CS 744: PIPEDREAM

Shivaram Venkataraman
Fall 2020
- 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

Worker 1  Worker 2  Worker 3  Worker 4
Input stage  Output stage

Worker 1
Worker 2
Worker 3
Worker 4

1 1 2 2
1 1 1 1
1 1
1

Forward Pass
Backward Pass
Idle

Time
Advantages?
CHALLENGE 1: WORK PARTITIONING

Goal: Balanced stages in the pipeline. Why?

Stages can be replicated!
WORK PARTITIONING

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 \( \approx \) num_workers / num_replicas_input

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

Round-robin for replicated stages \(\rightarrow\)  
   same worker for fwd, backward
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: IFIB
Weight management: Stashing, vertical sync
DISCUSSION

https://forms.gle/GdVRuE8rBHH2vPPW6
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
NEXT STEPS

Next class: TVM
Assignment 2 is out!
Course project deadlines
  Today! (titles, groups)
  Oct 16 (introductions)