Hello!

CS 744: PIPEDREAM

Shivaram Venkataraman

Spring 2024

ADMINISTRIVIA

Assignment 2 is due on 2/23

Project Proposal (2 pages)

- Introduction → 1 page
- Related Work → 0.5 to 0.75 pages
- Timeline (with eval plan)

WRITING AN INTRODUCTION

1-2 paras: what is the problem you are solving → Scale ML Training better

why is it important (need citations) → CLAIM → back this up

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

→ related work section

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

→ workloads or machines what metrics
datasets

RELATED WORK, EVAL PLAN

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

Explain what the papers / projects do

8 or 9 prior works

Why are they different / insufficient

Eval Plan

Describe what datasets, hardware you will use

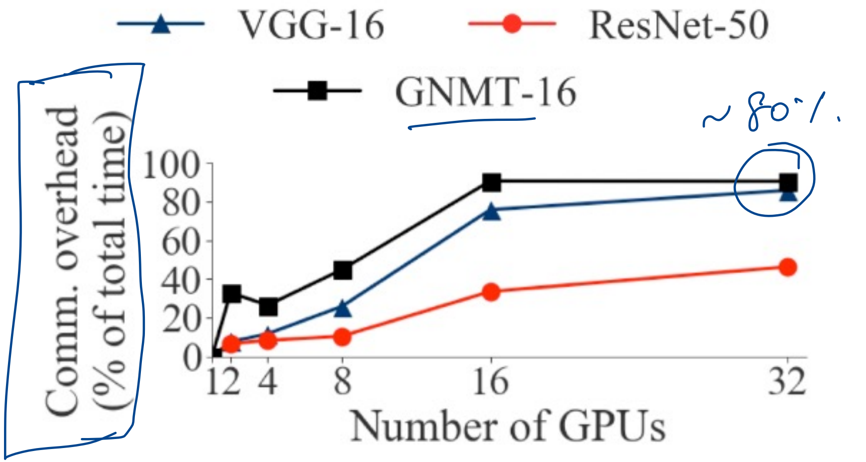
Available: Cloudlab, Google Cloud (~$150), Jetson TX2 etc.

LIMITATIONS OF DATA PARALLEL

- Does not scale well with more GPUs
- Comm overhead increase with num GPUs

  as high as ~80% for some models with 32 GPUs
- Replicate model on all machines

  ⇒ Memory footprint



8xV100s with NVLink (AWS)
PyTorch + NCCL 2.4

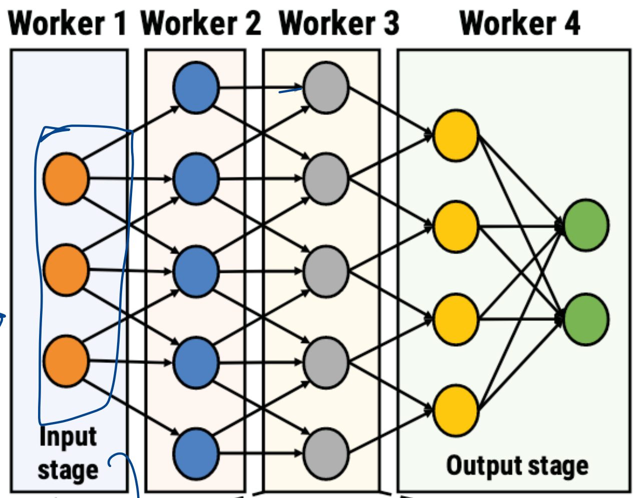
"fraction of training time spent in communication stalls"

MODEL PARALLEL TRAINING

first layer

Memory requirement goes down

Comm is constant and not dependent on total num workers



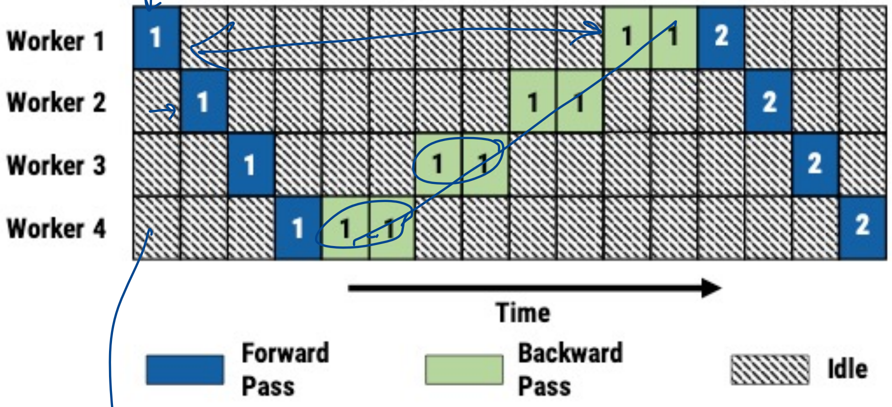
batch =32

activations or output

gradients

gradient

32



under utilization of hardware

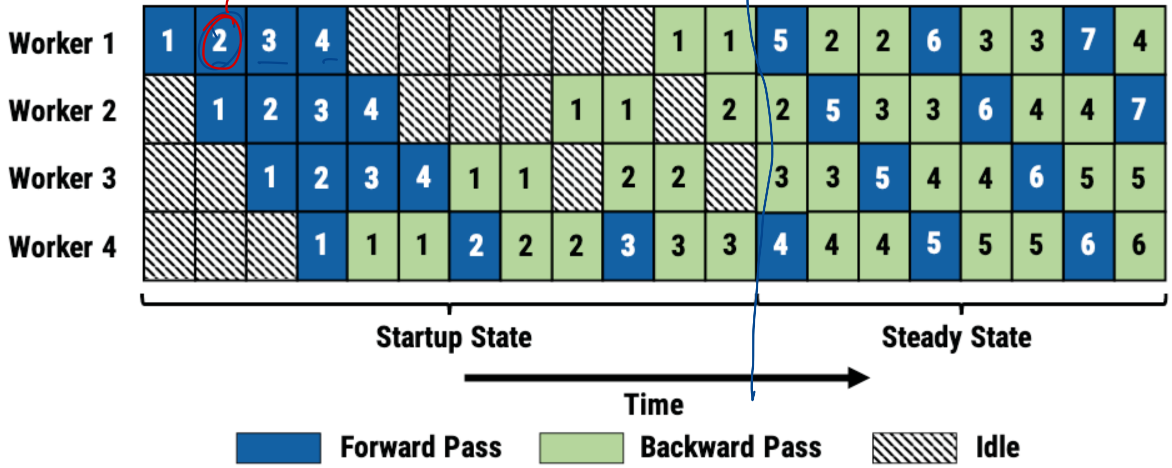
PIPELINE PARALLEL

Instead of 1 input batch
what if pipeline k batches

while retain
MP benefits

batch 2 is sent to w1

100% utilization



Advantages?

→ Better utilization

Partitioning →
Scheduling
Accuracy or Learning

Challenges

CHALLENGE 1: WORK PARTITIONING

each stage approx same amount of time

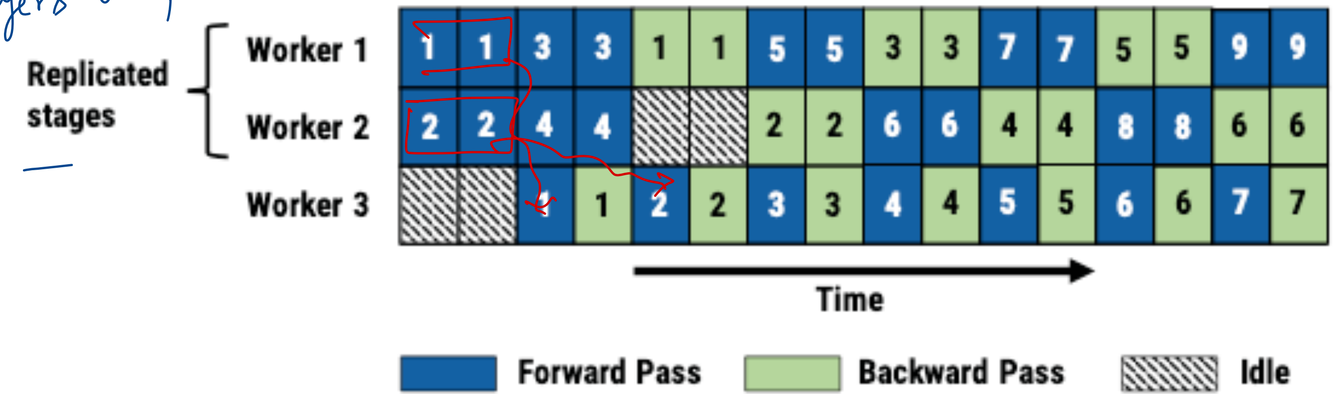
Goal: Balanced stages in the pipeline. Why?

Steady state throughput is the throughput of the slowest stage

Stages can be replicated! Ex: Two stage pipeline, but first stage is replicated

layers or params are replicated

Hybrid



WORK PARITIONING

Offline → before you start training

Profiler: computation time for forward, backward for each layer

size of output activations, gradients (network transfer)

size of parameters (memory)

Dynamic programming algorithm → sub problems that solve the bigger problem

Intuition: Find optimal partitions within a server,

Then find best split across servers using that

CHALLENGE 2: WORK SCHEDULING

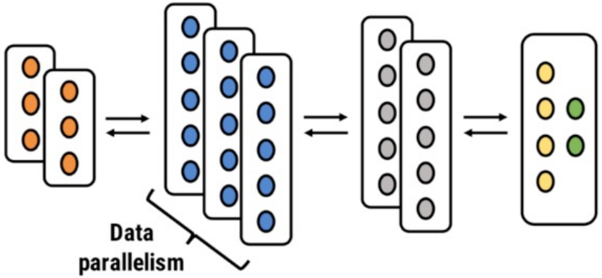
Traditional data parallel

forward iter(i)

backward iter(i)

forward iter(i+1)

… bw i+1



Pipeline parallel: Worker can

Forward pass to push to downstream

Backward pass to push to upstream

CHALLENGE 2: WORK SCHEDULING

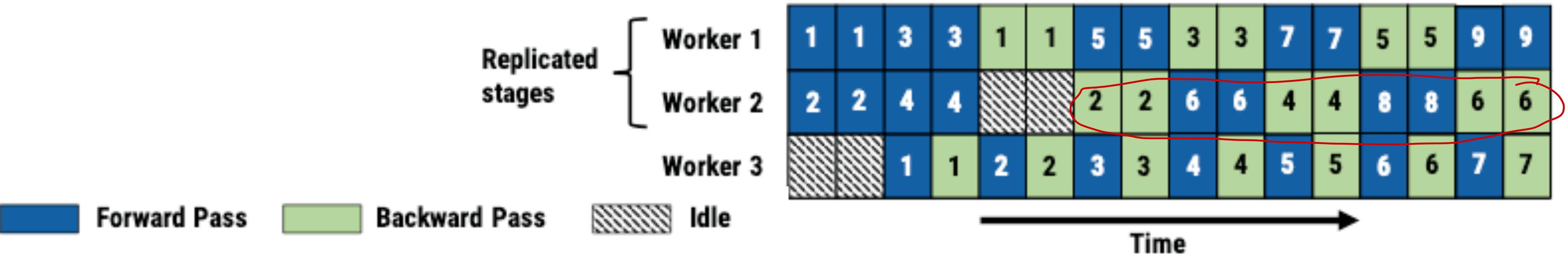
Num active batches ~= num_workers / num_replicas_input

different batches

Schedule one-forward-one-backward (1F1B) – Worker 3

Round-robin for replicated stages → Worker 2

same worker for fwd, backward - " stickiness "



CHALLENGE 3: EFFECTIVE LEARNING

Naïve pipelining

Different model versions forward and backward

Fwd pass for batch 5 uses version 2

Back pass for batch 5 use version 4

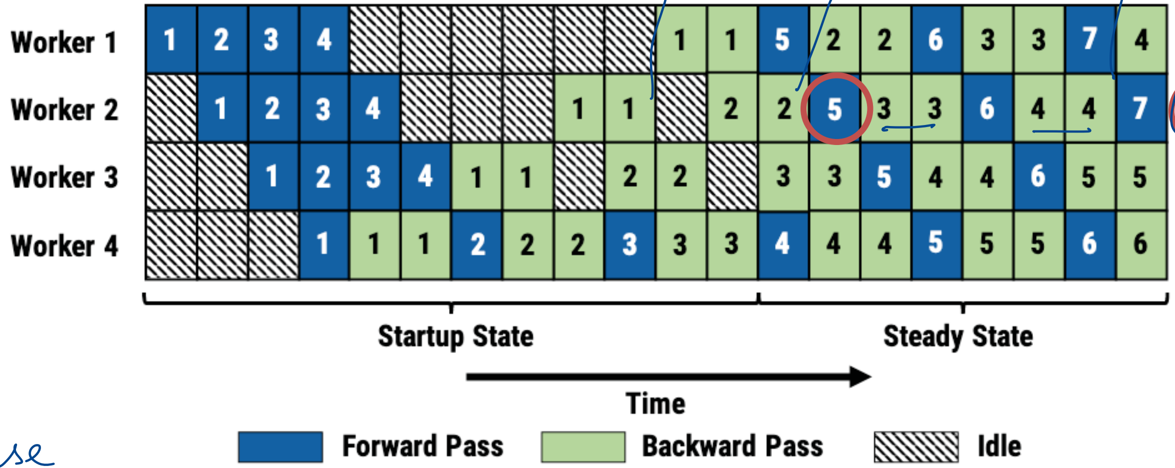
W_2^1 W_2^2 W_2^4

W_1

W_2^0

W_3

W_4



5

multiple versions of weights

use same version for fwd / bwd

increases mem use

CHALLENGE 3: EFFECTIVE LEARNING

Weight stashing

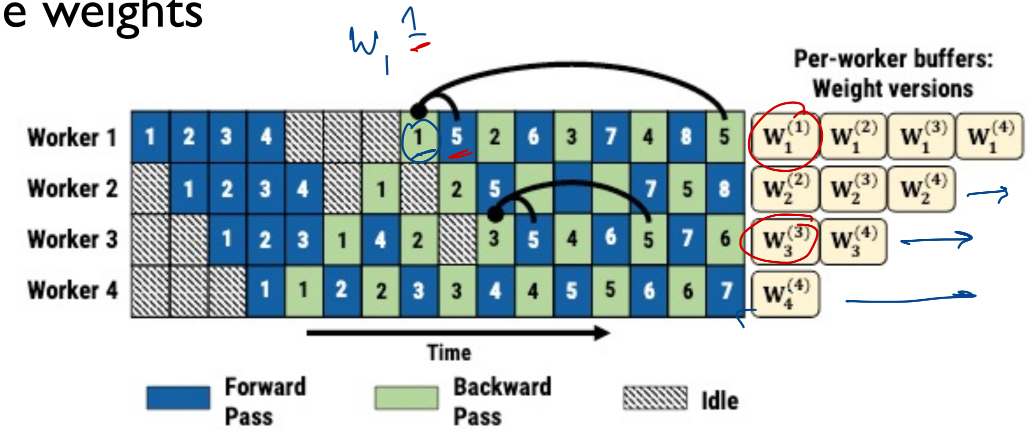
Maintain multiple versions of the weights

One per active mini-batch

Use latest version for forward pass.

Retrieve for backward

No guarantees across stages!



STALENESS, MEMORY OVERHEAD

How to avoid staleness:

Vertical sync → first worker note down the version number used and send it along the pipeline

Memory overhead

Similar to data parallel? → matches Data Parallel semantics

→ Further increases mem requirement

SUMMARY

Pipeline parallelism: Combine inter-batch and intra-batch

Partitioning: Replication, dynamic programming

Scheduling: 1F1B

Weight management: Stashing, vertical sync

DISCUSSION

https://forms.gle/BNwx6Nnmoh6EKAwJ9

Data Parallel is best for small models

Convolutions ⟷ FC

number of param

List two takeaways from the following table

PCIe, 10 Gbps

Model Name	Model Size	GPUs (#Servers x #GPUs/Server)	PipeDream Config	Speedup over DataParallel (Epoch Time)
Resnet-50	97MB	4x4 2x8	16 16	1x 1x
VGG-16	528MB	4x4 2x8	15-1 15-1	5.28x 2.98x
GNMT-8	1.1GB	3x4 2x8	Straight 16	2.95x 1x

NVLink, 25Gbps

5x

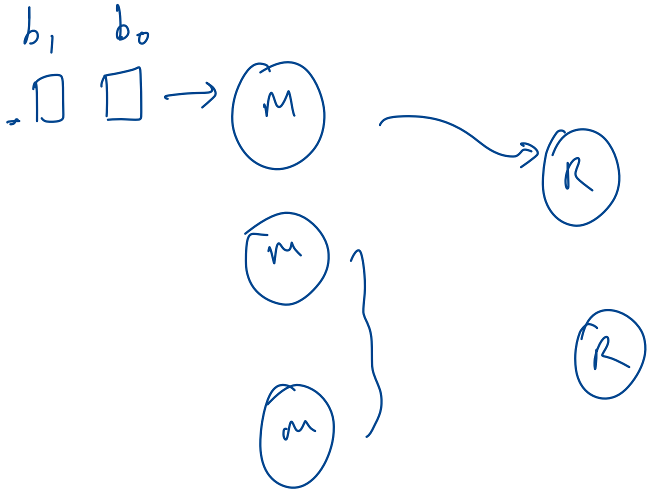
wins are smaller for larger model

What are some other workload scenarios (e.g. things we discussed for MapReduce or Spark) that could use similar ideas of pipelined parallelism? Develop such one example and its execution

Page Rank

- pipeline across iterations ?

  ↳ ready for next iteration



Streaming

NEXT STEPS

Next class: LLMs!

Work on Assignment 2!