CS 744: PIPEDREAM

Shivaram Venkataraman

Fall 2020

ADMINISTRIVIA

- Assignment 2 is due Oct 5th!
- Course project groups due today!
- Project proposal aka Introduction (10/16)

  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)

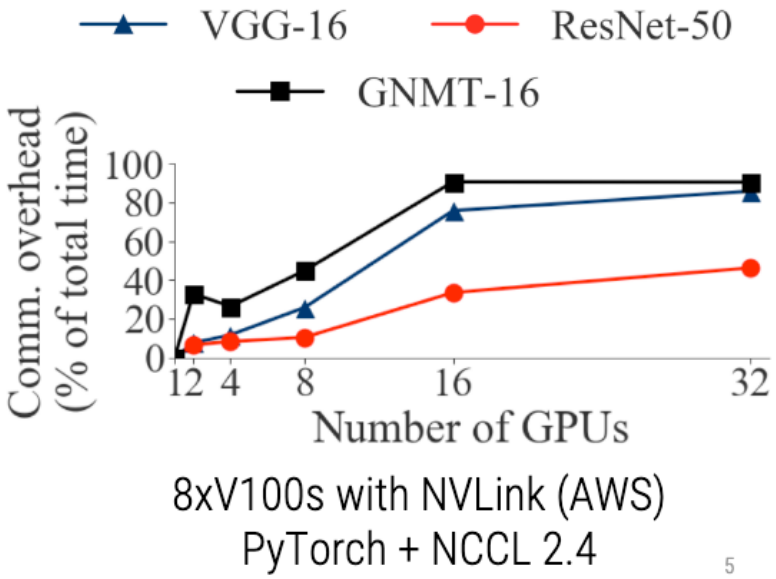
Explain what the papers / projects do

Why are they different / insufficient

Eval Plan

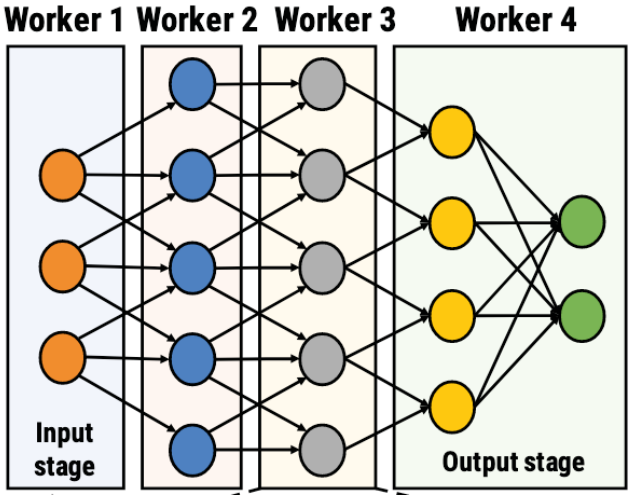
- Describe what datasets, hardware you will use
- Available: Cloudlab, Google Cloud (~$150), Jetson TX2 etc.

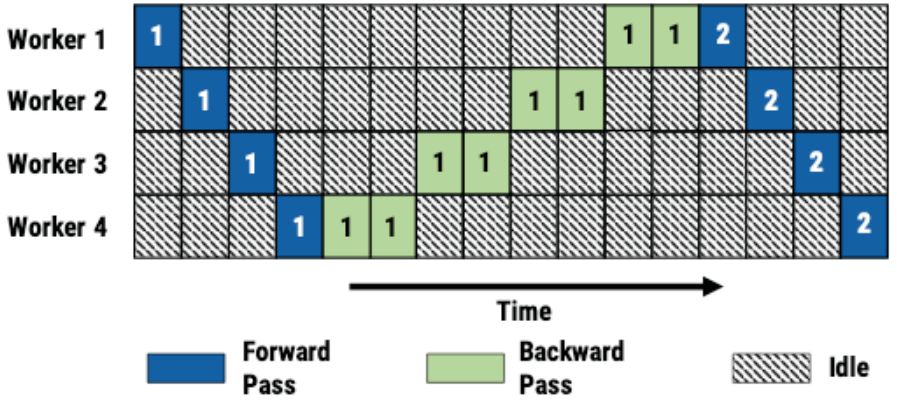
LIMITATIONS OF DATA PARALLEL



"fraction of training time spent in communication stalls"

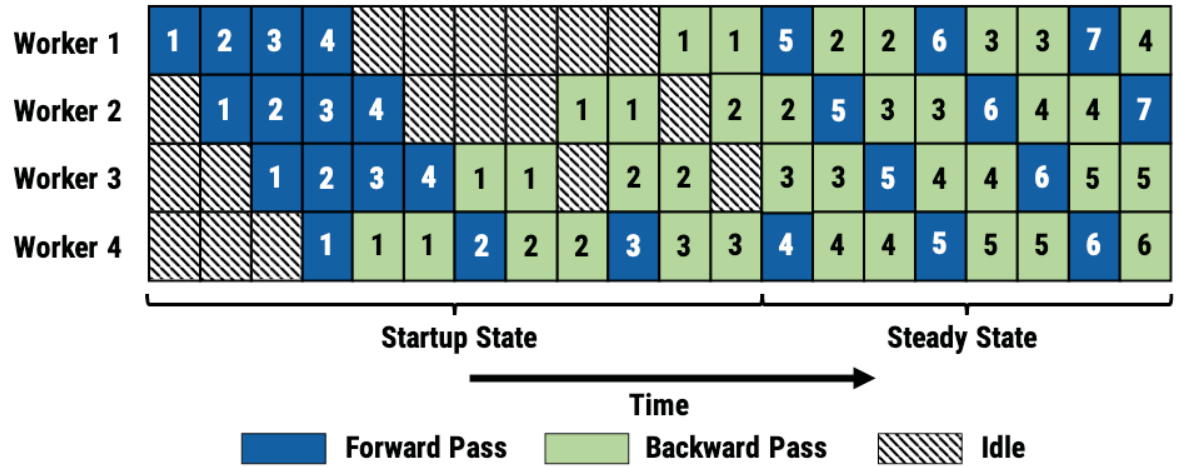
MODEL PARALLEL TRAINING





PIPELINE PARALLEL

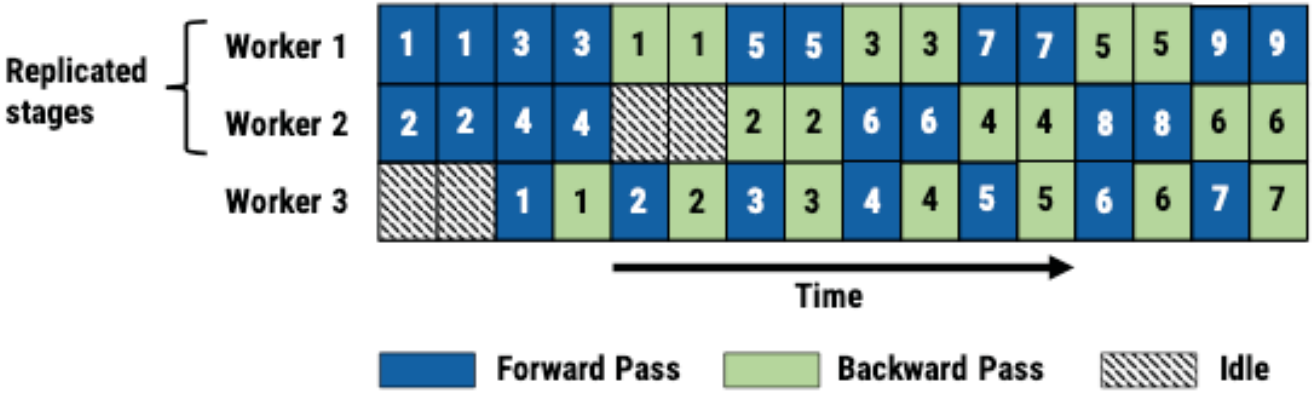
Advantages?



CHALLENGE 1: WORK PARTITIONING

Goal: Balanced stages in the pipeline. Why?

Stages can be replicated!



WORK PARITIONING

Profiler: computation time for forward, backward
size of output activations, gradients (network transfer)
size of parameters (memory)

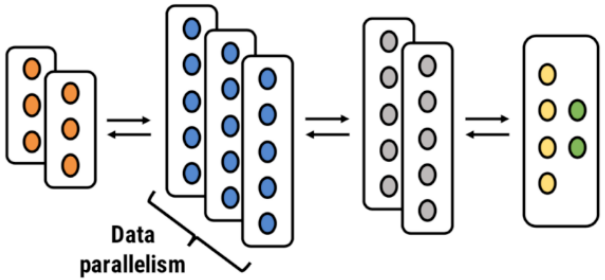
Dynamic programming algorithm

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)
- …



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

Schedule one-forward-one-backward (1F1B)

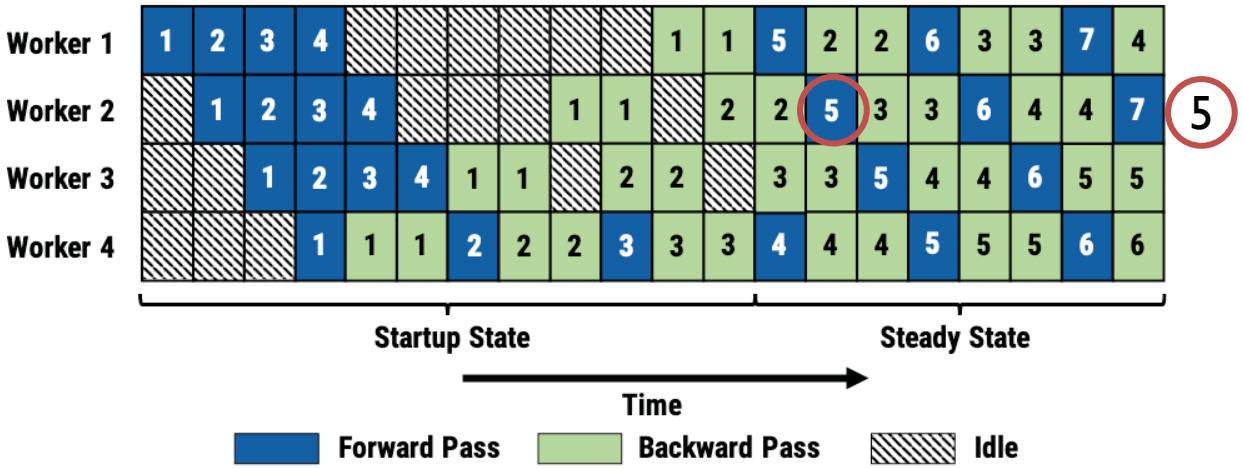
Round-robin for replicated stages →

same worker for fwd, backward

CHALLENGE 3: EFFECTIVE LEARNING

Naïve pipelining

Different model versions forward and backward



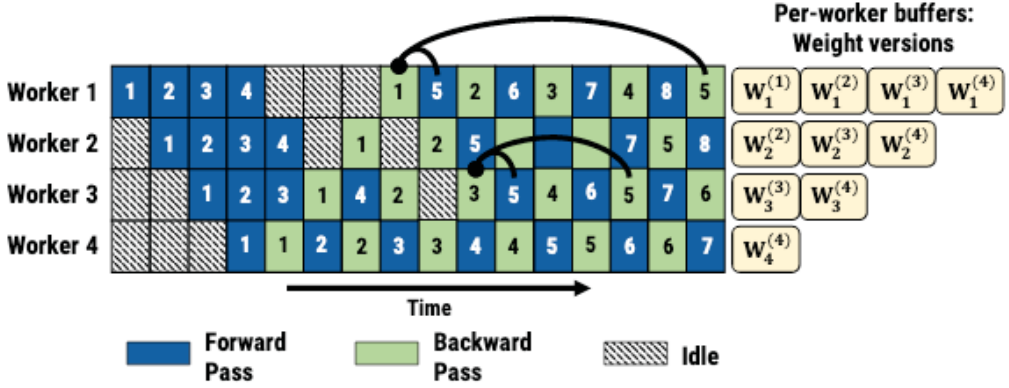
CHALLENGE 3: EFFECTIVE LEARNING

Weight stashing

- Maintain multiple versions of the weights
- One per active mini-batch

Use latest version for forward pass.

Retrieve for backward



STALENESS, MEMORY OVERHEAD

How to avoid staleness:

- Vertical sync

Memory overhead

- Similar to data parallel?

SUMMARY

Pipeline parallelism: Combine inter-batch and intra-batch

Partitioning: Replication, dynamic programming

Scheduling: 1F1B

Weight management: Stashing, vertical sync

DISCUSSION

https://forms.gle/GdVRuE8rBHH2vPPW6

List two takeaways from the following table

Model Name	Model Size	GPUs (#Servers x #GPUs/Server)	PipeDream Config	Speedup over DataParallel (Epoch Time)
Resnet-50	97MB	4x4 2x8	16 16	1× 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

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

NEXT STEPS

Next class: TVM

Assignment 2 is out!

Course project deadlines

- Today! (titles, groups)
- Oct 16 (introductions)