Assignment 2 out! Due Oct 13th early AM!

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

Title confirmed – Oct 14

Project Proposal (2 pages) – Oct 25

Introduction
Related Work
Timeline (with eval plan)
WRITING AN INTRODUCTION

1-2 paras: what is the problem you are solving why is it important (need citations)

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

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
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

Eval Plan
Describe what datasets, hardware you will use
Available: Cloudlab, Google Cloud (~$150), Jetson TX2 etc.
Scalable Storage Systems

Datacenter Architecture

Resource Management

Computational Engines

Machine Learning

SQL

Streaming

Graph

Applications

MapReduce

Spa Ic

GFS

Mesos

DRF

Datacenter Architecture
EMPIRICAL RISK MINIMIZATION

Train a ML model

Supervised learning

Function

Data (Examples)

Model

Regularization

Minimize $\|Z_i - z_i\|_2$

$f(x, w) = \sum_{i=1}^{N} f(w, z_i) + P(w)$

Avoids overfitting
DEEP LEARNING

ResNet18

- Convolution
- ReLU
- MaxPool
- Fully Connected
- SoftMax

$\ell_2 \left( \ell_1 \left( x_i, w_1 \right), w_2 \right)$

weights corresponding to each layer
STOCHASTIC GRADIENT DESCENT

Initialize $w$ ← $w_0$

For many iterations:

Loss = Forward pass
Gradient = backward pass
Update model

End

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

4 machine
forward
backward
$V_f(B, w_i)$
gradient

$W_i$ → $W_{i+1}$
aggregate (average)

$B_1$
$B_2$
$B_3$
$B_4$

Make sure updated model is on all machines

Parameter or Model averaging

$\nabla f(B, w_i) \Rightarrow$ update the model

to get
COLLECTIVE COMMUNICATION

Broadcast, Scatter

MPI_Bcast

Gather, Reduce

MPI_Gather

MPI_Scatter

MPI_Reduce

From https://mpitutorial.com/tutorials/
ALL REDUCE USING A RING

MPI_Allreduce

reduce, but output of reduce is present on all the machines
downside: latency is high

A single machine has constant number of messages = better than reduce + broadcast

From https://mpitutorial.com/tutorials/
DISTRIBUTED DATA PARALLEL API

# 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 BUCKETING

Why do we need gradient bucketing?

L When do we do aggregation
L larger aggregations → more efficient

Perf. model

\[ \text{Time}_{\text{comm}} = \alpha \text{ Time}_{\text{lat}} + \beta \text{Time}_{\text{bw}} \]

L fixed overhead
L scales with data size

Total number = 1B

Number of Parameters per AllReduce

1K 10K 100K 1M 10M
GRADIENT BUCKETING + ALL REDUCE

- Overlap computation with communication
- Bucket fixed size 25 MB
- Put grads into bucket until it fills up
- Trigger bucket AllReduce to happen in the background
Gradient Accumulation

def 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()

This is when all sync happens.
IMPLEMENTATION

Bucket_cap_mb = Profiling network topology + libraries
                25 MB

Parameter-to-bucket mapping → walk back from the end
                             and add layers to
                             buckets

Round-robin ProcessGroups

PCIe

                  □ → □ → □
                  ↑   ↓
□ ← □

                  D → D
                  ↑   ↓
                  D ← D

NVLink
Data vs. Compute vs. Communication

Imagenet \(\sim 100s \text{ of GB} \rightarrow 100s \text{ of epoch} \sim \text{ Days} \)

Web search logs \(\sim 100s \text{ of TB} \sim \text{ hours} \)
SUMMARY

Pytorch: Framework for deep learning
DistributedDataParallel API
Gradient bucketing, AllReduce
Overlap computation and communication
DISCUSSION

https://forms.gle/YnZC8PKQy1CDFJRF9
Figure 7: Per Iteration Latency vs Bucket Size on 16 GPUs

Figure 8: Per Iteration Latency vs Bucket Size on 32 GPUs
Figure 7: Per Iteration Latency vs Bucket Size on 16 GPUs

Figure 8: Per Iteration Latency vs Bucket Size on 32 GPUs
What could be some challenges in implementing similar optimizations for AllReduce in Apache Spark?

- Spark job could have many stages
- Reduction tree + broadcast
  - Slower than this paper?
- How mutable models can be handled

Spark: Many more tasks than cores / Assuming that all tasks are running at some time
NEXT STEPS

Next class: PipeDream
Assignment 2 is out!

Project Proposal
Preferences, Groups by Oct 11
2 pager by Oct 25
Figure 6: Per Iteration Latency Breakdown