Name: Srikar Chundury Date: 6th Oct, 2022

Distributed Deep Learning with Iterative Step-wise Averaging

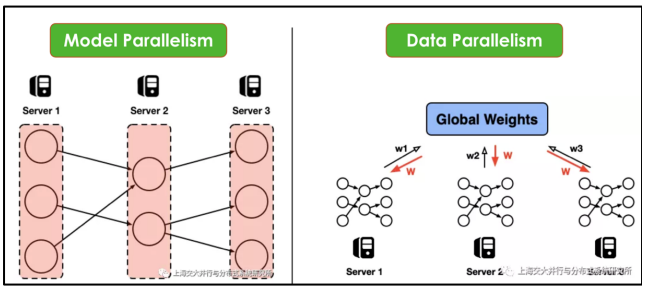
Problem Statement

A comprehensive study of present-day averaging techniques in distributed training of deep neural networks. Implement a new idea that involves step-wise averaging for plausible performance gains with respect to network bandwidth. Contrast this new idea with the study.

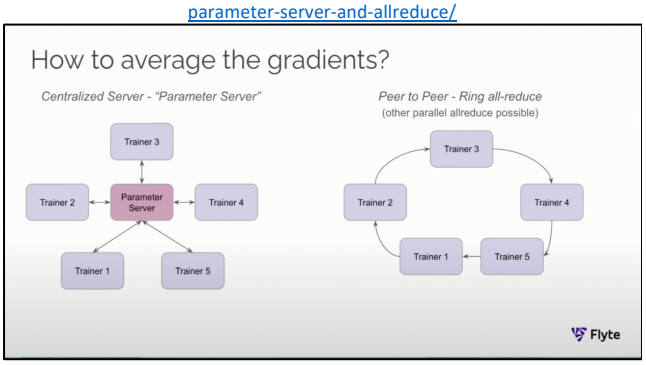
Motivation

2 ways to achieve parallelism (picture-1)-

1) **Data** Parallelism: Splitting data into multiple chunks, training the model on all nodes. Each model learns a different set of weights. Hence, the workers communicate with each other to make sure they are training a consistent model- "Synchronous training".
2) **Model** Parallelism: Splitting the network in multiple sub-networks, training the same data on each sub-network.



Picture(1) Source: http://www.juyang.co/distributed-model-training-ii-parameter-server-and-allreduce/



Picture(2) Source: https://www.youtube.com/watch?v=gF3cVTdgLUY

Data parallelism is preferred over model parallelism mostly because of the ease of implementation. But it does have one drawback- the sync step after every iteration i.e., gradient averaging step. The parameter sever approach [1,2] mitigates this problem by storing the model parameters centrally and allowing worker nodes to read/write to it. But this creates a single point of failure, the parameter server itself. It has to be highly available

A **peer-to-peer** alternative approach (picture-2) (called ring all-reduce variants [3,4,7]) mitigates the above bottleneck problem and is currently used by the industry. In this approach, the weights learned by each worker travel in a ring format **twice**: once to calculate the average and then to propagate the average. Although, it's significant improvement over its predecessors; solving the bottleneck issue, it still has a long synchronous step. This takes finite time and is directly **proportional to the number of worker nodes** in the ring, to calculate aggregated weights.

Deliverables

1. An end-to-end automated pipeline to measure and report performance metrics like CPU%, MEMORY, NETWORK I/0, DISK I/O per worker for a distributed training task of any neural network. Also reports model performance metrics like convergence time, accuracy.
2. A comparative study of the metrics collected in deliverable-1 for parameter-server, ring reduce and ring all-reduce techniques for the weight averaging step.
3. Implementing the step-wise reduce technique (start the next iteration with the average till that worker without the second pass in the ring all-reduce technique.
4. Include step-wise technique into the study in deliverable-2.

Experimental Setup

1. 3 container spark cluster using docker. Host machine: Macbook Pro (M1, 32GB).
2. Horovod[5], Pytorch for distributed deep neural network training.
3. Datasets: MNIST, ImageNet.
4. Models: Basic convolution NN for MNIST, ResNet-50 for ImageNet.
5. Will **not** use costly tools like dPRO [7]. Instead, will rely on docker stats.

References

1. [info] http://learningsys.org/papers/LearningSys_2015_paper_14.pdf (2015)
2. [paper] https://proceedings.neurips.cc/paper/2014/hash/1ff1de774005f8da13f42943881c655f-Abstract.html (NIPS, 2014)
3. [paper] https://arxiv.org/abs/2202.01158 (IEEE INFOCOM, 2022)
4. [paper] https://arxiv.org/abs/1802.05799 (Uber, 2018)
5. [paper] https://arxiv.org/abs/2209.12769 (IEEE TPDS, 2022)
6. [paper] https://arxiv.org/abs/2208.02498 (under review, 2022)
7. [paper] https://arxiv.org/abs/2205.02473 (MLSys, 2022)
8. [info] http://www.cs.fsu.edu/~xyuan/paper/09jpdc.pdf (really old paper on reduce algorithms)
9. [blog] https://www.uber.com/blog/horovod/