix work → increase processors



(a) ResNet50 on NCCL

(b) ResNet50 on Gloo

(c) BERT on NCCL

(d) BERT on Gloo

Figure 9: Scalability

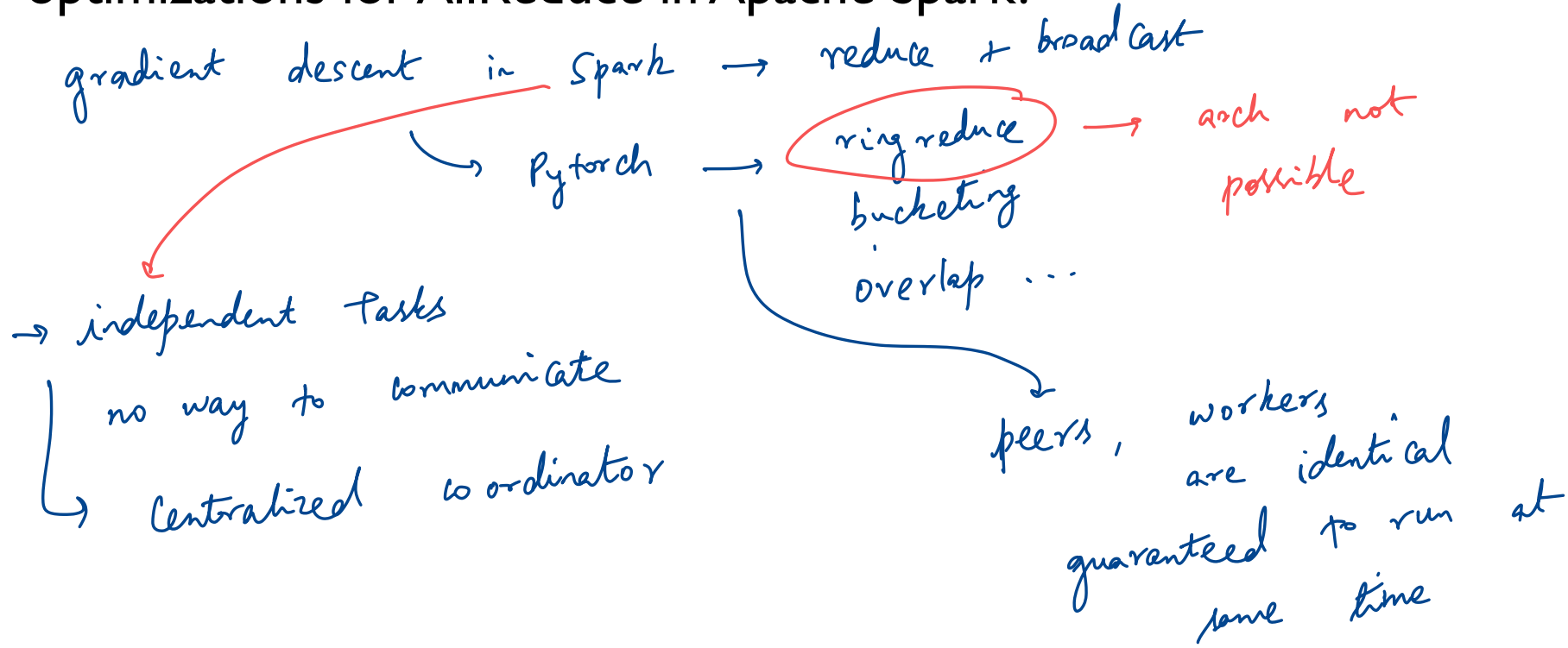
Variance is higher for gloo

NCCL scales better

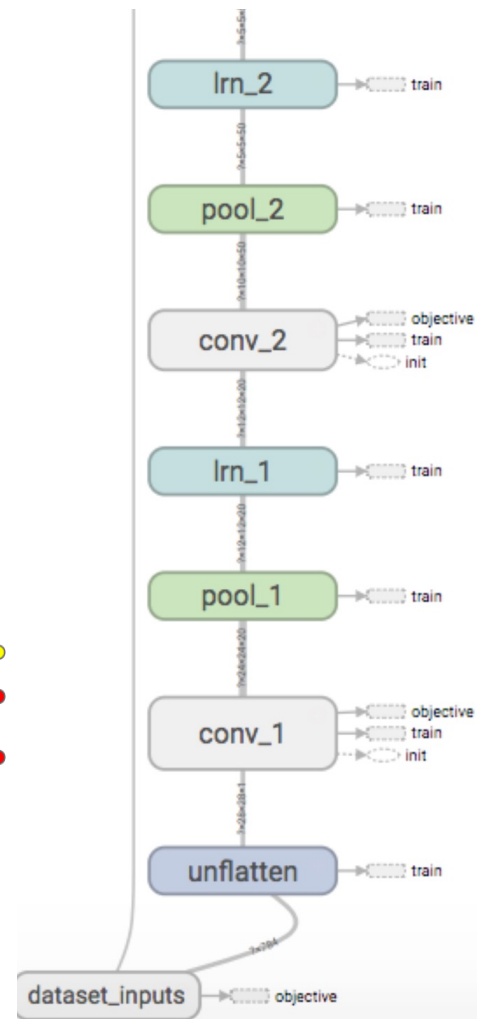
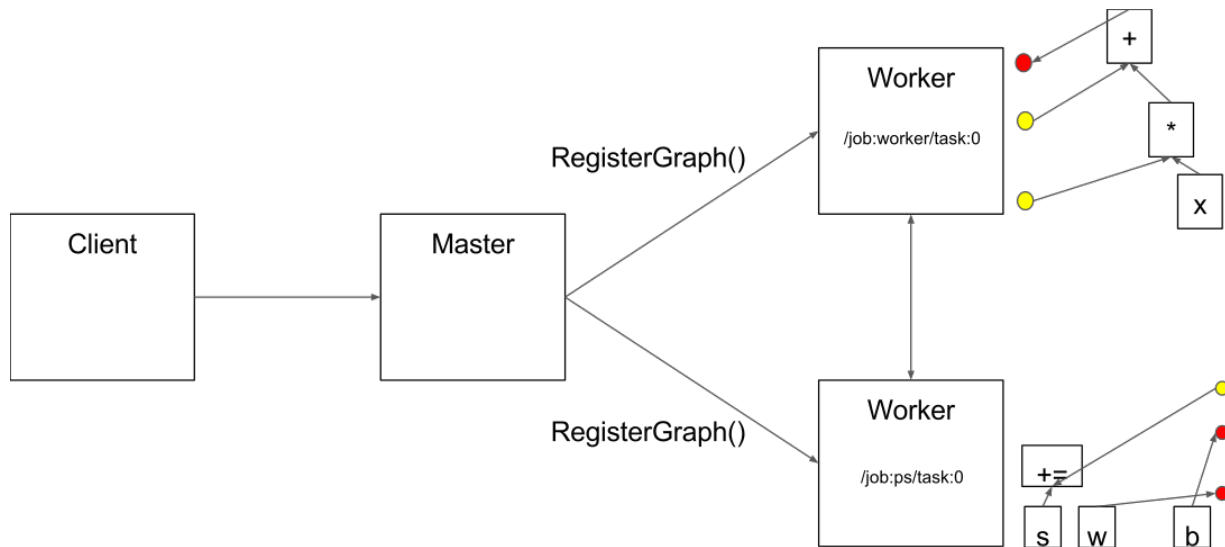
larger model

ideal "weak scaling"

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



# TENSORFLOW (2016)

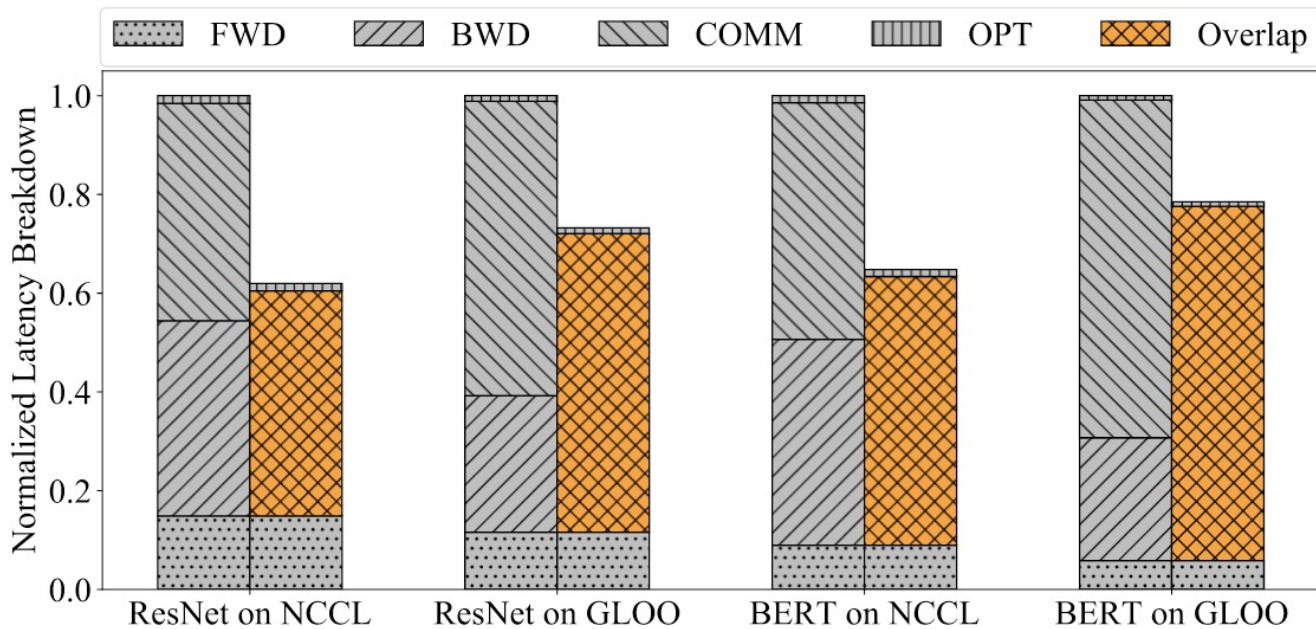


# NEXT STEPS

Next class: PipeDream

Assignment 2 is out!

# BREAKDOWN



**Figure 6: Per Iteration Latency Breakdown**