Hello!

CS 744: PYTORCH

Shivaram Venkataraman
Fall 2022
Assignment 2 out! Due Oct 12\textsuperscript{th} 11 AM!

Paper review deadline – 12:59pm on Tue/Thu

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

Title confirmed – Oct 15

Project Proposal (2 pages) – Oct 25

\begin{itemize}
  \item Introduction
  \item Related Work
  \item Timeline (with eval plan)
\end{itemize}

\textcolor{blue}{Maybe start soon?}

\textcolor{blue}{Survey in one or two weeks}
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

→ 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

Apache Spark

→ 1 or 2 page

→ interactive data analytics

→ Many industries need this

→ 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

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

Datacenter Architecture

Resource Management

Computational Engines

SQL

Streaming

Graph

Applications

MapReduce

Spark

GFS

Mesos

DRF

Hardware

Machine Learning

Applications

Applications

Applications
EMPIRICAL RISK MINIMIZATION

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

- Function
- Regularization
- Model
- Data (Examples)

Supervised learning

Training data, labels

Linear Regression

Logistic
DEEP LEARNING

\[ t_1 = l_1 \left( \text{input} \right) \]
\[ t_2 = l_2 \left( t_1 \right) \]

forward pass

weights

parameters

ResNet18

Convolution
ReLU
MaxPool
Fully Connected
SoftMax

classify images or other tasks
STOCHASTIC GRADIENT DESCENT

Initialize $w$

For many iterations:

Loss = Forward pass

Gradient = backward

Update model

End

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

Initialize model $w$

For many iterations:

Loss = Forward pass

Gradient = backward

Update model

End
DATA PARALLEL MODEL TRAINING

\[ \nabla = \nabla_B + \nabla_1 + \nabla_2 + \nabla_3 \]

same as grad with \( B \)

Mimic or reproduce behavior of single machine training!
COLLECTIVE COMMUNICATION

1991 first MPI standard

Broadcast, Scatter
- MPI_Bcast
- MPI_Scatter

Gather, Reduce
- MPI_Gather
- MPI_Reduce

→ gathering data from all machines

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

MPI_Allreduce

Reduce = Aggregate data from all workers

Agg data + Broadcast agg value to all workers

- All nodes broadcast to other nodes. Aggregate locally

bottleneck from one machine

reduce + broadcast

data transmitted per worker

number of connections

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()
Why do we need gradient bucketing?

- Bucketing
- Aggregate
- Every small layer Tensor
- Bucketing
- Aggregate whole gradient

![Graph showing the relationship between total NCCL execution time and number of parameters per AllReduce.](graph.png)
Gradient Bucketing + All Reduce

\[
T_{com} = \alpha T_{lat} + \beta T_{BW}
\]

You can pipeline backward & gradient aggregation first.
Gradient Accumulation

define 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

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

Fail latency is higher for larger num GPUs
What could be some challenges in implementing similar optimizations for AllReduce in Apache Spark?
Next class: PipeDream
Assignment 2 is out!

Project Proposal – Check Piazza!
Figure 6: Per Iteration Latency Breakdown