Hello!

CS 744: PIPEDREAM

Shivaram Venkataraman

Fall 2021

ADMINISTRIVIA

- Assignment 2 is due Wednesday AM! Tuesday evening!
- Course project groups due Oct 11, Monday!

  Rank 1 → highest pref

  ⋮

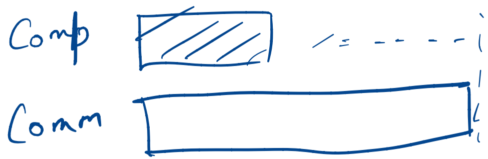
  Rank 10 lowest pref

- Project proposal aka Introduction (10/25)

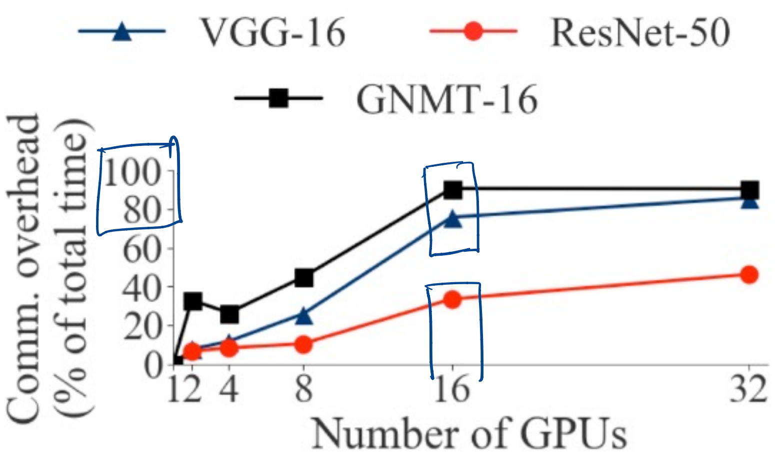
LIMITATIONS OF DATA PARALLEL

1. Overhead of comm. is high with data parallelism

2. Under utilization



3. Overhead / Scaling is different for GNMT vs. Resnet



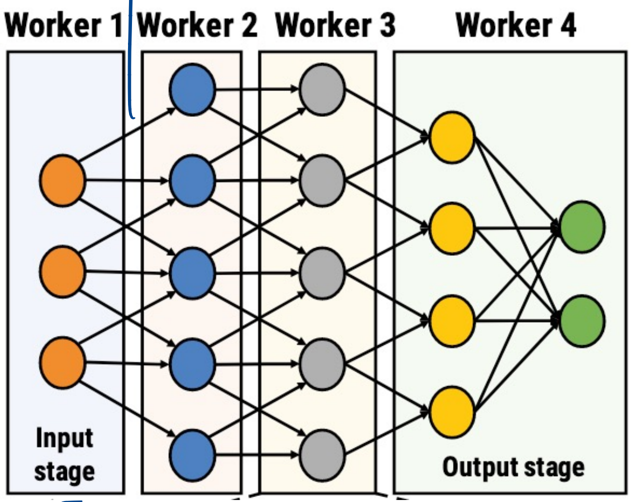
8xV100s with NVLink (AWS)
PyTorch + NCCL 2.4

"fraction of training time spent in communication stalls"

MODEL PARALLEL TRAINING

activations

data



First layer

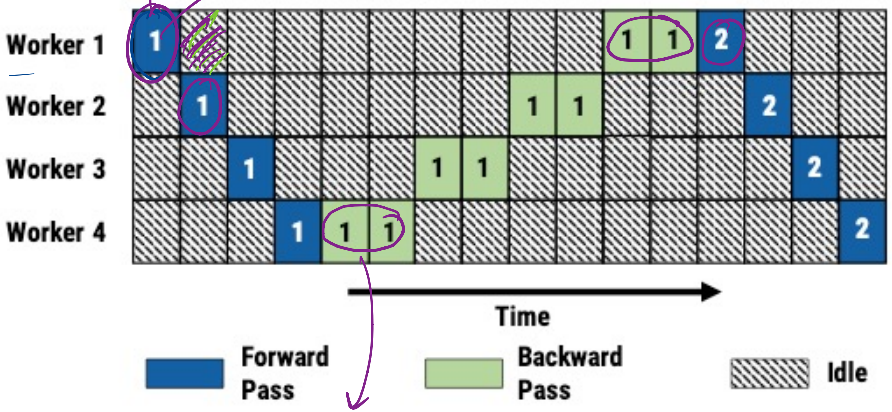
Last 2 layers

bs = 128 examples

CIFAR-10

first mini batch

one time unit



backwards pass takes 2 time unit

Low UTILIZATION

(a) Memory overhead is lower

(b) Comm. per worker is only related to activation size

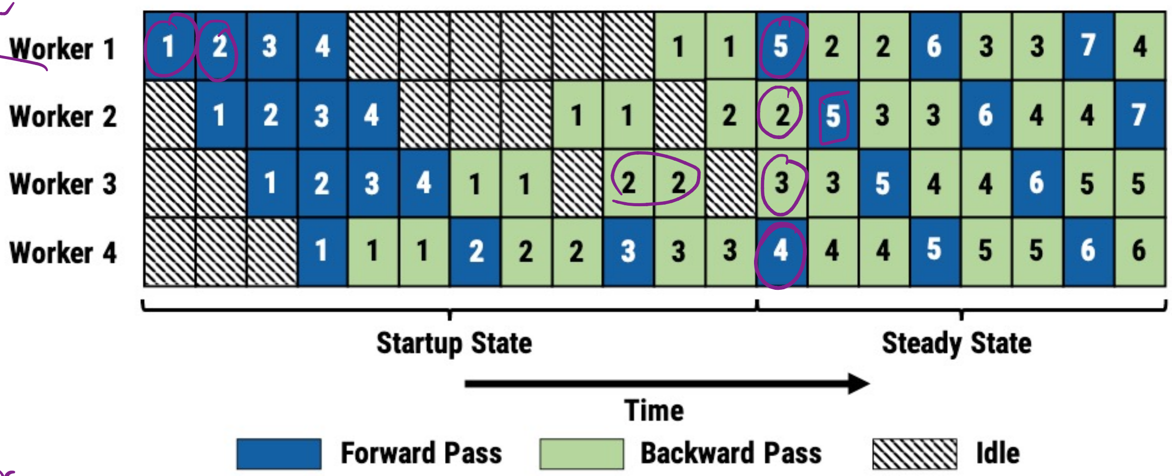
PIPELINE PARALLEL

→ Instead of having one mini batch in flight you can have four! →

① Partitioning

② Scheduling

③ Learning



Advantages?

→ High utilization after startup

→ The comm. per worker is proportional to activation size

CHALLENGE 1: WORK PARTITIONING

Stage: Collection of layers at a given worker

Goal: Balanced stages in the pipeline. Why?

↳ bad utilization if unbalanced

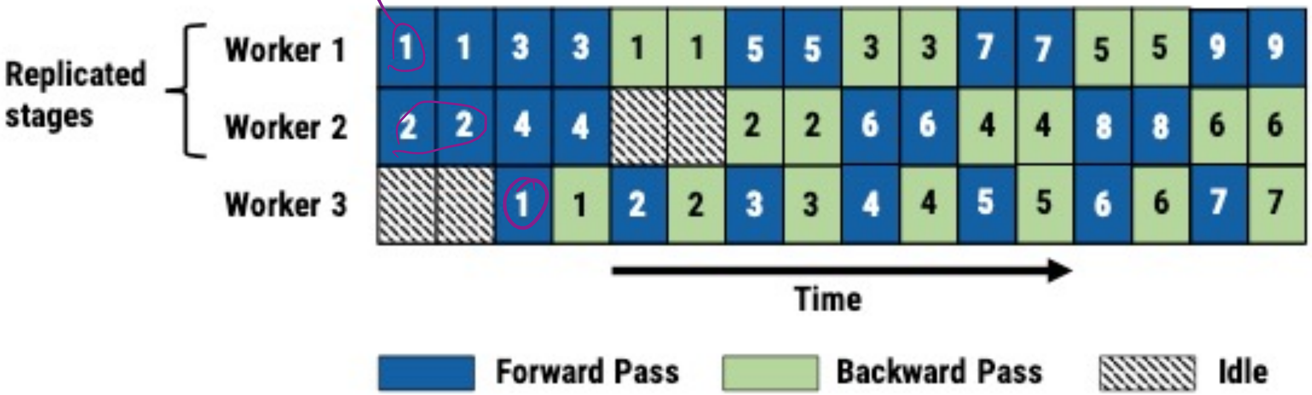
Steady state throughput is the throughput of the slowest stage

W_1 W_2 W_3 → layers are fast

Stages can be replicated!

mini batch

bs = 128



WORK PARITIONING

Given a model

Profiler: computation time for forward, backward for each layer

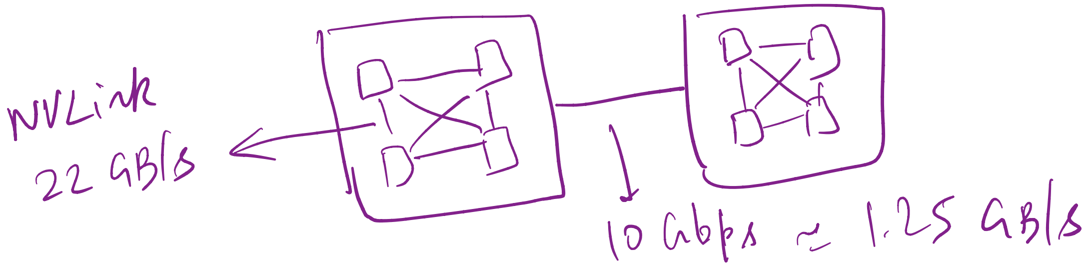
size of output activations, gradients (network transfer)

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

handling forward & backward pass for layer 1

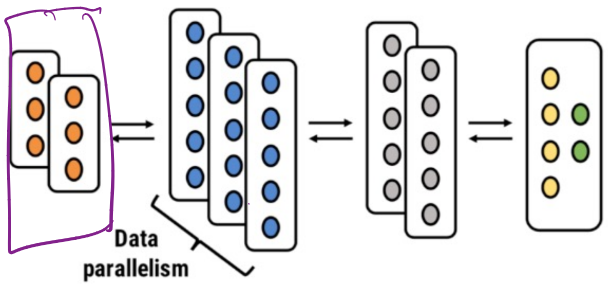
Traditional data parallel

forward iter(i)

backward iter(i)

forward iter(i+1)

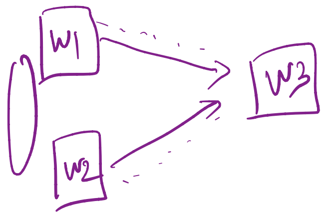
…



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

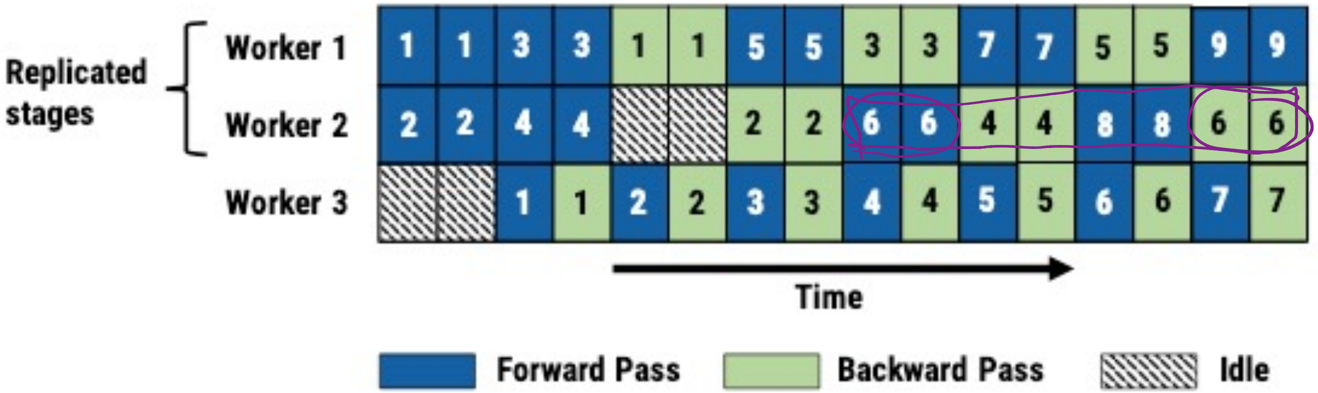
Schedule one-forward-one-backward (1F1B) → Worker 2

FW (6)
BW (4)
FW (8)
BW (6)

Round-robin for replicated stages →
same worker for fwd, backward

↳ mini batch

1 → W1
2 → W2
3 → W1
⋮

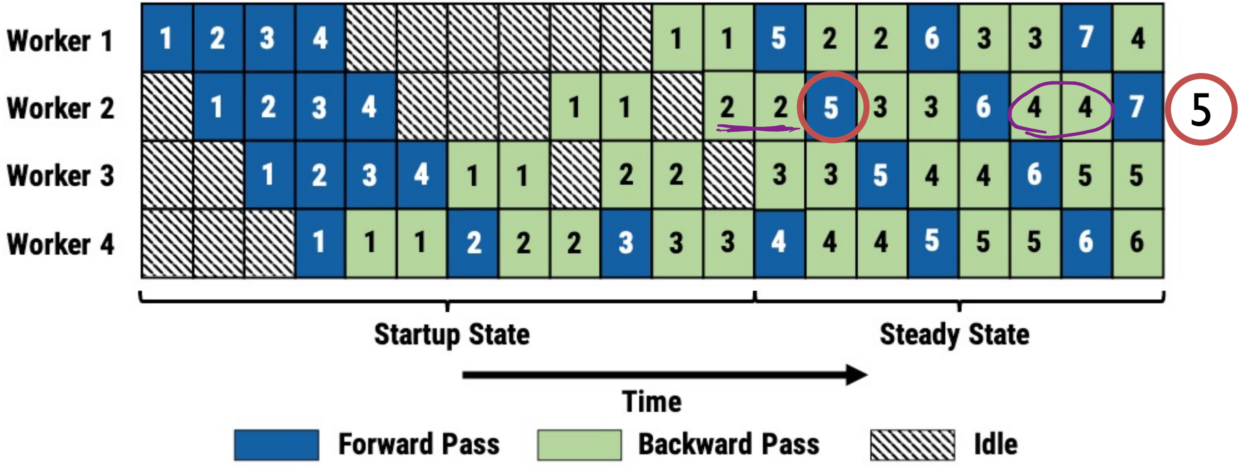


CHALLENGE 3: EFFECTIVE LEARNING

W2 : fwd pass of mini batch 5
↳ model version 2
: backward pass for mini batch 5
↳ model version 4

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 ≈ 4

Use latest version for forward pass.

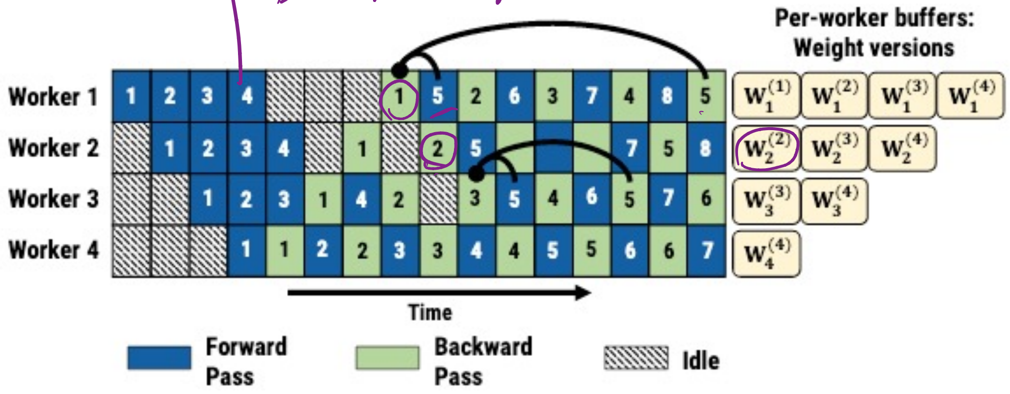
Retrieve for backward

No guarantees across stages!

W2 → save m_2, m_3, m_4 ....

Fw pass of MB 5 ⇒ use m_2

BW pass of MB 5 ⇒ retrieve m_2



STALENESS, MEMORY OVERHEAD

How to avoid staleness:

"Vertical sync" → when you first process mini batch note model version ← pass it along

$W_1, m_0 \rightarrow MB_1$

$W_1 ; m_0 \rightarrow MB\ 1$

$W_2 ; m_0 \rightarrow MB_1$

$W_4\ m_0 \rightarrow MB_1$ $W_4\ m_0 \rightarrow MB_1$

Memory overhead

Similar to data parallel?

→ if you have k mini batches in flight

⇒ same as $k \times b$ batchsize with DP

SUMMARY

Pipeline parallelism: Combine inter-batch and intra-batch

Partitioning: Replication, dynamic programming

Scheduling: 1F1B

Weight management: Stashing, vertical sync

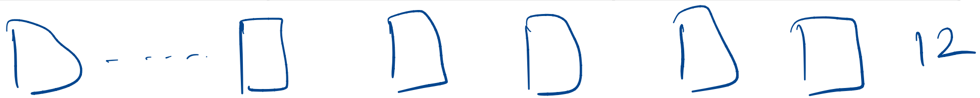
DISCUSSION

https://forms.gle/j2GCDyqCejBH8DaCA

List two takeaways from the following table

same as DataParallel

Model Name	Model Size	GPUs (#Servers x #GPUs/Server)	PipeDream Config	Speedup over DataParallel (Epoch Time)
Resnet-50	97MB	4x4 2x8	16 16 replication	1x 1x
VGG-16 Diff hardware topology → diff speedups	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

Sync between stages → can be pipelined

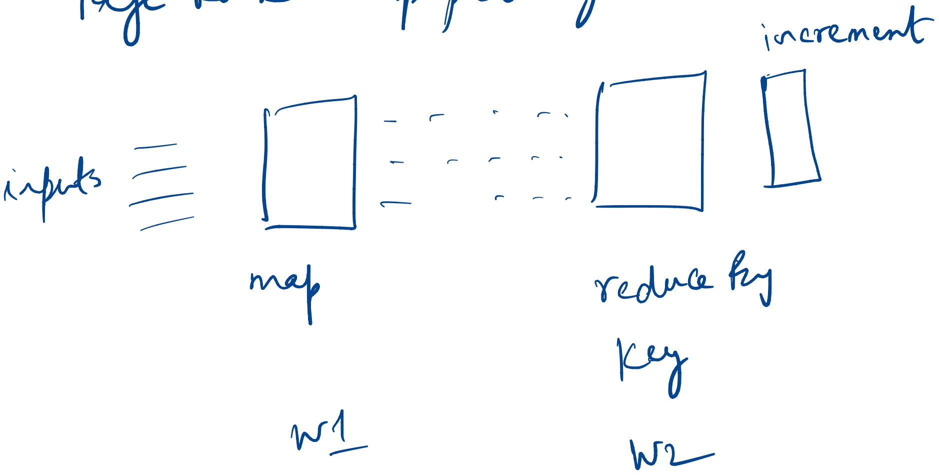
MR → if we dont have a dep. on all map tasks, can we start reducers early

PageRank* → where output (i-2) is input to iter (i)
(i-1) - - - - - (i+1)

Can we run two of these at the same time

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 pipelining within iteration



NEXT STEPS

Next class: Ray

Assignment 2 is due soon!

Course project: Oct 11 (Monday) Submit titles, groups