Welcome back!

CS 744: PIPEDREAM

Shivaram Venkataraman

Spring 2025

ADMINISTRIVIA

Assignment 2 is due on 2/13 → report

Project Proposal (2 pages) → ~2-3 pages

- Introduction → What is it that you want to do
- Related Work
- Timeline (with eval plan) → metrics, datasets, machines that you need

WRITING AN INTRODUCTION

1-2 paras: what is the problem you are solving → Problem Statement

why is it important (need citations)

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

↳ related work

1-2 paras: How do you plan on solving it and why your approach is better → Plan

1 para: Anticipated results or what experiments you will use

↳ change for final report

↓ validate your approach

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

~8-9 prior papers or projects that you summarize

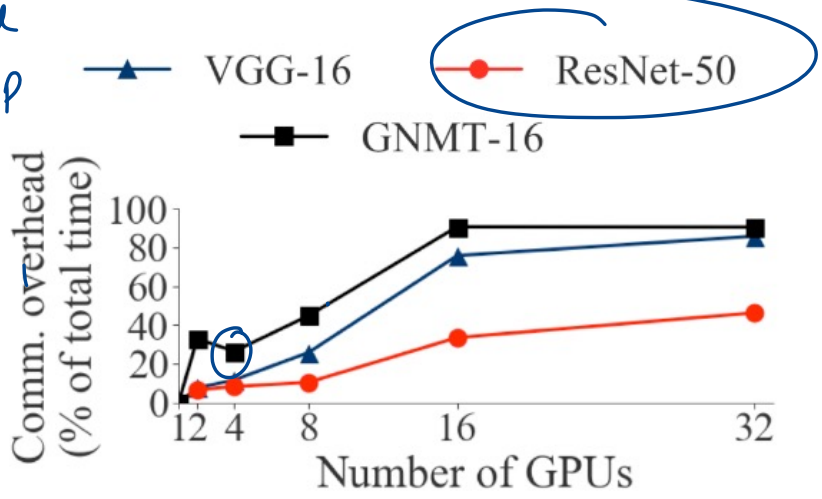
Eval Plan

Describe what datasets, hardware you will use

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

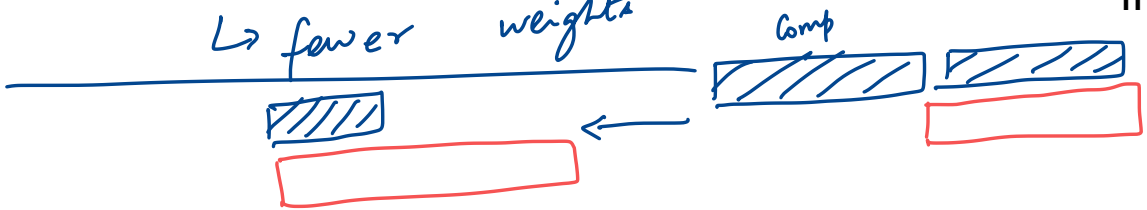
LIMITATIONS OF DATA PARALLEL

- Comm costs tend to dominate as num GPUs increases (eval DDP)
- These jumps are not linear (8 → 16 vs. 16 → 32)
- Convnets vs. VGG or GNMT-16
  - fewer weights



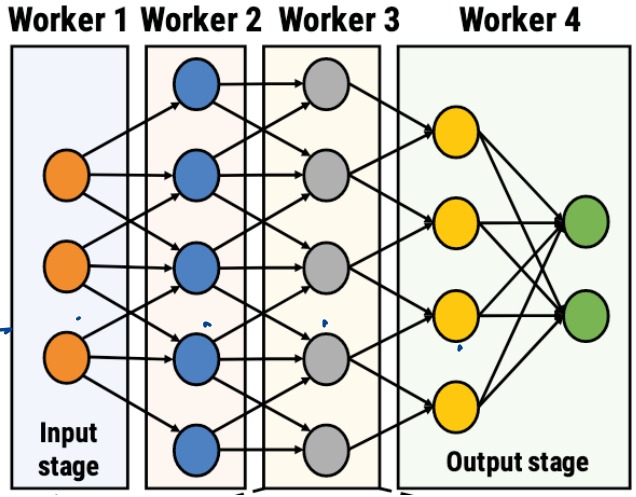
8xV100s with NVLink (AWS)
PyTorch + NCCL 2.4

"fraction of training time spent in communication stalls"



MODEL PARALLEL TRAINING

Partition the model across workers



batch = 32

1 2 3 (4, 5)

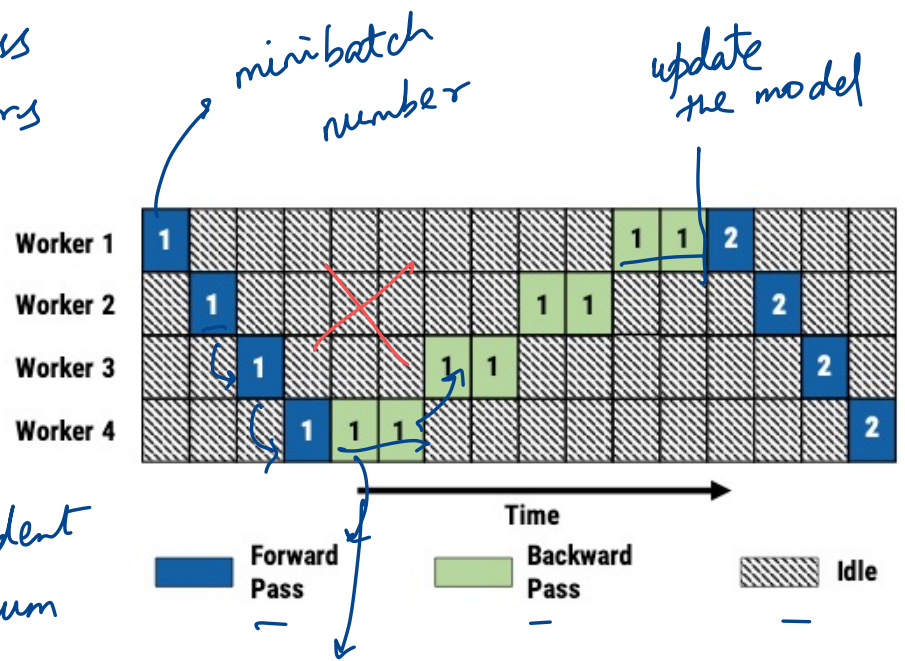
Fwd → activations

← Bwd gradients

① Mem req

② Comm independent of num workers

③ Idle boxes

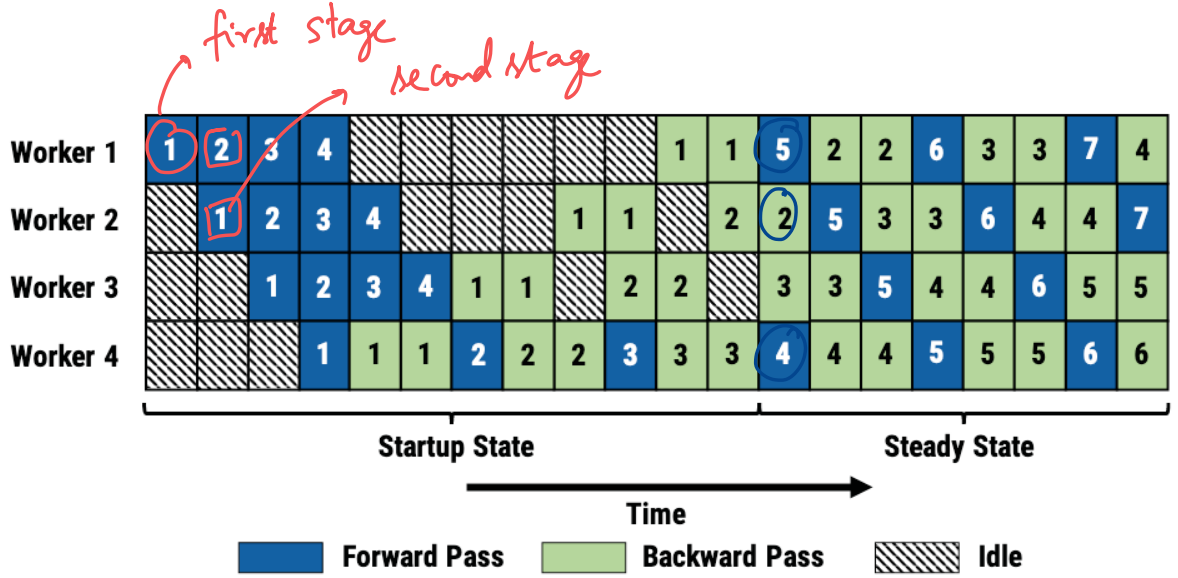


backward pass for mb 1

PIPELINE PARALLEL

Instead of 1 input batches → k input batches

↳ retain benefits from MP (memory, comm)



Advantages?

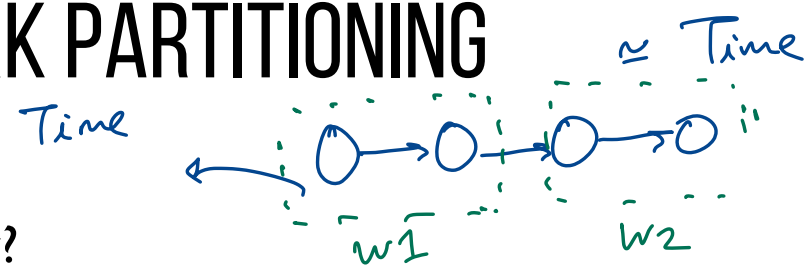
Better utilization from pipelining

Partitioning →

Schedule work at a worker

Accuracy ??

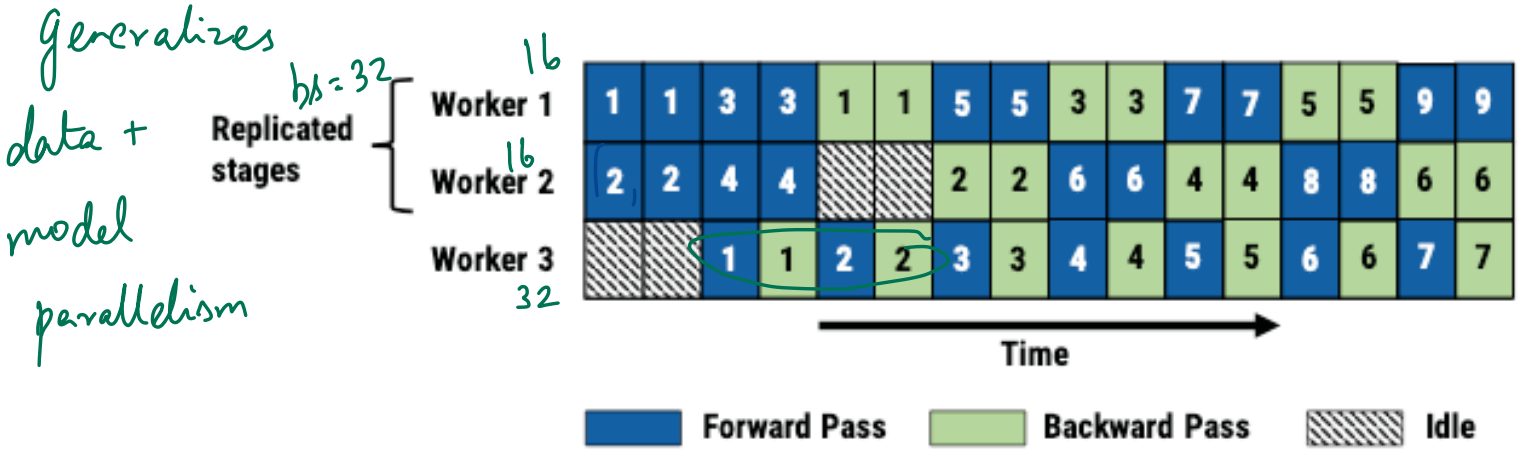
CHALLENGE 1: WORK PARTITIONING



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



WORK PARITIONING

Profiler: computation time for forward, backward for each layer
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

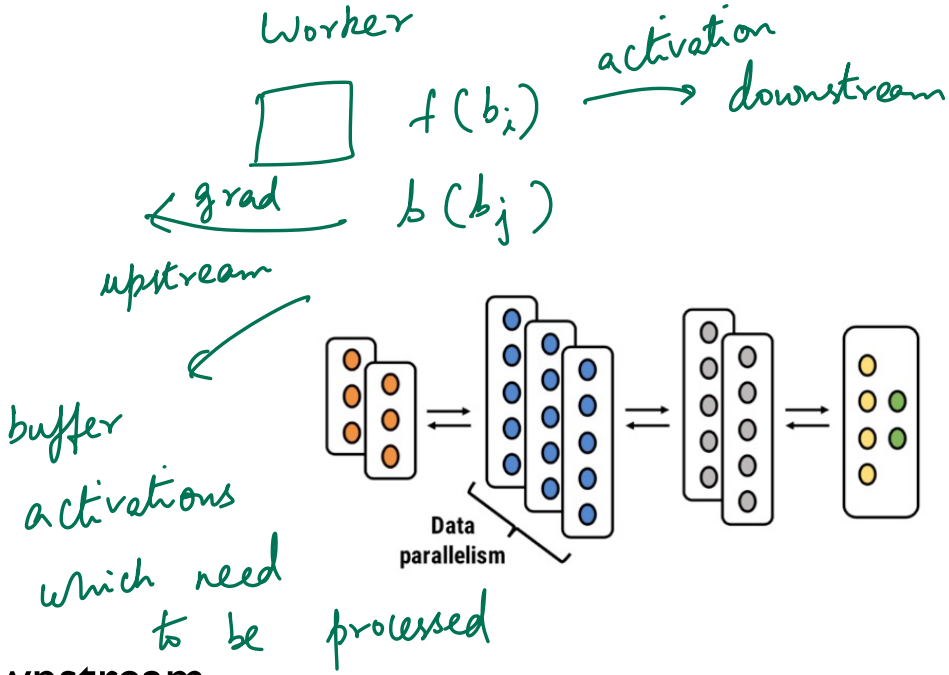
CHALLENGE 2: WORK SCHEDULING

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



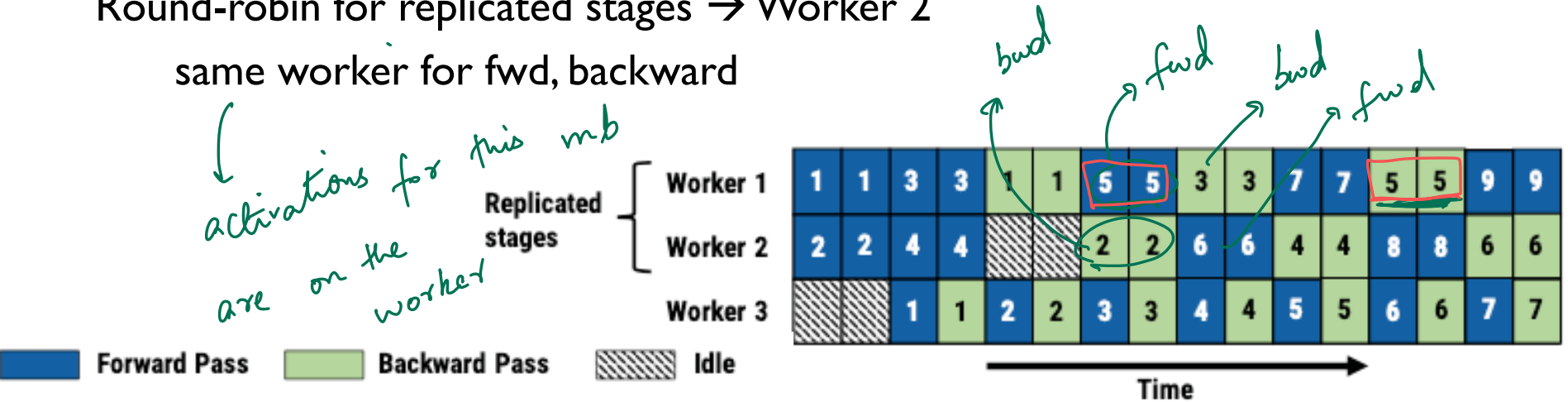
CHALLENGE 2: WORK SCHEDULING

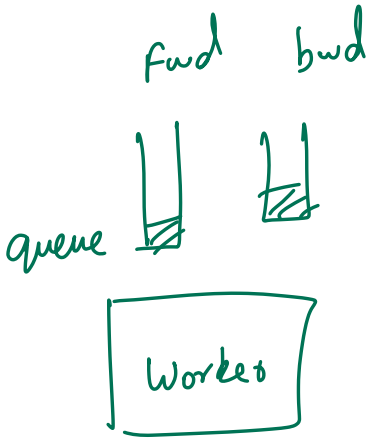
Num active batches ~= num_workers / num_replicas_input

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

Round-robin for replicated stages → Worker 2

same worker for fwd, backward



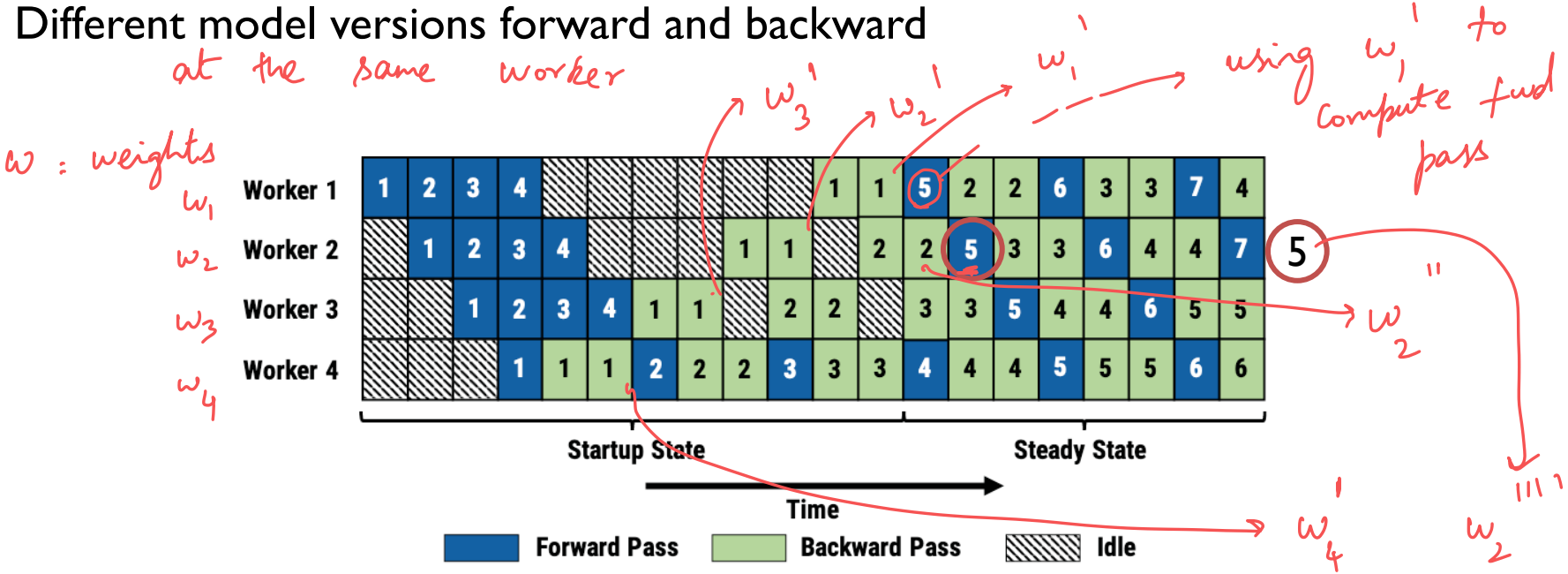


CHALLENGE 3: EFFECTIVE LEARNING

→ diff model versions at diff workers same batch $:= f(w_1^1, b_5)$

Naïve pipelining

→ Different model versions forward and backward at the same worker



CHALLENGE 3: EFFECTIVE LEARNING

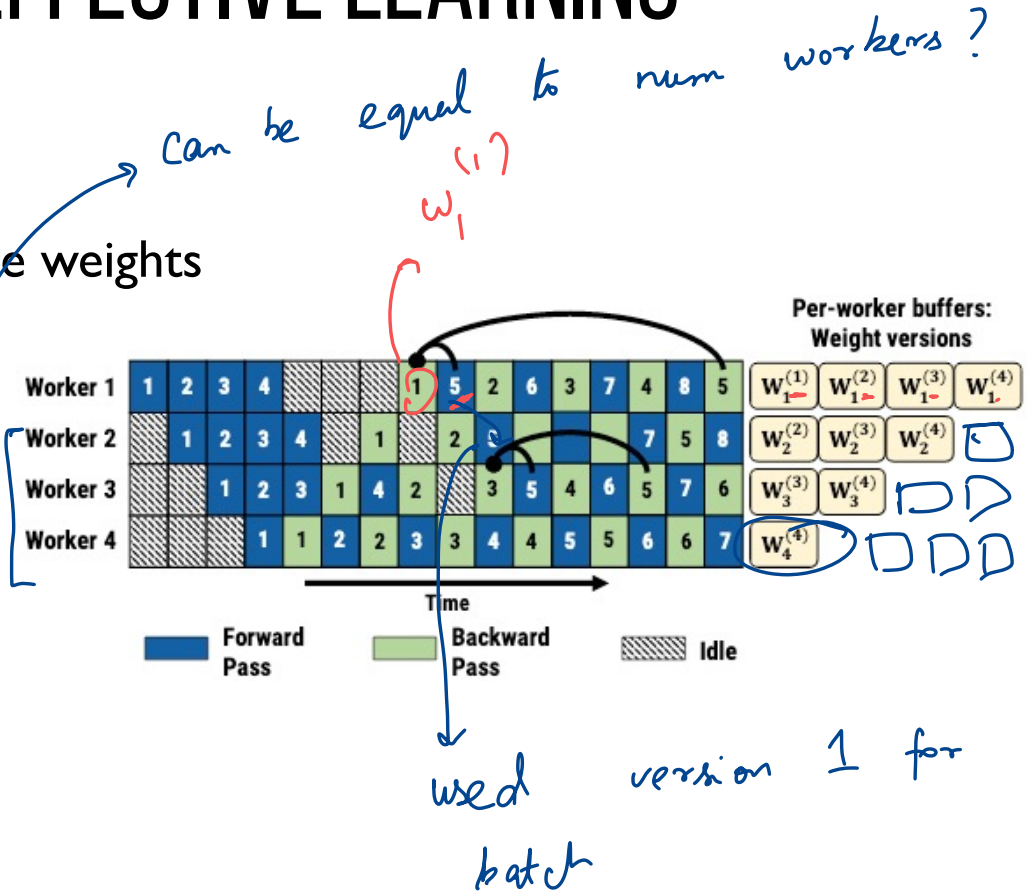
Weight stashing

- 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 → when you send activations include model version number used so far

Memory overhead

- Similar to data parallel? → 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/A2Kium67PBT8uHeTA

for VGG
more servers
=> higher speedup

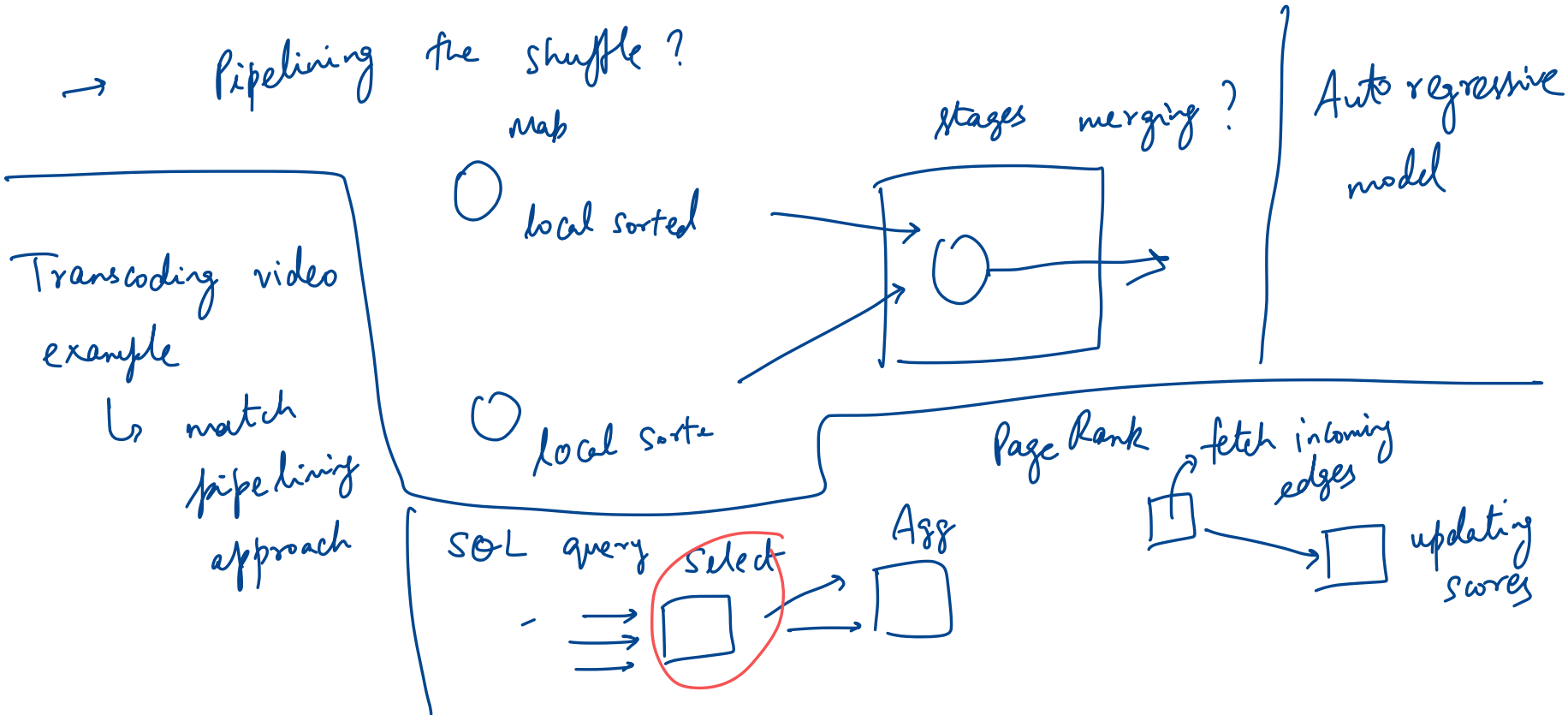
List two takeaways from the following table

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

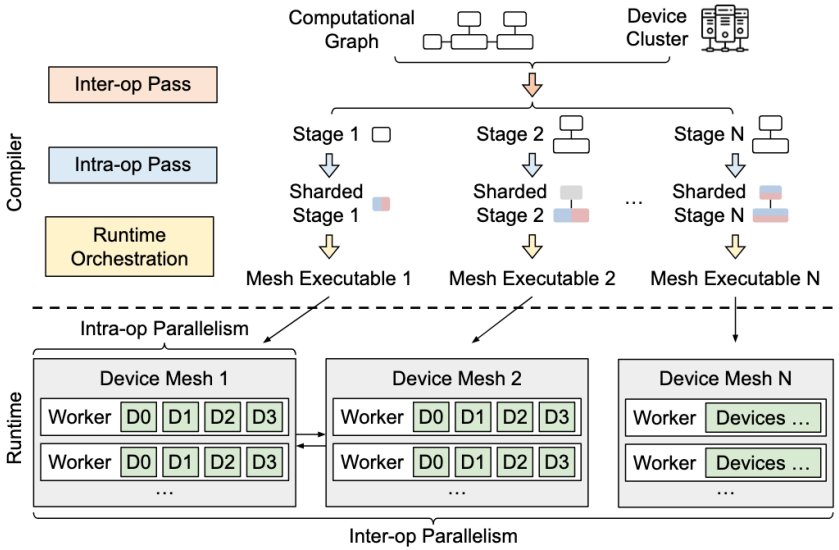
replicated

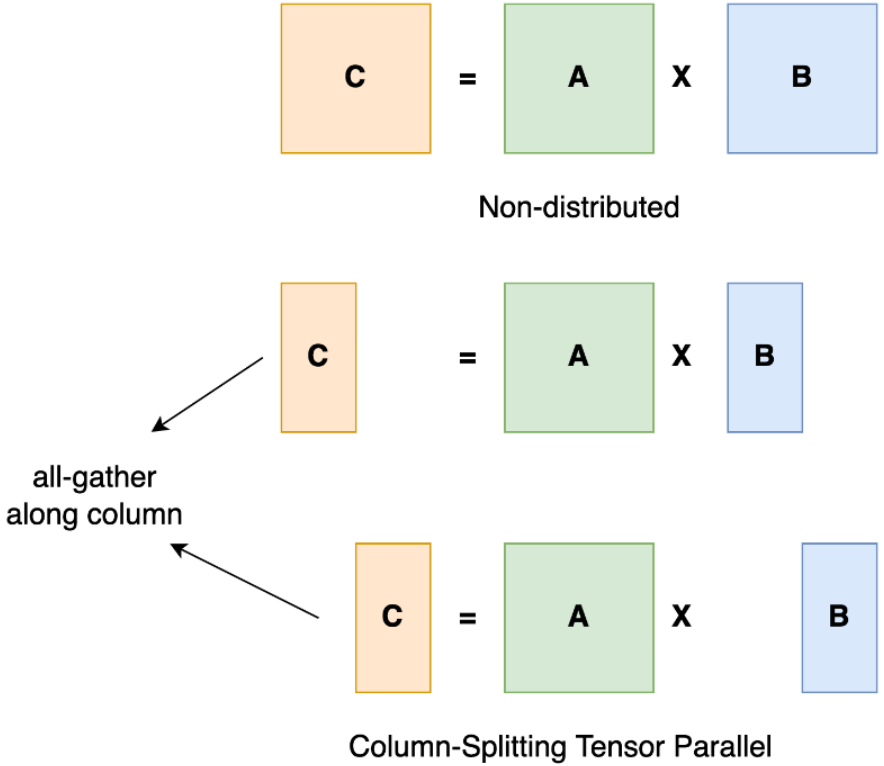
idle

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



3D PARALLELISM, ALPA





Tensor parallel illustration

NEXT STEPS

Next class: More LLMs!

Work on Assignment 2!