Asynchronous Distributed Stochastic Optimization with Double-pass Gradient Compression

Yonggui Yan **Chia-Yu Chen** **Pin-Yu Chen** **Xiaodong Cui** **Songtao Lu** **Lior Horesh** **Yangyang Xu**
Rensselaer Polytechnic Institute IBM Research

Abstract

As the size of data and/or machine learning models is increasing, implementing training algorithms over a single process becomes inefficient for large-scale problems. With the availability of multiple computing resources, e.g., GPU/CPU, distributed parallel computing has been studied over decades. However, this type of computing needs massive communication overheads to exchange information among processes, which will cost extra training time due to the tremendous transmitted information or the waiting for the communication especially when some slow processes present. Model compression techniques including the quantization and sparsification are perhaps the most efficient ways to reduce the amount of the exchanging messages. In this work, by leveraging the error-compensation method, an asynchronous parallel stochastic gradient compressed algorithm is proposed to deal with the staggler issue in distributed training problems, and the gradient compression process is implemented at both uplink (local to master) and downlink (master to local) transmissions. We show that the proposed asynchronous distributed algorithm with double-pass gradient compression can achieve the same convergence rate as the state-of-the-art training algorithms. Multiple numerical experiments are provided to support our theoretical analysis.

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

1 INTRODUCTION

A broad spectrum of machine learning problems are involved with the following stochastic optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) = \mathbb{E}_{\xi\in\Xi}[F(\mathbf{x},\xi)], \tag{1}$$

where $\mathbf{x}$ is the model parameters and ξ is a random data sample drawn from data set Ξ. $F(\mathbf{x},\xi)$ is the loss function given the model $\mathbf{x}$ and data sample ξ. We will assume that $f(\mathbf{x})$ is a smooth (but not necessarily convex) function.

The state of the art machine learning relies on large scale training data which may require long training time on a single computing process. Therefore, distributed training has been widely used under this condition to speed up the training process, among which parallel stochastic gradient descent (PSGD) based data parallel distributed training is most popular. Data parallel PSGD distributes data to multiple workers and computes the gradients in parallel before aggregating them for the model update [Abadi et al., 2016, Seide and Agarwal, 2016]. It may be realized in a centralized [Ghadimi et al., 2016] or decentralized architecture [Jin et al., 2016, Lian et al., 2017, Shen et al., 2018, Tang et al., 2018]. The centralized setting employs a central server as a hub to coordinate the training with the workers. Each worker only communicates with the hub. The decentralized setting has no central server and workers form a communication topology (e.g. a ring) to communicate among themselves.

The communication cost is a crucial factor to consider when designing a distributed training strategy. The current machine learning oftentimes resorts to models with a large number of parameters, which indicates a high communication cost when transferring models or gradients from the workers to the server or among the workers. Given the bandwidth of the communication network, the gradients of large models may become a bottleneck and incur significant latency

in training. Hence, communication efficient algorithms such as gradient compression have been actively sought after [Seide et al., 2014, Alistarh et al., 2017, Aji and Heafield, 2017, Tang et al., 2019].

In this work, we focus on gradient compression in a centralized distributed setting. We employ the error-compensated double-pass gradient compression called Double Squeeze (DS) in [Tang et al., 2019] for bandwidth-reduction training. In the meantime, we introduce asynchrony to DS. In [Tang et al., 2019], DS is realized in a synchronous mode where the master parameter server has to idle until all workers finish their local gradient evaluation for aggregation. If one worker is slow in computation, it will become a "straggler" and slow down the whole training process. In the extreme case, if one worker stops responding, the training will fail. The proposed Asynchronous Double Squeeze (A-DS) will effectively overcome this straggler issue and accelerate the training. The main contributions of this work are highlighted as follows.

- We provide the theoretical convergence guarantees of A-DS for non-convex problems. We show that as long as the staleness induced by the asynchrony is bounded by $K^{1/4}$, the asymptotical convergence rate $O(\sqrt{\frac{1}{MK}})$ can be achieved.
- We evaluated A-DS for image classifcation on CIFAR10 [Krizhevsky et al., 2009] using a ResNet model [He et al., 2016]. The results validate the superiority of A-DS compared with a varity of existing PSGD algorithms.

Notation. Throughout this paper, all-zero vector is denoted as $\mathbf{0}$. $\mathbf{x}^*$ denotes the global optimal solution of problem (1). $\nabla F(\mathbf{x}, \xi)$ represents the gradient of F at $\mathbf{x}$ with the sample ξ and $\nabla f(\mathbf{x})$ is the gradient of f. In a centralized architecture, the number of computing processes is denoted as n. Among these n nodes, one is the master and the remaining $N = n - 1$ are workers. We denote the index of workers by $i \in \{1, 2, ..., N\}$. We specify the variable on worker i by adding a superscript $^{(i)}$, such as the stochastic gradients $\mathbf{g}^{(i)}$ and compression error vector $\delta^{(i)}$. $\mathbb{E}[\cdot]$ takes the full expectation.

2 RELATED WORK

Gradient Compression Gradient compression can be categorized into two groups of techniques: quantization and sparsification. The quantization approaches include 1-bit stochastic gradient descent (SGD) [Seide et al., 2014], SignSGD [Bernstein et al., 2018], QSGD [Alistarh et al., 2017], TernGrad [Wen et al., 2017]. The sparsification approaches include Random-k [Stich et al., 2018], Top-k [Aji and Heafield, 2017], and Threshold-v [Dutta et al., 2019]. An extensive survey on the gradient compression techniques can be found in [Xu et al., 2020].

Residual and Error-feedback To achieve high compression ratio, most today's compression methods induces certain information loss. Large compression error may slow down the convergence. Several approaches had been proposed to mitigate compression error. One way to reduce the compression error is to compress the residue (or delta quantity) rather than the gradient itself due to its smaller dynamic range, for instance, Choco-SGD [Koloskova et al., 2019a, Koloskova et al., 2019b] and DIANA [Mishchenko et al., 2019]. Another way to alleviate the issue is error compensation or error-feedback which saves the error of compression in one SGD step and compensates it in the next SGD step before another compression [Seide et al., 2014]. In DS [Tang et al., 2019], workers send compressed gradients to the master and the master averages the received compressed gradients, then compresses and sends it to all workers. Two-side error compensations are used on both master and workers.

Asynchronous distributed training The asynchronous methods just need some available workers participating in the communication instead of all workers [Tsitsiklis et al., 1986, Zhou et al., 2018]. For example, in the centralized architecture, the communication can only be finished when all workers get and send the stochastic gradients in the synchronous methods, while in asynchronous methods, the communication can be finished as long as some workers get and send the stochastic gradients regardless of others. Thus when the workers compute at different speeds, the asynchronous algorithms would reduce much times for each round of communication. A-PSGD [Lian et al., 2015] solves (1) asynchronously on the centralized architecture. The master repeatedly collects M stochastic gradients from part of the workers, averages them and sends the averaged gradient to workers. The workers repeatedly, without coordination to each other, update the model with the received gradients from the master and then calculate a new gradient based on the updated model. Because of the different staleness caused by asynchrony, the gradients that the master collects may be calculated at different points.

In Table 1, we compare the different algorithms on a centralized architecture. Compression and asynchrony are used in DS and A-PSGD, respectively. The asymptotical convergence rates of PSGD, DS and A-

Table 1: Comparison of different parallel algorithms on the centralized architecture with N workers. In each iteration of A-PSGD and A-DS, M stochastic gradients are collected and averaged. K is the total number of iterations. **Comp** (Compression), **Asy** (Asynchrony), **ACR** (Asymptotical Convergence Rate).

Algorithm	**Comp**	**Asy**	**ACR**
PSGD	×	×	$O(\sqrt{\frac{1}{NK}})$
DS	✓	×	$O(\sqrt{\frac{1}{NK}})$
A-PSGD	×	✓	$O(\sqrt{\frac{1}{MK}})$
A-DS	✓	✓	$O(\sqrt{\frac{1}{MK}})$

PSGD are $O(\sqrt{\frac{1}{NK}}), O(\sqrt{\frac{1}{NK}}), O(\sqrt{\frac{1}{MK}})$, where K is the total number of iterations, N is the number of workers, and M is the number of the stochastic gradients collected and averaged in each iteration in the asynchronous algorithms. DS and A-PSGD speed up PSGD from two different aspects. One natural question is whether we take the advantages of both compression and asynchrony in one algorithm. We give an affirmative answer to this question.

3 ASYNCHRONOUS DOUBLE SQUEEZE

In this section, we introduce the asynchronous variant of Double Squeeze (A-DS) by first describing the algorithm details locally and then providing the mathematical update from a global view.

3.1 Local View of A-DS

We consider the centralized distributed architecture like DS and A-PSGD. Similar to DS, all information communicated from the master to workers or from workers to the master are compressed with error compensation. The main difference is that the communication protocol is no longer synchronous. Analogous to A-PSGD, in each update, A-DS collects M out of N stochastic gradients that may be calculated at different points.

The master and the workers are fed with the same initial value $\mathbf{x}$, and all initial compression errors are set to zero, i.e.,

$$\delta \leftarrow \mathbf{0}, \quad \delta^{(i)} \leftarrow \mathbf{0}, \forall i.$$

Then the master repeatedly performs the following steps:

- Select M compressed stochastic gradients from workers. Denote the indices of these M workers as $\{i_m\}_{m=1}^M$ and the corresponding compressed stochastic gradients as $\{\widehat{\mathbf{g}}^{(i_m)}\}_{m=1}^M$;
- Average the selected compressed stochastic gradients:
$$\Delta \leftarrow \frac{1}{M}\sum_{i=1}^{M} \widehat{\mathbf{g}}^{(i_m)};$$
- Compress the master's error-compensated stochastic gradient and update the error:
$$\widehat{\Delta} \leftarrow Q_\omega[\delta + \Delta], \quad \delta \leftarrow \delta + \Delta - \widehat{\Delta};$$
Here, Q_ω denotes the compressor on the master and ω indicates the stochasticity of the compressor.
- Send the compressed averaged gradient $\widehat{\Delta}$ back to all workers no matter whether they send a compressed stochastic gradient to the master in this update or not.

Each worker (using worker i as an example) basically repeats the following steps:

- Update the model parameters $\mathbf{x}$ with the received compressed averaged $\widehat{\Delta}$:
$$\mathbf{x} \leftarrow \mathbf{x} - \gamma\widehat{\Delta},$$
where γ denotes the learning rate;
- Compute the local stochastic gradient with the local sample $\xi^{(i)}$ (mini-batch could be used here):
$$\mathbf{g}^{(i)} \leftarrow \nabla F(\mathbf{x}, \xi^{(i)}),$$
where $\mathbf{x}$ is the current model parameters on the worker i, and each random sample ξ_i is independent with the random sample on other workers $\xi_j, j \neq i$, the model parameters $\mathbf{x}$ and the compressor $Q_{\omega^{(i)}}$
- Compress the worker's error-compensated stochastic gradient and update the error:
$$\widehat{\mathbf{g}}^{(i)} \leftarrow Q_{\omega^{(i)}}[\delta^{(i)} + \mathbf{g}^{(i)}], \quad \delta^{(i)} \leftarrow \delta^{(i)} + \mathbf{g}^{(i)} - \widehat{\mathbf{g}}^{(i)},$$
Here, $Q_{\omega^{(i)}}$ is the compressor on the worker i and $\omega^{(i)}$ indicates its stochasticity;
- Send $\widehat{\mathbf{g}}^{(i)}$ to the master.

Due to the asynchrony of the transmission, one worker may receive more than one or zero compressed averaged gradients from the master when updating the

model parameters. When the master selects M compressed stochastic gradients, the number of received compressed stochastic gradients from workers may already be larger than M and then the master selects M out of them, or it may be smaller than M and the master needs to wait until M compressed stochastic gradients are received. Also, among the selected M stochastic gradients for one time, more than one gradients may be from the same worker. Which case happens will depend on the relation between the computation and communication time, and the relation between M and N. Under the most ideal M, the master does not need to wait and also all received gradients will be used.

3.2 Global View of A-DS

In the following, let count iteration by the number of updates from the master. For each variable, the subscript $_k$ indicate its value after k iteration on the master, for example $\mathbf{x}_k$ denotes the value of parameter $\mathbf{x}$ after k updates with the averaged gradients.

Due to the asynchrony, the stochastic gradients the master receives at the kth iteration can be based on stale model parameters. Let $\tau_k^{(i)}$ denote the staleness for $\mathbf{x}$ that is used in computing the stochastic gradient by worker i at the kth iteration. We suppose the value of $\mathbf{x}$ used to compute the stochastic gradient is a real state of $\mathbf{x}$ no matter at which time point. At iteration k, some worker i may not be selected, and we let $\delta_k^{(i)} = \delta_{k-1}^{(i)}$; some worker i may be selected more than once, then $\delta^{(i)}$ is updated multiple times, and we denote the final one as $\delta_k^{(i)}$. Without loss of generality, we suppose at each iteration k, any worker i is selected at most once in the analysis for simplifying the notations. The global view of the algorithm is described in Algorithm 1.

Remark 1. In the algorithm, each worker has a copy of model parameters. The model parameters at different workers are initialized with the same values and are updated by using the same averaged gradients received from the master. Suppose that each worker will use all the averaged gradients that it receives to perform updates sequentially. Then the workers will have the same model parameters after performing the same number of updates, though the k-th update to the model parameters on all workers may happen at different time. Hence, we use the same notation $\mathbf{x}_k$ to denote the model parameters after the k-th iteration for all workers.

From the updates in Algorithm 1, we have the following lemma to show that how the optimization variable is updated through one round update of the proposed A-DS .

Algorithm 1 Global view of Asynchronous Double Squeeze (A-DS)

1: **Initialization:** Choose a non-increasing learning rate sequence $\{\gamma_k\}_{k\geq 1}$; set the same initial values $\mathbf{x}_1$ and compression error $\delta_0^{(i)} = \mathbf{0}$ for all workers $i = 1, 2, ..., N$, and the compression error $\delta_0 = \mathbf{0}$ for the master.
2: **for all** $k = 1, 2, ..., K-1$ **do**
3: Select M workers $\{i_m\}_{m=1}^M$;
4: **for all** $m = 1, 2, ..., M$ **do**

$$\mathbf{g}_k^{(i_m)} = \nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}}, \xi_k^{(i_m)}), \tag{2}$$

$$\mathbf{v}_k^{(i_m)} = \delta_{k-1}^{(i_m)} + \mathbf{g}_k^{(i_m)}, \tag{3}$$

$$\widehat{\mathbf{g}}_k^{(i_m)} = Q_{\omega_k^{(i_m)}}[\mathbf{v}_k^{(i_m)}], \tag{4}$$

$$\delta_k^{(i_m)} = \mathbf{v}_k^{(i_m)} - \widehat{\mathbf{g}}_k^{(i_m)}. \tag{5}$$

5: **end for**
6: The master receives $\{\widehat{\mathbf{g}}_k^{(i_m)}\}_{m=1}^M$ and does

$$\Delta_k = \frac{1}{M}\sum_{m=1}^M \widehat{\mathbf{g}}_k^{(i_m)}, \tag{6}$$

$$\mathbf{v}_k = \delta_{k-1} + \Delta_k, \tag{7}$$

$$\widehat{\Delta}_k = Q_{\omega_k}[\mathbf{v}_k], \tag{8}$$

$$\delta_k = \mathbf{v}_k - \widehat{\Delta}_k. \tag{9}$$

7: The master sends $\widehat{\Delta}_k$ to all workers and the workers update the model parameters

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k \widehat{\Delta}_k. \tag{10}$$

8: **end for**

Lemma 1. *The update rule from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ defined in Algorithm 1 satisfies*

$$\begin{aligned}\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k \frac{1}{M}\sum_{m=1}^M \nabla f\left(\mathbf{x}_{k-\tau_k^{(i_m)}}\right) \\ + \gamma_k \eta_k - \gamma_k \Omega_{k-1} + \gamma_k \Omega_k,\end{aligned} \tag{11}$$

where

$$\Omega_k = \delta_k + \frac{1}{M}\sum_{m=1}^M \delta_k^{(i_m)}, \tag{12}$$

$$\eta_k = \frac{1}{M}\sum_{m=1}^M \left(\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}}) - \nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}}, \xi_k^{(i_m)})\right). \tag{13}$$

Here, δ_k and $\delta_k^{(i_m)}$ are the compression errors, computed according to (5) *and* (9).

Our convergence analysis will be based on the updating rule (11). We can view η_k and Ω_k as some perturbations in an iterative method with delayed gradients. η_k is the averaged difference between the true gradients and the stochastic gradients that are computed at delayed points. Ω_k accumulates the master's compression error and the averaged compression errors of the M selected workers. Note that the staleness related errors as well as the compression errors are both coupled in these terms so that disentangling the errors and then quantifying them in the convergence analysis imposes more challenges than either synchronous or non-compressed settings. In order to have the convergence of $\mathbf{x}_k$, we will need to bound η_k and Ω_k, as the bounds are required to quantify the term $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2$ during the steps of proving the convergence rate of A-DS. In the next section, we will bound η_k and Ω_k respectively in Lemma 2 and Lemma 3. Then, we use these bounds to prove the convergence results.

4 CONVERGENCE ANALYSIS

In this section, we provide convergence guarantees of A-DS. We first give the assumptions that are necessary for our theoretical analysis. Based on these assumptions, we then show some Lemmas that serve as the pillar stones in establishing the final convergence results. All the detailed proofs are provided in the supplementary materials.

4.1 Assumptions

Through the analysis, we make the following assumptions.

Assumption 1 (**Gradient Lipschitz continuity**). *$f(\cdot)$ has L-Lipschitz continuous gradient, namely,*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

Assumption 2 (**Bounded gradient**). *∇f is uniformly bounded, i.e., there exists a constant $G > 0$ such that*

$$\|\nabla f(\mathbf{x})\|^2 \leq G^2, \forall \mathbf{x} \in \mathbb{R}^d.$$

Assumption 3 (**Bounded dalay**). *There exists an integer number T such that $\tau_k^{(i)} \leq T, \forall\, k, \forall\, i$.*

Assumption 4 (**Stochastic gradient**). *Each stochastic gradient is unbiased, and its variance is bounded, i.e., there is a constant $\sigma \geq 0$ such that*

$$\mathbb{E}_\xi[\nabla F(\mathbf{x}, \xi)] = \nabla f(\mathbf{x}),$$
$$\mathbb{E}_\xi[\|\nabla F(\mathbf{x}, \xi) - \nabla f(\mathbf{x})\|^2] \leq \sigma^2.$$

Assumption 5 (**Contraction property**). *There is a constant $\alpha \in [0, 1)$ such that for any iterations k, worker i, and $\mathbf{v} \in \mathbb{R}^d$, the following relations hold*

$$\mathbb{E}_{\omega_k^{(i)}}[\|Q_{\omega_k^{(i)}}[\mathbf{v}]\|^2] \leq \|\mathbf{v}\|^2, \tag{14}$$
$$\mathbb{E}_{\omega_k^{(i)}}[\|Q_{\omega_k^{(i)}}[\mathbf{v}] - \mathbf{v}\|^2] \leq \alpha^2\|\mathbf{v}\|^2, \tag{15}$$
$$\mathbb{E}_{\omega_k}[\|Q_{\omega_k}[\mathbf{v}] - \mathbf{v}\|^2] \leq \alpha^2\|\mathbf{v}\|^2. \tag{16}$$

The inequalities in (15) and (16) are commonly assumed for compression operators in the literature. They claim that the expected norm of the compression error of any vector $\mathbf{v}$ is smaller than the norm of $\mathbf{v}$. Many mentioned compressors in section 2 satisfy the condition include 1-bit SGD [Seide et al., 2014], Top-k [Alistarh et al., 2017], QSGD [Aji and Heafield, 2017], Random-k [Stich et al., 2018]. The inequality in (14) means that the compression process will not produce a vector with a larger norm. It may not hold for the quantization but holds for the sparsification. For example, on a given vector, the Top-k sparsification keeps its elements with the largest k absolute values and drops all other elements. Hence, the compressed vector and the compression error both contain a part of the original vector, and thus Top-k automatically satisfies all conditions in Assumption 5. In order to satisfy Assumption 5, we can set $Q_{\omega_k^{(i)}}$ to a sparsification compressor in our algorithm, and for Q_{ω_k}, we can choose a quantization or sparsification compressor.

4.2 Preliminary Lemmas

In this subsection, we provide two lemmas regarding the upper bound of $\mathbf{g}_k^{(i)}$, η_k and Ω_k, which will be used to show our main theorem.

By Assumptions 2 and 4, we have the following lemma.

Lemma 2. *For $\mathbf{g}_k^{(i)}$ given in (2) and η_k defined in (13), we have*

$$\mathbb{E}[\|\mathbf{g}_k^{(i)}\|^2] \leq \sigma^2 + G^2, \quad \forall k, i; \tag{17}$$
$$\mathbb{E}_{\xi_k}[\eta_k] = \mathbf{0}, \quad \mathbb{E}_{\xi_k}[\|\eta_k\|^2] \leq \frac{\sigma^2}{M}, \quad \forall k. \tag{18}$$

where $\mathbb{E}_{\xi_k}$ takes the expectation with respect to the random samples in the k-th iteration, i.e., $\xi_k = \{\xi_k^{(i_m)}\}_{m=1}^M$.

From Assumptions 2, 4 and 5, we can have the upper bound of $\mathbb{E}[\|\Omega_k\|^2]$ as the following.

Lemma 3. *For Ω_k defined in (12), we have*

$$\mathbb{E}[\|\Omega_k\|^2] \leq 2\widehat{G}^2, \tag{19}$$

where

$$\widehat{G} = \frac{\alpha^2(1+\frac{1}{\beta})}{1-\alpha^2(1+\beta)}(\sigma^2 + G^2 + \frac{(1+\frac{1}{\beta})(\sigma^2+G^2)}{1-\alpha^2(1+\beta)}), \tag{20}$$

and $\beta \in (0, \frac{1}{\alpha^2} - 1)$ *is a constant.*

Note that the upper bound of $\mathbb{E}[\|\Omega_k\|^2]$ can be obtained from the bounds of $\mathbb{E}[\|\delta_k\|^2]$ and $\mathbb{E}[\|\delta_k^{(i)}\|^2]$. In the convergence analysis of the classic DS methods, e.g., [Tang et al., 2019], it is claimed that $\mathbb{E}[\|\delta_k\|^2]$ and $\mathbb{E}[\|\delta_k^{(i)}\|^2]$ can be bounded without the condition in (14). However, the discussion in [Tang et al., 2019] focuses on the bound of $\mathbb{E}[\|\delta_k^{(i)}\|^2]$ but not much about the bound of $\mathbb{E}[\|\delta_k\|^2]$. We notice that the new condition in (14) is actually necessary to bound $\mathbb{E}[\|\delta_k\|^2]$. To quantify $\mathbb{E}[\|\delta_k^{(i)}\|^2]$, the bound of $\mathbb{E}[\|\mathbf{g}_k^{(i)}\|^2]$ is needed; similarly, to bound $\mathbb{E}[\|\delta_k\|^2]$, the bound of $\mathbb{E}[\|\Delta_k\|^2]$ is required as well. With the condition $\mathbb{E}_{\omega_k^{(i)}}[\|Q_{\omega_k^{(i)}}[\mathbf{v}]\|^2] \leq \|\mathbf{v}\|^2$, we can show that the upper bound of $\mathbb{E}[\|\Delta_k\|^2]$ is measured by the following arguments:

$$\begin{aligned}
&\mathbb{E}[\|\Delta_k\|^2] = \mathbb{E}\left[\left\|\frac{1}{M}\sum_{m=1}^{M}\widehat{\mathbf{g}}_k^{(i_m)}\right\|^2\right] \\
\leq &\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[\|\widehat{\mathbf{g}}_k^{(i_m)}\|^2] = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[\|Q_{\omega_k^{(i_m)}}[\mathbf{v}_k^{(i_m)}]\|^2] \\
\leq &\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[\|\mathbf{v}_k^{(i_m)}\|^2],
\end{aligned}$$

where the first inequality holds by the Cauchy-Schwarz inequality. The explicit expression of the upper bound of $\mathbb{E}[\|\mathbf{v}_k^{(i_m)}\|^2]$ and the detailed proof of Lemma 3 are given in the supplementary materials.

4.3 Convergence Guarantees

Now we are ready to present the main theorem of the convergence performance of A-DS.

Theorem 4. *Under Assumptions 1–5, if the learning rate sequence* $\{\gamma_k\}_{k=1}^K$ *in Algorithm 1 satisfies*

$$\frac{\gamma_k}{4} - \frac{L\gamma_k^2}{2} - 2L^2T\gamma_k^2\sum_{l=1}^{T}\gamma_{k+l} \geq 0, \forall\, k, \tag{21}$$

then we have

$$\begin{aligned}
&\frac{\sum_{k=1}^K \gamma_k \mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2]}{\sum_{k=1}^K \gamma_k} \\
\leq &\frac{2}{\sum_{k=1}^K \gamma_k}\Bigg(f(\mathbf{x}_1) - f(\mathbf{x}^*) \\
&+ \sum_{k=1}^{K}\left(\frac{L\gamma_k^2}{2} + 2\gamma_k L^2\sum_{j=k-T}^{k-1}\gamma_j^2\right)\frac{\sigma^2}{M} \\
&+ \sum_{k=1}^{K}\left(2\gamma_k^3L^2 + 8\gamma_k L^2 T\sum_{j=k-T}^{k-1}\gamma_j^2\right)\widehat{G}^2\Bigg),
\end{aligned} \tag{22}$$

where $\widehat{G}$ *is given in* (20) *and we use the convention* $\gamma_j = 0$, $\tau_j^{(i_m)} = 0$ *and* $\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}}) = \mathbf{0}$, *for* $j \leq 0$.

Like the DS method [Tang et al., 2019] and A-PSGD [Lian et al., 2015], we use the ergodic convergence as the metric to measure the convergence behaviours of the stochastic algorithms. Let $\tilde{K}$ be randomly selected from $\{1, \ldots, K\}$ by following the distribution

$$\text{prob}(\tilde{K} = k) = \frac{\gamma_k}{\sum_{k=1}^K \gamma_k}, \forall\, k = 1, \ldots, K.$$

Then $\mathbb{E}[\|\nabla f(\mathbf{x}_{\tilde{K}})\|^2]$ is bounded by the right side of (22).

Remark 2. If we do not use compression in A-DS, then the compression errors $\delta_k = \mathbf{0}$ and $\delta_k^{(i)} = \mathbf{0}$. Also, we have $\Omega_k = \mathbf{0}$. In this case, we can let $\widehat{G} = 0$ in Lemma 3, and the condition in (21) and the bound in (22) coincide with those analyzed in [Lian et al., 2015] about A-PSGD.

From the generic result in Theorem 4, we obtain the following corollary by choosing the learning rate appropriately.

Corollary 5. *Under Assumptions 1–5, let the learning rate in Algorithm 1 be a constant* γ

$$\gamma := \sqrt{\frac{M}{KL\sigma^2}}. \tag{23}$$

If the maximum staleness T *satisfies*

$$4LT\sigma^2 + M(4L + 16LT^2)\widehat{G}^2 \leq \sigma^3\sqrt{\frac{KL}{M}}, \tag{24}$$

then we have

$$\frac{\sum_{k=1}^K \mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2]}{K} \leq (2 + 2(f(\mathbf{x}_1) - f(\mathbf{x}^*)))\sqrt{\frac{L\sigma^2}{MK}}.$$

Remark 3. The corollary claims that the convergence rate is $O(\sqrt{\frac{1}{MK}})$, which excatly matches the rate of A-PSGD. If $\widehat{G} > 0$, the inequality in (24) requires that the maximum staleness $T = O(K^{1/4})$. If $\widehat{G} = 0$, i.e., there is no compression, then we need $T = O(K^{1/2})$, which is the same as the requirement in [Lian et al., 2015].

Remark 4. Note that when $M = N$ and $T = 0$, A-DS reduces to DS. In this case, the condition in (24) naturally holds, and our convergence rate becomes $O(\sqrt{\frac{1}{NK}})$ that will match the bound shown in [Tang et al., 2019].

5 EXPERIMENT RESULTS

We conduct numerical experiments on the proposed algorithm A-DS with comparing several state-of-the-art distributed training methods. We first compare A-DS to the synchronous DS and the non-compressing

PSGD and A-PSGD. Then we compare the results of A-DS with varying M.

All the experiments are about solving the image classification problem on data set Cifar10 by training a simplified Resnet. Cifar10 [Krizhevsky et al., 2009] includes 60000 training samples and 10000 testing samples, and each sample has 3072 features and belongs to one of the 10 classes. Resnet is a model proposed in [He et al., 2016] and the simplified Resnet only includes 9 convolutions talked in `https://myrtle.ai/learn/how-to-train-your-resnet/` and the corresponding code is referenced to `https://github.com/davidcpage/cifar10-fast`.

All the algorithms are implemented in Python with the package Pytorch for the tensor computing and the package MPI4PY for the communication between different learners. Specifically, in our code, the master posts "irecv" for all the workers, and at each interation the master receives M compressed stochastic gradients from the M workers and posts new "irecv" for the M workers. Thus in the code, we need to set $M \leq N$. And when $M = N$, A-DS becomes the synchronous DS.

For all the compressors Q_{ω_k} and $Q_{\omega_k^{(i)}}$ for $i \in 1, 2..., N$ in A-DS and DS, we take the elements in the input vector with the biggest 30% absolute values and set others as 0 i.e., Top-k(0.3) [Aji and Heafield, 2017]. And when we set the compressors Q_{ω_k} and $Q_{\omega_k^{(i)}}$ for $i \in 1, 2..., N$ as identical function, A-DS and DS reduce to A-PSGD and PSGD respectively.

All the experiments were run on the IBM DCS supercomputer, AiMOS where each node is equipped with two IBM Power 9 processors clocked at 3.15 GHz. Each processor contains 20 cores with 4 hardware threads. And each node can have at most six NVIDIA Tesla V100 GPUs with 32 GiB of memory each. We stop all experiments when finishing 50 epochs or arriving the the maximum running time 6 hours. We set the learning rate as the function of iteration number k. The learning rate is set as 0 at the beginning of the 1st epoch and the end of the 50th epoch, and is set as the maximum $\gamma_{\max}$ at the end of the 5th epoch. The learning rate at the other iterations is the linear interpolation of the learning rate at the above three iterations.

5.1 Comparison with State-of-the-art Methods

In this subsection, we compare training/testing accuracy and running time of A-DS with A-PSGD, DS and PSGD. We run the algorithms with five processes (one is the master and four are workers) on five AiMOS nodes and each with one GPU. In this setting, it takes much time to transfer the information between different nodes because of the big transferring cost. We set the maximum learning rate as $\gamma_{\max} = 0.5$ and the batch size as 640 for all algorithms. And for A-DS and A-PSGD, we set $M = 2$.

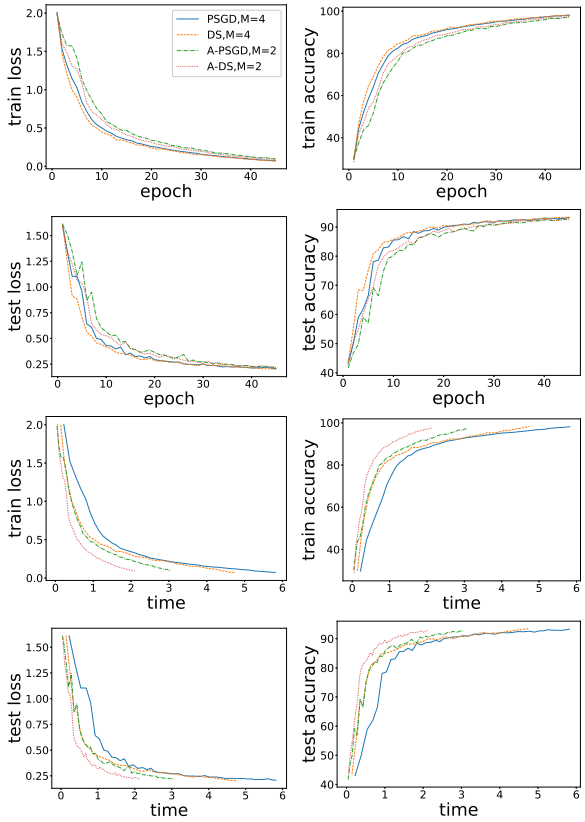

Figure 1: Results of PSGD, DS, A-PSGD, A-DS on five processes (one master and four workers).

When the algorithms stops, the four algorithms all achieve above 93% test accuracy. PSGD finished 45 epochs in the maximum running time 6 hours and other three algorithms finished 50 epochs. In Table 2, we gave the training and testing accuracy at the end of each algorithm and also the time for running 45 epochs. In Figure 1, we plot the results with respect to the epoch number and running time until finishing the 45th epoch. With respect to the epoch number, all the algorithms converge at a similar speed. PSGD and DS with $M = 4$ converge slightly faster then A-PSGD and A-DS with $M = 2$, which coincide with the asymptotical convergence rate $O(\sqrt{\frac{1}{MK}})$. But with respect to the running time, we can see the predominate advantages of the asynchrony and compressions over the synchronous and non-compressed algorithms. Because of the asynchrony, A-PSGD is faster than PSGD, and A-DS is faster than DS. Regarding the compressions, DS is faster than PSGD, and A-DS is faster than A-

Table 2: The results of PSGD, DS, A-PSGD, A-DS on five processes (one master and four workers). The training and testing accuracy (**Train/Test Acc**) on the finial model for each algorithm and the running time (**Run Time**) for 45 epochs.

Algorithm	**Train/Test Acc** (at the end)	**Run Time** (for 45 epochs)
PSGD	97.99/93.31	5.36
DS	98.71/93.49	4.74
A-PSGD	98.05/93.38	3.35
A-DS	97.86/93.25	2.12

PSGD. Among the four algorithms, it can be observed obviously that A-DS takes the least time to achieve the same accuracy.

5.2 Effect of M in A-DS

In this subsection, we compare the results of A-DS with different $M = 2, 4, 6$ and DS with $M = 8$ on nine processes (one master and eight workers). These nine processes are run on one AiMOS node with three GPUs. In this setting, it is very fast to transmit messages among the processes. The communication time for synchronous DS is mainly about the waiting for all workers finishing the communication with the master. The asynchronous A-DS just needs some workers finishing the communication in each iteration to update the model parameters. Here, M decides the levels of the asynchrony of A-DS. The smaller M is, the more asynchronous A-DS is. In the experiments, we set the maximum learning rate as $\gamma_{\max} = 0.1$ and the batch size as 128.

Table 3: The training and testing accuracy (**Train/Test Acc**), and the running time (**Run Time**) for 50 epochs of A-DS on nine processes (one master and eight workers) with different M.

Algorithm	M	**Train/Test Acc**	**Run Time**
A-DS	2	98.56/93.52	2.85
A-DS	4	98.05/93.38	3.75
A-DS	6	97.91/93.44	4.92
DS	8	97.99/93.31	5.96

All algorithms finished 50 epochs within the maximum running time. In Table 3, we provide the training and testing accuracy and also the running time for 50 epochs. In Figure 2, we plot the results with respect to the epoch number and running time for 50 epochs. With respect to the epoch number, we can

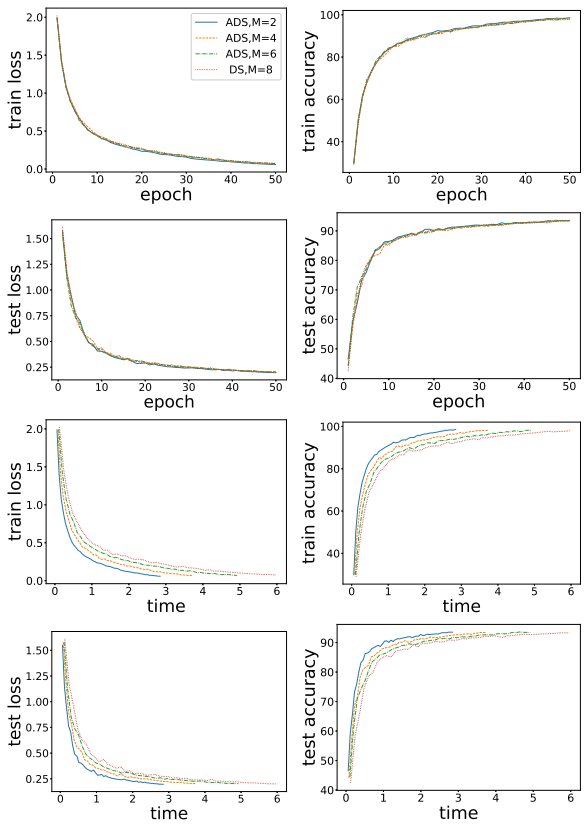

Figure 2: Results of A-DS on nine processes (one master and eight workers) with $M = 2, 4, 6, 8$.

see that all the algorithms still converge at a more or less the same rate for different M. But with respect to the running time, we can find the efficiency of the asynchrony. The smaller M is, the more asynchronous the algorithm is and the less time is consumed, implying that the asynchronous communication protocol is more practical without loss of any optimality in terms of both iteration complexity and training/testing accuracy.

6 CONCLUDING REMARK

In this paper, we studied an asynchronous distributed stochastic gradient algorithm with Double-pass gradient compression, namely A-DS. The integration of asynchrony and model compressions makes A-DS converges faster then other algorithms in the centralized stochastic parallel computing architecture. We also provide the theoretical convergence guarantees of A-DS to the first-order stationary points of general nonconvex training problems. If the maximum staleness T is bounded by $T = O(K^{1/4})$, it is shown that A-DS is able to achieve the same asymptotical convergence rate as A-PSGD. Multiple numerical experiments on real datasets with multiple GPUs showcase the strength of asynchrony and model compressions in training neural networks.

References

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

[Aji and Heafield, 2017] Aji, A. F. and Heafield, K. (2017). Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*.

[Alistarh et al., 2017] Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. (2017). Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720.

[Bernstein et al., 2018] Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. (2018). signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*.

[Dutta et al., 2019] Dutta, A., Bergou, E. H., Abdelmoniem, A. M., Ho, C.-Y., Sahu, A. N., Canini, M., and Kalnis, P. (2019). On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. *arXiv preprint arXiv:1911.08250*.

[Ghadimi et al., 2016] Ghadimi, S., Lan, G., and Zhang, H. (2016). Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[Jin et al., 2016] Jin, P. H., Yuan, Q., Iandola, F., and Keutzer, K. (2016). How to scale distributed deep learning? *arXiv preprint arXiv:1611.04581*.

[Koloskova et al., 2019a] Koloskova, A., Lin, T., Stich, S. U., and Jaggi, M. (2019a). Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*.

[Koloskova et al., 2019b] Koloskova, A., Stich, S. U., and Jaggi, M. (2019b). Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*.

[Krizhevsky et al., 2009] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[Lian et al., 2015] Lian, X., Huang, Y., Li, Y., and Liu, J. (2015). Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745.

[Lian et al., 2017] Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. (2017). Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340.

[Mishchenko et al., 2019] Mishchenko, K., Gorbunov, E., Takáč, M., and Richtárik, P. (2019). Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*.

[Seide and Agarwal, 2016] Seide, F. and Agarwal, A. (2016). Cntk: Microsoft's open-source deep-learning toolkit. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135.

[Seide et al., 2014] Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. (2014). 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Annual Conference of the International Speech Communication Association*.

[Shen et al., 2018] Shen, Z., Mokhtari, A., Zhou, T., Zhao, P., and Qian, H. (2018). Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication. *arXiv preprint arXiv:1805.09969*.

[Stich et al., 2018] Stich, S. U., Cordonnier, J.-B., and Jaggi, M. (2018). Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pages 4447–4458.

[Tang et al., 2018] Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. (2018). D 2: Decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*.

[Tang et al., 2019] Tang, H., Yu, C., Lian, X., Zhang, T., and Liu, J. (2019). Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *International Conference on Machine Learning*, pages 6155–6165. PMLR.

[Tsitsiklis et al., 1986] Tsitsiklis, J., Bertsekas, D., and Athans, M. (1986). Distributed asynchronous

deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812.

[Wen et al., 2017] Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. (2017). Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in Neural Information Processing Systems*, pages 1509–1519.

[Xu et al., 2020] Xu, H., Ho, C.-Y., Abdelmoniem, A. M., Dutta, A., Bergou, E. H., Karatsenidis, K., Canini, M., and Kalnis, P. (2020). Compressed Communication for Distributed Deep Learning: Survey and Quantitative Evaluation. Technical report, KAUST. http://hdl.handle.net/10754/662495.

[Zhou et al., 2018] Zhou, Z., Mertikopoulos, P., Bambos, N., Glynn, P. W., Ye, Y., Li, L.-J., and Li, F.-F. (2018). Distributed asynchronous optimization with unbounded delays: How slow can you go? In *International Conference on Machine Learning*.

Supplementary Materials: Asynchronous Distributed Stochastic Optimization with Double-pass Gradient Compression

A NUMERICAL EXPERIMENTS ON CIFAR100

We conduct all numerical experiments in Section 5 again with data set Cifar100. The data size of Cifar100 is the same as Cifar10 except each sample belongs to one of the 100 classes. The model is still the simplified Resnet except the output size is changed from 10 to 100. We also use Top-k(0.3) for all compressors Q_{ω_k} and $Q_{\omega_k^{(i)}}$ for $i \in 1, 2..., N$ in A-DS and DS.

We stop all experiments when finishing 60 epochs or arriving at the maximum running time 6 hours. We set the learning rate as the function of iteration number k. A little different from that in Section 5. The learning rate γ_k is the bigger one of $\gamma_{\min}$ and $\tilde{\gamma}_k$. Here, $\gamma_{\min}$ is a constant and we take $\gamma_{\min} = 0.001$. $\tilde{\gamma}_k$ is the linear interpolation at four iterations: $\tilde{\gamma}_k = 0, \gamma_{\max}, 0, 0$, respectively, at the beginning of the 1st epoch, the end of the 10th, 50th and 60th epochs. The constant $\gamma_{\max}$ is tuned for different batch sizes. Notice we stop the learning rate decreasing to 0 by setting the minimum learning rate $\gamma_{\min}$. $\gamma_{\min}$ is used at least for the iterations in the last 10 epochs. This setting helps us see the convergence, i.e., the flat parts of the curves in the later figures.

A.1 Comparison with State-of-the-art Methods

In this section, we compare training/testing accuracy and running time of A-DS with A-PSGD, DS and PSGD on Cifar100. We run the algorithms with five processes (one is the master and four are workers) on five AiMOS nodes and each with one GPU. We set the maximum learning rate as $\gamma_{\max} = 0.5$ and the batch size as 640 for all algorithms. For A-DS and A-PSGD, we set $M = 2$.

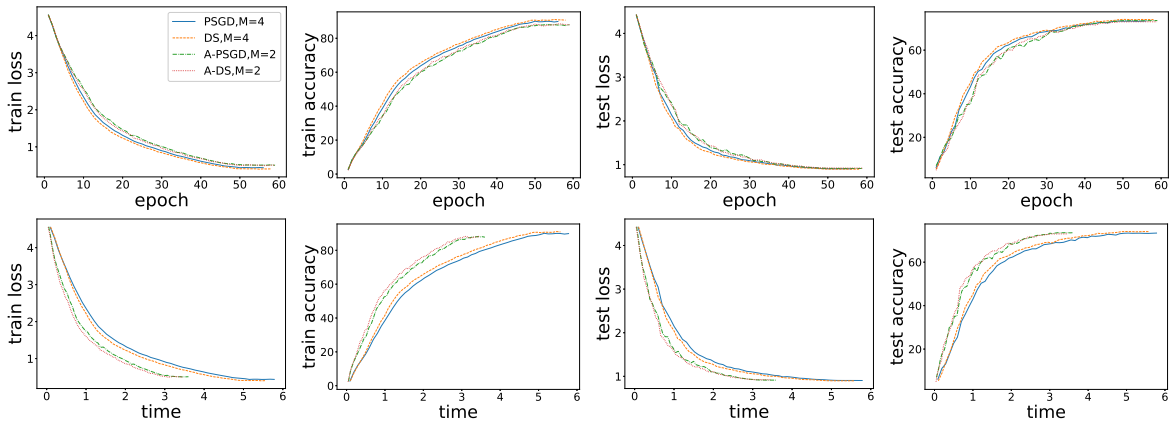

Figure 3: Results on Cifar100 by PSGD, DS, A-PSGD, A-DS on five processes (one master and four workers).

When the four algorithms stop, they all achieve above 73% test accuracy. Within the maximum running time 6 hours, PSGD, DS, A-PSGD and A-DS finished $57, 59, 60, 60$ epochs, respectively. In Table 4, we give the results at the end of each algorithm and after 57 epochs. In Figure 3, we plot the results to the epoch number until the end of each algorithm and to the running time until finishing the 57 epochs. We can get the similar conclusion as the results of Cifar10: among the four algorithms, A-DS takes the least time to achieve the same accuracy because of the asynchrony and the double-pass compression. Also, we can see that the curves are almost flat after 50 epochs, which means the convergence of the algorithms.

Table 4: Results on Cifar100 by PSGD, DS, A-PSGD, A-DS on five processes (one master and four workers). At the finial model, the finished epochs numbers (**Epochs Num**), training and testing accuracy (**Train/Test Acc**). After 57 epochs, the training and testing accuracy (**Train/Test Acc**) and the running time (**Run Time**).

Algorithm	At the final model		After 57 epochs	
	Epochs	**Train/Test Acc**	**Train/Test Acc**	**Run Time**
PSGD	57	89.89/73.39	89.81/73.29	5.89
DS	59	90.97/74.06	90.86/74.02	5.67
A-PSGD	60	87.47/73.55	87.83/73.55	3.67
A-DS	60	87.99/73.02	88.42/73.04	3.52

A.2 Effect of M in A-DS

In this subsection, we compare the results of A-DS with different $M = 2, 4, 6$ and DS with $M = 8$ on Cifar100. These algorithms are run on nine processes (one master and eight workers). These nine processes are run on one AiMOS node with three GPUs. In the experiments, we set the maximum learning rate as $\gamma_{\max} = 0.1$ and the batch size as 128.

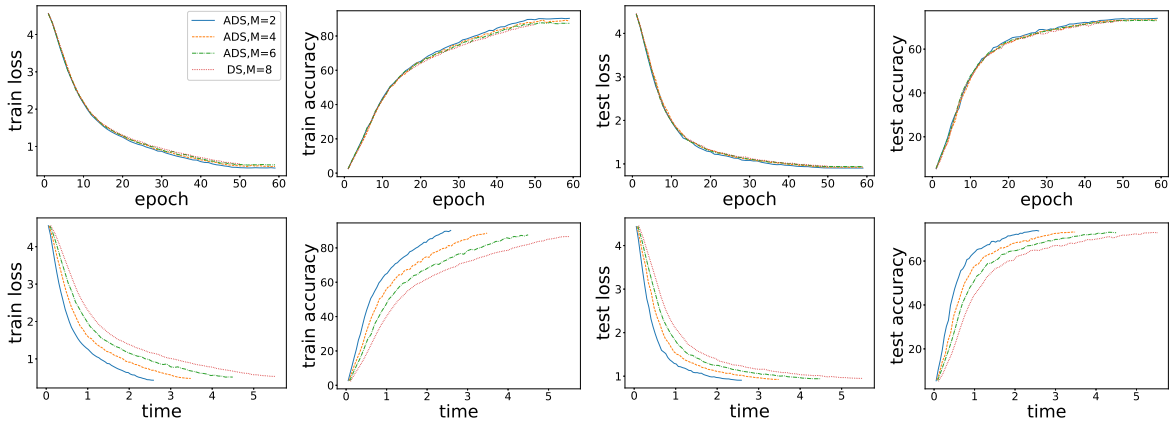

Figure 4: Results on Cifar100 by A-DS on nine processes (one master and eight workers) with $M = 2, 4, 6, 8$.

Table 5: Results on Cifar100 by A-DS on nine processes (one master and eight workers) with different M. At the finial model, the finished epochs numbers (**Epochs**), training and testing accuracy (**Train/Test Acc**). After 51 epochs, the training and testing accuracy (**Train/Test Acc**) and the running time (**Run Time**)

Algorithm	**M**	At the final model		After 51 epochs	
		Epochs	**Train/Test Acc**	**Train/Test Acc**	**Run Time**
A-DS	2	60	90.02/73.90	90.14/73.70	2.58
A-DS	4	60	89.11/73.34	88.58/73.18	3.48
A-DS	6	60	86.96/73.11	87.61/72.88	4.50
DS	8	51	86.57/72.89	86.57/72.89	5.51

When the algorithms stop within the maximum running time, A-DS with $M = 2, 4, 6$ all finish 60 epochs while DS with $M = 8$ only finishes 51 epochs. In Table 5, we give the results at the end of each algorithm and after 51 epochs. In Figure 4, we plot the results to the epoch number until the end of each algorithm and to the running time until finishing 51 epochs. All algorithms converge at a more or less the same rate for different M with respect to the epoch number. With respect to the running time, the efficiency of the asynchrony can be shown. The smaller M is, the more asynchronous the algorithm is and the less time is consumed.

B THEORETICAL PROOF

B.1 Proof of Lemma 1

Proof. The update rule (11) can be obtained according to updates (2)-(10) in Algorithm 1. The details are in the following. Replacing $\widehat{\Delta}_k$ in (10) by (9) and $\mathbf{v}_k$ by (7) gives $\mathbf{x}_{k+1}-\mathbf{x}_k=-\gamma_k(\mathbf{v}_k-\delta_k)=-\gamma_k(\delta_{k-1}+\Delta_k-\delta_k)$. Then by (6), we have $\mathbf{x}_{k+1}-\mathbf{x}_k=-\gamma_k(\delta_{k-1}+\frac{1}{M}\sum_{m=1}^M \widehat{\mathbf{g}}_k^{(i_m)}-\delta_k)$. Further replacing $\widehat{\mathbf{g}}_k^{(i_m)}$ by (5), $\mathbf{v}_k^{(i_m)}$ by (3) and $\mathbf{g}_k^{(i_m)}$ by (2) gives

$$\mathbf{x}_{k+1}-\mathbf{x}_k=\gamma_k\delta_k-\gamma_k\delta_{k-1}-\gamma_k\frac{1}{M}\sum_{m=1}^M\left(\mathbf{v}_k^{(i_m)}-\delta_k^{(i_m)}\right)=\gamma_k\delta_k-\gamma_k\delta_{k-1}-\gamma_k\frac{1}{M}\sum_{m=1}^M\left(\delta_{k-1}^{(i_m)}+\mathbf{g}_k^{(i_m)}-\delta_k^{(i_m)}\right)$$

$$=\gamma_k\delta_k-\gamma_k\delta_{k-1}-\gamma_k\frac{1}{M}\sum_{m=1}^M\left(\delta_{k-1}^{(i_m)}-\delta_k^{(i_m)}\right)-\gamma_k\frac{1}{M}\sum_{m=1}^M\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}},\xi_k^{(i_m)}).$$

Then with the notations in (12) and (13), we finish the proof. □

B.2 Proof of Lemma 2

Proof. Firstly, we prove the bound in (17) of $\mathbb{E}[\|\mathbf{g}_k^{(i)}\|^2]$. By (2), for any k and $i\in\{i_m\}_{m=1}^M$,

$$\|\mathbf{g}_k^{(i)}\|^2=\|\nabla F(\mathbf{x}_{k-\tau_k^{(i)}},\xi_k^{(i)})-\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})+\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\|^2$$
$$=\|\nabla F(\mathbf{x}_{k-\tau_k^{(i)}},\xi_k^{(i)})-\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\|^2+\|\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\|^2+2\langle\nabla F(\mathbf{x}_{k-\tau_k^{(i)}},\xi_k^{(i)})-\nabla f(\mathbf{x}_{k-\tau_k^{(i)}}),\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\rangle.$$

Then take the expectation with respect to the random samples $\xi_k^{(i)}$. Note $\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})$ does not depend on $\xi_k^{(i)}$.

$$\begin{aligned}\mathbb{E}_{\xi_k^{(i)}}[\|\mathbf{g}_k^{(i)}\|^2]=&\mathbb{E}_{\xi_k^{(i)}}\left[\|\nabla F(\mathbf{x}_{k-\tau_k^{(i)}},\xi_k^{(i)})-\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\|^2\right]+\|\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\|^2\\&+2\left\langle\mathbb{E}_{\xi_k^{(i)}}\left[\nabla F(\mathbf{x}_{k-\tau_k^{(i)}},\xi_k^{(i)})-\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\right],\nabla f(\mathbf{x}_{k-\tau_k^{(i)}})\right\rangle\\\le&\sigma^2+G^2,\end{aligned}$$

where Assumptions 2 and 4 are used in the inequality. Taking the full expectation gives (17).

Secondly, let us prove (18) about η_k. From (13) and the unbiased stochastic gradient, we have

$$\mathbb{E}_{\xi_k}[\eta_k]=\frac{1}{M}\sum_{m=1}^M\left(\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})-\mathbb{E}_{\xi_k^{(i_m)}}[\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}},\xi_k^{(i_m)})]\right)=\mathbf{0}.$$

We can bound $\mathbb{E}_{\xi_k}[\|\eta_k\|^2]$ by the independence between random samples $\{\xi_k^{(i)}\}$ and the variance boundness of the stochastic gradient.

$$\mathbb{E}_{\xi_k}[\|\eta_k\|^2]=\frac{1}{M^2}\mathbb{E}_{\xi_k}\left[\|\sum_{m=1}^M[\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})-\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}},\xi_k^{(i_m)})]\|^2\right]$$
$$=\frac{1}{M^2}\sum_{m=1}^M\mathbb{E}_{\xi_k^{(i_m)}}\left[\|\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})-\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}},\xi_k^{(i_m)})\|^2\right]$$
$$+\frac{1}{M^2}\sum_{m\neq m'}^M\left\langle\mathbb{E}_{\xi_k^{(i_m)}}[\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})-\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}},\xi_k^{(i_m)})],\mathbb{E}_{\xi_k^{(i_{m'})}}\left[\nabla f(\mathbf{x}_{k-\tau_k^{(i_{m'})}})-\nabla F(\mathbf{x}_{k-\tau_k^{(i_{m'})}},\xi_k^{(i_{m'})})\right]\right\rangle$$
$$=\frac{1}{M^2}\sum_{m=1}^M\mathbb{E}_{\xi_k^{(i_m)}}\left[\|\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})-\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}},\xi_k^{(i_m)})\|^2\right]\le\frac{1}{M^2}\sum_{m=1}^M\sigma^2=\frac{\sigma^2}{M}.$$

□

B.3 Proof of Lemma 3

Proof. From the definition of Ω_k in (12), let us bound $\delta_k^{(i)}$ and δ_k in order to bound Ω_k.

At each update, M workers are selected. At the k-th update, if worker i is selected as one of the M workers, then $\delta_k^{(i)}$ is renewed following (2)-(5). In this case, we have

$$\mathbb{E}[\|\delta_k^{(i)}\|^2] = \mathbb{E}[\mathbb{E}_{\omega_k^{(i)}}[\|\delta_k^{(i)}\|^2]] = \mathbb{E}[\mathbb{E}_{\omega_k^{(i)}}[\|\mathbf{v}_k^{(i)} - Q_{\omega_k^{(i)}}[\mathbf{v}_k^{(i)}]\|^2]] \leq \alpha^2\mathbb{E}[\|\mathbf{v}_k^{(i)}\|^2] = \alpha^2\mathbb{E}[\|\delta_{k-1}^{(i)} + \mathbf{g}_k^{(i)}\|^2], \tag{25}$$

where the inequality holds by Assumption 5. With the Cauchy-Schwarz inequality, we have

$$\mathbb{E}[\|\delta_k^{(i)}\|^2] \leq \alpha^2\mathbb{E}[\|\mathbf{v}_k^{(i)}\|^2] \leq \alpha^2(1+\beta)\mathbb{E}[\|\delta_{k-1}^{(i)}\|^2]+\alpha^2(1+\frac{1}{\beta})\mathbb{E}[\|\mathbf{g}_k^{(i)}\|^2] \leq \alpha^2(1+\beta)\mathbb{E}[\|\delta_{k-1}^{(i)}\|^2]+\alpha^2(1+\frac{1}{\beta})(\sigma^2+G^2), \tag{26}$$

where β can be any positive constant and we use Lemma 2 to $\mathbb{E}[\|\mathbf{g}_k^{(i)}\|^2]$. However, if worker i is not selected as one of the M workers at the k-th update, $\mathbf{v}_k^{(i)} = \mathbf{v}_{k-1}^{(i)}$ and $\delta_k^{(i)} = \delta_{k-1}^{(i)}$.

Without loss of generality, we suppose worker i is selected a times (at the updates $\{k_1, k_2, ..., k_{a-1}, k_a\}$ which satisfy $0 = k_0 < k_1 < k_2 < ... < k_{a-1} < k_a \leq k$) until the k-th update. Then by (26), we have

$$\begin{aligned}
&\mathbb{E}[\|\delta_k^{(i)}\|^2] = \mathbb{E}[\|\delta_{k_a}^{(i)}\|^2] \leq \alpha^2\mathbb{E}[\|\mathbf{v}_k^{(i)}\|^2] = \alpha^2\mathbb{E}[\|\mathbf{v}_{k_a}^{(i)}\|^2] \\
\leq&\alpha^2(1+\beta)\mathbb{E}[\|\delta_{k_a-1}^{(i)}\|^2] + \alpha^2(1+\frac{1}{\beta})(\sigma^2+G^2) = \alpha^2(1+\beta)\mathbb{E}[\|\delta_{k_a-1}^{(i)}\|^2] + \alpha^2(1+\frac{1}{\beta})(\sigma^2+G^2).
\end{aligned} \tag{27}$$

Further applying (27) to $\mathbb{E}[\|\delta_{k_{a-1}}^{(i)}\|^2]$, we can recursively get

$$\begin{aligned}
\mathbb{E}[\|\delta_k^{(i)}\|^2] \leq \alpha^2\mathbb{E}[\|\mathbf{v}_k^{(i)}\|^2] \leq&(\alpha^2(1+\beta))^2\mathbb{E}[\|\delta_{k_{a-2}}^{(i)}\|^2] + \Big(\alpha^2(1+\beta)+1\Big)\alpha^2(1+\frac{1}{\beta})(\sigma^2+G^2) \\
\leq&(\alpha^2(1+\beta))^a\mathbb{E}[\|\delta_0^{(i)}\|^2] + \Big(\sum_{j=0}^{a-1}(\alpha^2(1+\beta))^j\Big)\alpha^2(1+\frac{1}{\beta})(\sigma^2+G^2) \\
\leq&\frac{1}{1-\alpha^2(1+\beta)}\alpha^2(1+\frac{1}{\beta})(\sigma^2+G^2),
\end{aligned} \tag{28}$$

where we use $\delta_0^{(i)} = \mathbf{0}$ and $\beta \in (0, \frac{1}{\alpha^2} - 1)$. From (28), we also have

$$\mathbb{E}[\|\mathbf{v}_k^{(i)}\|^2] \leq \frac{1}{1-\alpha^2(1+\beta)}(1+\frac{1}{\beta})(\sigma^2+G^2) := \widetilde{G}^2.$$

Now, let bound δ_k given in (9). While bounding $\delta_k^{(i)}$, we use the bound of $\mathbf{g}_k^{(i)}$. Here, in order to bound δ_k, we need the bound of Δ_k in (6) which can be gotten by

$$\mathbb{E}[\|\Delta_k\|^2] = \mathbb{E}\left[\left\|\frac{1}{M}\sum_{m=1}^{M}\widehat{\mathbf{g}}_k^{(i_m)}\right\|^2\right] \leq \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[\|\widehat{\mathbf{g}}_k^{(i_m)}\|^2] = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[\|Q_{\omega_k^{(i_m)}}[\mathbf{v}_k^{(i_m)}]\|^2] \leq \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[\|\mathbf{v}_k^{(i_m)}\|^2] \leq \widetilde{G}^2,$$

where the first inequality holds by the Cauchy-Schwarz inequality and the second inequality holds by (14) in Assumption 5. With the above bound of $\mathbb{E}[\|\Delta_k\|^2]$, we can bound δ_k by the similar way of bounding $\delta_k^{(i)}$.

$$\begin{aligned}
\mathbb{E}[\|\delta_k\|^2] =&\mathbb{E}[\mathbb{E}_{\omega_k}[\|\delta_k\|^2]] = \mathbb{E}[\mathbb{E}_{\omega_k}[\|\mathbf{v}_k - Q_{\omega_k}[\mathbf{v}_k]\|^2]] \leq \alpha^2\mathbb{E}[\|\mathbf{v}_k\|^2] = \alpha^2\mathbb{E}[\|\delta_{k-1}+\Delta_k\|^2] \\
\leq&\alpha^2(1+\beta)\mathbb{E}[\|\delta_{k-1}\|^2] + \alpha^2(1+\frac{1}{\beta})\mathbb{E}[\|\Delta_k\|^2] \leq \alpha^2(1+\beta)\mathbb{E}[\|\delta_{k-1}\|^2] + \alpha^2(1+\frac{1}{\beta})\widetilde{G}^2 \\
\leq&(\alpha^2(1+\beta))^k\mathbb{E}[\|\delta_0\|^2] + \Big(\sum_{j=0}^{k-1}(\alpha^2(1+\beta))^j\Big)\alpha^2(1+\frac{1}{\beta})\widetilde{G}^2 \leq \frac{1}{1-\alpha^2(1+\beta)}\alpha^2(1+\frac{1}{\beta})\widetilde{G}^2.
\end{aligned}$$

Finally, we can bound Ω_k in (12) with the bounds of $\delta_k^{(i)}$ and δ_k. Using the Cauchy-Schwarz inequality twice gives

$$\mathbb{E}[\|\Omega_k\|^2] = \mathbb{E}[\|\delta_k + \frac{1}{M}\sum_{m=1}^{M}\delta_k^{(i_m)}\|^2] \le 2\mathbb{E}[\|\delta_k\|^2] + 2\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\delta_k^{(i_m)}\|^2]$$
$$\le 2\mathbb{E}[\|\delta_k\|^2] + 2\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[\|\delta_k^{(i_m)}\|^2] \le \frac{2\alpha^2(1+\frac{1}{\beta})}{1-\alpha^2(1+\beta)}(\sigma^2 + G^2 + \widetilde{G}^2).$$

The proof is finished. □

B.4 Proof of Theorem 4

Proof. The theorem is proved with an auxiliary sequence $\{\mathbf{y}_k\}_{k\ge 0}$ defined as

$$\mathbf{y}_k = \mathbf{x}_k - \gamma_k \Omega_{k-1}. \tag{29}$$

From (11), the update of $\mathbf{y}_k$ satisfies

$$\mathbf{y}_{k+1} - \mathbf{y}_k = -\gamma_k \frac{1}{M}\sum_{m=1}^{M}\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}}, \xi_k^{(i_m)}).$$

Now we prove the theorem from the auxiliary sequence. By Assumption 1, we have

$$f(\mathbf{y}_{k+1}) - f(\mathbf{y}_k) \le \langle \nabla f(\mathbf{y}_k), \mathbf{y}_{k+1} - \mathbf{y}_k\rangle + \frac{L}{2}\|\mathbf{y}_{k+1} - \mathbf{y}_k\|^2$$
$$= \Big\langle \nabla f(\mathbf{y}_k), -\gamma_k \frac{1}{M}\sum_{m=1}^{M}\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}}, \xi_k^{(i_m)})\Big\rangle + \frac{L}{2}\| - \gamma_k \frac{1}{M}\sum_{m=1}^{M}\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}}, \xi_k^{(i_m)})\|^2.$$

Taking the expectation with respect to ξ_k on both sides of the above inequality gives

$$\mathbb{E}_{\xi_k}[f(\mathbf{y}_{k+1}) - f(\mathbf{y}_k)] \le -\gamma_k \Big\langle \nabla f(\mathbf{y}_k), \frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\Big\rangle + \frac{L\gamma_k^2}{2}\mathbb{E}_{\xi_k}\Big[\|\frac{1}{M}\sum_{m=1}^{M}\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}}, \xi_k^{(i_m)})\|^2\Big], \tag{30}$$

where we use the independence of $\nabla f(\mathbf{y}_k)$ on ξ_k and Assumption 4.

The seond term in the right side of (30) could be rewritten with (13) and bounded by Lemma 2.

$$\mathbb{E}_{\xi_k}\Big[\|\frac{1}{M}\sum_{m=1}^{M}\nabla F(\mathbf{x}_{k-\tau_k^{(i_m)}}, \xi_k^{(i_m)})\|^2\Big] = \mathbb{E}_{\xi_k}\Big[\|\eta_k - \frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2\Big]$$
$$= \mathbb{E}_{\xi_k}[\|\eta_k\|^2] + \|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2 - 2\Big\langle \mathbb{E}_{\xi_k}[\eta_k], \frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\Big\rangle$$
$$\le \frac{\sigma^2}{M} + \|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2. \tag{31}$$

About the first term in the right side of (30), we have

$$
\begin{aligned}
&-\gamma_k\Big\langle\nabla f(\mathbf{y}_k),\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\Big\rangle\\
=&-\gamma_k\Big\langle\nabla f(\mathbf{y}_k)-\nabla f(\mathbf{x}_k),\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\Big\rangle-\gamma_k\Big\langle\nabla f(\mathbf{x}_k),\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\Big\rangle\\
\le&\gamma_k\Big(\|\nabla f(\mathbf{y}_k)-\nabla f(\mathbf{x}_k)\|^2+\frac{1}{4}\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2\Big)\\
&-\frac{\gamma_k}{2}\Big(\|\nabla f(\mathbf{x}_k)\|^2+\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2-\|\nabla f(\mathbf{x}_k)-\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2\Big)\\
=&\gamma_k\|\nabla f(\mathbf{y}_k)-\nabla f(\mathbf{x}_k)\|^2-\frac{\gamma_k}{4}\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2-\frac{\gamma_k}{2}\|\nabla f(\mathbf{x}_k)\|^2+\frac{\gamma_k}{2}\|\nabla f(\mathbf{x}_k)-\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2.
\end{aligned}
\tag{32}
$$

From (29) and Assumption 1 , the first term in the right side of (32) can be bounded by $\|\Omega_{k-1}\|^2$.

$$
\|\nabla f(\mathbf{x}_k)-\nabla f(\mathbf{y}_k)\|^2\le L^2\|\mathbf{x}_k-\mathbf{y}_k\|^2=L^2\|\gamma_k\Omega_{k-1}\|^2=\gamma_k^2L^2\|\Omega_{k-1}\|^2. \tag{33}
$$

Next, consider the last term in the right side of (32). Denote $T_1:=\|\nabla f(\mathbf{x}_k)-\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2$.

$$
\begin{aligned}
T_1=&\frac{1}{M^2}\|\sum_{m=1}^{M}(\nabla f(\mathbf{x}_k)-\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}}))\|^2\le\frac{1}{M}\sum_{m=1}^{M}\|\nabla f(\mathbf{x}_k)-\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2\\
\le&\frac{L^2}{M}\sum_{m=1}^{M}\|\mathbf{x}_k-\mathbf{x}_{k-\tau_k^{(i_m)}}\|^2\le L^2\max_{m\in\{1,2,\ldots,M\}}\|\mathbf{x}_k-\mathbf{x}_{k-\tau_k^{(i_m)}}\|^2=L^2\|\mathbf{x}_k-\mathbf{x}_{k-\tau_k^{(i_{m^*})}}\|^2,
\end{aligned}
$$

where the first inequality holds by the Cauchy-Schwarz inequality and the second inequality holds by Assumption 1 and $m^*=\mathrm{argmax}_{m\in\{1,2,\ldots,M\}}\|\mathbf{x}_k-\mathbf{x}_{k-\tau_k^{(i_m)}}\|^2$. Then we rewrite $\mathbf{x}_k-\mathbf{x}_{k-\tau_k^{(i_m)}}$ as sum of $\mathbf{x}_k-\mathbf{x}_{k-1}$ and plug (11).

$$
\begin{aligned}
T_1\le&L^2\|\sum_{j=k-\tau_k^{m^*}}^{k-1}(\mathbf{x}_{j+1}-\mathbf{x}_j)\|^2=L^2\|\sum_{j=k-\tau_k^{m^*}}^{k-1}(-\gamma_j\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})+\gamma_j\eta_j-\gamma_j\Omega_{j-1}+\gamma_j\Omega_j)\|^2\\
\le&4L^2\Bigg(\|\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})\|^2+\|\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j\eta_j\|^2+\|\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j\Omega_{j-1}\|^2+\|\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j\Omega_j\|^2\Bigg),
\end{aligned}
$$

where the Cauchy-Schwarz inequality is used. Then we expand the second term in the right side of the above inequality and apply the Cauchy-Schwarz inequality again to the first, third and fourth terms.

$$
\begin{aligned}
T_1\le&4L^2\Bigg(T\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\gamma_j\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})\|^2+\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\gamma_j\eta_j\|^2+\sum_{j\neq j'}\langle\gamma_j\eta_j,\gamma_{j'}\eta_{j'}\rangle\\
&+(\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j^2)(\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\Omega_{j-1}\|^2)+(\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j^2)(\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\Omega_j\|^2)\Bigg),
\end{aligned}
\tag{34}
$$

where we suppose $\gamma_j=0$, $\tau_j^{(i_m)}=0$ and $\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})=\mathbf{0}$ for all $j\le 0$.

Plug (33) and (34) back to (32), then (31) and (32) back to (30).

$$\begin{aligned}\mathbb{E}_{\xi_k}[f(\mathbf{y}_{k+1})-f(\mathbf{y}_k)] \le&\gamma_k^3L^2\|\Omega_{k-1}\|^2-(\frac{\gamma_k}{4}-\frac{L\gamma_k^2}{2})\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2-\frac{\gamma_k}{2}\|\nabla f(\mathbf{x}_k)\|^2+\frac{L\gamma_k^2}{2}\frac{\sigma^2}{M}\\ &+2\gamma_kL^2\bigg(T\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\gamma_j\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})\|^2+\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\gamma_j\eta_j\|^2+\sum_{j\neq j'}\langle\gamma_j\eta_j,\gamma_{j'}\eta_{j'}\rangle\\ &+(\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j^2)(\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\Omega_{j-1}\|^2)+(\sum_{j=k-\tau_k^{m^*}}^{k-1}\gamma_j^2)(\sum_{j=k-\tau_k^{m^*}}^{k-1}\|\Omega_j\|^2)\bigg).\end{aligned}$$

Then take full expectation. By Lemma 2 and Lemma 3, we have

$$\begin{aligned}&\mathbb{E}[f(\mathbf{y}_{k+1})-f(\mathbf{y}_k)]\\ \le&2\gamma_k^3L^2\widehat{G}^2-\frac{\gamma_k}{2}\mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2]-(\frac{\gamma_k}{4}-\frac{L\gamma_k^2}{2})\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2]+\frac{L\gamma_k^2}{2}\frac{\sigma^2}{M}\\ &+2\gamma_kL^2\Big(T\sum_{j=k-T}^{k-1}\gamma_j^2\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})\|^2]+(\frac{\sigma^2}{M}+4T\widehat{G}^2)\sum_{j=k-T}^{k-1}\gamma_j^2\Big)\\ =&-\frac{\gamma_k}{2}\mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2]-(\frac{\gamma_k}{4}-\frac{L\gamma_k^2}{2})\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2]+(\frac{L\gamma_k^2}{2}+2\gamma_kL^2\sum_{j=k-T}^{k-1}\gamma_j^2)\frac{\sigma^2}{M}\\ &+2\gamma_kL^2T\sum_{j=k-T}^{k-1}\gamma_j^2\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})\|^2]+(2\gamma_k^3L^2+8\gamma_kL^2T\sum_{j=k-T}^{k-1}\gamma_j^2)\widehat{G}^2.\end{aligned}$$

Summarizing the above inequality from $k=1$ to $k=K$ gives

$$\begin{aligned}&\mathbb{E}[f(\mathbf{y}_{K+1})-f(\mathbf{y}_1)]\\ \le&-\sum_{k=1}^{K}\frac{\gamma_k}{2}\mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2]-\sum_{k=1}^{K}(\frac{\gamma_k}{4}-\frac{L\gamma_k^2}{2})\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2]+\sum_{k=1}^{K}(\frac{L\gamma_k^2}{2}+2\gamma_kL^2\sum_{j=k-T}^{k-1}\gamma_j^2)\frac{\sigma^2}{M}\\ &+\sum_{k=1}^{K}2\gamma_kL^2T\sum_{j=k-T}^{k-1}\gamma_j^2\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})\|^2]+\sum_{k=1}^{K}(2\gamma_k^3L^2+8\gamma_kL^2T\sum_{j=k-T}^{k-1}\gamma_j^2)\widehat{G}^2.\end{aligned}$$

Exchange the order of summations in the fourth term of the right side of the above inequality.

$$\sum_{k=1}^{K}2\gamma_kL^2T\sum_{j=k-T}^{k-1}\gamma_j^2\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{j-\tau_j^{(i_m)}})\|^2]\le\sum_{k=1}^{K}(2L^2T\gamma_k^2\sum_{l=1}^{T}\gamma_{k+l})\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}\nabla f(\mathbf{x}_{k-\tau_k^{(i_m)}})\|^2],$$

where each element of the summations is nonnegative, and $\nabla f(\mathbf{x}_j)=\mathbf{0},\forall j\le 0$. Thus by (21) we have

$$\begin{aligned}&\mathbb{E}[f(\mathbf{y}_{K+1})-f(\mathbf{y}_1)]\\ \le&-\sum_{k=1}^{K}\frac{\gamma_k}{2}\mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2]+\sum_{k=1}^{K}(\frac{L\gamma_k^2}{2}+2\gamma_kL^2\sum_{j=k-T}^{k-1}\gamma_j^2)\frac{\sigma^2}{M}+\sum_{k=1}^{K}(2\gamma_k^3L^2+8\gamma_kL^2T\sum_{j=k-T}^{k-1}\gamma_j^2)\widehat{G}^2.\end{aligned}$$

Note $\mathbf{y}_1=\mathbf{x}_1$ and the global optimization solution $\mathbf{x}^*$ satisfies $f(\mathbf{x}^*)\le f(\mathbf{y}_{K+1})$. Rearranging the above inequality gives (22). □

B.5 Proof of Corollary 5

Proof. Let first verify the condition (21) in Theorem 4. From (24), the constant γ in (23) satisfies

$$\gamma\le\frac{\sigma^2}{4LT\sigma^2+M(4L+16LT^2)\widehat{G}^2}. \tag{35}$$

Here, $\gamma \leq \frac{1}{4LT}$ holds and the condition (21) holds for any $T \geq 1$, i.e., $\frac{1}{4} - \frac{L\gamma}{2} - 2L^2T^2\gamma^2 \geq \frac{1}{4} - \frac{1}{8T} - \frac{1}{8} = \frac{1}{8} - \frac{1}{8T} \geq 0$. With the constant γ given in (23), the conclusion (22) in Theorem 4 becomes

$$\begin{aligned}\frac{\sum_{k=1}^K \mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2]}{K} \leq & \frac{2}{\gamma K}\left(f(\mathbf{x}_1) - f(\mathbf{x}^*) + K(\frac{L\gamma^2}{2} + 2L^2T\gamma^3)\frac{\sigma^2}{M} + K\gamma^3(2L^2 + 8L^2T^2)\widehat{G}^2\right) \\ = & \frac{2(f(\mathbf{x}_1) - f(\mathbf{x}^*))}{\gamma K} + \gamma\frac{L\sigma^2}{M} + \gamma^2\left(4L^2T\frac{\sigma^2}{M} + (4L^2 + 16L^2T^2)\widehat{G}^2\right) \\ \leq & \frac{2(f(\mathbf{x}_1) - f(\mathbf{x}^*))}{\gamma K} + 2\gamma\frac{L\sigma^2}{M} = 2(f(\mathbf{x}_1) - f(\mathbf{x}^*))\sqrt{\frac{L\sigma^2}{MK}} + 2\sqrt{\frac{L\sigma^2}{MK}},\end{aligned}$$

where the last inequality holds because of (35). □