Efficient Distributed Transfer Learning Using Pipelined Model Parallelism

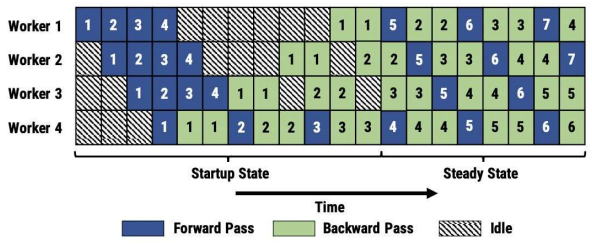
Daniel McNeela, Mohil Patel, Surabhi Gupta

Problem Statement

- Our goal is to build a system that can perform efficient distributed transfer learning.
- Transfer learning: incrementally build knowledge on top of a pretrained model - by re-training just the last few layers!
- Problem: Learn multiple new models from one! We aim to leverage pipelined model parallelism for this problem.

Related work

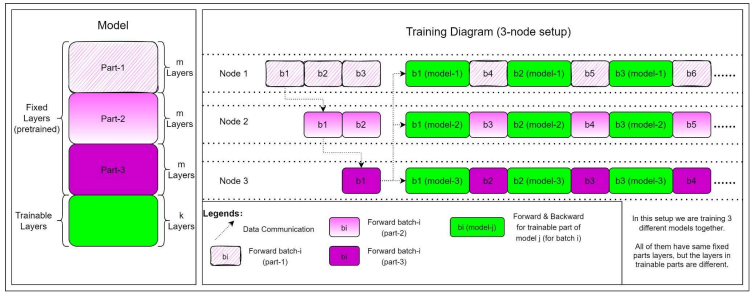
- Existing systems implement pipeline parallelism to achieve training speedups and better CPU/GPU utilization for training deep neural networks.
- Problems: pipedream suffers from staleness.
- Existing solutions to staleness need lots of extra memory to store old weights.
- Our question: How can we use this idea without having to deal with staleness.



Our Proposal

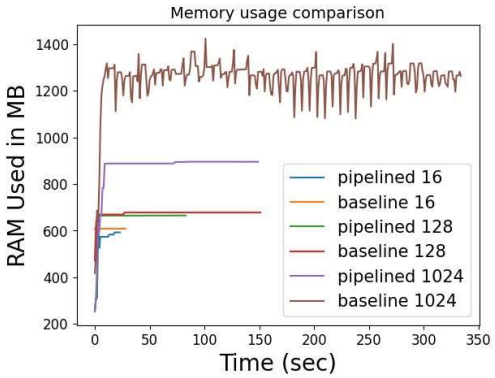
- One basic approach is to train each model on a separate machine. (our baseline)
- Our hypothesis -> Could do better if all machines 'collaborate' to compute the forward tensors of the fixed part. This avoids repeating forward computation!
- Could reduce the training time and the memory usage!
- Proposal: Use model pipelining to achieve this!

System Design



- Setup: Three models being fine-tuned in parallel. Each has the same fixed part and different trainable layers!
- Infra Design: Split the fixed layers across machines and assign trainable layers to each one separately.
- Algorithm: Perform forward passes in a pipelined fashion by communicating intermediate data across machines. The training part happens in parallel on each machine.
- Scheduling: 1-F, 1-B to ensure progress!

Early results



- Metrics: Training time, Memory, CPU, network usage.
- Pipelined transfer learning requires less than half the time and much lower memory compared to the baseline.

Current Status

- Implementation of entire pipelined setup from scratch!
- Still working on a better way to propagate labels for the training phase.
- More experimentation:
  - Study the I/O impact of our distributed setup and investigate its variation with batch sizes.
  - Comparison of CPU utilization w.r.t. baseline.
- An interesting future direction: Think about more efficient model splits for the forward pass.