Distributed Accelerated Proximal Optimization Algorithms with Variance Reduction

***Abstract*—The proximal stochastic gradient descent (ProxSGD) has been widely used to solve composite convex optimization problems. However, the random sampling in ProxSGD introduces noisy gradient updates with high variance, which causes to use a vanishing step size and so slows down the convergence as the iterates approach the optimum. In addition, although ProxSGD with variance-reduction enjoys great success in applications at small and moderate scales, but distributed versions of these algorithms are crucially demanded for larger-scale training datasets. In this paper, we propose a synchronous method, Sync-AcPSVRG, and an asynchronous method, Async-AcPSVRG, which integrate variance-reduction and momentum acceleration techniques in a distributed manner to speed up ProxSGD updates and scale nearly linearly with the number of workers. Both Sync-AcPSVRG and Async-AcPSVRG enjoy lower iteration complexity than existing accelerated stochastic variance reduction methods, which need more updates per iteration. Furthermore, we allow the number of updates to increase with the epochs which decreases the number of communication frequencies between local node and master node. We compare distributed AcPSVRG with existing parallel stochastic proximal optimization methods on a few datasets. The empirical results verify our theoretical findings and indicate that our proposed distributed accelerated methods with variance reduction are more efficient and effective than other ProxSGD variants.**

I. Introduction

In this paper, we consider the following composite optimization problem

$$\min_{x\in\mathbb{R}^d} F(x) = h(x) + f(x) = h(x) + \frac{1}{n}\sum_{i=1}^{n} f_i(x), \qquad (1)$$

where $f_i(x)$, $i = 1,2,\ldots,n$, are smooth convex loss functions, and $h(x)$ is a non-smooth regularization term. Proximal stochastic gradient descent (ProxSGD) [20] is a general approach for solving minimization problem in (1), which employs the fact that the objective function decomposes into a sum over many terms.

For problems where each f_i in (1) corresponds to a single data observation, ProxSGD selects a single data index i_k on each iteration k, and then performs update by solving the proximal optimization subproblem

$$x^{k+1} = \mathrm{Prox}_{\eta,h}\{x^k - \eta\nabla f_{i_k}(x_k)\},$$

where the proximal mapping is defined as

$$\mathrm{Prox}_{\eta,h}(y) = \ \mathrm{argmin}_{x\in\mathbb{R}^d}\{\frac{1}{2\eta}\|x-y\|^2 + h(x)\}.$$

In the above notation $\|\cdot\|$ is the L_2-norm and η is the step size. Typically, i_k is chosen uniformly at random from $\{1,2,\ldots,n\}$ on each iteration k, thus making the gradient approximation unbiased. However, because random sampling in ProxSGD introduces variance, vanishing step sizes are needed to guarantee the algorithm's convergence, and the convergence rate is only sublinear [11, 24]. In [29], a variance reduction technique for proximal algorithms was introduced and it is proved that it can achieve linear convergence. The convergence rate of ProxSGD with variance reduction can be further improved with the so-called momentum acceleration technique [2].

Another major drawback of ProxSGD is the sequential nature of algorithm. For massive datasets or training datasets distributed over a cluster of multiple nodes, parallel or distributed algorithms are sorely needed, making more practical impacts on parallel SGD variants to solve large-scale or distributed problems. There is quite a bit of recent works in the area of distributed optimization that have been successfully applied to accelerate many optimization algorithms including SGD [1, 16, 25] and ProxSGD [14, 18].

In this paper, we propose distributed algorithms to enhance the scalability of proximal algorithms using variance reduction and momentum acceleration techniques, yielding ProxSGD methods that scale near linearly and can provide better convergence rate.

II. Backgrounds

Since SGD estimates the gradient from only one or a few samples, the variance of the stochastic gradient estimator may be large [9, 30], which leads to slowdown and poor performance. Recently there has been a lot of interest in methods which reduce the variance incurred due to stochastic gradients. Variance reduction (VR) methods [5, 9] reduce the variance in the stochastic gradient estimates, and are able to alleviate the effects of vanishing step sizes that usually hit SGD, and yield methods that improve convergence rates both theoretically and empirically. Inspired by the success of these methods in reducing the gradient noise in stochastic gradient based optimization, recently many variants of variance reduction methods have been proposed [3, 10, 13, 28, 29]. Variance reduction methods begin with an initial estimate $\widetilde{x}$, and then generate a sequence of iterates x_k from

$$x_k = x_{k-1} - \eta\left[\nabla f_{i_k}(x_{k-1}) - \nabla f_{i_k}(\widetilde{x}) + \widetilde{\nabla} f(\widetilde{x})\right],$$

where η is the step size, $\widetilde{\nabla} f(\widetilde{x})$ is an approximation of the true gradient, and $\widetilde{x}$ is chosen to be a recent iterate from algorithm history. In particular, an error correction term is subtracted from regular update rules in stochastic optimization to reduce

the variance of gradients in order to deal with the problems of instability and slower convergence hit SGD.

Recently several acceleration techniques based on momentum compensations were introduced to further speed up the VR methods mentioned above [2, 8, 15, 22]. The momentum acceleration technique, which is basically proposed by Nesterov [21] for gradient methods, introduces one or multiple auxiliary variables to record the momentum, and updates the parameters according to the gradient at the auxiliary variable. To be specific, with momentum acceleration, the update rule becomes

$$\begin{aligned} x_{k+1} &= z_k - \eta \nabla f_{i_k}(z_k), \\ z_{k+1} &= x_{k+1} + \beta(x_{k+1} - x_k), \end{aligned} \tag{2}$$

where β is the momentum weight. It can be proven that the convergence rate of SGD is improved by using the Nesterov acceleration technique [8]. After that, many accelerated algorithms have been designed to achieve faster convergence rates for stochastic optimization methods [2, 6, 15, 27].

Although these acceleration techniques have great value in general, for large-scale problems, however, with the availability of very big training data, sequential SGD is very inefficient. Therefore, integrating the VR algorithms and acceleration techniques to distributed settings remain indispensable. In [16, 26] SVRG is extended to the parallel asynchronous setting. In particular, in [17] a parallel algorithm mitigated by variance reduction, coordinate sampling, and Nesterov's method is introduced. In this paper, we introduce distributed synchronous and asynchronous SGD-based algorithms with variance reduction integrated with acceleration techniques, which have not been well studied in the literature to the best of our knowledge. We proved that the proposed distributed algorithms have desirable convergence property and perform empirical investigations on it. The parallel algorithms are highly scalable, uses a constant learning rate, and converges linearly to the optimal solution in the strongly convex case.

III. Our Approaches

In this work, we attempt to introduce momentum to accelerate distributed variance reduction SGD algorithms for convex problems. We demonstrate that one step of momentum with only one weight provides a faster convergence rate in expectation, while we assume an upper bound for delay τ between delayed gradient and the latest one.

By applying this technique in distributed stochastic algorithms, we allow many local nodes to run simultaneously, while communicating with the server node through the exchange of updates. This new algorithm allows many processes to work towards a central solution which are faster by order than existing variance-reduced ProxSGD algorithms.

This work has three main contributions:

1) We propose distributed algorithms for convex problems adopted with variance reduction methods and momentum acceleration techniques with only one momentum weight, built on [2, 17, 28]. We show that synchronous and asynchronous variations of proposed algorithm can scale nearly linearly with the number of processors.
2) We theoretically study the convergence for general proximal algorithm and obtain improved convergence rate for convex and strongly convex functions. The analysis could be applied to other stochastic asynchronous algorithms like ASCD.
3) Finally, we present several empirical results over several distributed VR algorithms to demonstrate that the new accelerated algorithm can lead to better performance improvements than competing options, which agrees with our theory.

IV. Single-Worker Accelerated ProxSVRG

We begin by proposing our new accelerated proximal variance reduction scheme with momentum acceleration (AcPSVRG), in the single-worker case. As we will see later, the proposed method has a natural generalization to the distributed setting that has low communication requirement. Similar to the epoch gradient descent algorithm [7], we increase the number of iterations by a constant γ for every epoch.

A. Algorithm Overview

Our proposed VR scheme is divided into S epochs, with m_s updates taking place in each epoch. Throughout the paper, we will use the superscript for the index of each epoch, and the subscript for the index of iterations within each epoch.

Let the iterates generated in the s-th epoch be written as $\{x_k^s\}_{k=0}^{m_s-1}$. The update rule for AcPSVRG can be formulated as follows:

In each iteration the following type of proximal subproblem is solved,

$$z_{k+1}^s = \mathrm{Prox}_{\frac{\eta}{\beta_s} h}\{z_k^s - \frac{\eta}{\beta_s}\widetilde{\nabla}_k^s\}, \tag{3}$$

where

$$\widetilde{\nabla}_k^s = \nabla f_{i_k^s}(x_k^s) - \nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \nabla f(\widetilde{x}^{s-1}). \tag{4}$$

We let snapshot $\widetilde{x}^s$ be a weighted average of x_k^s in the most recent epoch s, which is updated at the end of each epoch, i.e., after every m_s parameter updates. We compensate the momentum term and introduce a new extrapolation rule to update x_k^s,

$$x_{k+1}^s = \widetilde{x}^s + \beta_s(z_{k+1}^s - \widetilde{x}^s),$$

where $\beta_s \in [0,1]$ is the momentum weight. Indeed, z_{k+1}^s is a momentum which adds a weighted sum of gradient history into x_k^s. Comparably, this update has only one momentum weight β_s versus two weights τ_1 and τ_2 in [2] through introducing two momentum vectors. Similar to [7], we increase the number of iterations by a constant γ for every epoch, i.e., $m_{s+1} = \gamma m_s$. The choice of growing the iterations per epoch can reduce the number of full gradient calculations and the frequency of communication between the server and the local nodes in the distributed setting, while it can speed up the convergence [4].

Note that if i_k^s in (4) is chosen uniformly at random from the set $\{1,2,\ldots,n\}$ on each iteration k, then, conditioning on all history,

$$\mathbb{E}[\nabla f_{i_k^s}(\widetilde{x})] = \nabla f(\widetilde{x}).$$

Thus, the error correction term has expected value 0, and $\mathbb{E}[\widetilde{\nabla}_k^s] = \nabla f(x_k^s)$, i.e., the approximate gradient $\widetilde{\nabla}_k^s$ is unbiased.

B. Algorithm Details for AcPSVRG

The detailed steps of AcPSVRG are listed in Algorithm 1. Note, the number of updates per epoch m_s and $\widetilde{x}^s$ are initialized with m_1 and $\widetilde{x}^0$. At the beginning of each epoch we initialize $x_0^s = z_0^s = \widetilde{x}^{s-1}$, where $\widetilde{x}^{s-1}$ is the average of the past m_{s-1} stochastic iterations.

AcPSVRG builds on the Katyusha momentum compensation method [2], but with the lower space complexities. Namely, on every iteration, one gradient computation is required and one optimization subproblem solved with one auxiliary variable. Furthermore, at the end of each epoch, n gradients $\{\nabla f_i(\widetilde{x})\}_1^n$ also should be calculated.

Algorithm 1 AcPSVRG

Input: The number of epochs S and the step size η
Initialize: $\widetilde{x}^0$, m_1, θ_1, and $\rho > 1$
  for $s = 1$ to S **do**
    $\widetilde{x} = \widetilde{x}^{s-1}$
    $\nabla f(\widetilde{x}) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\widetilde{x})$
    $x_0^s = z_0^s = \widetilde{x}$
    for $k = 0$ to $m_s - 1$ **do**
      Pick i_k^s uniformly at random from $\{1,\ldots,n\}$.
      $\widetilde{\nabla}_k^s = \nabla f_{i_k^s}(x_k^s) - \nabla f_{i_k^s}(\widetilde{x}) + \nabla f(\widetilde{x})$
      $\delta_k^s = \operatorname{argmin}_\delta h(z_k^s + \delta) + \langle \widetilde{\nabla}_k^s, \delta\rangle + \frac{\beta_s}{2\eta}\|\delta\|^2$
      $z_{k+1}^s = z_k^s + \delta_k^s$
      $x_{k+1}^s = \widetilde{x} + \beta_s(z_{k+1}^s - \widetilde{x})$
    end for
    $\widetilde{x}^s = \frac{1}{m_s}\sum_{k=0}^{m_s-1} x_k^s$, $m_{s+1} = \gamma m_s$
  end for
Output: $\widetilde{x}^S$

V. Distributed Algorithms

We now consider the distributed setting, on a master-slave framework with a master machine and p local clients, each of which contains a portion of the data set. In this setting, the whole data Ω is decomposed into disjoint subsets $\{\Omega_k\}$, where k denotes a particular local worker, and $\cup_{l=1}^p \Omega_l = \Omega$ and if $i \neq j$, $\Omega_i \cap \Omega_j = \emptyset$. Different clients can not communicate with each other and the clients can only communicate with the central server. Our model of computation is similar to the ones used in Parameter Server [12] and Mllib [19]. Our goal is to derive stochastic algorithms that scale linearly to high p.

A. Synchronous Accelerated SGD

AcPSVRG naturally extends to the distributed synchronous setting, and is presented in Algorithm 2. To distinguish the algorithm from the single worker case, we call it Sync-AcPSVRG. On each epoch, the local nodes first retrieve a copy of the central iterate $\widetilde{x}^{s-1}$, compute the sum of the gradients on its local data, i.e., $\sum_{i\in\Omega_k}\nabla f_i(\widetilde{x}^{s-1})$ and send it to the master node. The accelerated VR method is then locally performed on each node by only using the local data, and each worker sends the most recent parameter denoted as x_k for k-th worker to the master. Then parameter $\widetilde{x}^s$ is updated by the master after all the locally updated parameters have been gathered. As we put now constraint on how training data are partitioned on different workers, by sharing full gradient $\nabla f(\widetilde{x})$ across nodes, we ensure that the local gradient updates utilize global gradient information from remote nodes. We ensure also local updates are not far away from global updates through including momentum. This speeds up the convergence of stochastic optimization and controls the difference between the local update and global update, even if each local node runs for one whole epoch before communicating back with the central node.

Algorithm 2 Sync-AcPSVRG

Input: The number of epochs S and the step size η
Initialize: $\widetilde{x}^0$, m_1, θ_1, and $\rho > 1$
  for Worker l **do**
    for $s = 1$ **to** S **do**
      Receive $\widetilde{x} = \widetilde{x}^{s-1}$ from master node.
      Send $\nabla_l f(\widetilde{x}) = \sum_{i\in\Omega_l}\nabla f_i(\widetilde{x})$ to master node.
      Receive $\nabla f(\widetilde{x})$ from master node.
      $x_{l,0}^s = z_{l,0}^s = \widetilde{x}$
      for $k = 0$ **to** $m_s - 1$ **do**
        Pick $i_{l,k}^s$ uniformly at random from Ω_l.
        $\widetilde{\nabla}_{l,k}^s = \nabla f_{i_{l,k}^s}(x_{l,k}^s) - \nabla f_{i_{l,k}^s}(\widetilde{x}) + \nabla f(\widetilde{x})$
        $\delta_{l,k}^s = \operatorname{argmin}_\delta h(z_{l,k}^s + \delta) + \langle \widetilde{\nabla}_{l,k}^s, \delta\rangle + \frac{\beta_s}{2\eta}\|\delta\|^2$
        $z_{l,k+1}^s = z_{l,k}^s + \delta_{l,k}^s$
        $x_{l,k+1}^s = \widetilde{x} + \beta_s(z_{l,k+1}^s - \widetilde{x})$
      end for
      $\widetilde{x}_l^s = \frac{1}{m_s}\sum_{k=0}^{m_s-1} x_{l,k}^s$, $m_{s+1} = \gamma m_s$
      Send $\widetilde{x}_l^s$ to master node.
    end for
  end for
  Master Node:
      Average $\widetilde{x}_l$ received from workers.
      Broadcast averaged $\widetilde{x}$ to local workers.
      Average $\nabla f(\widetilde{x}_l)$ received from workers.
      Broadcast averaged $\nabla f(\widetilde{x})$ to local workers.
Output: $\widetilde{x}^S$

In Sync-AcPSVRG, as we mentioned earlier, we assume that each worker has access to a subset and not the entire data set and performs local updates for one epoch, or iterations, before communicating with the server. This is a rather low communication frequency compared to a mini-batch parameter server model such as parallel implementation of SVRG [31] or mS2GD [10] in which stochastic optimization is performed on the master node based on the whole dataset, and updates and gradients are frequently transferred between workers and master. This makes a significant difference in runtimes when

the number of local nodes is large. We also increase the number of inner updates per epoch by a constant γ which reduces the communication rate between server and worker because there is no communication during the inner iterations of each epoch and yields fastest convergence. We also let local workers make updates according to the proximal solver.

B. Asynchronous Accelerated SGD

The synchronous algorithm could be extended to the asynchronous one called Async-AcPSVRG as shown in Algorithm 3. We adopt asynchronous update in each inner loop and there is synchronization operation after each epoch. In Async-AcPSVRG, the master node keeps a copy of the averaged x. We make workers do proximal mapping step, and server is responsible for element-wise addition operations, average x, aggregate full gradient and then broadcast it to workers at the end of each epoch.

Algorithm 3 Async-AcPSVRG

Input: The number of epochs S and the step size η.
Initialize: $\widetilde{x}^0$, m_1, θ_1, and $\rho > 1$
for $s = 1$ to S **do**
  $t = 0$;
  Worker l
  Wait until it receives $\widetilde{x} = \widetilde{x}^{s-1}$ from master node.
  Send $\nabla_l f(\widetilde{x}) = \sum_{i\in\Omega_l} \nabla f_i(\widetilde{x})$ to master node.
  Wait until it receives $\nabla f(\widetilde{x})$ from master node.
  $x^s_{l,0} = z^s_{l,0} = \widetilde{x}$
  for $k = 0$ to $m_s - 1$ **do**
    Receive $x^s_{l,k-\tau_k} := x^s_t$ from master node.
    Pick $i^s_{l,k}$ uniformly at random from Ω_l.
    $\widetilde{\nabla}^s_{l,k} = \nabla f_{i^s_{l,k}}(x^s_{l,k-\tau_k}) - \nabla f_{i^s_{l,k}}(\widetilde{x}) + \nabla f(\widetilde{x})$
    $\delta^s_{l,k} = \text{argmin}_\delta \; h(z^s_{l,k} + \delta) + \langle \widetilde{\nabla}^s_{l,k}, \delta\rangle + \frac{\beta_s}{2\eta}\|\delta\|^2$
    $z^s_{l,k+1} = z^s_{l,k} + \delta^s_{l,k}$
    Send $z^s_{l,k+1}$ to master node.
    Master Node:
    Receive $z^s_{l,k+1}$ from worker l.
    Update $x^s_{t+1} = \widetilde{x} + \beta_s(z^s_{l,k+1} - \widetilde{x})$, $t = t+1$.
  end for
  Master Node:
  Calculate average $\widetilde{x}^s = \frac{1}{t}\sum_{i=1}^t x^s_i$ and broadcast averaged $\widetilde{x}^s$.
  Receive $\nabla_l f(\widetilde{x}^s)$ from workers and calculate average $\nabla f(\widetilde{x}^s)$.
  Broadcast averaged $\nabla f(\widetilde{x}^s)$ to local workers.
  $m_{s+1} = \gamma m_s$
end for
Output: $\widetilde{x}^S$

The key idea for Async-AcPSVRG is that, once a local node solves the proximal update, it sends the update to the master. The master thread will use the update received by the local node to update x. Each processor repeatedly runs these procedures concurrently, without any synchronization. We use $k - \tau_k$ to denote the state of x at the reading time by the local worker. For asynchronous algorithms, the partial gradient calculated by the local worker will be delayed, and at iteration k the worker l could only obtain $\nabla f_{i^s_{l,k}}(x^s_{k-\tau_k})$ instead of $\nabla f_{i^s_{l,k}}(x^s_k)$. We assume that the delay between the time of evaluation and updating is bounded by a non-negative integer τ, i.e., $\tau_k \le \tau$. The parameter τ captures the degree of delay. When there are more threads, the delay accumulates and results in larger τ. Furthermore, we also assume that the system is synchronized after every epoch.

For the purpose of our analysis, we assume a consistent read model, i.e., when x is read or updated in the central node, it will be locked. However, the proposed asynchronous algorithms can be easily implemented in a lock-free setting, leading to further speedups. We leave the analysis of inconsistent read (wild) model as future work.

VI. Convergence Analysis

In this section, we provide convergence analysis for distributed AcPSVRG. For further analysis, throughout this paper, we make the following assumptions, which are commonly used in previous works [17, 26].

Assumption 1 (Lipschitz Gradient). The components $f_i(x), i = 1,2,\ldots,n$ are differentiable and have Lipschitz continuous partial gradients, i.e., for $x, y \in \mathbb{R}^d$, we have,

$$\|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x - y\|. \tag{5}$$

Assumption 2 (Convexity). The components $f_i(x), i = 1,2,\ldots,n$ and function $h(x)$ are convex. The objective function $F(x)$ could be strongly convex with parameter μ, i.e., $\forall x, y \in \mathbb{R}^d$

$$F(y) \ge F(x) + \langle \xi, y - x\rangle + \frac{\mu}{2}\|x - y\|^2, \qquad \forall \xi \in \partial F(x), \tag{6}$$

and for non-strongly convex functions $\mu = 0$.

Assumption 3 (Bounded Delay). The delays $\tau_1, \tau_2, \ldots$ are independent random variables, and $\tau_k \le \tau$ for all k. As we mentioned earlier, we use $k - \tau_k$ to denote the read state at iteration k in the asynchronous algorithm.

Let $p(w)$ be a convex function over a convex set X. Let $\hat{w} = \text{argmin}_{w\in X}\{p(w) + \alpha\|w - \bar{y}\|^2\}$ for some $\bar{y} \in X$ and $\alpha \ge 0$. Due to the fact that the sum of a convex and a strongly convex function is also strongly convex, we have

$$p(w) + \alpha\|w - \bar{y}\|^2 \ge p(\hat{w}) + \alpha\|\hat{w} - \bar{y}\|^2 + \alpha\|w - \hat{w}\|^2. \tag{7}$$

Lemma VI.1. *[2, 9] If $f_i(x)$, $i = 1,2,\ldots,n$ have L-Lipschitz continuous gradients, then with defining*

$$\widetilde{\nabla} f(w) = \nabla f_k(w) - \nabla f_k(\widetilde{x}) + \nabla f(\widetilde{x}),$$

we have

$$\mathbb{E}\left(\left\|\widetilde{\nabla} f(w) - \nabla f(w)\right\|^2\right) \le 2L(f(\widetilde{x}) - f(w) + \langle \nabla f(w), w - \widetilde{x}\rangle). \tag{8}$$

Lemma VI.2. *Under Assumptions 1-3, if x^* is the optimal solution of Problem (1), for the Algorithm 3 and $\eta < \frac{1}{3L}$, we have*

$$\mathbb{E}\left[F(\widetilde{x}^s)-F(x^*)\right] \le \lambda_s \mathbb{E}[F(\widetilde{x}^{s-1})-F(x^*)] + \frac{\beta_s^2}{2\eta m_s(1-\alpha_s)}\mathbb{E}[\|z_0^s-x^*\|^2-\|z_{m_s}^s-x^*\|^2], \quad (9)$$

with $\lambda_s=\frac{(1-\beta_s+\alpha_s)}{1-\alpha_s}$, $\alpha_s=\frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2}$, $\theta_\eta=\frac{1-\eta L}{2\eta L}$ *and* β_s *is chosen such that* $\beta_s<1-\theta_\eta^{-1}$.

Proof. See Appendix A. □

We have the following convergence result.

Theorem VI.3. *Suppose the assumptions of Lemma VI.2 and further,* β_s *and* η *are chosen such that* $\beta_s \le \frac{1}{s+2}$, $\lambda_s/\beta_s^2 \le 1/\beta_{s-1}^2$ *and* $\alpha_s \le \frac{1}{2}$. *Then, we have*

$$\mathbb{E}[F(\widetilde{x}^S)-F(x^*)] \le \frac{1}{\beta_0^2(S+2)^2}[F(\widetilde{x}^0)-F(x^*)] + \frac{1}{\eta m_1(S+2)^2}\mathbb{E}[\|z_0^1-x_*\|^2]. \quad (10)$$

If function $F(x)$ *is strongly convex with parameter* μ, *and* β_s *and* m_s *are chosen such that*

$$\lambda_s' := \lambda_s + \frac{2\beta_s^2}{\mu\eta m_s} < 1,$$

we have,

$$\mathbb{E}\left[F(\widetilde{x}^S)-F(x^*)\right] \le \lambda_{max}^S\left[F(\widetilde{x}^0)-F(x^*)\right], \quad (11)$$

where $\lambda_{max}=\max_s \lambda_s'$.

Proof. From Lemma VI.2 we have,

$$\mathbb{E}[F(\widetilde{x}^s)-F(x^*)] \le \lambda_s[F(\widetilde{x}^{s-1})-F(x^*)] + \frac{\beta_s^2}{2\eta m_s(1-\alpha_s)}\mathbb{E}[\|z_0^s-x_*\|^2-\|z_{m_s}^s-x_*\|^2]. \quad (12)$$

Dividing both sides of the above inequality by β_s^2, and using the fact $\lambda_s/\beta_s^2 \le 1/\beta_{s-1}^2$ we have

$$\begin{aligned}\frac{\mathbb{E}[F(\widetilde{x}^s)-F(x^*)]}{\beta_s^2} &\le \frac{\lambda_s}{\beta_s^2}[F(\widetilde{x}^{s-1})-F(x^*)] + \frac{1}{2\eta m_s(1-\alpha_s)}\mathbb{E}[\|z_0^s-x^*\|^2-\|z_{m_s}^s-x^*\|^2] \\ &\overset{a}{\le} \frac{1}{\beta_{s-1}^2}[F(\widetilde{x}^{s-1})-F(x^*)] + \frac{1}{2\eta m_s(1-\alpha_s)}\mathbb{E}[\|z_0^s-x^*\|^2-\|z_{m_s}^s-x^*\|^2],\end{aligned}$$

where in inequality $\overset{a}{\le}$ we used $\frac{\lambda_s}{\beta_s^2} \le \frac{1}{\beta_{s-1}^2}$. By using $z_0^{s+1}=z_{m_s}^s$, and adding the above inequality from $s=1$ to S, we have

$$\frac{\mathbb{E}[F(\widetilde{x}^S)-F(x^*)]}{\beta_S^2} \le \frac{1}{\beta_0^2}[F(\widetilde{x}^0)-F(x^*)] + \frac{1}{\eta m_0}\mathbb{E}\left[\|z_0^1-x_*\|^2\right],$$

where we used $m_1 \le m_s$ and $\alpha_s \le \frac{1}{2}$. Then we have

$$\begin{aligned}&\mathbb{E}[F(\widetilde{x}^S)-F(x^*)] \\ &\le \frac{\beta_S^2}{\beta_0^2}[F(\widetilde{x}^0)-F(x^*)] + \frac{\beta_S^2}{\eta m_1}\mathbb{E}[\|z_0^1-x_*\|^2 \\ &\le \frac{1}{\beta_0^2(S+2)^2}[F(\widetilde{x}^0)-F(x^*)] + \frac{1}{\eta m_1(S+2)^2}\mathbb{E}[\|z_0^1-x_*\|^2].\end{aligned}$$

Now suppose $F(x)$ is μ-strongly convex. Since x^* is the optimal solution, by inequality (6) we have

$$\nabla F(x^*)=0, \qquad F(x)-F(x^*) \ge \frac{\mu}{2}\|x-x^*\|^2. \quad (13)$$

Using the inequality in (13) and $z_0^s=\widetilde{x}^{s-1}$, we have

$$\begin{aligned}&\mathbb{E}[F(\widetilde{x}^s)-F(x^*)] \\ &\overset{a}{\le} \lambda_s\mathbb{E}[F(\widetilde{x}^{s-1})-F(x^*)] + \frac{\beta_s^2}{2m_s\eta(1-\alpha_s)}\mathbb{E}[\|z_0^s-x_*\|^2-\|z_m^s-x_*\|^2] \\ &\overset{b}{\le} \lambda_s\mathbb{E}[F(\widetilde{x}^{s-1})-F(x^*)] + \frac{2\beta_s^2}{\mu\eta m_s}\mathbb{E}[F(\widetilde{x}^{s-1})-F(x^*)] \\ &= (\lambda_s+\frac{2\beta_s^2}{\mu\eta m_s})\mathbb{E}[F(\widetilde{x}^{s-1})-F(x^*)], \quad (14)\end{aligned}$$

where the inequality $\overset{a}{\le}$ follows from Lemma VI.2, and $\overset{b}{\le}$ is due to the inequality (13) and the inequality $\alpha_s \le \frac{1}{2}$. □

As a corollary, we immediately obtain an expected linear rate of convergence for the synchronous algorithm Sync-AcPSVRG for the strongly convex case.

Corollary VI.4. *Suppose function* F *is strongly convex and* β_s, m_s *and step size* η *are chosen such that the following condition is satisfied*

$$\lambda_s' := 1-\beta_s+\frac{2\beta_s^2}{\eta\mu m_s} < 1,$$

Then, for iterates of algorithm Sync-AcPSVRG, we have

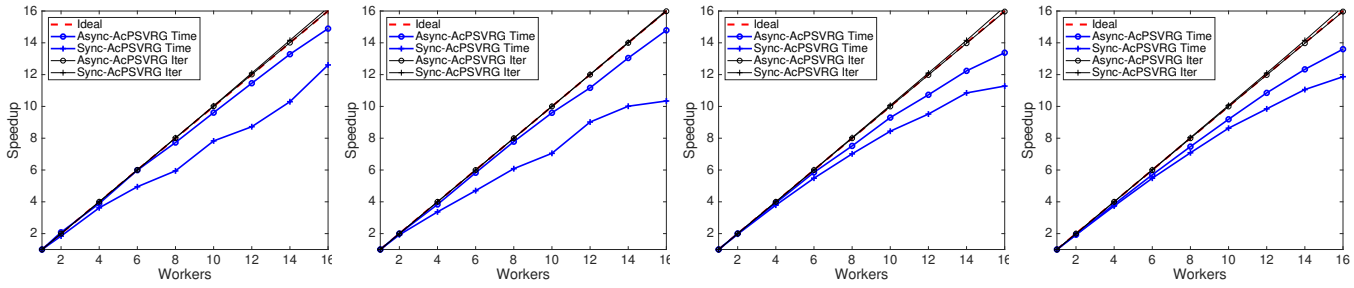
$$\mathbb{E}\left[F(\widetilde{x}^S)-F(x^*)\right] \le \lambda_{max}^S\left[F(\widetilde{x}^0)-F(x^*)\right], \quad (15)$$

with $\lambda_{max}=\max_s\lambda_s'$. *The optimal value of* λ *is* $\lambda_{max}=1-\frac{\mu m_s\eta}{8}$ *which is obtained by choosing* $\beta_s=\frac{\mu m_s\eta}{4}$.

Proof. For the synchronous algorithm, $\tau=0$, and consequently $\alpha_s=0$. Then, from Theorem VI.3, we have $\lambda_s=1-\beta_s$ and

$$\lambda_s'=1-\beta_s+\frac{2\beta_s^2}{\mu\eta m_s}.$$

The minimal value of λ_{max} is obtained by minimizing λ_s' with respect to β_s. □



(a) speedup on ijcnn (b) speedup on covtype (c) speedup on real-sim (d) speedup on rcv1

Fig. 1: Scalability analysis of Sync-AcPSVRG and Async-AcPSVRG with iteration and time speedup.

TABLE I: Summary of training datasets.

Datasets	Data	Features	Non-zeros
ijcnn	35,000	22	455,000
covtype	581,012	54	7,521,450
real-sim	72,309	20,958	3,709,083
rcv1	20,242	47,236	50,233,657

VII. Experiments

We present our empirical results in this section. For our experiments, we study logistic regression loss function with L_2 regularization. The problem can be formulated as optimization problem (1) with $f_i(x) = \log(1+e^{-y_i z_i^T x})$, $h(x) = \frac{\tau}{2}\|x\|^2$, where $z_i \in \mathbb{R}^d$ and y_i is the corresponding label for each i. Further, the L_2 regularization weight τ is set 10^{-4} in all experiments. We evaluate the following state-of-the-art variance reduction algorithms for our experiments: 1) SVRG [9], 2) ProxASAGA[23] and 3) Katyusha [2]. The learning rates are tuned for these algorithms in our experiments, and the results shown in this section are based on the best learning rate for each algorithm we achieved.

All the algorithms were implemented in C++ and Open-MPI[1]. We run our experiments on datasets from LIBSVM website[2], as shown in Table I . The epoch size m_1 initially is chosen as 125 in all our experiments and it grows in each epoch by a constant of $\gamma = 1.5$. In AcPSVRG, we set step sizes η and β_s to satisfy our assumptions in lemmas and theorems with $\tau \sim P$, where P is the number of workers.

In the first experiment, we compare the speedup achieved by our synchronous and asynchronous algorithm. To this end, for each dataset we first measure the time required for the algorithm to reach the accuracy of 10^{-10}. The iteration speedup with P processes is defined as

$$\text{iteration speedup} = \frac{\text{number of iterations with single worker}}{\text{avg. number of iterations with } P \text{ workers}}$$

and the time speedup is defined as

$$\text{time speedup} = \frac{\text{runtime with single worker}}{\text{runtime with } P \text{ workers}}.$$

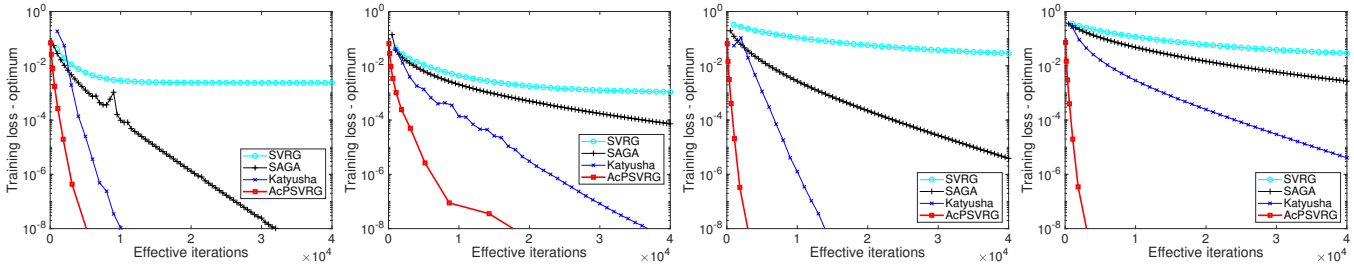
[1]All simulations were performed on Amazon EC2 t2.large instances with 8 GB RAM.

[2]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html

Results in Figure (1) show the speedup on various datasets. As shown by Figure (1), we achieve remarkable speedups for all the datasets. Both synchronous and asynchronous versions show linear iteration speedup in all data sets. Furthermore, the highest time speedup for Async-AcPSVRG is achieved for ijcnn dataset which has the fewest number of features compared to other datasets we considered in this experiment. Evidently, the time speedup achieved by Async-AcPSVRG is higher than ones obtained by Sync-AcPSVRG. The speedup for Sync-AcPSVRG improves with the size of datasets, since the synchronization cost, which includes both communication time and waiting time, is dominant compared to computation time on small datasets. The results for Async-AcPSVRG show better scalability compared with [17, 26].

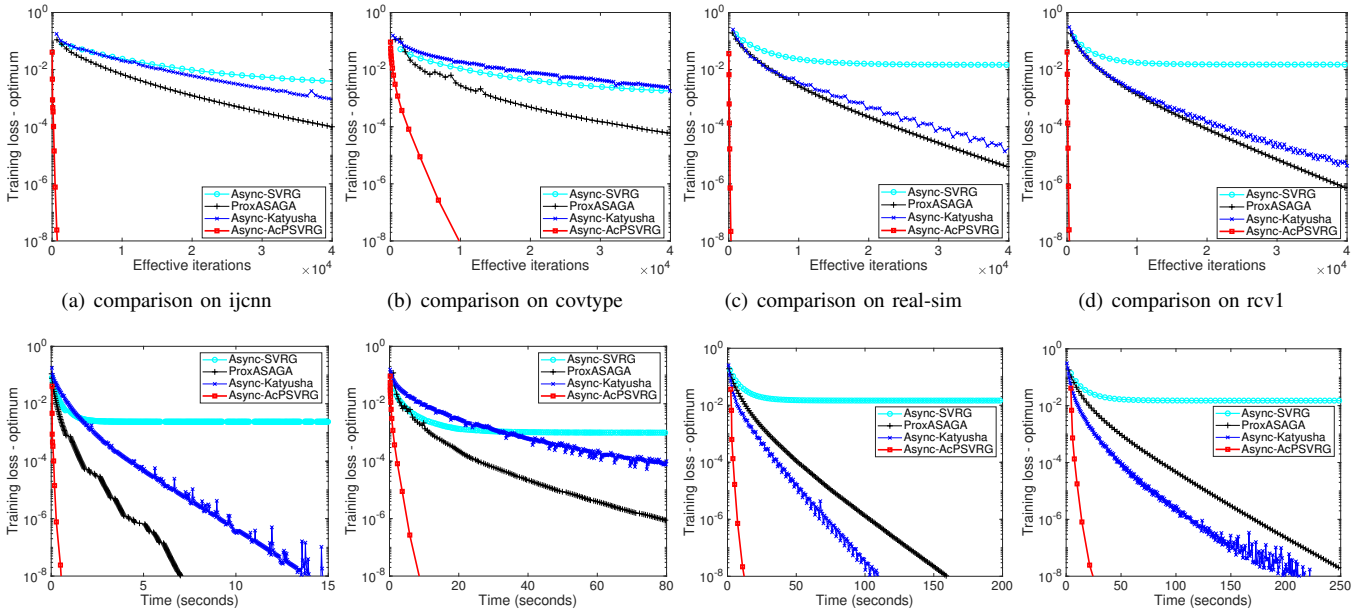
For the second set of experiments we compare the performance of AcPSVRG with the variants of variance reduction stochastic gradient descent described earlier in this section. Figure 2 shows the performance of AcPSVRG, SVRG, ProxASAGA and Katyusha with single worker. It is seen that the performance of AcPSVRG method outperforms other variants of SGD methods in all cases. We would like to verify in experiments if such benefits preserve for the asynchronous variants of these algorithms. We use 16 workers to compare the algorithms in this experiment. Figure 3 (top) shows the performance of asynchronous SVRG (Async-SVRG), ProxASAGA and asynchronous Katyusha (Async-Katyusha) regarding effective iterations on server which are similar to update steps in sequential setting. The performance gains are qualitatively similar to these algorithms on single worker in Figure 2. It is observed that Async-AcPSVRG always exhibits better convergence than other SGD variants. In particular, compared to Async-Katyusha, as a momentum acceleration method, Async-AcPSVRG shows significantly better convergence. Since the computation complexity of each epoch of these algorithms is different, in Figure 3 (bottom) we directly plot the objective value versus the runtime for each of these algorithms. As seen in the figure, Async-AcPSVRG outperforms variance-reduced ProxSGD algorithms in all the cases. In particular compared to Async-Katyusha, as AcPSVRG uses only one auxiliary variable for calculation of momentum, it has lower complexity per iteration and so shows better convergence speed.

Our proposed algorithms outperform Katyusha in several



(a) comparison on ijcnn (b) comparison on covtype (c) comparison on real-sim (d) comparison on rcv1

Fig. 2: Convergence performance of SVRG, SAGA, Katyusha, and AcPSVRG with single worker.



(a) comparison on ijcnn (b) comparison on covtype (c) comparison on real-sim (d) comparison on rcv1

(e) comparison on ijcnn (f) comparison on covtype (g) comparison on real-sim (h) comparison on rcv1

Fig. 3: Training loss residual $f(x) - f(x^*)$ versus iteration (top) and time (bottom) plot of Async-SVRG, ProxASAGA, Async-Katyusha, and Async-AcPSVRG.

aspects. First, the plots of Async-Katyusha in asynchronous experiments show much more oscillations compared to our algorithms. This appearance is caused by the sensitivity of Katyusha to delayed gradients when calculating momentum terms with local partial data, which introduces variance in training process. Our algorithms do not show sensitivity on delayed gradients. Secondly, AcPSVRG needs to tune only one parameter for momentum acceleration for different datasets while Katyusha needs adjustment of two acceleration parameters in order to achieve a comparable level of performance.

VIII. Conclusion

In this paper, we have studied the distributed proximal stochastic gradient algorithms by integrating with variance reduction and momentum acceleration techniques. Using constant learning rate, we have proved their convergence rates, discussed their speedups, and empirically verified our theoretical findings. Our distributed proximal algorithms can achieve nearly linear speedup. The proposed algorithms can reduce the communication cost significantly by increasing the length of epoch by a constant after each epoch.

As for future work, we will extend the study in this paper to the non-convex case, and investigate AcPSVRG in shared memory architectures, both theoretically and empirically.

References

[1] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[2] Z. Allen-Zhu. Katyusha: The first truly accelerated stochastic gradient method. *ArXiv e-prints, abs/1603.05953*, 2016.

[3] Z. Allen-Zhu and E. Hazan. Variance reduction for faster non-convex optimization. In *ICML*, pages 699–707, 2016.

[4] Z. Allen-Zhu and Y. Yuan. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *ICML*, pages 1080–1089, 2016.

[5] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[6] O. Fercoq and P. Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

[7] E. Hazan and S. Kale. An optimal algorithm for stochastic strongly-convex optimization. *arXiv preprint arXiv:1006.2425*, 2010.

[8] C. Hu, W. Pan, and J. T. Kwok. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems*, pages 781–789, 2009.

[9] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[10] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

[11] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(Mar):777–801, 2009.

[12] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pages 583–598, 2014.

[13] X. Li, T. Zhao, R. Arora, H. Liu, and J. Haupt. Stochastic variance reduced optimization for nonconvex sparse learning. In *ICML*, pages 917–925, 2016.

[14] Y. Li, L. Xu, X. Zhong, and Q. Ling. Make workers work harder: Decoupled asynchronous proximal stochastic gradient descent. *arXiv preprint arXiv:1605.06619*, 2016.

[15] H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

[16] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv preprint arXiv:1507.06970*, 2015.

[17] Q. Meng, W. Chen, J. Yu, T. Wang, Z. Ma, and T.-Y. Liu. Asynchronous accelerated stochastic gradient descent. In *IJCAI*, pages 1853–1859, 2016.

[18] Q. Meng, W. Chen, J. Yu, T. Wang, Z. Ma, and T.-Y. Liu. Asynchronous stochastic proximal optimization algorithms with variance reduction. In *AAAI*, pages 2329–2335, 2017.

[19] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

[20] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[21] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2013.

[22] A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

[23] F. Pedregosa, R. Leblond, and S. Lacoste-Julien. Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. In *Advances in Neural Information Processing Systems*, pages 55–64, 2017.

[24] A. Rakhlin, O. Shamir, K. Sridharan, et al. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, 2012.

[25] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[26] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. J. Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*, pages 2647–2655, 2015.

[27] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *ICML*, pages 64–72, 2014.

[28] F. Shang, Y. Liu, K. Zhou, J. Cheng, K. K. Ng, and Y. Yoshida. Guaranteed sufficient decrease for stochastic variance reduced gradient optimization. *arXiv preprint arXiv:1802.09933*, 2018.

[29] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[30] P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, pages 1–9, 2015.

[31] T. Zhao, M. Yu, Y. Wang, R. Arora, and H. Liu. Accelerated mini-batch randomized block coordinate descent method. In *Advances in neural information processing systems*, pages 3329–3337, 2014.

APPENDIX A
PROOF OF LEMMA VI.2

Proof. By choosing $\eta < \frac{1}{3L}$ we have,

$$\begin{aligned}
f(x_{k+1}^s) &\le f(x_k^s) + \left\langle \nabla f(x_k^s), x_{k+1}^s - x_k^s \right\rangle + \frac{L}{2}\left\|x_{k+1}^s - x_k^s\right\|^2 \\
&= f(x_k^s) + \left\langle \nabla f(x_k^s), x_{k+1}^s - x_k^s \right\rangle \\
&\quad + \frac{1}{2\eta}\left\|x_{k+1}^s - x_k^s\right\|^2 - \theta_\eta L\left\|x_{k+1}^s - x_k^s\right\|^2 \\
&\overset{a}{=} f(x_k^s) + \left\langle \widetilde{\nabla}_k^s + \nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s), x_{k+1}^s - x_k^s \right\rangle \\
&\quad + \frac{1}{2\eta}\left\|x_{k+1}^s - x_k^s\right\|^2 + \left\langle \nabla f(x_{k-\tau_k}^s) - \widetilde{\nabla}_k^s, x_{k+1}^s - x_k^s \right\rangle \\
&\quad - \theta_\eta L\left\|x_{k+1}^s - x_k^s\right\|, \qquad (16)
\end{aligned}$$

where the inequality follows from Lipschitz continuous nature of the gradient of function f. In $\overset{a}{=}$, we add and subtract $\left\langle \widetilde{\nabla}_k^s - \nabla f(x_{k-\tau_k}^s), x_{k+1}^s - x_k^s \right\rangle$. From Cauchy-Schwarz inequality, we also have,

$$\begin{aligned}
&\mathbb{E}\left\langle \nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s), x_{k+1}^s - x_k^s \right\rangle \\
&\le \frac{1}{2\theta_\eta L}\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2 + \frac{L\theta_\eta}{2}\left\|x_{k+1}^s - x_k^s\right\|^2. \qquad (17)
\end{aligned}$$

To bound the last term in equation (16), we obtain,

$$\begin{aligned}
&\left\langle \nabla f(x_{k-\tau_k}^s) - \widetilde{\nabla}_k^s, x_{k+1}^s - x_k^s \right\rangle \\
&\overset{a}{\le} \mathbb{E}\left[\frac{1}{2\theta_\eta L}\left\|\nabla f(x_{k-\tau_k}^s) - \widetilde{\nabla}_k^s\right\|^2 + \frac{\theta_\eta L}{2}\left\|x_{k+1}^s - x_k^s\right\|^2\right] \\
&\overset{b}{\le} \frac{1}{\theta_\eta}\left[f(\widetilde{x}^{s-1}) - f(x_{k-\tau_k}^s) + \left\langle \nabla f(x_{k-\tau_k}^s), x_{k-\tau_k}^s - \widetilde{x}^{s-1} \right\rangle\right] \\
&\quad + \frac{\theta_\eta L}{2}\left\|x_{k+1}^s - x_k^{s-1}\right\|^2, \qquad (18)
\end{aligned}$$

where inequality $\overset{a}{\le}$ follows from the Cauchy-Schwarz inequality, and inequality $\overset{b}{\le}$ follows from Lemma VI.1.

Substituting the inequalities (17) and (18) in (16), and taking expectation over i_k^s, we obtain,

$$\begin{aligned}
&\mathbb{E}\left[F(x_{k+1}^s) - f(x_k^s)\right] \\
&\le \mathbb{E}\left[h(x_{k+1}^s) + \left\langle \widetilde{\nabla}_k^s, x_{k+1}^s - x_k^s \right\rangle + \frac{1}{2\eta}\left\|x_{k+1}^s - x_k^s\right\|^2\right] \\
&\quad + \frac{1}{\theta_\eta}\mathbb{E}\left[f(\widetilde{x}^{s-1}) - f(x_{k-\tau_k}^s) + \left\langle \nabla f(x_{k-\tau_k}^s), x_{k-\tau_k}^s - \widetilde{x}^{s-1} \right\rangle\right] \\
&\quad + \frac{1}{2\theta_\eta L}\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2 \\
&\overset{a}{\le} \mathbb{E}\left[\left\langle \beta_s \widetilde{\nabla}_k^s, z_{k+1}^s - z_k^s \right\rangle\right] + \frac{\beta_s^2}{2\eta}\left\|z_{k+1}^s - z_k^s\right\|^2 \\
&\quad + \mathbb{E}\left[\beta_s h(z_{k+1}^s) + (1-\beta_s)h(\widetilde{x}^{s-1})\right] \\
&\quad + \frac{1}{\theta_\eta}\mathbb{E}\left[f(\widetilde{x}^{s-1}) - f(x_{k-\tau_k}^s) + \left\langle \nabla f(x_{k-\tau_k}^s), x_{k-\tau_k}^s - \widetilde{x}^{s-1} \right\rangle\right] \\
&\quad + \frac{1}{2\theta_\eta L}\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2, \qquad (19)
\end{aligned}$$

where in inequality $\overset{a}{\leq}$ we use the update $x_k^s = z_k^s + (1-\beta_s)\widetilde{x}^{s-1}$ and convexity of h. Since $p(\delta) = h(z_k^s + \delta) + \langle \widetilde{\nabla}_k^s, \delta \rangle$ is a convex function, using inequality (7) with $w = x^* - z_k^s$, $\bar{y} = 0$, it follows

$$\begin{aligned} h(z_{k+1}^s) + \left\langle \widetilde{\nabla}_k^s, z_{k+1}^s - z_k^s \right\rangle + \frac{\beta_s}{2\eta} \left\| z_{k+1}^s - z_k^s \right\|^2 \\ \leq h(x^*) + \left\langle \widetilde{\nabla}_k^s, x^* - z_k^s \right\rangle \\ + \frac{\beta_s}{2\eta} (\|z_k^s - x^*\|^2 - \left\|z_{k+1}^s - x^*\right\|^2). \end{aligned} \tag{20}$$

By replacing the above inequality in (19) we have,

$$\begin{aligned} &\mathbb{E}\left[F(x_{k+1}^s) - f(x_k^s)\right] \\ \leq &\mathbb{E}\left[\left\langle \beta_s \widetilde{\nabla}_k^s, x^* - z_k^s \right\rangle + \frac{\beta_s^2}{2\eta} (\|z_k^s - x^*\|^2 - \left\|z_{k+1}^s - x^*\right\|^2)\right] \\ &+ \mathbb{E}\left[\beta_s h(x^*) + (1-\beta_s) h(\widetilde{x}^{s-1})\right] \\ &+ \frac{1}{\theta_\eta} \mathbb{E}\left[f(\widetilde{x}^{s-1}) - f(x_{k-\tau_k}^s) + \left\langle \nabla f(x_{k-\tau_k}^s), x_{k-\tau_k}^s - \widetilde{x}^{s-1} \right\rangle\right] \\ &+ \frac{1}{2\theta_\eta L} \mathbb{E}\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2 \\ \overset{a}{=} &\mathbb{E}\left[\frac{\beta_s^2}{2\eta} (\|z_k^s - x^*\|^2 - \left\|z_{k+1}^s - x^*\right\|^2) + \beta_s h(x^*)\right] \\ &+ \mathbb{E}\left[\left\langle \nabla f(x_{k-\tau_k}^s), \beta_s x^* + (1-\beta_s)\widetilde{x}^{s-1} - x_k^s \right\rangle\right] \\ &+ \mathbb{E}\left[\left\langle -\nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \nabla f(\widetilde{x}^{s-1}), \beta_s x^* + (1-\beta_s)\widetilde{x}^{s-1} - x_k^s \right\rangle\right] \\ &+ \mathbb{E}\left[\left\langle \nabla f(x_{k-\tau_k}^s), \theta_\eta^{-1}(x_{k-\tau_k}^s - \widetilde{x}^{s-1}) \right\rangle\right] \\ &+ (1-\beta_s)\mathbb{E}[h(\widetilde{x}^{s-1})] + \frac{1}{\theta_\eta}\mathbb{E}[f(\widetilde{x}^{s-1})] - \frac{1}{\theta_\eta}\mathbb{E}[f(x_{k-\tau_k}^s)] \\ &+ \frac{1}{2\theta_\eta L} \mathbb{E}\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2. \end{aligned} \tag{21}$$

The equality $\overset{a}{=}$ is obtained by definition of $\widetilde{\nabla}_k^s$ and rearranging terms. By using

$$\mathbb{E}\left\langle -\nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \nabla f(\widetilde{x}^{s-1}), \beta_s x^* + (1-\beta_s)\widetilde{x}^{s-1} - x_k^s \right\rangle = 0,$$

in (21), we have

$$\begin{aligned} &\mathbb{E}\left[F(x_{k+1}^s) - f(x_k^s)\right] \\ &\leq \mathbb{E}\left[\frac{\beta_s^2}{2\eta} (\|z_k^s - x^*\|^2 - \left\|z_{k+1}^s - x^*\right\|^2)\right] \\ &\quad + \mathbb{E}\left[\beta_s h(x^*) + (1-\beta_s) h(\widetilde{x}^{s-1})\right] \\ &\quad + \mathbb{E}\left[\left\langle \nabla f(x_{k-\tau_k}^s), \beta_s x^* + (1-\beta_s)\widetilde{x}^{s-1} - x_k^s \right\rangle\right] \\ &\quad + \mathbb{E}\left[\left\langle \nabla f(x_{k-\tau_k}^s), \theta_\eta^{-1}(x_{k-\tau_k}^s - \widetilde{x}^{s-1}) \right\rangle\right] \\ &\quad + \frac{1}{\theta_\eta} \mathbb{E}\left[f(\widetilde{x}^{s-1}) - f(x_{k-\tau_k}^s)\right] \\ &\quad + \frac{1}{2\theta_\eta L} \mathbb{E}\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2. \end{aligned} \tag{22}$$

Additionally, we have the following,

$$\begin{aligned} &\left\langle \nabla f(x_{k-\tau_k}^s), \beta_s x^* + (1-\beta_s)\widetilde{x}^{s-1} - x_k^s + \theta_\eta^{-1}(x_{k-\tau_k}^s - \widetilde{x}^{s-1}) \right\rangle \\ &= \left\langle \nabla f(x_{k-\tau_k}^s), \beta_s x^* + (1-\beta_s-\theta_\eta^{-1})\widetilde{x}^{s-1} \right\rangle \\ &\quad + \left\langle \nabla f(x_{k-\tau_k}^s), \theta_\eta^{-1} x_{k-\tau_k}^s - x_{k-\tau_k}^s + x_{k-\tau_k}^s - x_k^s \right\rangle \\ &\overset{a}{\leq} \beta_s f(x^*) + (1-\beta_s-\theta_\eta^{-1}) f(\widetilde{x}^{s-1}) + \theta_\eta^{-1} f(x_{k-\tau_k}^s) \\ &\quad - f(x_{k-\tau_k}^s) + \left\langle \nabla f(x_{k-\tau_k}^s) - \nabla f(x_k^s), x_{k-\tau_k}^s - x_k^s \right\rangle \\ &\quad + \left\langle \nabla f(x_k^s), x_{k-\tau_k}^s - x_k^s \right\rangle \\ &\overset{b}{\leq} \beta_s f(x^*) + (1-\beta_s-\theta_\eta^{-1}) f(\widetilde{x}^{s-1}) + \theta_\eta^{-1} f(x_{k-\tau_k}^s) \\ &\quad + \left\langle \nabla f(x_{k-\tau_k}^s) - \nabla f(x_k^s), x_{k-\tau_k}^s - x_k^s \right\rangle - f(x_k^s), \end{aligned} \tag{23}$$

where in inequalities $\overset{a}{\leq}$ and $\overset{b}{\leq}$ we used the convexity of f.

Since z_{k+1}^s is the optimal solution of proximal subproblem in the Algorithm 3, there exists $\xi_{k+1}^s \in \partial h(z_{k+1}^s)$ satisfying

$$\beta_s(z_{k+1}^s - z_k^s) + \eta\, \widetilde{\nabla}_k^s + \eta\, \xi_{k+1}^s = 0. \tag{24}$$

We next bound the term $\mathbb{E}[\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2]$ using the fact that gradient of the function f is Lipschitz continuous in the following manner,

$$\begin{aligned} &\mathbb{E}[\left\|\nabla f(x_k^s) - \nabla f(x_{k-\tau_k}^s)\right\|^2] \\ &\leq L^2 \tau \sum_{i=k-\tau}^{k-1} \mathbb{E}[\left\|x_{i+1}^s - x_i^s\right\|^2] \\ &\overset{a}{=} L^2 \tau \beta_s^2 \sum_{i=k-\tau}^{k-1} \mathbb{E}[\left\|z_{i+1}^s - z_i^s\right\|^2] \\ &\overset{b}{=} L^2 \tau \eta^2 \sum_{i=k-\tau}^{k-1} \mathbb{E}[\left\|\widetilde{\nabla}_i^s + \xi_{i+1}^s\right\|^2], \end{aligned} \tag{25}$$

and similarly,

$$\begin{aligned} &\left\langle \nabla f(x_{k-\tau_k}^s) - \nabla f(x_k^s), x_{k-\tau_k}^s - x_k^s \right\rangle \\ &\leq L\tau \sum_{i=k-\tau}^{k-1} \mathbb{E}[\left\|x_{i+1}^s - x_i^s\right\|^2] \\ &\overset{c}{=} L\tau \beta_s^2 \sum_{i=k-\tau}^{k-1} \mathbb{E}[\left\|z_{i+1}^s - z_i^s\right\|^2] \\ &\overset{d}{=} L\tau \eta^2 \sum_{i=k-\tau}^{k-1} \mathbb{E}[\left\|\widetilde{\nabla}_i^s + \xi_{i+1}^s\right\|^2], \end{aligned} \tag{26}$$

where in equalities $\overset{a}{=}$ and $\overset{c}{=}$ we use the definition for the updates of x_{i+1}^s in Async-AcPSVRG and equalities $\overset{b}{=}$ and $\overset{d}{=}$ follow from (24).

To proceed, we define the following quantities,

$$\begin{aligned} u_k^s &= \nabla f_{i_k^s}(x_{k-\tau_k}^s) - \nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \nabla f(\widetilde{x}^{s-1}) + \xi_{k+1}^s \\ &= \widetilde{\nabla}_k^s + \xi_{k+1}^s, \\ v_k^s &= \nabla f_{i_k^s}(x_k^s) - \nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \nabla f(\widetilde{x}^{s-1}) + \xi_{k+1}^s, \end{aligned}$$

where $\xi_{k+1}^s \in \partial h(x_{k+1}^s)$. By substituting equation (26) in (23) combined with (25), we obtain from (22)

$$\begin{aligned}\mathbb{E}[F(x_{k+1}^s)] \leq & \beta_s F(x^*) + (1-\beta_s)F(\widetilde{x}^{s-1}) \\ & + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_k^s - x^*\|^2 - \|z_{k+1}^s - x^*\|^2] \\ & + (1+\frac{1}{2\theta_\eta})L\tau\eta^2 \sum_{i=k-\tau}^{k-1}\mathbb{E}[\|u_i^s\|^2]. \end{aligned} \tag{27}$$

Equivalently, we have the following

$$\begin{aligned}\mathbb{E}[F(x_{k+1}^s) - F(x^*)] \leq & (1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ & + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_k^s - x^*\|^2 - \|z_{k+1}^s - x^*\|^2] \\ & + (1+\frac{1}{2\theta_\eta})L\tau\eta^2 \sum_{i=k-\tau}^{k-1}\mathbb{E}[\|u_i^s\|^2]. \end{aligned} \tag{28}$$

We next bound the term $\mathbb{E}[\|u_k^s\|^2]$ in terms of $\mathbb{E}[\|v_k^s\|^2]$ in the following,

$$\begin{aligned}\mathbb{E}[\|u_k^s\|^2] &\leq 2\mathbb{E}[\|u_k^s - v_k^s\|^2 + \|v_k^s\|^2] \\ &\leq 2\mathbb{E}\left[\left\|\nabla f_{i_k^s}(x_{k-\tau_k}^s) - \nabla f_{i_k^s}(x_k^s)\right\|^2\right] + 2\mathbb{E}\|v_k^s\|^2 \\ &\overset{a}{\leq} 2L^2\tau \sum_{i=k-\tau+1}^{k-1}\mathbb{E}\left\|x_{i+1}^s - x_i^s\right\|^2 + 2\mathbb{E}\|v_k^s\|^2 \\ &= 2L^2\beta_s^2\tau \sum_{i=k-\tau+1}^{k-1}\mathbb{E}\left\|z_{i+1}^s - z_i^s\right\|^2 + 2\mathbb{E}\|v_k^s\|^2 \\ &= 2L^2\tau\eta^2 \sum_{i=k-\tau+1}^{k-1}\mathbb{E}\left\|\widetilde{\nabla}_i^s + \xi_i^s\right\|^2 + 2\mathbb{E}\|v_k^s\|^2 \\ &= 2L^2\tau\eta^2 \sum_{i=k-\tau+1}^{k-1}\mathbb{E}\|u_i^s\|^2 + 2\mathbb{E}\|v_k^s\|^2, \end{aligned} \tag{29}$$

where in $\overset{a}{\leq}$ we use the Lipschitz continuity of the gradient. Adding the above inequalities from $k=0$ to m_s-1, we get

$$\begin{aligned}\sum_{k=0}^{m_s-1}\mathbb{E}\left[\|u_k^s\|\right]^2 &\leq \sum_{k=0}^{m_s-1}\left[2L^2\tau\eta^2\sum_{i=k-\tau+1}^{k-1}\mathbb{E}[\|u_i^s\|^2] + 2\mathbb{E}[\|v_k^s\|^2]\right] \\ &\leq 2L^2\tau^2\eta^2\sum_{k=0}^{m_s-1}\mathbb{E}[\|u_k^s\|^2] + 2\sum_{k=0}^{m_s-1}\mathbb{E}[\|v_k^s\|^2]. \end{aligned} \tag{30}$$

The last inequality follows from that fact that the delay is at most τ and in particular for each $\mathbb{E}[\|u_i^s\|^2]$, there are at most τ terms. From the above inequality, we get

$$\sum_{k=0}^{m_s-1}\mathbb{E}[\|u_k^s\|]^2 \leq \frac{2}{1-2L^2\tau^2\eta^2}\sum_{k=0}^{m_s-1}\mathbb{E}[\|v_k^s\|^2]. \tag{31}$$

Adding the inequality (28) from $k=0$ to $k=m_s-1$, we get

$$\begin{aligned}\sum_{k=0}^{m_s-1}\mathbb{E}[F(x_{k+1}^s) - F(x^*)] &\leq m_s(1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ &\quad + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s - x^*\|^2 - \|z_{m_s}^s - x^*\|^2] \\ &\quad + (1+\frac{1}{2\theta_\eta})L\tau^2\eta^2\sum_{k=0}^{m_s-1}\mathbb{E}[\|u_k^s\|^2] \\ &\overset{a}{\leq} m_s(1-\beta_s)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ &\quad + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s - x^*\|^2 - \|z_{m_s}^s - x^*\|^2] \\ &\quad + \frac{(2+\theta_\eta^{-1})L\tau^2\eta^2}{1-2L^2\tau^2\eta^2}\sum_{k=0}^{m_s-1}\mathbb{E}[\|v_k^s\|^2], \end{aligned} \tag{32}$$

where in inequality $\overset{a}{\leq}$ we used (31).

From Lemma 3 of [9] (see also [26]), we have

$$\mathbb{E}[\|v_k^s\|^2] \leq 4L\mathbb{E}[F(x_k^s) - F(x^*) + F(\widetilde{x}^{s-1}) - F(x^*)].$$

Substituting the above bound on $\mathbb{E}[\|v_k^s\|^2]$ in equation (32), we get the following

$$\begin{aligned}&(1 - \frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2})\sum_{k=0}^{m_s-1}\mathbb{E}[F(x_{k+1}^s) - F(x^*)] \\ &\leq m_s\left((1-\beta_s) + \frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2}\right)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ &\quad + \frac{\beta_s^2}{2\eta}\mathbb{E}[\|z_0^s - x_*\|^2 - \|z_{m_s}^s - x_*\|^2]. \end{aligned} \tag{33}$$

From convexity of function F and the definition $\widetilde{x}^s = \frac{1}{m_s}\sum_{k=0}^{m_s-1}x_{k+1}^s$, we get

$$F(\widetilde{x}^s) = F(\frac{1}{m_s}\sum_{k=0}^{m_s-1}x_{k+1}^s) \leq \frac{1}{m_s}\sum_{k=0}^{m_s-1}F(x_{k+1}^s).$$

From the above inequality and (33), we get the following

$$\begin{aligned}&\left(1 - \frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2}\right)\mathbb{E}[F(\widetilde{x}^s) - F(x^*)] \\ &\leq \left((1-\beta_s) + \frac{4(2+\theta_\eta^{-1})L^2\tau^2\eta^2}{1-2L^2\tau^2\eta^2}\right)[F(\widetilde{x}^{s-1}) - F(x^*)] \\ &\quad + \frac{\beta_s^2}{2\eta m_s}\mathbb{E}[\|z_0^s - x^*\|^2 - \|z_{m_s}^s - x^*\|^2]. \end{aligned} \tag{34}$$

□