GRADIENT DELAY ANALYSIS IN ASYNCHRONOUS DISTRIBUTED OPTIMIZATION

Haider Al-Lawati and Stark C. Draper

University of Toronto

ABSTRACT

Gradient-based algorithms play an important role in solving a wide range of stochastic optimization problems. In recent years, implementing such schemes in parallel has become the new paradigm. In this work, we focus on the asynchronous implementation of gradient-based algorithms. In asynchronous distributed optimization, the gradient delay problem arises since optimization parameters may be updated using stale gradients. We consider a hub-and-spoke system and derive the expected gradient staleness in terms of other system parameters such as the number of nodes, communication delay, and the expected compute time. Our derivations provide a means to compare different algorithms based on the expected gradient staleness they suffer from.

1. INTRODUCTION

Gradient-based algorithms are popular techniques used in solving stochastic optimization problems of the form

$$\min_{w\in\mathcal{W}} F(w) \text{ where } F(w) := \mathbb{E}_X\left[f(w,x)\right] \tag{1}$$

where X is a random variable that follows a distribution P over the set $\mathcal{X}$. The purpose of such algorithms is to find $w^* = \arg\min_{w\in\mathcal{W}} F(w)$. One of the widely used such algorithms is the *stochastic gradient descent* (SGD). In SGD, in each iteration j, the gradient $\nabla f(w(j), x(j))$ is calculated with respect to the current point $w(j)$ based on the observed sample $x(j)$ to obtain $w(j+1)$. One can also sample b data points $x(j,1), x(j,2), \dots, x(j,b)$ in each iteration j at a time and calculate $g(j)$ as an average of the b gradients. i.e., $g(j) = \frac{1}{b}\sum_{s=1}^{b} f(w(j), x(j,s))$.

This variant of SGD is known as *minibatch* SGD. Assuming that over the iterations $1, 2, \dots T$, the system observes $M = bT$ data points, the performance of these algorithms is measured by the convergence rate which measures how fast the sequence $w(1), w(2), \dots$ converges to the optimal solution w^*. It has been shown that for convex smooth objective functions, both SGD and minibatch SGD converge at a rate $\mathcal{O}(1/\sqrt{M})$ [1].

1.1. Parallelizing Gradient-Based Algorithms

In many modern day applications, solving stochastic optimization on a single machine may not be feasible and therefore must be parallelized. Various computing platforms are deployed to parallelize gradient-based algorithms such as shared memory architecture, clusters and grid computing [2]. In shared memory architecture such as multicore processors (e.g., multi-core CPUs), the optimization parameter is stored in a memory that can be accessed by all the processors whose job is to calculate the gradients and update the parameter. In grid and cluster computing platforms, several computing nodes are deployed to perform gradient calculations. These nodes are allowed to communicate with a single node known as the *master node* using communication protocols such as the *Message Passing Interface* (MPI). The master node keeps and updates the optimization parameter. Such setup is called *hub-and-spoke*. It is also possible to have a masterless setup in which nodes keep their own version of optimization parameter and communicate only with their neighbors [1, 3].

Distributed optimization methods can be categorized as synchronous or asynchronous. In synchronous methods [4, 1, 5, 3, 6], in each iteration j, all workers calculate gradients with respect to the same optimization parameter. For instance in synchronous minibatch SGD, in the j-th iteration, worker i calculates b/n gradients and sends their average denoted by $g_i(j)$ to the master. The master receives and aggregates these n messages from all workers resulting in a minibatch of size b. The master then updates the optimization parameter according to

$$w(j+1) = w(j) - \eta(j)\sum_{i=1}^{n} g_i(j)$$

where $\eta(j)$ is the learning rate or step size. In practical distributed system, the rate at which the optimization parameter is updated is affected by the speed of the slow computing nodes known as "stragglers" [7, 8, 9]. To alleviate this problem, asynchronous methods have been proposed [10, 11, 12, 2, 13]. In asynchronous methods, each worker or processor may use a different version of the optimization parameter. This leads to what is known as the "stale" (i.e., delayed) gradient problem as workers mostly carry out computations based on out-of-date global optimization parameters. For instance, in distributed SGD, the master updates the optimization parameter as soon as it receives a gradient from any worker and the resulting update rule is

$$w(j+1) = w(j) - \eta(j)\nabla f(j-\tau(j), x(j-\tau(j))) \quad (2)$$

where $\tau(j)$ is the number of steps by which the gradient is delayed in the j-th iteration. As we discuss in the next section, gradient staleness has a negative impact on convergence.

1.2. Related Works

Gradient staleness is one of the important quantities that shows up in many theoretical convergence analyses for asynchronous distributed optimization and has impact on its convergence. The impact of staleness on convergence rate has been studied in many recent works. In [2], the authors show that the delayed SGD algorithm with a fixed staleness τ after T iterations converges at a rate $\mathcal{O}\big(\sqrt{\tau/T}\big)$ for convex smooth objective functions. In [14], the authors derive the same for nonconvex optimization problems. When $\tau(j)$ is random with $\mathbb{E}[\tau(j)] \leq B < \infty$, the convergence rate becomes $\mathcal{O}\big(\sqrt{B/T}\big)$ [15]. Often τ is assumed to be a bounded quantity when analyzing convergence rate for asynchronous schemes.

We analyze staleness from a different perspective. In particular, we are interested in exploring the relationship between the staleness and other system parameters such as the number of nodes, compute time, communication time, etc. We study asynchronous optimization in a hub-and-spoke setup assuming a deterministic communication time between the master and workers. Our setup resembles one of the schemes studied in [16]. While [16] derives the expected inter-update time for that scheme using renewal theory, our work also applies results from renewal theory to derive the expected staleness. To the best of our knowledge, this is the first work that provides theoretical analysis of gradient staleness and establishes the connection between staleness and other system parameters. Past works such as [2, 15] observed that in certain settings gradient staleness scales linearly with the number of compute nodes in the system. However, no theoretical analysis or mathematical proofs were provided.

1.3. Contribution

In this work, we study gradient staleness that arises in asynchronous distributed stochastic optimization. We characterize asynchronous distributed optimization as a delayed renewal process and derive the expected rate at which the master updates the optimization parameter. Based on this delayed renewal process, we analyze gradient staleness and derive its expected value in terms of other system parameters such as the number of nodes, the expected compute time and the fixed communication time. Our derivation shows that the expected staleness scales linearly with the number of workers. The expected staleness is also proportional to the communication time. The expected staleness expression we derive is applicable to various asynchronous schemes such as asynchronous SGD and asynchronous minibatch SGD using hub-and-spoke setup. Our results provide a means to choose system parameters that help reduce staleness and improve convergence.

2. SYSTEM MODEL

In this work, we consider a hub-and-spoke distributed system consisting of a master and n compute nodes known as workers to solve the optimization problem (1) in an asynchronous manner. Workers compute gradients of the objective function while the master aggregates the gradients and updates the optimization parameter. In the j-th compute round, worker i takes $X_i(j)$ seconds to process a local minibatch of size b, where the sequence $\{X_i(j), i \in [n], j = 1, 2, \cdots\}$ are i.i.d. with $\mathbb{E}[X_i(j)] = \mu$. As soon as the worker completes the computation, it sends the local minibatch to the master, a transmission that takes $T_c/2$ seconds for some fixed nonnegative real T_c. The master, on the other hand, waits for K local minibatches to update the optimization parameter. Once the optimization parameter is updated, the master broadcasts the update to all workers. The updated parameter $w(j)$ takes $T_c/2$ seconds to reach the i-th worker.

Let $N_i(t)$ denote the number of times the master received local minibatches from worker i by time t, where $t \geq 0$. Let $N(t)$ denote the total number of local minibatches the master has received by time t. Then, $N(t) = \sum_{i=1}^{n} N_i(t)$. Let $M(t)$ be the number of times the master has updated the optimization parameter by time t. Furthermore, let $T(1), T(2), \cdots$ be the inter-update times of the optimization parameter. $T(m)$ is the time between the $(m-1)$-st and m-th updates. Hence the time when the m-th update takes place is $T_{up}(m) := \arg\min_{t\geq 0} \{M(t) \geq m\} = \sum_{i=1}^{m} T(i)$.

Suppose that worker i starts computing the j-th local minibatch using $w(m)$ at time t_1. This implies that $w(m)$ has been broadcasted by the master at most by time $t_2 = t_1 - T_b^i(m)$. Since all updates prior to $w(m)$ have also been broadcasted prior to t_2, then m is the largest value satisfying $T_{up}(m) \leq t_1 - T_b^i(m)$. Equivalently, m is the largest integer such that $\sum_{l=1}^{m} T(l) + T_b^i(m) \leq t_1$. The j-th minibatch by worker i is sent to the master at time $t_1 + X_i(j)$ and received by the master at time $t = t_1 + X_i(j) + T_w^i(j)$. The actual parameter at the master at time t is $w\big(M(t)\big)$. Hence the staleness experienced by these gradients is

$$\tau(t) = m - M(t). \quad (3)$$

3. MAIN RESULTS

In this section, we present our main theoretical results. First we show that the random process $N(t)$ we defined in Sec. 2 is a delayed renewal process and use that to derive the asymptotic expected rate of parameter updates. Next, we state our main theorem in which derive the limit of $\mathbb{E}[\tau(t)]$ in terms of other system's parameters such as n, μ, and T_c. For compactness, we will often drop the argument t and use τ.

Lemma 1. *Let $M(t)$ be as defined in Sec. 2, then the limit of the expected rate at which the master updates the optimization parameter is*

$$\lim_{t\to\infty} \frac{\mathbb{E}\big[M(t)\big]}{t} = \frac{n}{K\mu}. \tag{4}$$

Proof. Let $N_i(t)$, $N(t)$, $M(t)$, $\big\{X_i(j)\big\}_{j\geq 1}$, and T_c be as defined in Sec. 2. Define the sequence $Z_i(j)$ as follows. Let $Z_i(1) = X_i(j) + T_c/2$ and $Z_i(j) = X_i(j)$ for $j \geq 2$. Note that the j-th minibatch sent by worker i reaches the master at time $\sum_{l=1}^{j} X_i(l) + T_c/2$. Hence, the sequence $\big\{Z_i(j)\big\}_{j\geq 1}$ represents the inter-arrival times for $N_i(t)$. Observe that $\big\{Z_i(j)\big\}_{j\geq 1}$ is an independent sequence of random variables and that $\big\{Z_i(j)\big\}_{j\geq 2}$ are also identically distributed. Thus, for each $i \in [n]$, $N_i(t)$ is a delayed renewal process since it is a counting process with independent inter-arrival times $Z_i(1), Z_i(2), \cdots$ while $Z_i(2), Z_i(3), \cdots$ are identically distributed as well. Using renewal theorem results for the delayed renewal processes we get that for each $i \in [n]$, the rate of arrivals of minibatch from worker i in the limit is $\lim_{t\to\infty} \mathbb{E}\big[N_i(t)\big]/t = \frac{1}{\mu}$. Hence, the expected rate of minibatch arrivals at the master is

$$\lim_{t\to\infty} \frac{\mathbb{E}\big[N(t)\big]}{t} = \sum_{i=1}^{n} \lim_{t\to\infty} \frac{\mathbb{E}\big[N_i(t)\big]}{t} = \frac{n}{\mu}$$

Since the master updates the parameter for every K minibatches it receives,

$$\lim_{t\to\infty} \frac{\mathbb{E}\big[M(t)\big]}{t} = \lim_{t\to\infty} \frac{\mathbb{E}\big[N(t)\big]}{Kt} = \frac{n}{K\mu}.$$

□

Theorem 2. *Let τ be the gradient staleness of the optimization scheme introduced in Sec. 2. Then in the limit*

$$\lim_{t\to\infty} \mathbb{E}\big[\tau\big] = \frac{n}{K}\frac{T_c + \mu}{\mu}. \tag{5}$$

The proof of Theorem 2 is omitted due to space limitation but will appear in the full version of this work.

Observe that the term $\frac{n}{K\mu}$ is the asymptotic expectation of the rate at which the master updates the optimization parameter. Hence, Theorem 2 looks like it should follow by an application of Little's Law to an appropriately defined queuing system. However, Little's Law is applicable for queues without preemption, while our system allows preemption. In fact, in our analysis we assume that workers may receive multiple updates while gradients computation is in progress. However, only the most updated parameter is considered in the next compute round while older updates are discarded. Hence, Little's Law is not applicable in this context.

Corollary 3. *For the SGD and minibatch SGD schemes, the limit of the expected gradient staleness is*

$$\lim_{t\to\infty} \mathbb{E}\big[\tau\big] = n\frac{T_c + \mu}{\mu}. \tag{6}$$

Proof. The proof follows directly from Theorem 2 by setting $K = 1$, since in both SGD and minibatch SGD, the master updates the parameter as soon as it receives gradients from any worker. □

Our results above confirms that gradient staleness scales linearly with the number of workers which was observed in previous literature. To the best of our knowledge our work is the first one to formally provide mathematical proof. Moreover, Theorem 2 and Corollary 3 show that as the staleness increases with communication time. Additionally, our results can be used to compare two schemes in terms of expected staleness they suffer from even if both schemes have the same expected inter-update times as well as the same per-epoch minibatch size. In Sec. 4, we demonstrate this with an example.

4. NUMERICAL RESULTS

In this section, we present some numerical results to validates our theoretical analysis. We simulate the systems by modeling the inter-arrival times $\big\{X_i(j)\big\}$ as an i.i.d. process according to the shifted exponential distribution with parameters λ and α. In other words, for each i and each j,

$$P(X_i(j) \leq t) = \lambda \exp\big\{-\lambda(t-\alpha)\big\},\ t \geq \alpha, \alpha \geq 0.$$

The exponential and shifted exponential distributions are widely used in literature to model the variable compute times [16, 17, 18, 19]. We choose the shifted exponential distribution as the shift part α ensures a minimum amount of time that is nonzero to complete the compute task. We note the reader that our theoretical results are not restricted to these distributions. In fact, any distribution model will provide the same results as we made no assumptions on the distribution type when we derived our theoretical results in Section 3.

In our simulations, we set $\alpha = 1$, $\lambda = 2/3$, $T_c = 1$ and $K = 10$ while we vary the number of workers $10 \leq n \leq 20$ since K must satisfy $K \leq n$. Figure 1 plots the theoretical values for the limit of the expected staleness versus the average staleness from the simulation.

We note the reader that our theoretical derivations apply to the limit of $\mathbb{E}[\tau(t)]$ while in our numerical result above, we compare the limit values with the empirical averages. To understand why we are interested in this limiting behavior, consider the following example. Suppose that for a system

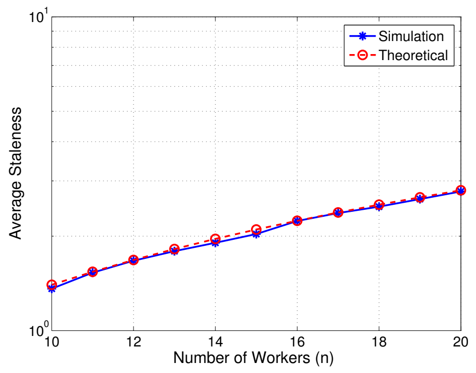

Fig. 1: Simulation vs. theoretical results for expected staleness with $K = 10$, $T_c = 1$.

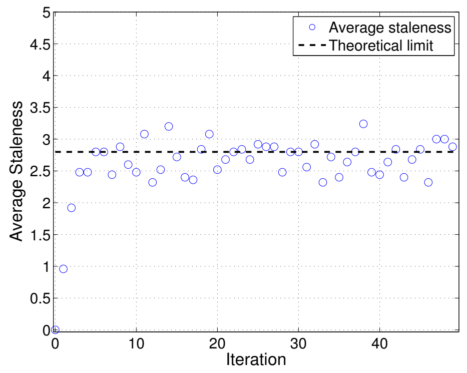

Fig. 2: Average gradient staleness per iteration with $K = 5$ and $n = 10$ workers.

consisting of a master and $n = 10$ workers with $K = 5$. All workers initially start with $w(1)$. Hence, the first update of the optimization parameter to will evaluate $w(2)$ using staleness-free gradients since they are with respect to $w(1)$. In other words, $\tau = 0$. Since $K = 5$, the master uses the first 5 local minibatches from the 10 workers to evaluate $w(2)$. In the second iteration, the master uses the remaining 5 local minibatches of gradients with respect to $w(1)$ to evaluate $w(3)$. Therefore, $\tau(2) = 1$. However, as the process continues, gradient staleness increases till it fluctuates around some value. Figure 2 depicts the average gradient delay per iteration for the first 50 iterations. Observe the transient behavior in the first few iterations, but eventually the system reaches its steady-state and the average staleness per iteration fluctuates around the theoretical limit we derived.

Finally, we compare the performance of two cases that have the same average per-epoch runtime, communication time, and per-epoch minibatch size. For both cases, we fix $n = 10$. In Case 1, we set $b_1 = 1$ and $K_1 = 10$ and assume that $\mu_1 = 2$. In Case 2, we set $b_2 = 10b_1$ and hence $\mu_2 = 10\mu_1$ assuming that computation time grows linearly with respect to the task load. This means that μ is a proportional to b. For Case 2, we choose $K_2 = 1$. Furthermore, we assume $T_c = 0$ in both cases. Note that the expected inter-update time is 2 and the per-epoch minibatch size is 10 gradients in both cases. However, when using our results, $\mathbb{E}[\tau_1] = 1$ while $\mathbb{E}[\tau_2] = 10$ in the limit, where τ_i is the staleness for Case i. This suggests that Case 2 suffers from larger gradient staleness and hence we expect it to overall perform worse than Case 1. Figure 3 compares the performance of Case 1 versus Case 2 when using both systems to solve a linear regression problem using a synthetic dataset. The system in Case 1 achieves a lower error rate than the system in Case 2 in the same amount of time.

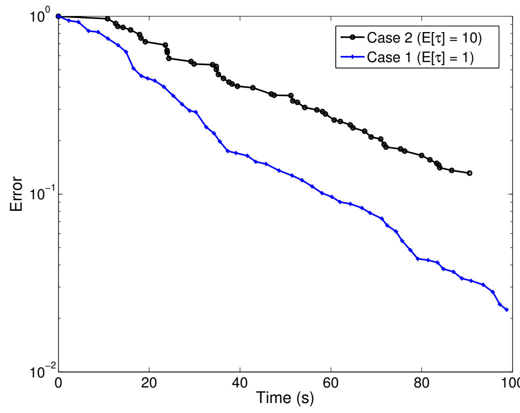

Fig. 3: Performance comparison between Case 1 and Case 2

5. CONCLUSION

In this paper, we present a novel analysis to evaluate the expected value of gradient staleness process in asynchronous distributed stochastic optimization. Our analysis uses renewal theory to derive the theoretical limits of gradient staleness. We present our results assuming the hub-and-spoke setup with deterministic communication time between the master and workers. Our theoretical analyses are validated by numerical simulations. Moreover, our theoretical results show that staleness scales linearly with the number of nodes, a phenomenon that has been observed and reported previously.

6. ACKNOWLEDGEMENT

This work was supported by Huawei Technologies Canada, the Natural Science and Engineering Research Council (NSERC) of Canada through a Discovery Research Grant, and the Omani Government Postgraduate Scholarship. We thank Jason Lam and Zhenhua Hu of Huawei Technologies Canada and Tharindu Adikari from the University of Toronto for technical discussions.

7. REFERENCES

[1] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, “Optimal distributed online prediction using mini-batches,” *Journal of Machine Learning Research*, pp. 165–202, Jan. 2012.

[2] M. Zinkevich, J. Langford, and A. J. Smola, “Slow learners are fast,” in *Advances in Neural Inf. Process. Sys.*, 2009, pp. 2331–2339.

[3] Konstantinos I. T. and M. G. Rabbat, “Efficient distributed online prediction and stochastic optimization with approximate distributed averaging,” *IEEE Trans. Signal and Inf. Process. Nets*, pp. 489–506, 2016.

[4] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz, “Revisiting distributed synchronous sgd,” in *Proc. of Int. Conf. on Learning and Representation*, 2016.

[5] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, “More effective distributed ml via a stale synchronous parallel parameter server,” in *Advances in Neural Inf. Process. Sys.*, pp. 1223–1231. 2013.

[6] M. Nokleby and W. U. Bajwa, “Distributed mirror descent for stochastic learning over rate-limited networks,” in *Proc. of IEEE Int. Workshop on Comp.Advances in Multi-Sensor Adaptive Process.*, Dec. 2017, pp. 1–5.

[7] J. Dean and L. A. Barroso, “The tail at scale,” *Communications of the ACM*, pp. 74–80, 2013.

[8] P. Garraghan, X. Ouyang, R. Yang, D. McKee, and J. Xu, “Straggler root-cause and impact analysis for massive-scale virtualized cloud datacenters,” *IEEE Trans. on Services Computing*, pp. 91–104, Jan. 2019.

[9] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, “Improving mapreduce performance in heterogeneous environments,” in *Proc. of USENIX Conf. on Operating Sys. Design and Imp.*, 2008.

[10] H. Al-Lawati, N. Ferdinand, and S. C. Draper, “Anytime minibatch with stale gradients,” in *Proc. of Conf. on Inf. Sciences and Sys.*, Mar. 2019, pp. 1–5.

[11] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, “An asynchronous mini-batch algorithm for regularized stochastic optimization,” *IEEE Trans. on Automatic Cont.*, pp. 3740–3754, 2016.

[12] N. Feng, B. Recht, C. Re, and S. J. Wright, “Hogwild: A lock-free approach to parallelizing stochastic gradient descent,” in *Advances in Neural Inf. Process. Sys.*, pp. 693–701. 2011.

[13] X. Lian, Y. Huang, Y. Li, and J. Liu, “Asynchronous parallel stochastic gradient for nonconvex optimization,” in *Advances in Neural Inf. Process. Sys.*, 2015, pp. 2737–2745.

[14] A. Nedich, D. P. Bertsekas, and V. S. Borkar, “Distributed asynchronous incremental subgradient methods,” *Studies in Comp. Math.*, pp. 381–407, 2001.

[15] A. Agarwal and J. C. Duchi, “Distributed delayed stochastic optimization,” in *Advances in Neural Inf. Process. Sys.*, pp. 873–881. 2011.

[16] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, “Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD,” in *Proc. of Int. Conf. on Artificial Intelligence and Statistics*, 2018, pp. 803–812.

[17] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” *IEEE Trans. on Inf. Theory*, pp. 1514–1529, Mar. 2018.

[18] E. Ozfatura, D. Gündüz, and S. Ulukus, “Speeding up distributed gradient descent by utilizing non-persistent stragglers,” *CoRR*, 2018.

[19] S. Dutta, V. Cadambe, and P. Grover, “Short-dot: Computing large linear transforms distributedly using coded short dot products,” in *Advances in Neural Inf. Process. Sys.*, pp. 2092–2100. 2016.