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

Worker 1 Worker 2 Worker 3 Worker 4

First layer

Input stage Output stage

activations

data

First mini batch

one time unit

Time

Forward Pass

Backward Pass

Idle

Worker 1
Worker 2
Worker 3
Worker 4

bₜ = 128 examples
CIFAR-10

User lasers

⊥

User lasers

⊥

takes 2 time units

① Memory overhead is lower
② Comm. per worker is only related to activation size

Low

UTILIZATION

Backward pass

Low

UTILIZATION
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 communication 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?
Steady state throughput is the throughput of the slowest stage

Stages can be replicated!

\[ bs = 128 \]

- Replicated stages
  - Worker 1
  - Worker 2
  - Worker 3

Diagram:
- Forward Pass
- Backward Pass
- Idle
WORK PARITIONING

Given a model

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

NVLink
22 GB/s

10 Gbps = 1.25 GB/s
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 $\approx \text{num\_workers} / \text{num\_replicas\_input}$

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

Round-robin for replicated stages $\rightarrow$ same worker for fwd, backward

Mini batch

$1 \rightarrow W_1$

$2 \rightarrow W_2$

$3 \rightarrow W_1$
Naïve pipelining
Different model versions forward and backward

W2: forward pass of mini-batch 5
L model version 2
: backward pass for mini-batch 5
L model version 4
Weight stashing
Maintain multiple versions of the weights
One per active mini-batch \( \approx 4 \)

Use latest version for forward pass.
Retrieve for backward
No guarantees across stages!
STALENESS, MEMORY OVERHEAD

How to avoid staleness:

Vertical sync

Memory overhead
Similar to data parallel?

if you have $k$ mini batches in flight

$\Rightarrow$ same as $k/2$ batchsize with DP
SUMMARY

Pipeline parallelism: Combine inter-batch and intra-batch
Partitioning: Replication, dynamic programming
Scheduling: IF1B
Weight management: Stashing, vertical sync
DISCUSSION

https://forms.gle/j2GCDyqCejBH8DaCA
List two takeaways from the following table:

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Model Size</th>
<th>GPUs (#Servers x #GPUs/Server)</th>
<th>PipeDream Config</th>
<th>Speedup over DataParallel (Epoch Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resnet-50</td>
<td>97MB</td>
<td>4x4 2x8</td>
<td>16 16</td>
<td>1x 1x</td>
</tr>
<tr>
<td>VGG-16</td>
<td>528MB</td>
<td>4x4 2x8</td>
<td>15-1 15-1</td>
<td>5.28x 2.98x</td>
</tr>
<tr>
<td>GNMT-8</td>
<td>1.1GB</td>
<td>3x4 2x8</td>
<td>Straight 16</td>
<td>2.95x 1x</td>
</tr>
</tbody>
</table>
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
- If we don’t 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.
Next class: Ray

Assignment 2 is due soon!

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