EXPOSITION ON THE CASE STUDY FOR DECENTRALIZED PARALLEL STOCHASTIC GRADIENT DESCENT

Po Hao Chen
Boston University
Boston, MA
bupochen@bu.edu

ABSTRACT

In this self-contained exposition, we explore the analysis on Parallel Stochastic Gradient Descent by Lian et al. [LZZ+17] We summarize contributions from their work and examine its improvement over prior results.

1 Introduction

In recent years, massively parallel computations are becoming increasingly critical to machine learning analysis as dataset grows larger. Mainstream machine learning frameworks, including Tensorflow, Torch and CNTK, which adopt supports for distributed computations are built in a centralized fashion. Traditionally, a master node aggregates the independent computations across a network of clusters. However, it incurs communication overheads and creates latency in our compute stack.

In the past few decades, stochastic optimization problems have become essential in modeling machine learning problems. They involve randomness when maximizing or minimizing objective function subjected to constraints.

$$\min_{x \in R^n} f(x) := \mathop{\mathbb{E}}_{\xi \sim D} [F(x; \xi)] \tag{1}$$

Equation (1) formulates a stochastic optimization problem that summarizes deep learning model, linear regression, and logistic regressions. Formally, let R^n be the feasible set that we search to find a $x \in R^n$ that approximately minimizes cost function F. Denote ξ as a random variable drawn from an arbitrary distribution D. However, observe that ξ is only available after x is chosen; we cannot minimize cost function F directly, we instead minimizes its expectation $\mathbb{E}_{\xi \sim D} [F(x; \xi)]$.

We consider the problem in the context of machine learning problems in which we draw a data sample ξ from a large-scale dataset. Parallel stochastic gradient descent (PSGD) is a prominent algorithm used for the task. The existing implementations are based on centralized cluster topology where a master node updates the model by aggregating the stochastic gradients computed by all other nodes.

However, a bottleneck occurs for centralized topology when all the other nodes attempt to communicate with the master node concurrently. This causes the performance to suffer with lower network bandwidth. Motivated by this idea, Lian et al. studied algorithms for *decentralized* topology where no master node exists and only communication with neighboring node is allowed.

[LZZ+17] answers the following: ***Can dectranlized algorithms be faster than its centralized counterpart?***

They provided not only the theoretical analysis, but also empirical evidences, to support a positive answer.

1.1 Contribution

Prior to [LZZ+17], decentralized algorithms were studied by [SRNV09] as consensus optimization and [YVQ10] showed an autonomous distributed online learning algorithm with privacy-preserving properties. However, it remains an open question whether **decentralized methods could have advantages over centralized algorithms** in scenarios where only the decentralized network is available.

Consider a High-Performance Computing (HPC) datacenter, *should the application centralize communication across all nodes*? The existed theory [[SN11], [BFH12] [SRNV09]] either implicitly suggest the computational and communication complexity is better for centralized methods or did not make such analysis.

[LZZ+17] indicates a positive result for decentralized algorithm by showing the decentralized PSGD (D-PSGD) admits similar computational complexity to centralized PSGD (C-PSGD) and requires much less communication on the busiest node. In particular, they demonstrated:

- Identification of the cases in which decentralized algorithms are faster than its centralized counterpart.
- Scalability of D-PSGD in terms of number of nodes. With more nodes available, decentralized algorithm speeds up asymptotically linearly with respect to computational complexity. This is the first speedup result related to decentralized algorithms.
- Validation of theoretical analysis of D-PSGD and different C-PSGD variants through empirical study across multiple frameworks (CNTK and Torch). On network with low-bandwidth and high latency, D-PSGD outperforms C-PSGD x10 faster.

Their findings can be summarized with the following table. We denote n as the number of nodes and the computational complexity is the number of stochastic gradient evaluations it takes to converge to a ϵ-approximation solution.

Algorithm	Comm. Complexity	Comp. Complexity
C-PSGD	$O(n)$	$O(\frac{n}{\epsilon}+\frac{1}{\epsilon^2})$
D-PSGD	O (deg(network))	$O(\frac{n}{\epsilon}+\frac{1}{\epsilon^2})$

2 Related Work

In this section we look at related iterative algorithms for stochastic optimization problem to build up an understanding for our analysis. We denote K and n as the number of iterations and nodes, respectively.

2.1 Stochastic Gradient Descent (SGD)

In practice, SGD is much faster than its deterministic counterpart Gradient Descent (GD). It requires low $O(1)$ memory overheads for large dataset and is very fast in finding minimum in various settings. The convergence rate is known to be $O(\frac{1}{\sqrt{K}})$ in the convex case shown by [MB11] and $O(\frac{1}{K})$ for strongly convex case by [NJLS09]. In non-convex problems, [GL13] proved that SGD is expected to converge in $O(\frac{1}{\sqrt{K}})$.

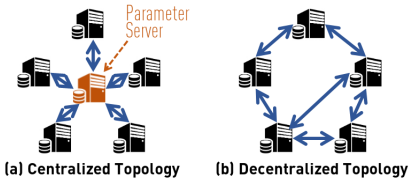


Figure 1: Server Topology

2.2 Centralized parallel SGD (C-PSGD)

C-PSGD algorithms are implemented based on the parameter server topology seen in Figure 1. In this framework, a master node distribute mini-batches of data and the workload to n worker nodes. The *central* parameter server updates the globally shared parameters from the n stochastic gradients evaluated in each iterations. [AD11] showed the method admits a convergence rate of $O(\frac{1}{\sqrt{Kn}})$ and implies a linear speedup to its serial analogue.

Since the model parameters has to be updated in order of the gradient computed, we cannot guarantee the calculation is always accurate in an asynchronous setting. If the master receives a calculation τ-step delay away from some worker A and passes an updated global parameter back to A, the other workers may not see this information and compute on *stale* parameters. [AD11] proved that as long as the staleness is bounded, asynchronous C-PSGD gurrantees linear speedup on convex, strongly convex, and non-convex and all other objectives.

2.3 Decentralized parallel stochastic algorithms

Decentralized algorithms do not rely on a *central* master node to orchestrate the information passing the network. Each node maintains its own model and only communicates with its neighbors as shown in Figure 1.

[LLZ17] showed a decentralized stochastic optimization algorithm for general convex and strongly convex objectives with computational complexity $O(n/\epsilon^2)$ and $O(n/\epsilon)$, respectively. In asychronous setting, [SY16] demonstrated an algorithm with $O(n/\epsilon^2)$ complexity for convex objectives. Further, [RNV08] proposed an algorithm for problems subjected to a convex constrained set.

Similar algorithms to D-PSGD proposed by Lian et al. has been studied in [SRNV09], [RNV08] and [SN11], however, their algorithms does not allow the nodes to communicate and compute simultaneously. In addition, the analysis in the previous works also require the gradient to be bounded by a constant. HogWild++ [ZHA16] implemented decentralized model parameters for PSGD on multi-socket system and empirically demonstrated superior performance to some centralized algorithm. However, neither the convergence nor the rate is clear.

The aforementioned work has a common issue, that is, that we cannot understand the scalability and efficiency of the decentralized algorithm with only a single node.

2.4 Other decentralized algorithm

Decentralized algorithms are often studied as method for solving consensus problems in control, privacy and networking communities. Although studies have been done in various settings, they do not demonstrate a clear advantage over their centralized counterparts.

3 Decentralized parallel stochastic gradient descent (D-PSGD)

We begin the analysis on D-PSGD in this section. Consider a cluster with n compute nodes, we can represent the decentralized communication topology as an undirected graph $G(V, W)$. Let V denote the set of nodes $\{1, 2, ..., n\}$, $W \in R^{nxn}$ denote a symmetric doubly stochastic matrix where $W_{ij} = W_{ji}$ indicates there is a connection between node i and j. $W_{ij} \in [0, 1], \forall i, j$ can be interpreted as the influence node j has on node i, and the sum of each row equals 1. If $W_{ij} = 0$ then node i and j are disconnected.

In order to distribute our workloads to each node $i \subseteq [n]$, we rewrite the objective in Equation (1) as the following:

$$\min_{x \in R^n} f(x) := \frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{\xi \sim D_i} [F_i(x; \xi)]$$

Recall that F the cost function and ξ is a random variable drawn from the distribution D of the entire dataset. We assume all nodes have access to a shared dataset such that $D_i = D$ for all i. Alternatively, we may partition the dataset for n local storage on the nodes, and approximately define a distribution for sampling local data. This can be achieved by defining D_i to have the same distribution as D but over the local data. Both of the approaches are sufficient for $F_i = F, \forall i$

Now, we define the synchronous D-PSGD algorithm where each node i executes the program concurrently. Note our notation follows the format of $x_{k,i}$ where the subscript indicate iterate k followed by index of node.

Algorithm 1 Decentralized parllel stochastic gradient descent (D-PSGD) on i-th node

Require: $x_0 \in R^n$, learning rate γ, stochastic matrix W, and number of iterations K

1: **Initialization:** $x_{0,i} = x_0$
2: **for** $k = 0, 1, ..., K - 1$ **do**
3: Randomly sample $\xi_{k,i}$ from local data.
4: Compute local stochastic gradient across all nodes $\nabla F_i(x_{k,i}; \xi_{k,i}), \forall i \subseteq [n]$
5: Aggregate from the neighborhood to compute weighted average. $x_{k+\frac{1}{2},i} = \sum_{j=1}^{n} W_{ij} x_{k,j}$
6: Set $x_{k+1,i} = x_{k+\frac{1}{2},i} - \gamma \nabla F_i(x_{k,i}; \xi_{k,i})$
7: **end for**
8: **return** $\frac{1}{n} \sum_{i=1}^{n} x_{K,i}$

To summarize each k iteration for node i, we compute the stochastic gradient $\nabla F_i(x_{k,i}; \xi_{k,i})$ with local variable $x_{k,i}$ in step 4. Assuming synchronization is satisfied, the node exchange its own $x_{k,i}$ with the neighbors. We average $x_{k,i}$ with

the received variables and store it to an intermediate variable $x_{k+\frac{1}{2},i}$. Finally, in step 6, we update the local variable $x_{k+1,i}$ with the average in $x_{k+\frac{1}{2},i}$ and local stochastic gradient $\nabla F_i(x_{k,i};\xi_{k,i})$.

Note that alternative implementation is allowed and our theoretical analysis in the next section still holds. See Appendix A for details.

We may also define the algorithm from a global point of view by concatenating the variables, random samples, stochastic gradients into matrices.

Algorithm 2 D-PSGD on i-th node (global view)

Require: learning rate γ, stochastic matrix W, number of iterations K, tolerance parameter ϵ
  Initialize $X_0 = [x_{0,1}, x_{0,2}, ..., x_{0,n}]$
  for k=0,1,...,K-1 **do**
    $X_k := [x_{k,1}, x_{k,2}, ..., x_{k,n}] \in R^{Nxn}$
    $\xi_k := [\xi_{k,1}, \xi_{k,2}, ..., \xi_{k,n}]^T \in R^n$ where $\xi_{k,i} \sim D_i$
    $\partial F(x_k, \xi_k) := [\nabla F_1(x_{k,1};\xi_{k,1}), \nabla F_2(x_{k,2};\xi_{k,2}), ...\nabla F_n(x_{k,n};\xi_{k,n})] \in R^{Nxn}$
    $X_{k+1} \leftarrow X_k W - \gamma \partial F(X_k;\xi_k)$
  end for

  if $K^{-1}(\sum_{k=0}^{K-1} \mathbb{E}\left[||\nabla f(\frac{X_k 1_n}{n})||^2\right] \leq \epsilon$ **then**
    return X_{k+1} is an ϵ-approximation
  else
    return X_{k+1} is not an ϵ-approximation

4 Convergence Rate Analysis

We can now discuss the convergence rate of D-PSGD. The result of the section shows that D-PSGD has the same computational complexity as C-PSGD, however the communication complexity can be better. We define:

$$\partial f(X_k) := [\nabla f_1(x_{k,1}) \nabla f_2(x_{k,2}) \dots \nabla f_n(x_{k,n})] \in R^{Nxn}$$

where $f_i(x) = \mathbb{E}_{\xi \sim D_i}[F_i(x;\xi)]$ and $X_k := [x_{k,1}, ..., x_{k,n}]$
We make a few common assumption:

1. f_i is L-Lipschitz such that there exists $L > 0$ and $||\nabla f_i(x) - \nabla f_i(y)|| \leq L||x - y||, \forall x, y \in R^n$
2. spectral gap of W, $\rho := (\max\{|\lambda_2(W)|, |\lambda_n(W)|\})^2 < 1$
3. bounded variance: can we get the "bias" to decay to small value with more iteration? (see below)
4. Start from 0: $X_0 = 0$ simplies the proof W.L.O.G

The spectral gap is a quantity that tells us how "connected" the graph is. Imagine a piece of information being passed on the cluster where it transitions from node a to b with probability $W_{a,b}$. We are interested in the expected behaviour after iteration k, i.e the distribution of the stochastic matrix at iterate k. It is known that W^k converges to a stationary distribution (equal probability towards all nodes) as $k \to \infty$. A small value of ρ essentially tells us the time for W^k to converge is fast. With this idea, we can formally state the following:

Lemma 4.1 *Spectral Gap Bound*

$$||\frac{1_n}{n} - W^k e_i|| \leq \rho^k$$

Proof:

$$||\frac{1_n}{n} - W^k e_i|| = ||(W^\infty - W^k)e_i||^2 \leq ||W^\infty - W^k||^2 ||e_i||^2 \leq \rho^2$$

. We use the diagonalization argument to derive the final bound ρ^k.

Assume the stochastic gradient $\mathbb{E}_{i \sim \mathcal{U}([n])}\left[\mathbb{E}_{\xi \sim D_i}\left[||\nabla F_i(x;\xi) - \nabla f(x)||^2\right]\right] \leq \sigma^2$ is bounded. It implies two of the following. There exists constant σ, ς

- $\mathbb{E}_{\xi\sim D_i}\left[||\nabla F_i(x;\xi)-\nabla\nabla f_i(x)||^2\right]\leq\sigma^2,\forall i,\forall x$
- $\mathbb{E}_{i\sim\mathcal{U}([n])}\left[||f_i(x)-\nabla f(x)||^2\right]\leq\varsigma^2,\forall x$

The intuition here is that σ^2 is bounding the variance of the distribution on each node partition and ς^2 is bounding the variance of the final output distribution. And if we have access to a shared remote database, $\varsigma=0$. With stochastic gradient, if the variance is large it will be hard for us to decay to zero with constant learning rate, so we decay the learning rate instead. This means taking smaller steps as the number of iterations increase, which gives us bad convergence rate. Thus, it is more preferable to have small biases in our distribution.

Theorem 4.2 (Convergence Theorem for D-PSGD)

$$C_1:=(\frac{1}{2}-\frac{9\gamma^2L^2n}{(1-\sqrt{\rho})^2C_2})\qquad C_2:=(1-\frac{18\gamma^2}{(1-\sqrt{\rho})^2}nL^2)$$

$$\frac{1}{K}(\frac{1-\gamma L}{2}\sum_{k=0}^{K-1}\mathbb{E}\left[||\frac{\partial f(X_k)1_n}{n}||^2\right]+C_1\sum_{k=0}^{K-1}\mathbb{E}\left[||\nabla f(\frac{X_k1_n}{n})||^2\right])\leq\frac{f(0)-f^*}{\gamma K}+\frac{\gamma L}{2n}\sigma^2+\frac{\gamma^2L^2n\sigma^2}{(1-\rho)C_2}+\frac{9\gamma^2L^2n\varsigma^2}{(1-\sqrt{\rho})^2C_2}$$

Theorem 4.2 characterizes the convergence of the average of all local variable $X_{k,i}$. The full proof for the theorem can be found in the supplement materials in [LZZ+17]. I will highlight the important pieces here.

Lemma 4.3

$$\mathbb{E}\left[||\partial f(X_j)||^2\right]\leq\sum_{h=1}^{n}3\,\mathbb{E}\left[L^2||\frac{\sum_{i'=1}^{n}x_{j,i'}}{n}-x_{j,h}||^2\right]+3n\varsigma^2+3\,\mathbb{E}\left[||\nabla f(\frac{X_j1_n}{n})1_n^T||^2\right]$$

Proof: By triangle inequality we can break LHS into 3 terms.

$$\mathbb{E}\left[||\partial f(X_j)||^2\right]\leq 3\,\mathbb{E}\left[||\partial f(X_j)-\partial f(\frac{X_j1_n}{n}1_n^T)||^2\right]+3\,\mathbb{E}\left[\partial f(\frac{X_j1_n}{n}1_n^T)-\nabla f(\frac{X_j1_n}{n})1_n^T||^2\right]+3\,\mathbb{E}\left[||\nabla f(\frac{X_j1_n}{n})1_n^T||^2\right]$$

Notice the second term resembles the bounded variance assumption, we can rewrite it as:

$$\leq 3\,\mathbb{E}\left[||\partial f(X_j)-\partial f(\frac{X_j1_n}{n}1_n^T)||_F^2\right]+3n\varsigma^2+3\,\mathbb{E}\left[||\nabla f(\frac{X_j1_n}{n})1_n^T||^2\right]$$

Apply L-Lipschtiz assumption:

$$\leq\sum_{h=1}^{n}3\,\mathbb{E}\left[L^2||\frac{\sum_{i'=1}^{n}x_{j,i'}}{n}-x_{j,h}||^2\right]+3n\varsigma^2+3\,\mathbb{E}\left[||\nabla f(\frac{X_j1_n}{n})1_n^T||^2\right]$$

Proof of the Theorem 4.2

Let $\frac{X_{k+1}1_n}{n}=\frac{1}{n}\sum_{i=1}^{n}x_{K,i}$ which is the final output of our algorithm. We first rewrite it as the last step of the D-PSGD algorithm. We can cancel out W because $\rho<1$.

$$\mathbb{E}\left[f(\frac{X_{k+1}1_n}{n})\right]=\mathbb{E}\left[f(\frac{X_kW1_n}{n}-\gamma\frac{\partial F(X_k;\xi_k)1_n}{n})\right]=\ \mathbb{E}\left[f(\frac{X_k1_n}{n})-\gamma\frac{\partial F(X_k;\xi_k)1_n}{n})\right]$$

By linearity we pull out the terms, and apply descent lemma on the second erm.

$$\mathbb{E}\left[f(\frac{X_k1_n}{n})\right]-\gamma\,\mathbb{E}\left[\langle\nabla f(\frac{X_k1_n}{n}),\frac{\partial f(X_k)1_n}{n}\rangle\right]+\frac{\gamma^2L}{2}\,\mathbb{E}\left[||\frac{\sum_{i=1}^{n}\nabla F_i(x_{k,i};\xi_{k,i})}{n}||^2\right]$$

Splitting the last term into two parts. Observe that we can apply bounded variance on the term.

$$\mathbb{E}\left[f(\frac{X_{k+1}1_n}{n})\right]=\mathbb{E}\left[f(\frac{X_k1_n}{n})\right]-\gamma\,\mathbb{E}\left[\langle\nabla f(\frac{X_k1_n}{n}),\frac{\partial f(X_k)1_n}{n}\rangle\right]+$$
$$\underbrace{\frac{\gamma^2L}{2}\,\mathbb{E}\left[||\frac{\sum_{i=1}^{n}\nabla F_i(x_{k,i};\xi_{k,i})-\sum_{i=1}^{n}\nabla f_i(x_{k,i})}{n}||^2\right]}_{\leq\frac{\gamma^2L}{2n}\sigma^2}+\frac{\gamma^2L}{2}\,\mathbb{E}\left[||\frac{\sum_{i=1}^{n}\nabla f_i(x_{k,i})}{n}||^2\right]$$

By the fact that $2\langle a,b\rangle = ||a||^2 + ||b||^2 - ||a-b||^2$. Let $a = \nabla f(\frac{X_k 1_n}{n})$, $b = \frac{\partial f(X_k)1_n}{n}$

$$\leq \mathbb{E}\left[f(\frac{X_k 1_n}{n})\right] - \frac{\gamma - \gamma^2 L}{2}\mathbb{E}\left[||\frac{\partial f(X_k)1_n}{n}||^2\right] - \frac{\gamma}{2}\mathbb{E}\left[||\nabla f(\frac{X_k 1_n}{n})||^2\right] + \frac{\gamma^2 L}{2}\frac{\sigma^2}{n} + \frac{\gamma}{2}\underbrace{\mathbb{E}\left[||\nabla f(\frac{X_k 1_n}{n}) - \frac{\partial f(X_k)1_n}{n}||^2\right]}_{:=T_1}$$

Refer to the supplement materials on the details proof on the claims.

$$T_1 = \mathbb{E}\left[||\nabla f(\frac{X_k 1_n}{n}) - \frac{\partial f(X_k)1_n}{n}||^2\right] \leq \frac{L^2}{n}\sum_{i=1}^{n}\underbrace{\mathbb{E}\left[||\frac{\sum_{i'=1}^{n} x_{k,i'}}{n} - x_{k,i}||\right]}_{Q_{k,i}}$$

We define $Q_{k,i}$ as the squared distance of the local optimization variable on the i-th node from average local optimization variables. We claim:

$$Q_{k,i} \leq 2\gamma^2\underbrace{\mathbb{E}\left[||\sum_{j=0}^{k-1}(\partial F(X_j;\xi_j) - \partial f(X_j))(\frac{1_n}{n} - W^{k-j-1}e_i)^2||^2\right]}_{:=T_2} + 2\gamma^2\underbrace{\mathbb{E}\left[||\sum_{j=0}^{k-1}\partial f(X_j)(\frac{1_n}{n} - W^{k-j-1}e_i)||^2\right]}_{:=T_3}$$

$$T_2 = \mathbb{E}\left[||\sum_{j=0}^{k-1}(\partial F(X_j;\xi_j) - \partial f(X_j))(\frac{1_n}{n} - W^{k-j-1}e_i)||^2\right]$$

Use Cauchy-Schwartz and isolate the terms for the variance and the mixing time. Apply Lemma 4.1 and bounded variance. Notice $\sum_{j=0}^{k=1}\rho^{k-j-1}$ converges to $\frac{1}{1-\rho}$

$$\leq \sum_{j=0}^{k-1}\mathbb{E}\left[||\partial F(X_j;\xi_j) - \partial f(X_j)||_F^2\right]||\frac{1_n}{n} - W^{k-j-1}e_i||^2 \leq n\sigma^2\sum_{j=0}^{k-1}\rho^{k-j-1} \leq \frac{n\sigma^2}{1-\rho}$$

$$T_3 = \mathbb{E}\left[||\sum_{j=0}^{k-1}\partial f(X_j)(\frac{1_n}{n} - W^{k-j-1}e_i)||^2\right]$$

$$= \underbrace{\sum_{j=0}^{k-1}\mathbb{E}\left[||\partial f(X_j)(\frac{1_n}{n} - W^{k-j-1}e_i)||^2\right]}_{:=T_4} + \underbrace{\sum_{j\neq j'}\mathbb{E}\left[\langle\partial f(X_j)(\frac{1n}{n} - W^{k-j-1}e_i), \partial f(X_j')(\frac{1n}{n} - W^{k-j'-1}e_i)\rangle\right]}_{:=T_5}$$

$$T_4 = \sum_{j=0}^{k-1}\mathbb{E}\left[||\partial f(X_j)(\frac{1n}{n} - W^{k-j-1}e_i)||^2\right] \leq \sum_{j=0}^{k-1}\mathbb{E}\left[||\partial f(X_j)||^2\right]||\frac{1_n}{n} - W^{k-j}e_i||^2$$

Apply the Spectral gap bound and Inequality Lemma 4.3

$$\leq 3\sum_{j=0}^{k-1}\sum_{j=1}^{n}\mathbb{E}\left[L^2 Q_{j,h}||\frac{1_n}{n} - W^{k-j-1}e_i||^2\right] + 3n\varsigma^2\frac{1}{1-\rho} + 3\sum_{j=0}^{k-1}\mathbb{E}\left[||\nabla f(\frac{X_j 1_n}{n})1_n^T||^2\right]||\frac{1_n}{n} - W^{k-j-1}e_i||^2$$

$$T_5 = \sum_{j\neq j'}^{k-1}\mathbb{E}\left[\langle\partial f(X_j)(\frac{1n}{n} - W^{k-j-1}e_i), \partial f(X_j')(\frac{1n}{n} - W^{k-j'-1}e_i)\rangle\right]$$

$$\leq \sum_{j\neq j'}^{k-1}\underbrace{\mathbb{E}\left[||\partial f(X_j)(\frac{1_n}{n} - W^{k-j-1}e_i)||\right]}_{\leq\mathbb{E}[||\partial f(X_j)||]\cdot||\frac{1_n}{n} - W^{k-j-1}e_i||}\underbrace{||\partial f(x_j')(\frac{1_n}{n} - w^{k-j'-1}e_i)||}_{\leq||\partial f(x_j')||\cdot||(\frac{1_n}{n} - w^{k-j'-1}e_i)||}$$

$$\leq \sum_{j\neq j'}^{k-1} \mathbb{E}\left[\frac{||\partial f(X_j)||^2}{2} + \frac{||\partial f(X_{j'})||^2}{2}\right] \rho^{k-\frac{j-j'}{2}-1}$$

$$\underbrace{\leq 3\sum_{j\neq j'}^{k-1}(\sum_{h=1}^{n} \mathbb{E}\left[L^2 Q_{j,h}\right] + \mathbb{E}\left[||\nabla f(\frac{X_j 1_n}{n} 1_n^T)||^2\right])\rho^{k-\frac{j+j'}{2}-1}}_{=:T_6} + \underbrace{\sum_{j\neq j'}^{k-1} 3n\varsigma^2 \rho^{k-1-\frac{j+j'}{2}}}_{=:T_7}$$

Plugging T_6, T_7 into T_5, and T_5, T_4 into T_3. Then, T_2, T_3 into $Q_{k,i}$ bound to obtain the following.

$$Q_{k,i} \leq \frac{2\gamma^2 n\sigma^2}{1-\rho} + \frac{18\gamma^2 n\varsigma^2}{(1-\sqrt{\rho})^2} + 6\gamma^2 \sum_{j=0}^{k-1} \mathbb{E}\left[||\nabla f(\frac{X_j 1_n}{n}) 1_n^Y||^2\right](\rho^{k-j-1} + \frac{2\sqrt{\rho^{k-j-1}}}{1-\sqrt{\rho}})$$

$$+6\gamma^2 \sum_{j=0}^{k-1}\sum_{h=1}^{n} \mathbb{E}\left[L^2 Q_{j,h}(\frac{2\sqrt{\rho^{k-j-1}}}{1-\sqrt{\rho}} + \rho^{k-j-1})\right]$$

Define M_k as average of $Q_{k,i}$ on all nodes := $\frac{\mathbb{E}[\sum Q_{k,i}]}{n}$. We can bound T_1 with M_k. $\mathbb{E}[T_1] \leq \frac{L^2}{n}\sum_{i=1}^{n}\mathbb{E}[Q_{k,i}] = L^2\,\mathbb{E}[M_k]$

Finally, we put everything together.

$$\mathbb{E}\left[f(\frac{X_{k+1}1_n}{n})\right] \leq \mathbb{E}\left[f(\frac{X_k 1_n}{n})\right] - \frac{\gamma-\gamma^2 L}{2}\mathbb{E}\left[||\frac{\partial f(X_k)1_n}{n}||^2\right] - \frac{\gamma}{2}\mathbb{E}\left[\nabla f(\frac{X_k 1_n}{n})||^2\right] + \frac{\gamma^2 L}{2n}\sigma^2 + \frac{\gamma}{2}L^2\,\mathbb{E}[M_k]$$

Summing from $k=0$ to $K-1$.

$$\frac{\gamma-\gamma^2 L}{2}\mathbb{E}\left[||\frac{\partial f(X_k)1_n}{n}||^2\right] + \frac{\gamma}{2}\mathbb{E}\left[\nabla f(\frac{X_k 1_n}{n})||^2\right] \leq f(0) - f^* + \frac{\gamma^2 KL}{2n}\sigma^2 + \frac{\gamma}{2}L^2\sum_{k=0}^{K-1}\mathbb{E}[M_k]$$

$$\leq f(0) - f^* + \frac{\gamma^2 KL}{2n}\sigma^2 + \frac{\gamma^3 L^2 n\sigma^2}{(1-\rho)(1-\frac{18}{(1-\sqrt{\rho})^2}\gamma^2 nL^2)}K + \frac{9\gamma^3 L^2 n\varsigma^2}{(1-\rho)^2(1-\frac{18}{(1-\sqrt{\rho})^2}\gamma^2 nL^2)}K$$

$$+\frac{9n\gamma^3 L^2}{(1-\rho)^2(1-\frac{18}{(1-\sqrt{\rho})^2}\gamma^2 nL^2)}\sum_{k=0}^{K-1}\mathbb{E}\left[||\nabla f(\frac{X_k 1_n}{n})||^2\right]$$

Rearrange the derivation:

$$\leq \frac{f(0)-f^*}{\gamma K} + \frac{\gamma L}{2n}\sigma^2 + \frac{\gamma^2 L^2 n\sigma^2}{(1-\rho)C_2} + \frac{9\gamma^2 L^2 n\varsigma^2}{(1-\sqrt{\rho})^2 C_2}$$

If we appropriately pick a learning rate $\gamma = \frac{1}{2L+\sigma\sqrt{K/n}}$.

Corollary 4.3.1 (Convergence Rate of D-PSGD)

$$\frac{\sum k=0^{K-1}\mathbb{E}\left[||\nabla f(\frac{X_k 1_n}{n})||^2\right]}{K} \leq \frac{8(f(0)-f^*)L}{K} + \frac{(8f(0)-8f^*+4L)\sigma}{\sqrt{Kn}}$$

if K is sufficiently large, in particular,
$K \geq \frac{4L^4 n^5}{\sigma^6(f(0)-f^*+L)^2}(\frac{\sigma^2}{1-\rho} + \frac{9\varsigma^2}{(1-\sqrt{\rho})^2})^2$, *and* $K \geq \frac{72L^2 n^2}{\sigma^2(1-\sqrt{\rho})^2}$

This means the convergence rate for D-PSGD is $O(\frac{1}{K} + \frac{1}{\sqrt{nK}})$ given that we have enough iterations.

Implication of Corollary 4.3.1

- Linear Speedup
  - $\frac{1}{\sqrt{nK}}$ dominates $\frac{1}{K}$ for sufficiently large K.
  - total computational complexity $O(1/\epsilon^2)$ and on a single node is $O(1/n\epsilon^2)$
- Advantages
  - Computational complexity is the same as C-PSGD
  - Able to avoid network bottleneck since communication overhead is only $O(deg(G))$, whereas C-PSGD is $O(n)$. This means D-PSGD can yield a $O(1)$ complexity if the clusters is set up in a ring pattern.

5 Experiments

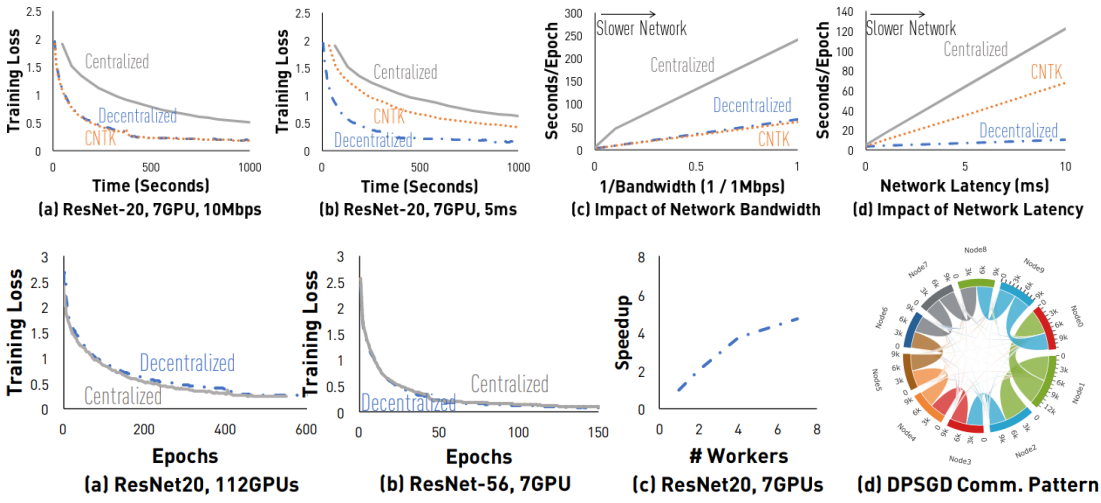
Lian et al. validated the convergence rate theorem dervived in section IV with experiments comparing D-PSGD against the other centralized methods. They evaluated D-PSGD on two different standard machine learning tasks. Here, we discuss the setup of their empirical data collection and the results on one of the tasks.

5.1 Cluster Setup

- 7GPUs. Local machine with 8 Nvidia TITAN Xp.
- 10 GPUs. EC2 instances, equipped with Nvidia K80 GPU.
- 16 GPUs. 16 local machines. Each node has two Xeon E5-2680 8-core procesors and Nvidia K20 GPU. connected by Gigabit Ethernet.
- 112 GPUS. EC2 intsances with Nvidia K80 GPUS.

5.2 Image Classification

- Datasets and Models: ResNet with different number of layers trained on CIFAR-10.
- Implementations
  - CNTK: synchronous C-PSGD based on MPI's AllReduce
  - Centralized: Implemented parameter-based synchronous SGD with MPI. Fixed one node as parameter server.
  - Decentralized: Implemented D-PSGD with MPI within CNTK.
- Results



In the leftmost plot, the network is intentionally slowd down. We see that the parameter-based C-PSGD suffers in performances from the communication overheads while D-PSGD converges more quickly. In terms of scalability, the second row of plots showed that the performance is similar for both D-PSGD and C-PSGD on 70 and 112 nodes.

bandwidth	high	low	high
latency	low	high	high
winner	similar	D-PSGD	D-PSGD

D-PSGD has more balanced communication pattern between nodes as noted in [LZZ+17], this is the main advantage allowing the algorithm to not be impacted by the network bottleneck as its centralized counterparts. CNTK's centralized AllReduce implementation was able to output close performance to D-PSGD when the bandwidth is low, however the communication overhead becomes more apparent with high latency.

6 Conclusion

The theoretical and empirical results on D-PSGD demonstrated that decentralized algorithm can outperform its centralized counterpart. [LZZ+17] proved that the asymptotic convergence rate is the same as C-PSGD. However, it is able to avoid the network bottleneck when the bandwidth is low. Further, the authors made mention to possible improvements.

The synchronization barrier incurs costs in the D-PSGD algorithm, breaking such limitations may improve the algorithm even further. In addition, it will be interesting to see how D-PSGD performs in a larger cluster with modern datacenter-grade GPU.

7 Acknowledgement

This work is submitted as the final paper for the course on Advanced Optimization Theory at Boston University.

Appendix A: Modification of D-PSGD

The authors made mentions to the following changes that are considered valid and does not hurt theoretical analysis.

- The stochastic gradient calculation can be replaced with mini-batch stochastic gradients
- Computing stochastic gradient $\nabla F_i(x_{k,i}; \xi_{k,i})$ and the neighborhood average can be run in parallel.
- We may update the local optimization variable before computing neighborhood weighted average. However, step 4,5 in the original algorithm can only be run in parallel if we leave this step to last.

References

[AD11] Alekh Agarwal and John C. Duchi. Distributed delayed stochastic optimization, 2011.

[BFH12] Pascal Bianchi, Gersende Fort, and Walid Hachem. Performance of a distributed stochastic approximation algorithm, 2012.

[GL13] Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming, 2013.

[LLZ17] Guanghui Lan, Soomin Lee, and Yi Zhou. Communication-efficient algorithms for decentralized and stochastic optimization, 2017.

[LZZ+17] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent, 2017.

[MB11] Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[NJLS09] Arkadii S. Nemirovski, Anatoli B. Juditsky, Guanghui Lan, and Alexander Shapiro. Robust Stochastic Approximation Approach to Stochastic Programming. *SIAM Journal on Optimization*, 19(4):1574–1609, January 2009.

[RNV08] S. Sundhar Ram, A. Nedich, and V. V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization, 2008.

[SN11] Kunal Srivastava and Angelia Nedic. Distributed asynchronous constrained stochastic optimization. *Selected Topics in Signal Processing, IEEE Journal of*, 5:772 – 790, 09 2011.

[SRNV09] S. Sundhar Ram, A. Nedić, and V. V. Veeravalli. Asynchronous gossip algorithms for stochastic optimization. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3581–3586, 2009.

[SY16] Benjamin Sirb and Xiaojing Ye. Consensus optimization with delayed and stochastic gradients on decentralized networks. *2016 IEEE International Conference on Big Data (Big Data)*, pages 76–85, 2016.

[YVQ10] Feng Yan, S. V. N. Vishwanathan, and Yuan (Alan) Qi. Cooperative autonomous online learning. *CoRR*, abs/1006.4039, 2010.

[ZHA16] Huan Zhang, Cho-Jui Hsieh, and Venkatesh Akella. Hogwild++: A new mechanism for decentralized asynchronous stochastic gradient descent. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 629–638, 2016.