Pipe-SGD: A Decentralized Pipelined SGD Framework for Distributed Deep Net Training
Youjie Li1, Mingchao Yu2, Songze Li2, Salman Avestimehr2, Nam Sung Kim1, Alexander Schwing1

Introduction

Motivation:
- Distributed training of deep nets is essential to address ever-increasing computational demands and memory consumption
- Main objective is to minimize the end-to-end training time

Prior Work:
- Centralized parameter-server framework, especially with asynchronous training (Fig. 1)
  - Pros: low synchronization overhead + interleaved training iterations
  - Cons: uncontrollable staleness of gradient + congested communication
- Decentralized framework with synchronous training (Fig. 2)
  - Cons: no stale gradient + balanced communication via AllReduce
  - Cons: high synchronization overhead + sequential execution of local iterations

Our Approach:
- To combine the best of both, our approach Pipe-SGD: decentralized pipelined training (Fig. 3)
  - Balances communication via AllReduce → low communication time
  - Pipelines local iterations to hide compute/communicate time → low execution time
  - Controls staleness of gradients → good convergence

Approach (Continued)

Finding the Optimal Staleness under Resource Constraints (Fig. 5–6):
- Assume ideal conditions where resources are unlimited and constant T
- Total train time: $T_{total} \geq T(l + k + 1) + (n + l + \frac{1}{p} + T_{allreduce} + T_{comput})$
- However, in reality, both communication and computation resources are strictly limited, thus:
  $T_{total} \geq T(l + k + 1) + (n + l + \frac{1}{p} + T_{allreduce} + T_{comput})$
where the total time is solely determined by compute or communication resources, regardless of $k$.
- Also, since gradient staleness is always $k$ iterations, increasing $k$ only harms. Thus the optimal value: $k = 1$.

Finding the Optimal Level of Pipelining (Fig. 7–8):
- The bandwidth-optional Ring-AllReduce is adopted for gradient communication (Fig. 8)
- When level of pipeline is 1 (i.e., all computation within each iteration) (Fig. 7), total time:
  $T_{total} = T(l + k + 1) + (n + l + \frac{1}{p} + T_{allreduce} + T_{comput})$
- If we increase the level to 2 (i.e., further pipelining gradient communication) (Fig. 7), then:
  $T_{total} = T(l + k + 1) + (n + \frac{1}{p} + T_{compute})$
- From $T_{allreduce}$ and $T_{compute}$, we note that increasing the level results in longer communication time, which worsens the already communication-dominant system in practice. Thus the optimal value: $level of pipeline = 1$.

Finding the Optimal Communication Ratio:
- Assume that: a) given a cluster we use the same batch size on each worker as in the single-node training
  - b) the single node and PixelNet train the same epochs on the dataset. Thus, the “scaling efficiency” $SE$ of Pipe-SGD is:
    $$SE = \frac{\text{Actual Speedup}}{\text{Ideal Speedup}} = \frac{\text{Batch Size}}{p}$$
- Hence, once system is compute bound, linear speedup can be achieved as the cluster size scales, i.e. $SE=1$, which means the optimal value: communication ratio $\approx 50\%$.

To sum up, Pipe-SGD is optimal for: $k = 1$, level of pipeline $= 1$, system is compute bound (after compression).

Compression
- For the desired communication ratio, simple and light-weight compression schemes are used in Pipe-SGD such as floating-point truncation on 16-bit LSBS, and scalar 8-bit quantization.
- We find that complex compression algorithms aren’t suitable for the bandwidth-optimal Ring-AllReduce:
  - Compression itself is compute-heavy
  - Compression is invoked repeatedly, complexity: $O(p)$