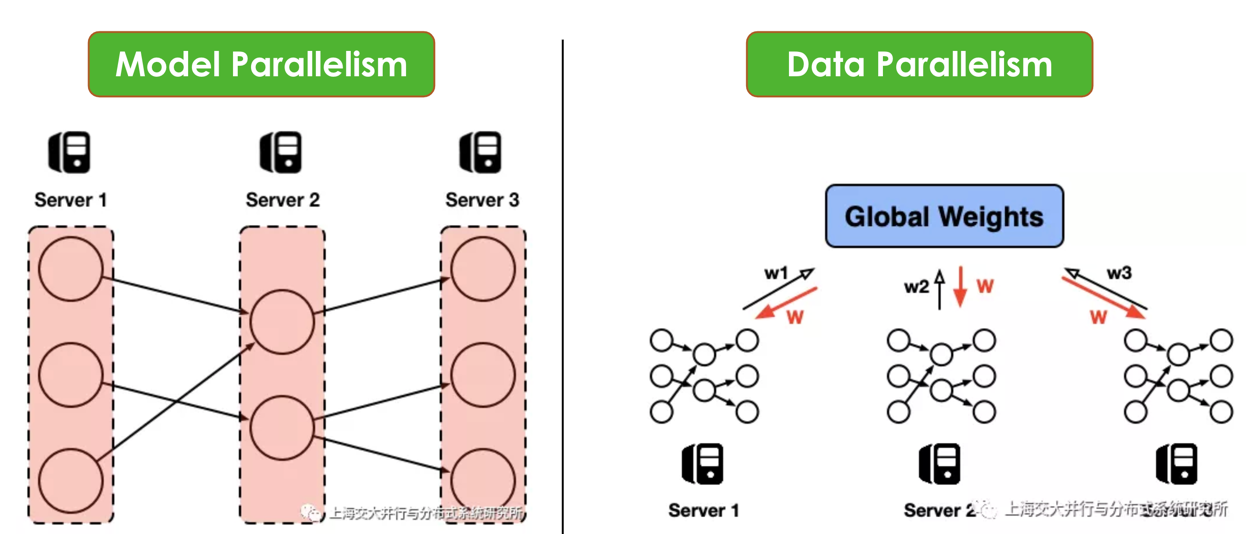
Name: Srikar Chundury Date: 16th Sept, 2022

# Abstract idea- Distributed deep learning using iterative pair-wise averaging

## About

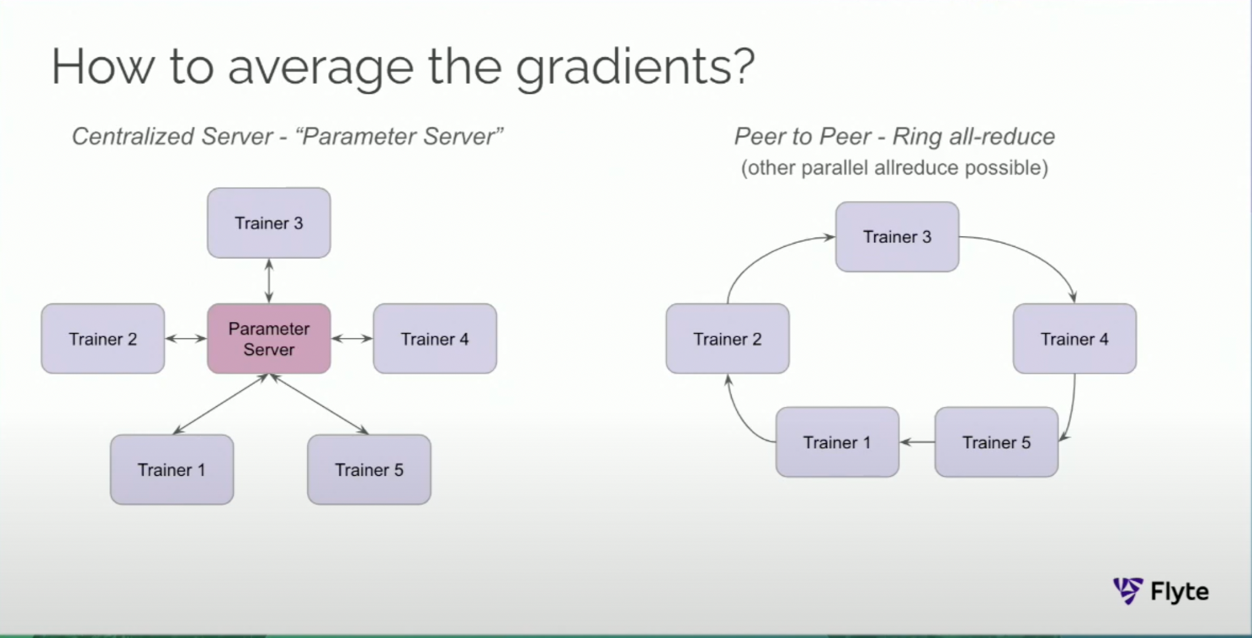
2 ways to achieve parallelism-

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.



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

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 [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 [1,2,3,4]. It has to be highly available etc.



Source: https://www.youtube.com/watch?v=gF3cVTdgLUY

A peer-to-peer alternative approach (called ring all-reduce [4,5,6,7,8]) 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.

## Proposition

I propose to study a *plausible* improvement over the above-described technique, where the weights in the iteration steps are an aggregation of its current weight and the previous node’s weight only.

Example- say workers A, B and C each produce a, b and c weights after the first iteration

Subset Data 1 Subset Data 2 Subset Data 3

Worker A Node 🡪 Worker B Node 🡪 Worker C Node

Weights: (first pass) a b c

(c+a)/2 (a+b)/2 (b+c)/2

(c+a+b+c)/4 (a+b+c+a)/4 (b+c+a+b)/4

… … … And so on…

The weights should eventually converge is my claim.

Pros: Network activity is expensive and can be reduced to pair wise communication only.

Cons: Same cons as for ring all-reduce. No new ones.

## Queries

* Is it in scope for this course? Yes
* Is a proof of concept with 3 VM spark cluster and a modification of the horvord sufficient? yes
  + Can I introduce a “sleep” between iterations to mimic network activity? yes

## References

1. <http://learningsys.org/papers/LearningSys_2015_paper_14.pdf> - Huawei’s asynchronous distributed learning
2. <https://www.cs.cmu.edu/~muli/file/ps.pdf> - parameter server related paper
3. <https://web.eecs.umich.edu/~mosharaf/Readings/Parameter-Server.pdf> - parameter server related paper
4. <https://www.uber.com/blog/horovod/> - Uber’s distributed deep learning
5. <https://arxiv.org/abs/1802.05799> - Uber’s paper
6. <http://www.cs.fsu.edu/~xyuan/paper/09jpdc.pdf> - Peer-to-peer model for distributed deep learning
7. <http://research.baidu.com/bringing-hpc-techniques-deep-learning/> - Baidu’s ring all-reduce on GPUs
8. <https://docs.flyte.org/projects/cookbook/en/stable/auto/case_studies/ml_training/spark_horovod/keras_spark_rossmann_estimator.html> - Flyte; Spark 3.0 + horovod

## Feedback points

* Try step wise averages (average till that node) and pair wise averages (averaging every 2 nodes).
* Report a comprehensive case-study showing merits and de-merits.

## Changes

* Pair-wise reduce algorithms have been tried out by people and claim only degradations as per this medium article: <https://roman-kazinnik.medium.com/machine-learning-distributed-ring-reduce-vs-all-reduce-cb8e97ade42e>
* Hence, the step-wise averaging approach seems like the most novel approach. That is, getting average at each worker and triggering the next iteration.