LOSP: Overlap Synchronization Parallel with Local Compensation for Fast Distributed Training

Haozhao Wang, Zhihao Qu, *Member, IEEE* Song Guo, *Fellow, IEEE* Ningqi Wang, Ruixuan Li, *Member, IEEE* and Weihua Zhuang, *Fellow, IEEE*

***Abstract*—When running in Parameter Server (PS), the Distributed Stochastic Gradient Descent (D-SGD) incurs significant communication delays and huge communication overhead due to the model synchronization. Moreover, considering the heterogeneity of computational capability among workers, traditional synchronization modes incur under-utilization of computational resources because fast workers have to wait for slow ones finishing the computation. To tackle these issues, we propose a new synchronization mechanism, named LOSP, which overlaps computation and communication procedures in distributed training and introduces local compensation to mitigate adverse effects caused by such non-strict synchronization. The advantages of LOSP come from three aspects. First, LOSP removes the waiting time for model synchronization by conducting computation and communication in an overlapped manner. Second, LOSP allows multiple local updates on workers to fully exploit the computation capability and to significantly reduce the communication cost. Third, the training accuracy can be improved by local compensation that prohibits an excessive deviation of model update direction during multiple local updates. We theoretically prove that LOSP (1) preserves the same convergence rate as the sequential SGD for non-convex problems, and (2) exhibits good scalability due to the linear speedup property with respect to both the number of workers and the average number of local updates. Evaluations show that LOSP significantly improves performance over the state-of-the-art ones in terms of both convergence accuracy and communication cost.**

***Index Terms*—Overlap Synchronization Parallel, Parameter Server, Local Compensation, Distributed Machine Learning**

I. Introduction

MACHINE learning has demonstrated great promises in a wide range of application domains, e.g., self-driving, smart city, language processing, etc., which are fundamentally altering the way individuals and organizations live, work and interact [2]–[4]. With the rapid growth of training data and machine learning model size, how to efficiently train machine learning model in a distributed manner has received much attention since computation can be parallelized in multiple nodes.

Haozhao Wang is with School of Computer Science and Technology, Huazhong University of Science and Technology and Department of Computing, The Hong Kong Polytechnic University. E-mail: hz_wang@hust.edu.cn

Zhihao Qu is with the School of Computer and Information, Hohai University, and Department of Computing, The Hong Kong Polytechnic University. E-mail: quzhihao@hhu.edu.cn

Song Guo is with with the Department of Computing, The Hong Kong Polytechnic University. E-mail: song.guo@polyu.edu.hk

Ningqi Wang is with Information Networking Institute, Carnegie Mellon University. E-mail: nq.maigre@gmail.com

Ruixuan Li is with School of Computer Science and Technology, Huazhong University of Science and Technology. E-mail: rxli@hust.edu.cn

Weihua Zhuang is with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. E-mail: wzhuang@uwaterloo.ca

This is an extended revision of the early version appeared in [1]

A widely used framework in distributed machine learning is data parallelism over Parameter Server (PS) architecture, i.e., data are distributed over multiple workers and a global model is cooperatively optimized with the coordination of servers [5], [6]. As to the algorithm running in PS for solving training problems, distributed Stochastic Gradient Descent (D-SGD) is usually adopted because it is applicable to various model optimizations with proven efficiency in terms of scalability [5], [7], [8].

One of the popular synchronization modes for D-SGD in PS is Bulk Synchronous Parallel (*BSP*). Specifically, the learning process proceeds by iteratively running two procedures in a sequential manner, i.e., computing the local gradients and synchronizing the global model. In the computation procedure, each worker computes local gradient separately according to its own dataset. In the synchronization procedure, after receiving gradients from workers, the server aggregates the gradients and sends back the updated parameters of the global model for the next step computation. Although the gradients are computed simultaneously in all workers, the aggregation step needs the synchronization of all workers which would incur a significant delay. We conduct preliminary experiments to show the computation time and the synchronization time of *BSP*. As illustrated in Fig. 1, we have the observation that *BSP* suffers serious synchronization delay especially when the number of workers is large.

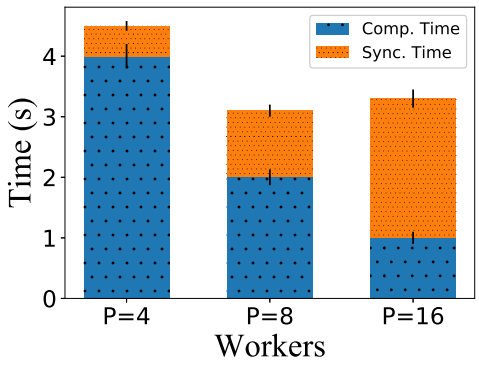


Fig. 1. The average time duration of computation and synchronization procedure in one iteration when training machine learning model ResNet20 on ImageNet. Each node in the cluster is equipped with 4 CPU cores and 10 G bps Ethernet interface card.

To reduce the synchronization delay, the mainstream is to design flexible synchronization mode. Considering the

synchronization delay is usually caused by slow workers, various synchronization modes have been proposed to reduce or eliminate the waiting time of fast workers [9]. *TAP* [10] allows each worker to push the gradient and pull the model from the server separately without synchronization. Though the *TAP* model is of low synchronization cost, it converges slowly due to the serious staleness problem. *SSP* [11], [12] establishes a framework of the *BSP* and *TAP* that makes a trade-off between convergence efficiency and synchronization efficiency. One major issue in these synchronization methods is that each worker still executes the computation and synchronization sequentially such that workers are idle from finishing the computation to receiving the updated global model from the server. Recently, it is observed that a higher computation efficiency can be achieved when the parameters in a neural network are sent layer by layer in parallel with computing [8], [13]. However, the workers in these methods are still idle when the synchronization process has a longer duration than the computation process.

To fully unleash the potential of a given distributed training system, we propose a framework named LOSP that overlaps the computation and synchronization. This work can be viewed as an extension of our previous work on OSP [1]. In LOSP, the overlap synchronization means that workers sequentially push updates and pull models while simultaneously calculating the gradient in a non-stop manner. In addition, to tackle the problem of slow workers, adaptively multiple local updates are allowed to fully exploit the computational capability of workers. Through these ways, not only the computing resource and communication resource can have nearly full utilization, but also the communication cost can be significantly reduced due to a reduced number of synchronization procedures.

However, the adverse effect of reducing the synchronization delay in such a way is that the gradients in each iteration are computed with respect to the stale global model of the last iteration, which yields a deviation from the correct direction of model update. Such deviation may cause an even more severe degradation on training accuracy when the number of local updates becomes large. Although traditional gradient compensation method [14] efficiently solves the deviation problem in the case of asynchronous mode, they bring extra computing cost and can hardly be applied to such overlapped mode when there exist multiple local updates. To tackle this problem, we devise a novel local compensation mechanism that prohibits an excessive deviation of local updates to further improve the training accuracy. The core idea of the mechanism is to compensate the stale model with the locally cached gradient to reduce its difference with the fresh model before computing the gradient. We revisit the theoretical analysis of the convergence rate and conduct extensive experiments to validate the feasibility and efficiency of the proposed method.

The main contributions of this paper are summarized as follows.

- We propose a novel synchronization framework, *LOSP*, that overlaps computation and communication procedures in distributed training and introduces local compensation to mitigate adverse effects caused by such non-strict synchronization. This method achieves high synchronization efficiency and significantly reduces the communication cost in large-scale distributed learning.
- We theoretically prove that the ergodic convergence of *LOSP* is the same as the sequential non-compression SGD. Meanwhile, LOSP has a linear speedup property with respect to both the number of workers and the average number of local updates, which demonstrates the efficiency and the scalability of LOSP.
- We conduct experiments on both convex and non-convex problems under clusters of different scales. The results show that *LOSP* improves the performance up to 1.6× for MnistCNN without losing the convergence result.

The rest of this paper is organized as follows. The related works about computation and communication efficiency in distributed training systems are introduced in Section 2. The problem formulation is presented in Section 3. Then, we describe *LOSP* in detail and analyze its convergence property in Section 4 and Section 5, respectively. In Section 6, performance evaluations are presented to show the efficiency of our proposed method. Finally, Section 7 concludes this paper.

II. Related Work

With the emerging of deep learning applications, machine learning systems for efficiently training machine learning models become imperative. To improve the training efficiency on a single machine, multiple works recently proposed reducing the time of memory access to accelerate the processing speed [15]–[18]. Though the performance improvement has been achieved, the performance still cannot satisfy the requirement for fast training models due to limited resources of a single machine. One of the most efficient ways of solving this issue is to extend the systems on single machine to distributed systems. Among all the distributed learning systems, Parameter Server [5], [6] is the most popular one and recently receives many attentions. However, since Parameter Server adopts a centralized paradigm to train the models, the server suffers from a huge communication delay when the number of workers is large. To solve the communication delay issue, many solutions are proposed recently.

To solve the communication bottleneck of Parameter Server, a straightforward method is to remove the server and group all workers into a graph. Recently, decentralized learning [19], [20] is proposed to train models without a centralized server. In this paradigm, all workers are organized in a form of graph and each worker only communicates with its neighbors. The machine learning models are synchronized in a gossip manner. The results in [19] show that decentralized learning achieves a comparable convergence result with Parameter Server but has less communication overhead.

Following the architecture of Parameter Server, another way to mitigate the communication issue is reducing the synchronization frequency [21]–[26]. Yu et al. [21] propose a method that runs multiple local updates in each worker before aggregating the models. Further, other approaches [23]–[25] adaptively adjust the number of local updates to balance the communication cost and the computation cost. Xinchen et.al. [26] proposes a method that guarantees the evenness of the data to reduce the overall synchronization times.

Different from the above methods that directly reduce the communication delay, there are some approaches to improve computation efficiency from the perspective of the relationship between computation and communication. This type of methods improve the training efficiency mainly by increasing the ratio of computation to communication. One method is to adopt a large batch size [27]–[29] to increase the computing efficiency. However, a large batch size is mostly an empirical method and can lead to accuracy reduction [30]. Another method is to break the synchronization of all workers, e.g., SSP [11], [12], [31], partial synchronization [32]–[34], and *TAP* [10], [35] such that the communication burst issue of forced synchronization can be mitigated and the corresponding computation-communication ratio is significantly increased. The extreme case is the one-shot synchronization [36] in which all workers are only synchronized one time.

Furthermore, there are many works that accelerate the distributed learning system [37]–[40] via optimizing the network. Qihua et.al. [38] proposes scheduling the data flow in the network to reduce the transmission time of model parameters. For the wireless network, Deniz et.al. [39] proposes a method to improve the coding and decoding effciency from the physical layer. Amedeo et.al. [40] proposes aggregating the parameters on the switch to reduce the communication data size in the network.

Different from the above methods, this paper takes one step ahead along the approach of parallelizing computation and communication in each worker. Although some recent works focus on overlapping the computation and communication [41], [42], they are totally different from ours. Assran et al. [41] design an asynchronous overlapping method for the decentralized architecture, while we propose an overlapping scheme for Parameter Server architecture with periodical synchronization. Jianyu et al. [42] propose an overlapping method for the Parameter Server system. However, they focus on solving the straggler problem in the system by adding an anchor model on each worker. Instead, we in this paper introduce the local compensation to mitigate the adverse effect of staleness.

III. Problem Definition and Preliminaries

For clarity, main symbols used in this paper are summarized in Table I.

In this paper, we focus on solving the sum optimization problem which is typically used in machine learning field. Specifically, let $f_i(\omega)$ be the loss of model parameter ω fitting the i-th data sample of the training set. Then, the objective of is to find a model ω that minimizes the fitting loss of all data samples in the training set, i.e.,

$$\min \ F(\omega) = \frac{1}{n}\sum_{i=1}^{n} f_i(\omega), \tag{1}$$

where n denotes the number of data samples.

The objective function in (1) is complicated since it is defined as the average loss of the parameter vector from the whole datasets. For most learning models, it is impossible to derive a closed-form solution to this optimization problem. Therefore, gradient descent based optimization methods are widely used to solve this problem, which could iteratively converge to the desired parameter under the meticulously designed training process. For various machine learning mode, SGD methods are applicable to solve this problem efficiently. Instead of using the whole dataset to compute the gradient, SGD randomly selects a mini-batch of data samples ξ and computes stochastic gradient $g(\omega;\xi)$ with the model ω. Compared to the batch method, the stochastic method is more efficient for large-scale learning problems according to practical and theoretical analysis. In SGD, the model update in iteration $t, t \in \mathbb{N}$ is formulated as:

$$\omega_{t+1} = \omega_t - \eta_t g(\omega_t;\xi_t), \tag{2}$$

TABLE I
A SUMMARY OF MAIN MATHEMATICAL SYMBOLS

Symbol	Definition
P	The number of workers
K_t^i	The number of local updates in t-th global iteration
ω_t	Parameter of t-th global iteration
$\omega_{t,k}^i$	Local parameter of worker i in k-th inner iteration of t-th global iteration
η_t	Learning rate in t-th global iteration
ξ_t^i	Mini-batch of Samples selected by worker i in iteration t
$g(\omega;\xi)$	Gradient with respect to samples ξ and parameter ω
G_t^i	Accumulated gradients of worker i in iteration t
$\nabla F(\omega)$	Full gradient with respect to the whole dataset
σ^2	Variance bound of stochastic gradient
M	Variance bound of the number of inner updates of workers
L	Lipschitz constant
$\omega_\star$	The optimal solution of the problem
d	The number of dimensions of the gradient vector
s	Quantization level

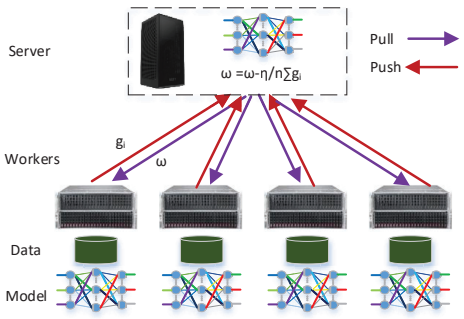


Fig. 2. Illustration of Parameter Server for Distributed Machine Learning.

Parameter Server (PS) architecture is one of the most popular learning paradigm for large scale machine learning. In PS, data are distributed over multiple workers and a global model are cooperatively optimized with the coordination of servers [5], [6]. As to the algorithm running in PS for solving training problems, distributed Stochastic Gradient Descent (D-SGD) is usually adopted. The typical PS based learning architecture and the distributed learning algorithm are illustrated in Fig. 2. In each iteration of D-SGD with BSP mode, each worker i independently computes the local stochastic gradient and pushes the gradient to the server. After receiving gradients

from all workers, the server computes the average of them and updates the global model as following.

$$\omega_{t+1} = \omega_t - \frac{\eta_t}{P}\sum_{i=1}^{P} g(\omega_t^i; \xi_t^i), \tag{3}$$

where P is the number of workers. Then, the model synchronization procedure distributes the updated model to all workers for the next step computation. Traditional BSP mode will introduce huge communication cost and large synchronization delay with the ever-increasing size of machine models and the constraint of bandwidth. It is unrealistic for large-scale distributed machine learning systems. Consequently, we focus on deriving an efficient synchronization mechanism with low communication complexity for solving the problem (1) with D-SGD.

IV. The Framework of Overlap Synchronous Parallel with Local Compensation

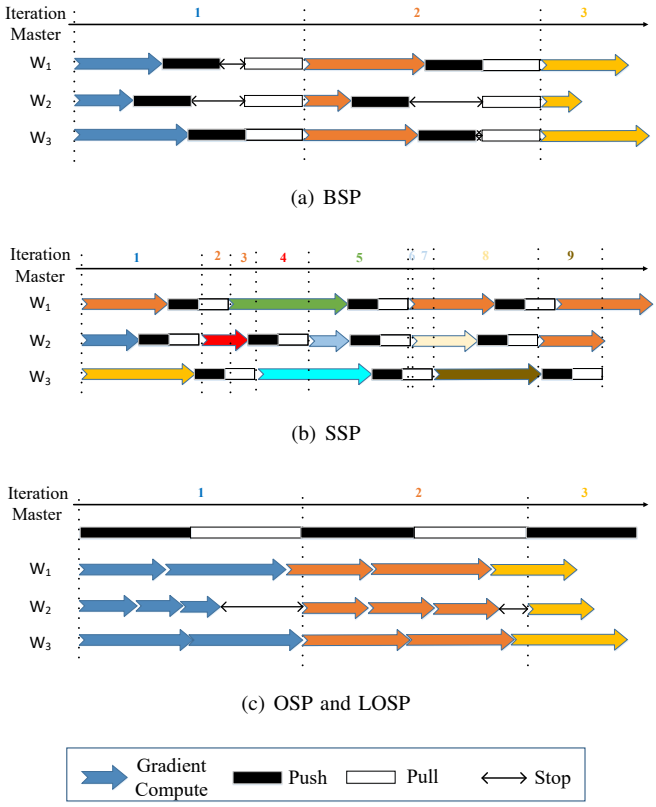
In this section, we propose an efficient distributed machine learning framework, LOSP, that applies local compensation in overlap synchronous parallel. By overlapping the computation and communication processes in distributed training, *LOSP* can significantly the waiting time of works caused by synchronization delay. Moreover, the number of communication rounds is significantly reduced due to multiple local updates. We firstly elaborate the design principle of *LOSP* based D-SGD in the PS architecture and then present the detailed algorithm design.

A. Overlap Synchronization with Local Compensation

The model updating process of D-SGD runs iteratively and has two steps in each iteration, i.e., computing gradient by each work and synchronizing the global model by the server. These two steps are executed sequentially in traditional parallel modes, e.g., *BSP* and *SSP*. Examples are shown in Figure 3(a) and Figure 3(b). Each worker firstly computes and transmits gradient to the server. Then, the server updates the global model according to the aggregation of gradients and transmits the updated models to each work. With the rapid growth of computing capacity of high-performance processing units, e.g., Graphics Processing Units (GPUs) and Neural Processing Units (NPUs), the synchronization procedure that mainly relies on network bandwidth becomes the main bottleneck in distributed machine learning.

Overlap. The two steps can de-facto be parallel as well such that the large delay incurred by synchronization can be totally diminished. We first propose the Overlap Synchronization Parallel (*OSP*) where the processes of local gradient computation and model synchronization are overlapped. Specifically, each worker creates two threads $\mathcal{P}$ and $\mathcal{M}$ that are responsible for computation and communication, respectively.

At the beginning of the $(t-1)$-th iteration, both the server and the workers hold the same model ω_{t-1}. The communication thread $\mathcal{M}$ in each worker pushes the computed gradient $g(\omega_{t-2}, \xi_{t-2}^i)$ obtained in the $(t-2)$-th iteration to the server.



(a) BSP

(b) SSP

(c) OSP and LOSP

(d) Legend

Fig. 3. Four different parallel models of D-SGD in parameter server. Workers of *OSP* and *LOSP* compute gradient and make synchronization in parallel. Fast workers will keep computing gradients until reaching a number, e.g., 3, of iterations. Specially, the accumulated gradients are quantized for transmission to further accelerate the training process.

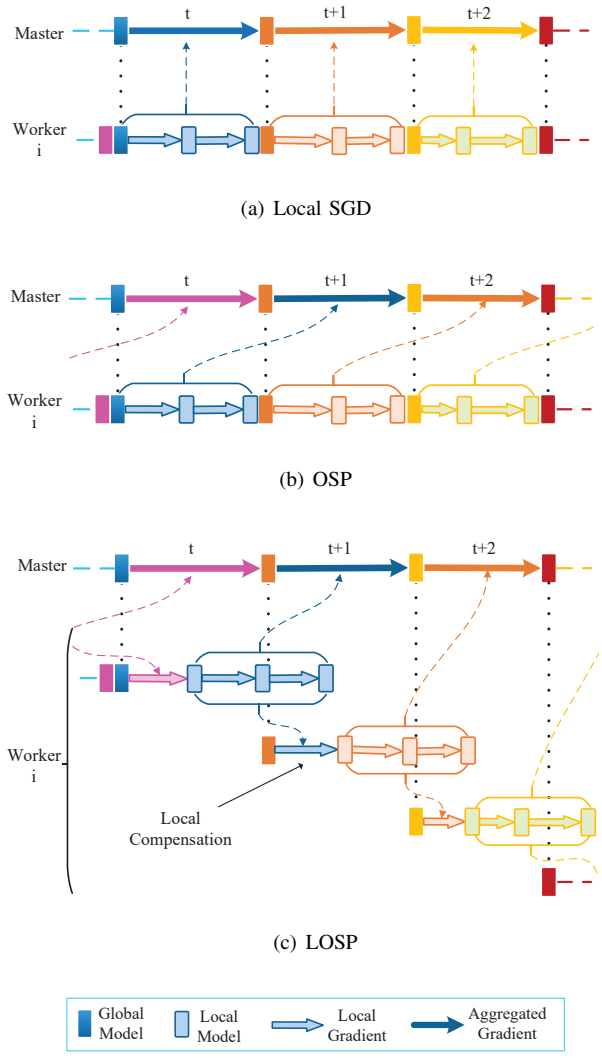
After receiving all gradients from all workers, the server updates the global model by

$$\omega_t = \omega_{t-1} - \frac{\eta_{t-1}}{P}\sum_{i=1}^{P} g(\omega_{t-2}, \xi_{t-2}^i), \tag{4}$$

and then returns it back. At the same time, the computation thread $\mathcal{P}$ in each worker i computes the local gradient $g(\omega_{t-1}, \xi_{t-1}^i)$ with respect to the model ω_{t-1}. Once finishing the overlapped two steps, the system goes into the next iteration. Likewise, the communication thread $\mathcal{M}$ updates the global model ω_t with the gradient obtained in the $(t-1)$-th iteration:

$$\omega_{t+1} = \omega_t - \frac{\eta_t}{P}\sum_{i=1}^{P} g(\omega_{t-1}, \xi_{t-1}^i). \tag{5}$$

Local Compensation. Although the overlapping method significantly reduces the overhead caused by communication, it incurs a staleness issue. By comparing the update formula (5) of *OSP* and the update rule (3) of *BSP*, it can be easily concluded that the gradient used in *OSP* is calculated from the stale model ω_{t-1} instead of the newest model ω_t, yielding a gap between the used model update of *OSP* and the desired one. Specially, when there exist multiple local updates in each iteration as specified later, the gap between two models is amplified, resulting in the worse performance.



(a) Local SGD

(b) OSP

(c) LOSP

(d) Legend

Fig. 4. Illustration of the differences among Local SGD, OSP and LOSP.

To fill this gap, we refine the *OSP* to *LOSP* by introducing a local compensation, as shown in Fig. 4. The core idea of *LOSP* is that each worker approximates the current global model ω_t using the received ω_{t-1} and its historical gradients. Specifically, the difference is $\frac{\eta_{t-1}}{P}\sum_{i=1}^{P} g(\omega_{t-2}, \xi_{t-2}^i)$ which is the average of all gradients as presented in (4). Fortunately, each worker i holds the local gradient $g(\omega_{t-2}, \xi_{t-2}^i)$ which can be exactly utilized to approximate the difference. Accordingly, we propose using the local gradient to fill the gap caused by the overlap. At the beginning of the $(t-1)$-th iteration, each worker i firstly compensates the received model ω_{t-1} with the locally cached gradient $g(\omega_{t-2}, \xi_{t-2}^i)$ as

$$\omega_{t-1,0}^i = \omega_{t-1} - \gamma\eta_{t-1} g(\omega_{t-2}, \xi_{t-2}^i), \tag{6}$$

where γ is the compensated step size that controls the proximity between the local gradient and the global average. Then, each worker computes the gradient $g(\omega_{t-1,0}^i, \xi_{t-1}^i)$ and sends it to the server. After receiving all gradients, the global model ω_t in the t-th iteration is updated by

$$\omega_{t+1} = \omega_t - \frac{\eta_t}{P}\sum_{i=1}^{P} g(\omega_{t-1,0}^i, \xi_{t-1}^i). \tag{7}$$

The *LOSP* has better performance than *OSP* because the model used by *LOSP* for computing the gradient is $\omega_{t-1,0}^i$, being much closer to ω_t than ω_{t-1} used by *OSP*. The principle behind the method is general fact for continuous function that two closer points have closer functional values.

Allowing Adaptive Local Updates. Considering that the communication process generally has a longer duration than the computation process, we further relax the strict synchronization rule by allowing adaptive local updates specific for the overlap method to fully leverage the computing resources. The proposed scheme is shown in the Fig.3(c) and Fig.4(c). In the computation thread $\mathcal{P}$, the gradients $g(\omega_{t-1,k}^i, \xi_{t-1,k}^i)$ are continuously computed based on the local model while the local model $\omega_{t-1,k}^i$ is updated with the gradients

$$\omega_{t-1,k+1}^i = \omega_{t-1,k}^i - \eta_{t-1} g(\omega_{t-1,k}^i, \xi_{t-1,k}^i), \tag{8}$$

where $k \in \mathcal{N}$ denotes the index of local iteration and $\omega_{t-1,0}^i$ is the compensated model. The local updating process runs until the number of local iterations reaching a bounded number τ, or the worker pulling a new global model from the server. At the same time, the gradients are accumulated locally. Denoting the total number of local iterations in worker i as K_t^i, the accumulated gradient can be written as

$$G_t^i = \sum_{k=0}^{K_t^i - 1} g(\omega_{t-1,k}^i, \xi_{t-1,k}^i). \tag{9}$$

In parallel with the computation thread, the communication thread pulls the updated model ω_t from the server and pushes the locally accumulated gradient G_t^i to the server. The global model in the server is thus updated by

$$\omega_{t+1} = \omega_t - \eta_t \frac{1}{P}\sum_{i=1}^{P} G_t^i, \tag{10}$$

and the local model is compensated in the worker by

$$\omega_{t,0}^i = \omega_t - \gamma\eta_t G_t^i. \tag{11}$$

The differences among *BSP* with multiple local updates, namely *Local SGD*, *OSP*, and *LOSP* are illustrated in Fig. 4. From this figure, we can find that *OSP* updates the global model with the gradient calculated under the stale model. *OSP* mitigates this issue by adding a local compensation in each iteration such that its convergence efficiency is close to the *Local SGD* but has higher communication efficiency.

Note that the computing cost of the synchronization step is negligible and thus little additional delay is introduced in the computation step. By elaborating the policy of local updates, i.e., adjusting the bounded number of iteration τ, the computation capability of workers will be fully exploited. Besides, the number τ in computation thread is not the same as the staleness threshold in *SSP*. The staleness threshold in *SSP* is designed to limit the lag between the local version of model in worker and the global version of model in server. Gradients

computed by the stale model have a significant impact on convergence properties, which has been widely investigated [9], [14], [43]. Here the number of bounded iterations, τ, is used to restrict the number of local updates in a worker, aiming to reduce the variance of local gradients and the diversity of searching directions between workers to guarantee the convergence rate of the algorithm.

B. Algorithm Description

The workflow of *LOSP* is shown in Algorithm 1. Each worker i creates two parallel threads $\mathcal{P}$ and $\mathcal{M}$ that are responsible for computation and communication respectively. In each iteration, Thread $\mathcal{M}$ pulls the current global model ω_t and then pushes the local accumulated gradient G_t^i to the server. Correspondingly, Thread $\mathcal{P}$ replaces its local model $\omega_{t,k}^i$ with global model ω_t and compensates it with locally accumulated gradient G^i as soon as the global model is pulled back. Otherwise, Thread $\mathcal{P}$ will update local model $\omega_{t,k}^i$ with the gradient calculated by itself. As to the server, it only interacts with Thread $\mathcal{M}$ of workers. It updates the model with the aggregated local updates G and then sends the updated model back to all workers.

Algorithm 1: LOSP: Overlap Synchronization Parallel with Local Compensation

Input : the learning rate η_t, the compensated step size γ, and the iteration threshold τ
Output : Final parameter ω

In worker i=1,...,P:
Initialize: $k = 0$, $isPull = false$, $G_0^i = 0$
Parallel *Thread* $\mathcal{M}$ and *Thread* $\mathcal{P}$:
Communication Thread $\mathcal{M}$*:*
repeat
  pulls ω_t from the server
  if G_{t+1}^i *is not* 0 **then**
    copies the local update $G^i \leftarrow G_{t+1}^i$
    sets the pulled flag $isPull \leftarrow True$
    clears the local update $G_{t+1}^i \leftarrow 0$
    pushes local update G^i to the server
  end
until $Convergence$;
Computation Thread $\mathcal{P}$*:*
repeat
  if $k \geq \tau$ **then**
    wait until $isPull$ is $True$
  end
  if $isPull$ *is* $True$ **then**
    resets the flag $isPull \leftarrow false$
    updates local model $\omega_{t,k}^i \leftarrow \omega_t - \gamma\eta_t G^i$
    resets the local clock $k \leftarrow 0$
    updates global clock $t \leftarrow t + 1$
  end
  computes stochastic gradient $g_{t,k}^i \leftarrow \nabla f_\xi(\omega_{t,k}^i)$
  accumulates the local gradients $G_t^i \leftarrow G_t^i + g_{t,k}^i$
  updates local model $\omega_{t,k+1}^i \leftarrow \omega_{t,k}^i - \eta_t g_{t,k}^i$
  updates local clock $k \leftarrow k + 1$
until $Convergence$;
In server:
Initialize: $t = 0$, initialize ω_0 randomly
repeat
  for $i = 1$ ***to*** P **do**
    receives G_t^i from worker i
  end
  updates global model $\omega_{t+1} \leftarrow \omega_t - \frac{\eta_t}{P}\sum_{i=1}^{P} G_t^i$
  updates global clock $t \leftarrow t + 1$
  broadcasts ω_t to all workers
until $Convergence$;

V. Theoretical Analysis

Since the convergence efficiency of *LOSP* is affected by both overlap synchronization and local compensation, we provide the theoretical analysis of the convergence rate to validate the efficiency of our method. According to the theoretical results, the *LOSP* essentially has the same convergence rate as non-compression SGD, but significantly improves the synchronization efficiency and reduces the communication cost. For the clarity of theoretical analysis, we first introduce some assumptions for SGD-based optimization and overlap settings.

A. Assumptions for LOSP

Similar to previous studies [44], we make the following assumptions, which are widely adopted for analyzing the convergence property of D-SGD.

Assumption 1. *Assumptions for D-SGD in machine learning optimization:*

(1) ***Lipschitz smooth***. *The objective function $F : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and the gradient function of F is Lipschitz continuous with Lipschitz constant $L > 0$, i.e.,*

$$\|\nabla F(\omega) - \nabla F(\tilde{\omega})\|_2 \leq L\|\omega - \tilde{\omega}\|_2$$

for all ω, $\tilde{\omega} \in \mathbb{R}$;

(2) ***Bounded loss***. *The sequence of iterations ω_t is contained in an open set over which F is bounded below by a scalar F_{inf};*

(3) ***Unbiased gradient with bounded variance***. *The stochastic gradient $g(\omega;\xi)$ computed from random samples ξ is unbiased for every parameter ω, i.e.,*

$$\mathbb{E}_\xi[g(\omega;\xi)] = \nabla F(\omega).$$

The variance of stochastic gradient is bounded

$$\mathbb{E}_\xi(\|g(\omega;\xi) - \nabla F(\omega)\|^2) \leq \sigma^2$$

where σ^2 is a constant.

Next, we introduce two conditions for each iteration of *LOSP*, i.e., the constant average number of local updates and bounded variance of the number of local updates. In practical systems, these two conditions can be easily satisfied given that the computational resources of the whole cluster are almost stable and robust.

Assumption 2. *Conditions for the number of local updates:*

(1) *The average number of local updates* $\bar{K} = \frac{1}{P}\sum_{i=1}^{P} K_t^i$ *does not change for each iteration* $t = 1, 2, \ldots, N$;
(2) *The variance of the number of local updates is bounded for each iteration* $t = 1, 2, \ldots, N$, *i.e.,*

$$\frac{1}{P}\sum_{i=1}^{P} \|K_t^i - \bar{K}\|_2^2 \leq M.$$

B. Convergence Rate of LOSP

As specified in the Section IV, the update rules of *LOSP* can be summarized as

$$\begin{aligned}
&\omega_{t-1,0}^i = \omega_{t-1} - \gamma\eta_{t-1}G_{t-1}^i,\\
&\omega_{t-1,k+1}^i = \omega_{t-1,k}^i - \eta_{t-1}g(\omega_{t-1,k}^i, \xi_{t-1,k}^i), k = 0, \ldots, K_t^i - 1,\\
&G_t^i = \sum_{k=0}^{K_t^i-1} g(\omega_{t-1,k}^i, \xi_{t-1,k}^i),\\
&\omega_{t+1} = \omega_t - \frac{\eta_t}{P}\sum_{i=1}^{P} G_t^i.
\end{aligned}$$

Jointly leveraging the above assumptions, we derive the convergence properties of *LOSP* without assuming the convexity of the loss function.

Theorem 1. *(LOSP, Nonconvex objective, fixed stepsize) Suppose algorithm runs with fixed learning rate* $\eta_t = \bar{\eta}$ *satisfying*

$$\bar{\eta}LK_{max}(1 + 3\bar{\eta}LK_{max}/2 + 3\bar{\eta}L\bar{K} + 3\bar{\eta}L\gamma^2K_{max} - 3\bar{\eta}L/2) \leq 1,$$

where $K_{max} \leq \tau$ *is defined as* $\max\{K_t^i,\ t = 1, 2, \ldots, N$ *and* $i = 1, 2, \ldots, P\}$. *The expected average squared gradient norms of* F *satisfy the following bound for all* $N \in \mathbb{N}$*:*

$$\frac{1}{N}\sum_{t=1}^{N}\mathbb{E}\|\nabla F(\omega_t)\|_2^2 \leq \frac{2(F(\omega_1) - F(\omega_\star))}{N\bar{\eta}\bar{K}} + \bar{\eta}L\sigma^2\Big(\frac{3\bar{\eta}LM}{2} + \frac{3\bar{\eta}L\bar{K}^2}{2} + 3\bar{\eta}L\gamma^2(\bar{K}^2 + M) - \frac{3\bar{\eta}L\bar{K}}{2} + \frac{3\bar{\eta}L\bar{K}^2}{P} + \frac{\bar{K}}{P}\Big). \quad (12)$$

Proof: See appendix. ∎

From equation (12), we know that *LOSP* converges to a non-zero constant under a fixed learning rate as $N \to \infty$. The final non-zero constant mainly comes from the second term because the first term being the initial distance approaches to zero gradually. As we can see, the second term is unrelated to the number of iterations N and it diminishes with the learning rate η. To show the relationship between the convergence result and the number of iterations, we have the following corollary.

Corollary 2. *Under the condition of Theorem 1, if we set*

$$\bar{\eta} = \sqrt{\frac{(F(\omega_1) - F(\omega_\star))P}{\bar{K}L\sigma^2N}}, \quad (13)$$

then for any iteration number

$$N \geq Max(A_2, A_4), \quad (14)$$

the output of Algorithm 1 satisfies the following ergodic convergence rate

$$\frac{1}{N}\sum_{t=1}^{N}\mathbb{E}\|\nabla F(\omega_t)\|_2^2 \leq 4\sqrt{\frac{(F(\omega_1) - F(\omega_\star))L\sigma^2}{\bar{K}P}} * \frac{1}{\sqrt{N}}, \quad (15)$$

where

$$\begin{aligned}
A_1 &= \frac{3LM}{2} + \frac{3L\bar{K}^2}{2} + 3L\gamma^2(\bar{K}^2 + M) - \frac{3L\bar{K}}{2} + \frac{3L\bar{K}^2}{P},\\
A_2 &= \frac{2(F(\omega_1) - F(\omega_\star))A_1^2P^3}{\bar{K}^3L\sigma^2},\\
A_3 &= 3K_{max}/2 + 3\bar{K} + 3\gamma^2K_{max} - 3/2,\\
A_4 &= \frac{K_{max}P(F(\omega_1) - F(\omega_\star))}{2\bar{K}\sigma^2} \cdot \Big(\sqrt{LK_{max}} + \sqrt{LK_{max} + 4A_3}\Big)^2.
\end{aligned}$$

Proof: See appendix. ∎

Corollary 2 shows that the average squared gradient converges to zero with a speed $O(\frac{1}{\sqrt{NP}})$ as the learning rate diminishes. The convergence rate of *LOSP* is in the same order as the sequential SGD when the iteration number N is sufficiently large. Besides, such convergence speed also indicates a linear speedup in terms of the number of workers. Though the convergence rate of *LOSP* is similar to that of *OSP*, its performance has a significant improvement empirically.

VI. Performance Evaluation

In this section, we present extensive experiments to evaluate both *OSP* and *LOSP*.

A. Experimental Setup

Baseline. We compare our methods with the following popular methods:

- Bulk Synchronous Parallel (*BSP*). *BSP* synchronizes all workers in each iteration. Specifically, the server updates the global model only when it receives gradients from all workers.
- K Batch Asynchronous Parallel (*KBatchAsync*) [45]. *KBatchAsync* can be viewed as a general version of Total Asynchronous Parallel (*TAP*) [10]. In each iteration, the server updates the global model after receiving any K mini-batches of gradient from all workers. *TAP* can tolerate stragglers when compared to *BSP*.
- Stale Synchronous Parallel (*SSP*) [11], [12], [46]. *SSP* is also an asynchronous parallel model but controls the maximum staleness of *TAP*. Specifically, the fast workers are forced to stop computing as they exceed the slowest worker by a given number of steps.
- Local-SGD [47] is an extension of *BSP* model in which each worker computes multiple local iterations before computing average of all workers.
- CoCoD-SGD [48] is an extension of Local-SGD but decouples the computation and communication.

It can be observed that the first three methods are various synchronization modes and the last two methods are convergence optimization techniques. As a consequence, we firstly compare our designed method to the first methods in terms of the synchronization efficiency, i.e., using *OSP* with only synchronization technique. Then, we compare the *OSP* with

local compensation, i.e., *LOSP*, to the last two methods in terms of convergence efficiency.

Datasets and Models. To guarantee the generality of evaluations, the experiments include both convex and non-convex models. Specifically, we adopt the Logistic Regression (LR) the deep neural network as the convex model and the non-convex model, respectively. Besides, the size of the machine learning models range from small to large.

- Mnist dataset [49]. It is composed of $60K$ 28×28 training images and has 10 classes. On top of the dataset, we run the LR model and the popular MnistCNN [50].
- Cifar10 dataset [51]. It includes $50K$ 32×32 training images and $10K$ test images and has 10 classes. We run the deep neural network ResNet34 [52] and AlexNet [53] on this dataset.
- Cifar100 dataset [51]. The dataset also consists of $50K$ 32×32 training images and $10K$ test images but has 100 classes. The neural network DenseNet121 [54] is executed on this dataset.
- Penn Treebank corpus dataset (PTB) [55]. The dataset consists of $923,000$ training and $82,000$ test words. We run a 2-layer LSTM [56] on it.

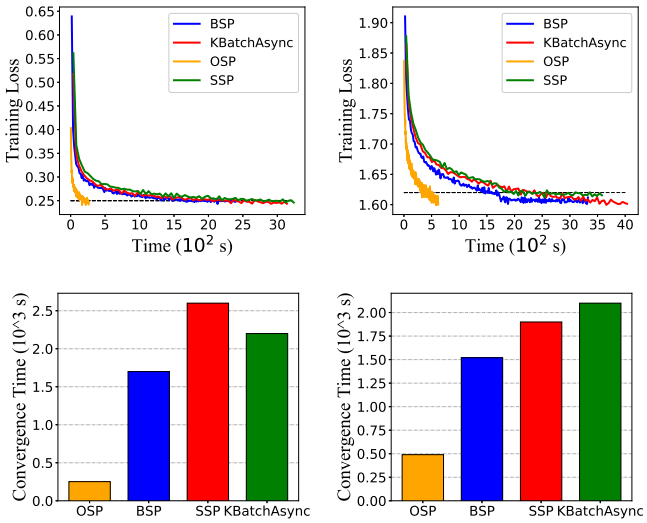
Clusters. For generality, we conduct experiments on both CPU and GPU clusters. Specifically, we build four types of clusters. *1) Cluster A*. It includes 8 Virtual Machines (VMs) as 8 workers and 1 VM as the server. Each VM is configured with 4 cores (2.6GHz) and 8GB RAM, and all VMs are connected by 10-Gbps Ethernet. *2) Cluster B*. It uses the same configurations as cluster A but consists of 16 VMs as workers. *3) Cluster C*. It includes 9 VMs and each VM is configured with a GTX 1080TI GPU. We employ one of the VMs as the server and others as the workers. The VMs are connected through the links with 10 Gbps bandwidth. *4) Cluster D*. The cluster is built on a real commercial cloud, i.e., Aliyun [57]. It consists of 9 homogeneous VMs including 8 workers and 1 server, in which each node is equipped with a Tesla P4 GPU and all VMs are connected through the 100 Mbps links. The main difference between cluster C and D is that their network bandwidth is different, which simulates different practical settings. We implement all methods on PyTorch and run them on top of 64-bit Centos 7.1 operating system. The source code can be found in [58].

Metrics. We adopt two general metrics, i.e., convergence speed and convergence rate, to evaluate the methods. The convergence speed is to measure the time required to reach some given error threshold. Similarly, the convergence rate is to measure the iterations required to reach some given error threshold. Beside of the two metrics, we in this paper also investigate the impact of our method on the computing resource usage by the ratio of computing time to the total execution time.

B. Experimental Results

1) Impact of Synchronization Mechanism: We firstly present the results on convex models and then non-convex models.

Results on Convex Problems. The mini-batch size and the learning rate of all methods are set to be $B = 200$ and $\eta = 0.01$ respectively. Besides, the staleness threshold of *SSP* and the batches of *KBatchAsync* are set to 16 and $K = 8$, respectively, which are the best setting in our tests. In the following figures, the black dashed line is an auxiliary line that indicates the convergence threshold of the training loss. It can help to clearly show the convergence speed, i.e., time duration or the number of iterations required to reach a given training loss.



(a) LR on Mnist (b) LR on Cifar10

Fig. 5. Comparison of different synchronization mechanisms for convex problems on the cluster B.

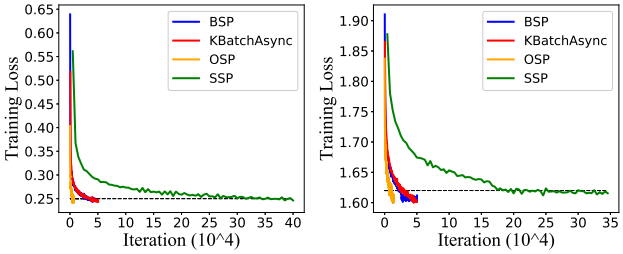
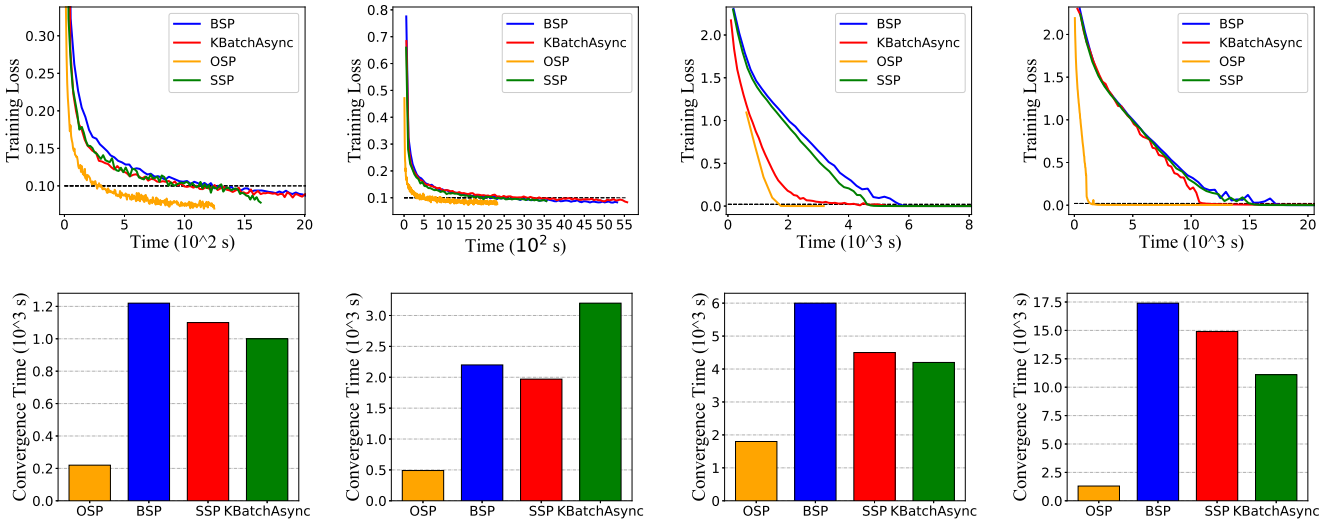


Fig. 6. Comparison of different parallel mechanisms for convex problems on the cluster B.

The results of convergence speed of using LR are shown in Fig. 5. Besides, our method *OSP* also outperforms the other methods in terms of the convergence efficiency. When reaching convergence for LR on Mnist and Cifar10, *BSP* costs nearly $1600s$ and $1500s$ that are far from the performance of *OSP* which only costs $250s$ and $400s$. Besides, in our tests, *BSP* even performs better than asynchronous parallel models, i.e., *SSP* and *KBatchAsync*. This phenomenon arises because the clusters used in the experiments are special-purpose that has little chances of having stragglers.

The results of convergence rate of using LR are shown in Fig. 6. We observe that *OSP* achieves a better converence quality than *BSP*. In fact, this is because each worker of *OSP* iterates more local steps in each global iteration than that of *BSP*, which is also verified theoratically in Corollary 2.



(a) MnistCNN on the cluster A. (b) MnistCNN on the cluster B. (c) AlexNet on the cluster A. (d) AlexNet on the cluster B.

Fig. 7. Convergence speed comparison of different parallel mechanisms.

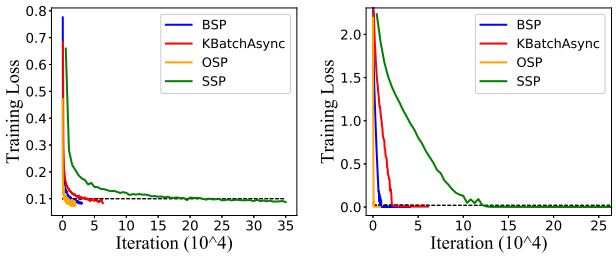


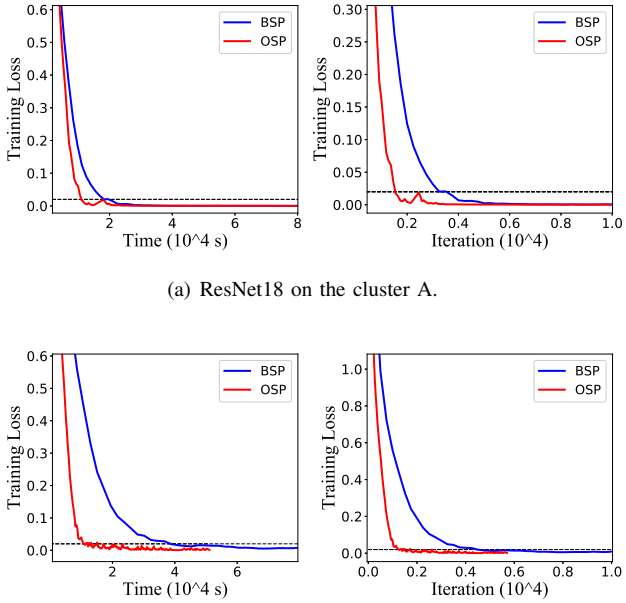
Fig. 8. Comparison of different parallel mechanisms for non-convex models on the cluster B.

Besides, both *SSP* and *KBatchAsync* have poorer convergence performance than *BSP*, not to mention in comparison with our method *OSP*.

Results on Non-convex Problems. The mini-batch size of all methods is set to$B = 200$. The learning rate is set to0.1 initially and decayed by multiplying 0.1 in $50th$ epoch ($25000th$ iteration for *BSP*). The configuration of the staleness threshold for *SSP* and the number of batches for *KBatchAsync* are the same as that in the convex case.

The results for convergence speed of non-convex models are shown in Fig. 7. Similar to convex models, *OSP* performs the best in all cases. Taking MnistCNN on 16-workers cluster as an example, the time reaching convergence required by *OSP* is approximate $500s$ which is only a quarter of best baseline, i.e., approximately $2000s$ of *SSP*. The difference from convex problems is that *BSP* performs worse than asynchronous methods in most cases. *BSP* causes communication burst in synchronizing phase that is more severe when the quantity of parameters in machine learning models is large.

The results for convergence rate of non-convex models are shown in Fig. 8. *SSP* performs the worst due to the staleness of the gradient and the smaller batch size than the other methods.



(a) ResNet18 on the cluster A.

(b) ResNet18 on the cluster B.

Fig. 9. Convergence speed comparison of different parallel mechanisms.

Similar to the results in Fig. 6, *BSP* performs better than *SSP* and *KBatchAsync*. However, it is still worse than *OSP*.

2) Impact of Synchronization Mechanism on Computing Resource Usage: In this section, we investigate if our method *OSP* can improve the computing resource usage of the cluster. Computing resource usage is presented in terms of the ratio of computing time to total time. Denote the computing time of worker i as t_i and the total time of worker i as T_i, then the computing resource usage is

$$\frac{\sum_{i=1}^{P} t_i}{\sum_{i=1}^{P} T_i}.$$

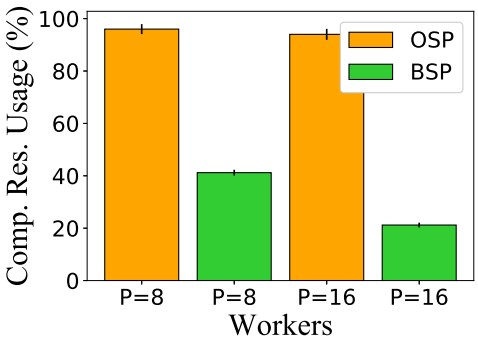
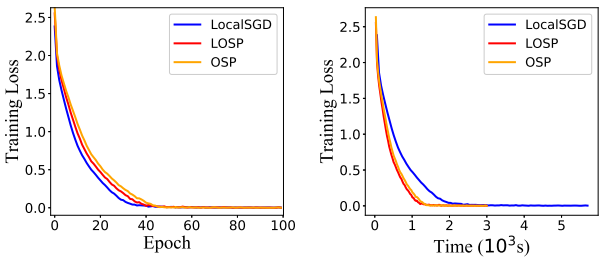


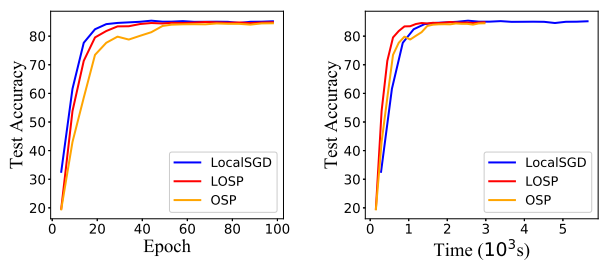
Fig. 10. Computing resource usage of *OSP* and *BSP*. Computing resource usage of *OSP* is nearly 100 percent.

We do experiments on ResNet18 with the mini-batch size of 200 and learning rate of 0.1. The results are shown in Fig. 10, from which we observe that the computing resource usage of *OSP* is nearly 100%. To understand the impact of computing resource usage, we compare the performance of *OSP* to *BSP* in both Fig. 10 and Fig. 9. As shown in Fig. 9 (a), the convergence time of *BSP* is approximately 1.5 times and its computing resource usage is 2/5 of *OSP*. As the computing resource usage of *BSP* decreases to be 1/5 of *OSP* in the 16-workers cluster, the convergence time also increases to be 3.5 times of *OSP*. Hence, *OSP* can really improve the training efficiency through improving the resource usage.

3) Impact of Local Compensation: In this section, we present the results of *LOSP* that investigates the effectiveness of local compensation on the convergence efficiency. The number of local iterations is set to be 64 for ResNet34 and DenseNet121 and 16 for LSTM. For generality, we set the compensated rate γ of LOSP to be 0.2 for all experiments. The learning rate is set to be 0.1 for ResNet34 and DenseNet121 and 20 for LSTM.

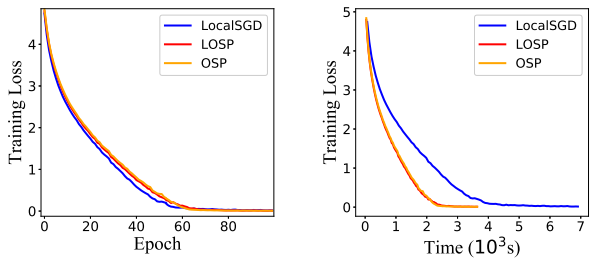


(a) Training Loss.

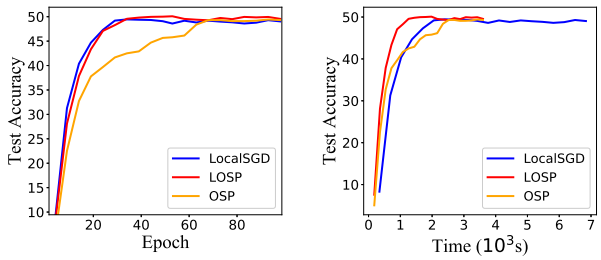


(b) Test Accuracy.

Fig. 11. ResNet34 on the cluster C.

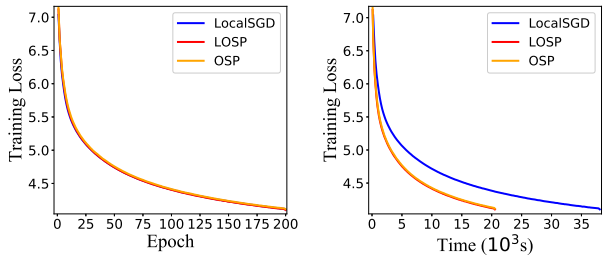


(a) Training Loss.

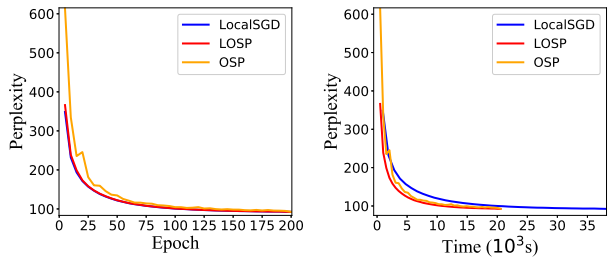


(b) Test Accuracy.

Fig. 12. Densenet121 on the cluster C.



(a) Training Loss.



(b) Test Accuracy.

Fig. 13. LSTM on the cluster C.

The results for ResNet34, DenseNet121, and LSTM are shown in Fig. 12-14. On one hand, the results show that all three methods perform almost the same in terms of the convergence rate, i.e., training loss vs. epoch. On the other hand, our methods LOSP and OSP perform slightly worse than LocalSGD in terms of the generalization performance, i.e., accuracy vs. epoch. This phenomenon exhibits that the staleness incurred by the overlap can deteriorate the generalization performance. In fact, the results have demonstrated the effectiveness of the local compensation for the staleness issue. LOSP achieves significant performance gain over OSP in terms of the generalization

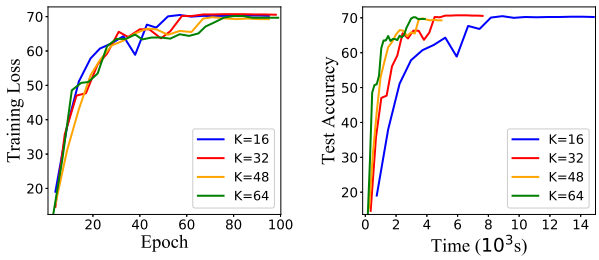
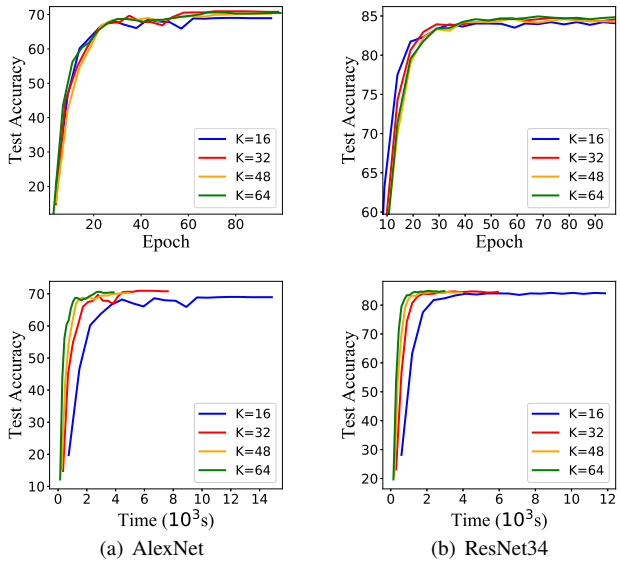


Fig. 14. Impact of K over OSP. AlexNet on Cifar10 dataset on the cluster A.



(a) AlexNet (b) ResNet34

Fig. 15. Impact of K over LOSP. (a) and (b) are results of AlexNet and ResNet34 with different K on the cluster C.

performance.

Furthermore, the results also show that LOSP performs the fastest in terms of the convergence speed for all models. Notably, LOSP achieves more than 2× improvement in terms of convergence speed as compared to the LocalSGD for DenseNet121. Besides, the results also show that OSP converges almost the same as the two other methods in terms of speed although it has worse generalization, which demonstrates the efficiency of the overlapping technique.

Notably, the training losses of OSP and LOSP are very similar, but their test accuracy is quite different, as shown in Fig.12. This is mainly because the inherent computing principles for the training loss and test accuracy is different. Specifically, the training loss is derived from training the local model over the local training dataset, while the test accuracy is calculated from the global over the global test dataset. *OSP* and *LOSP* possess the same local training process, hence yielding the similar training losses. By contrast, the global model of *OSP* is updated by the stale gradient, resulting in worse test accuracy than that of the *LOSP* with the mitigated staleness.

4) Impact of the Number of Local Updates K: To fully investigate the impact of the number of local updates, we conduct experiments for both *OSP* and *LOSP* with various local updates.

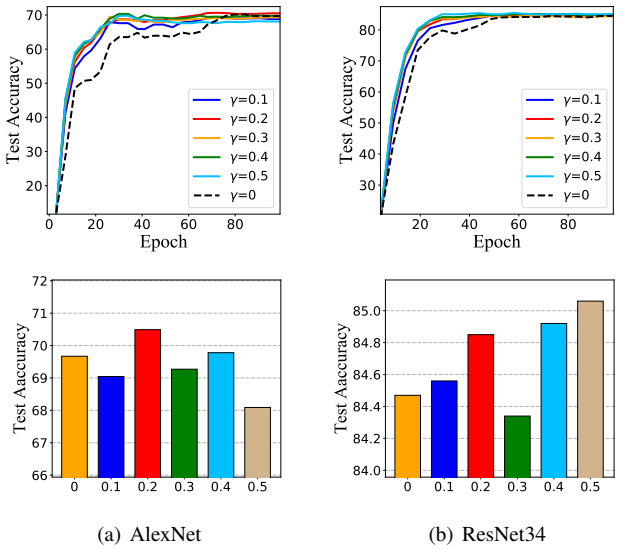
As shown in Fig. 14, the experiments for *OSP* are conducted on the cluster A. It can be seen that the convergence rate decreases as the number of local iterations K increases. For example, the OSP with $K = 16$ converges within fewer epochs than that with $K = 64$. The reason arises from the increased gradient variance and staleness incurred by the reduced number of synchronization and overlap, respectively. On the other hand, the convergence speed grows with the number of K where the OSP with $K = 64$ converges the fastest. It is also worthwhile to note that the performance between OSP with $K = 48$ and $K = 64$ is similar. In our experiment, the synchronization time is close to the computation time with 64 local iterations. As a consequence, as K gets closer to 64, the performance gain in reducing the communication time becomes small, while the convergence rate starts to dominate the performance. Hence, although OSP with $K = 64$ has better computing efficiency, the lower convergence rate slows it down.

The experiments for *LOSP* are shown in Fig. 15, in which the AlexNet are conducted on the cluster A and the ResNet34 are conducted on the cluster C. It can be observed that the LOSP with $K = 16$ performs the worst in terms of both convergence rate and speed for both two models. In fact, the reason for this phenomenon has been well presented by the right item of the inequality 12 in the Theorem 1, where the minimum is achieved as K is some value that is larger than 0. In the two figures, the optimal values of K for maximizing the final accuracy are 32. As K increases from 32, the accuracy and convergence rate starts to decrease.

5) Impact of Local Step Size γ: We investigate the impact of the γ on the convergence rate by uniformly ranging the value of γ from 0 to 0.5. The hyper-parameters adopted are the same as the experiments in Fig. 11. The results are shown in Fig. 16. We can find that the performance of LOSP receives a sudden change from $\gamma = 0$ to $\gamma = 0.1$. After that, LOSP is less sensitive to the γ. Such a phenomenon arises from the existence of the local compensation and indicates that we can simply employ any value of *gamma* between 0.1 and 0.5. Besides, we can also observe that the performance of LOSP with $\gamma = 0.2$ has higher final accuracy than its neighbors, i.e., $\gamma = 0.1$ and $\gamma = 0.3$. For simplicity, we can always take 0.2 as the value of γ in practice.

6) Impact of Straggler: The experiments on the impact of stragglers in LOSP are conducted on the cluster A. We generate the stragglers by randomly running the single-alone LR classification program in the background. In each interval, only one VM runs with the LR program and all VMs run in a round-robin manner. The experimental results show that the stragglers have a negative impact on the OSP model. We think that this phenomenon arises from the huge unbalanced number of local iterations. With local compensation, the influence of the unbalanced number is greatly mitigated. The final accuracy achieved by LOSP is even higher than that of LocalSGD with balanced k on workers.

7) Impact of Low-bandwidth Network: To better adapt to the low-bandwidth of the network, we have minor changes on some hyper-parameters. For the learning rate of MnistCNN and AlexNet, we adopt 0.0004 and 0.025 respectively. The batch size of each worker is set to32. The period for the averaging



(a) AlexNet (b) ResNet34

Fig. 16. Impact of γ over LOSP. (a) and (b) are results of AlexNet and ResNet34 with different γ on the cluster C.

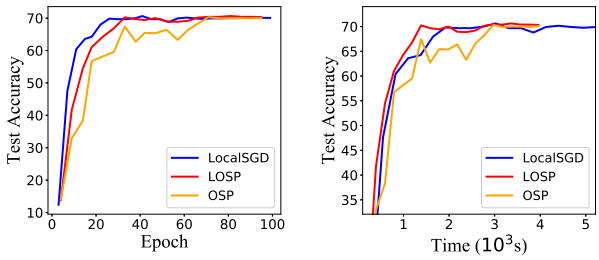


Fig. 17. Impact of straggler. AlexNet on Cifar10 dataset on the cluster A.

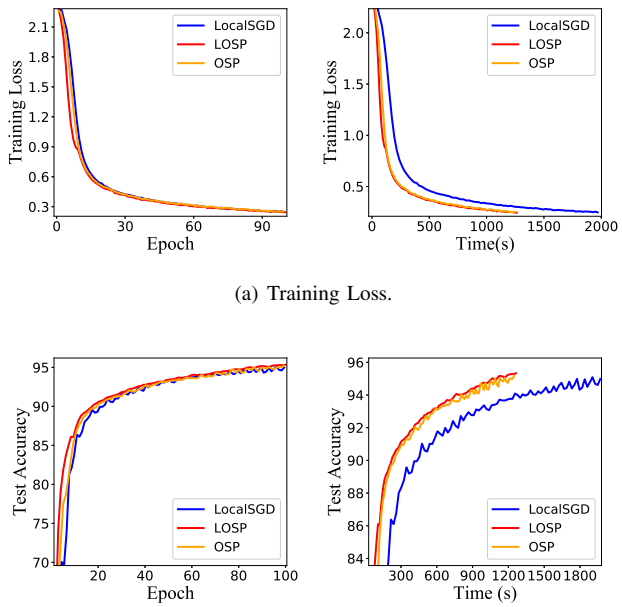
of LocalSGD nad CoCoD-SGD is set to 256. Similarly, the threshold of the number of local iterations of *LOSP* and *OSP* is also set to be 256. For MnistCNN, the compensated factor γ of *LOSP* is initially set to be 0.0006 and decayed to be 0.0001 at 10th epoch. For AlexNet, the compensated factor γ is initially set to be 0.025 and diminished to be 0.01 at 120-th epoch.

The results of different methods on MnistCNN are shown in Fig. 18. It can be observed that *LOSP* achieves the highest accuracy as 95.35%, and then the *LOSP* with 95.25%. Local-SGD obtains the lowest accuracy being 94.93%. In our experiment, CoCoD-SGD could not converge. Except the final accuracy, *LOSP* also converges faster than LocalSGD in terms of the time. Specifically, *LOSP* costs 1264 seconds which reduces by a factor of 36% when compared to the 1971 seconds of LocalSGD.

In fact, when considering the large machine learning model, the advantages of *LOSP* can be further delivered. The results

TABLE II
ACCURACY OF DIFFERENT METHODS

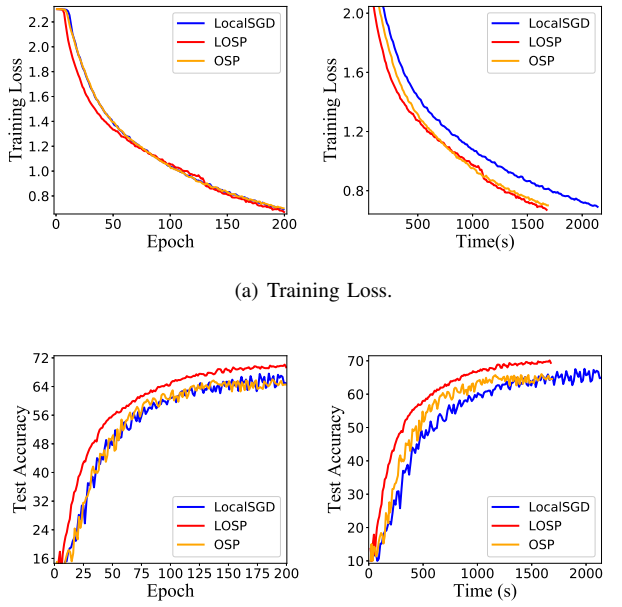
Models	Methods			
	Local-SGD	OSP	LOSP	CoCoD-SGD
MnistCNN	94.93	95.25	**95.35**	Diverge
AlexNet	66.01	64.928	**70.14**	Diverge



(a) Training Loss.

(b) Test Accuracy.

Fig. 18. Convergence Time of different methods on MnistCNN with running on the cluster D.



(a) Training Loss.

(b) Test Accuracy.

Fig. 19. Convergence Time of different methods on AlexNet with running on the cluster D.

of different methods running AlexNet are shown in Fig. 19. As expected, *LOSP* significantly outperforms the other methods with the highest accuracy 70.14%. *OSP* and LocalSGD only obtain 64.928% and 66.01% respectively. Similar to that of MnistCNN, CoCoD-SGD still diverges. The results of different methods are further summarized in Table II.

VII. Conclusion

In this paper, we propose a new distributed machine learning mechanism named *LOSP*. It accelerates distributed machine learning by overlapping the computation and communication of the training process. The convergence of *LOSP* is further established without assuming convexity. The theoretical results show that the convergence rate of *LOSP* is in the same order as sequential SGD. Finally, extensive experiments on both convex models and non-convex models are conducted to verify our results. Experimental results present that *LOSP* outperforms all the state-of-the-art methods.

Appendix A
Proof of General Lemmas

For the clarity of presentation, we introduce a notation for the local update in iteration t as $G_t^i = \sum_{k=0}^{K_t^i-1} g(\omega_{t-1,k}^i, \xi_{t,k}^i)$. Besides, we have the following facts for LOSP:

$$\omega_{t-1} - \omega_{t-1,0}^i = \gamma\eta_t G_{t-1}^i, \tag{16}$$

$$\omega_{t-1,0}^i - \omega_{t-1,k}^i = \eta_t \sum_{j=0}^{k-1} g(\omega_{t-1,j}^i, \xi_{t-1,j}^i). \tag{17}$$

To support the convergence analysis for LOSP, we derive the following lemmas.

Lemma 3. *The expected squared norm of the local stochastic gradients is bounded by*

$$\mathbb{E}\|G_t^i\|_2^2 \le K_t^i\sigma^2 + \mathbb{E}\|\sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2, \tag{18}$$

$$\mathbb{E}\|\sum_{i=1}^{P} G_t^i\|_2^2 \le \sum_{i=1}^{P} K_t^i\sigma^2 + \mathbb{E}\|\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2. \tag{19}$$

Proof. According to the definition of G_t^i, we have

$$\begin{aligned}
&\mathbb{E}\|G_t^i\|_2^2 = \mathbb{E}\|G_t^i - \sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i) + \sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 \\
&= \mathbb{E}\|G_t^i - \sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 + \mathbb{E}\|\sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 \\
&+ 2\mathbb{E} < G_t^i - \sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2, \sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 > \\
&= \mathbb{E}\|G_t^i - \sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 + \mathbb{E}\|\sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 \\
&= \sum_{k=0}^{K_t^i-1} \mathbb{E}\|g(\omega_{t-1,k}^i, \xi_{t-1,k}^i) - \nabla F(\omega_{t-1,k}^i)\|_2^2 \\
&+ \mathbb{E}\|\sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 + \sum_{k\ne j}^{K_t^i-1} \mathbb{E} < g(\omega_{t-1,k}^i, \xi_{t-1,k}^i) \\
&- \nabla F(\omega_{t-1,k}^i), g(\omega_{t-1,j}^i, \xi_{t-1,j}^i) - \nabla F(\omega_{t-1,j}^i) > \\
&= \sum_{k=0}^{K_t^i-1} \mathbb{E}\|g(\omega_{t-1,k}^i, \xi_{t-1,k}^i) - \nabla F(\omega_{t-1,k}^i)\|_2^2 \\
&+ \mathbb{E}\|\sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2 \\
&\le K_t^i\sigma^2 + \|\sum_{k=0}^{K_t^i-1} \nabla F(\omega_{t-1,k}^i)\|_2^2,
\end{aligned} \tag{20}$$

where the third and last equalities hold due to unbiased gradient of Assumption 1-(4), and the last inequality uses the bounded variance of Assumption 1-(4). The bound of $\mathbb{E}\|\sum_{i=1}^{P} G_t^i\|_2^2$ in (19) is similar to that of $\mathbb{E}\|G_t^i\|_2^2$. □

Lemma 4. *For any $k = 0, 1, \ldots, K_t^i$, the squared norm of local update is bounded by*

$$\mathbb{E}\|\omega_{t-1,0}^i - \omega_{t-1,k}^i\|_2^2 \le 3\eta_t^2L^2k\sigma^2 + 3\eta_t^2L^2\|\sum_{j=0}^{k-1} \nabla F(\omega_{t-1,j}^i)\|_2^2. \tag{21}$$

Proof. From the workflow of Algorithm 1, we have

$$\begin{aligned}
&\mathbb{E}\|\omega_{t-1,0}^i - \omega_{t-1,k}^i\|_2^2 = \mathbb{E}\|\eta_t \sum_{j=0}^{k-1} g(\omega_{t-1,j}^i, \xi_{t-1,j}^i)\|_2^2 \\
&= \eta_t^2\mathbb{E}\|\sum_{j=0}^{k-1}[g(\omega_{t-1,j}^i, \xi_{t-1,j}^i) - \nabla F(\omega_{t-1,j}^i)] + \sum_{j=0}^{k-1} \nabla F(\omega_{t-1,j}^i)\|_2^2 \\
&= \eta_t^2\mathbb{E}\|\sum_{j=0}^{k-1}[g(\omega_{t-1,j}^i, \xi_{t-1,j}^i) - \nabla F(\omega_{t-1,j}^i)]\|_2^2 \\
&+ \eta_t^2\mathbb{E}\|\sum_{j=0}^{k-1} \nabla F(\omega_{t-1,j}^i)\|_2^2 \\
&+ 2\eta_t^2\mathbb{E} < \sum_{j=0}^{k-1}[g(\omega_{t-1,j}^i, \xi_{t-1,j}^i) - \nabla F(\omega_{t-1,j}^i)], \sum_{j=0}^{k-1} \nabla F(\omega_{t-1,j}^i) > \\
&= \eta_t^2\mathbb{E}\|\sum_{j=0}^{k-1}[g(\omega_{t-1,j}^i, \xi_{t-1,j}^i) - \nabla F(\omega_{t-1,j}^i)]\|_2^2 \\
&+ \eta_t^2\mathbb{E}\|\sum_{j=0}^{k-1} \nabla F(\omega_{t-1,j}^i)\|_2^2 \\
&\le \eta_t^2k\sigma^2 + \eta_t^2\|\sum_{j=0}^{k-1} \nabla F(\omega_{t-1,j}^i)\|_2^2,
\end{aligned} \tag{22}$$

where the last equality holds because the unbiased gradient of Assumption 1-(3) and the last inequality uses the bounded variance of Assumption 1-(3). □

Lemma 5. *The difference of worker i between global model and locally compensated model is bounded by*

$$\begin{aligned}
&\mathbb{E}\|\omega_{t-1} - \omega_{t-1,0}^i\|_2^2 \\
&\le \gamma^2\eta_t^2\sigma^2K_{t-1}^i + \gamma^2\eta_t^2\|\sum_{k=0}^{K_{t-1}^i-1} \nabla F(\omega_{t-2,k}^i)\|_2^2.
\end{aligned} \tag{23}$$

Proof. With the workflow in Algorithm 1, we have

$$\mathbb{E}\|\omega_{t-1} - \omega_{t-1,0}^i\|_2^2 = \mathbb{E}\|\gamma\eta_t G_{t-1}^i\|_2^2$$

$$\leq \gamma^2\eta_t^2\sigma^2 K_{t-1}^i + \gamma^2\eta_t^2\| \sum_{k=0}^{K_{t-1}^i-1} \nabla F(\omega_{t-2,k}^i)\|_2^2, \quad (24)$$

where the inequality uses (19) in Lemma 3. □

Appendix B
Proof of Theorem 1

According to the algorithm, the one-step updated fomula of global parameter ω_{t+1} is

$$\omega_{t+1} = \omega_t - \eta_t \frac{1}{P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1} g(\omega_{t-1,k}^i, \xi_{t,k}^i), \quad (25)$$

where $\omega_{t-1,0}^i = \omega_{t-1} - \gamma\eta_t \sum_{k=0}^{K_{t-1}^i-1} g(\omega_{t-2,k}^i, \xi_{t-2,k}^i)$. The one-step update can be bounded by the following lemma.

Lemma 6. *The squared norm of the one-step update in Algorithm 1 is bounded by*

$$\mathbb{E}\|\omega_{t+1} - \omega_t\|_2^2 \leq \frac{\eta_t^2\sigma^2}{P^2}\sum_{i=1}^{P} K_t^i + \frac{\eta_t^2}{P^2}\|\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\nabla F(\omega_{t-1,k}^i)\|_2^2. \quad (26)$$

Proof. From (25), we have

$$\mathbb{E}\|\omega_{t+1} - \omega_t\|_2^2 = \mathbb{E}\|\eta_t\frac{1}{P}\sum_{i=1}^{P} G_t^i\|_2^2 \leq \frac{\eta_t^2\sigma^2}{P^2}\sum_{i=1}^{P} K_t^i + \frac{\eta_t^2}{P^2}\|\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\nabla F(\omega_{t-1,k}^i)\|_2^2, \quad (27)$$

where the last equality uses (3) in Lemma 3. □

Based on this lemma, we give the detailed proof of Theorem 1.

Proof. To prove the covergence of *LOSP*, we firstly bound the update of one step $F(\omega_{t+1}) - F(\omega_t)$ and then summarize all steps from 1 to N to achieve the overall convergence.

Based on Assumption 1 and the conclusion of Appendix B of [4], the bound of one iteration is

$$\begin{aligned} &F(\omega_{t+1}) - F(\omega_t) \\ &\leq \langle \nabla F(\omega_t), \omega_{t+1} - \omega_t\rangle + \frac{L}{2}\|\omega_{t+1} - \omega_t\|_2^2 \\ &= \left\langle \nabla F(\omega_t), -\eta_t\frac{1}{P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1} g(\omega_{t-1,k}^i, \xi_{t,k}^i)\right\rangle \\ &+ \frac{L}{2}\|\omega_{t+1} - \omega_t\|_2^2. \end{aligned} \quad (28)$$

Because $\xi_{t,k}^i$ in our algorithm are i.i.d. for all t, k, i, by taking expectation for both sides of (28) upon $\xi_{t,k}^i$ and combining Asssumption 3, we can immediately get

$$\begin{aligned} &\mathbb{E}F(\omega_{t+1}) - F(\omega_t) \\ &\leq \left\{\left\langle \nabla F(\omega_t), -\eta_t\frac{1}{P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\nabla F(\omega_{t-1,k}^i)\right\rangle\right\}_{T_1} \\ &+ \frac{L}{2}\mathbb{E}\|\omega_{t+1} - \omega_t\|_2^2. \end{aligned} \quad (29)$$

We present the bound of T_1 as follows:

$$T_1 = \left\langle \nabla F(\omega_t), -\eta_t\frac{1}{P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\nabla F(\omega_{t-1,k}^i)\right\rangle \quad (30)$$

$$= -\frac{\eta_t}{P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\left\langle \nabla F(\omega_t), \nabla F(\omega_{t-1,k}^i)\right\rangle \quad (31)$$

$$= -\frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}[\|\nabla F(\omega_t)\|_2^2 + \|\nabla F(\omega_{t-1,k}^i)\|_2^2] \quad (32)$$

$$+ \frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\left\{\|\nabla F(\omega_t) - \nabla F(\omega_{t-1,k}^i)\|_2^2\right\}_{T_2}, \quad (33)$$

where the third item is because $-2 < a, b >= \|a-b\|^2 - \|a\|^2 - \|b\|^2$. According to Assumption 1, T_2 can be bounded as

$$\begin{aligned} T_2 &= \|\nabla F(\omega_t) - \nabla F(\omega_{t-1,k}^i)\|_2^2 \\ &\leq L^2\|\omega_t - \omega_{t-1,k}^i\|_2^2 \\ &\leq L^2\|\omega_t - \omega_{t-1} + \omega_{t-1} - \omega_{t-1,0}^i + \omega_{t-1,0}^i - \omega_{t-1,k}^i\|_2^2 \\ &\leq 3L^2(\|\omega_t - \omega_{t-1}\|_2^2 + \|\omega_{t-1} - \omega_{t-1,0}^i\|_2^2 + \|\omega_{t-1,0}^i - \omega_{t-1,k}^i\|_2^2). \end{aligned}$$

Using Lemma 4, Lemma 5, and Lemma 6, we could obtain

$$\begin{aligned} T_2 \leq\ & \frac{3\eta_{t-1}^2L^2\sigma^2}{P^2}\sum_{i=1}^{P} K_{t-1}^i + \frac{3\eta_{t-1}^2L^2}{P^2}\|\sum_{i=1}^{P}\sum_{k=0}^{K_{t-1}^i-1}\nabla F(\omega_{t-2,k}^i)\|_2^2 \\ &+ 3\eta_t^2L^2k\sigma^2 + 3\eta_t^2L^2\|\sum_{j=0}^{k-1}\nabla F(\omega_{t-1,j}^i)\|_2^2 \\ &+ 3L^2\gamma^2\eta_t^2\sigma^2K_{t-1}^i + 3L^2\gamma^2\eta_t^2\|\sum_{k=0}^{K_{t-1}^i-1}\nabla F(\omega_{t-2,k}^i)\|_2^2. \end{aligned} \quad (34)$$

Putting T_2 back into T_1, we get

$$\begin{aligned} T_1 \leq\ & -\frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}[\|\nabla F(\omega_t)\|_2^2 + \|\nabla F(\omega_{t-1,k}^i)\|_2^2] \\ &+ \frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\Big[\frac{3\eta_{t-1}^2L^2\sigma^2}{P^2}\sum_{j=1}^{P}K_{t-1}^j + 3\eta_t^2L^2k\sigma^2 \\ &+ 3L^2\gamma^2\eta_t^2\sigma^2K_{t-1}^i + 3\eta_t^2L^2\|\sum_{j=0}^{k-1}\nabla F(\omega_{t-1,j}^i)\|_2^2 \\ &+ \frac{3\eta_{t-1}^2L^2}{P^2}\|\sum_{j=1}^{P}\sum_{l=0}^{K_{t-1}^j-1}\nabla F(\omega_{t-2,l}^j)\|_2^2 \\ &+ 3L^2\gamma^2\eta_t^2\|\sum_{k=0}^{K_{t-1}^i-1}\nabla F(\omega_{t-2,k}^i)\|_2^2\Big]. \end{aligned} \quad (35)$$

Now, following from (29), together with the bound of $\mathbb{E}\|\omega_{t+1} - \omega_t\|_2^2$ derived in Lemma 25, we have the following inequality:

$$\begin{aligned} &\mathbb{E}F(\omega_{t+1}) - F(\omega_t) \\ &\leq -\frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}[\|\nabla F(\omega_t)\|_2^2 + \|\nabla F(\omega_{t-1,k}^i)\|_2^2] \end{aligned}$$

$$
\begin{aligned}
&+\frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\Big[\frac{3\eta_{t-1}^2L^2\sigma^2}{P^2}\sum_{j=1}^{P}K_{t-1}^j+3\eta_t^2L^2k\sigma^2\\
&+3L^2\gamma^2\eta_t^2\sigma^2K_{t-1}^i+3\eta_t^2L^2\|\sum_{j=0}^{k-1}\nabla F(\omega_{t-1,j}^i)\|_2^2\\
&+\frac{3\eta_{t-1}^2L^2}{P^2}\|\sum_{j=1}^{P}\sum_{l=0}^{K_{t-1}^j-1}\nabla F(\omega_{t-2,l}^j)\|_2^2\\
&+3L^2\gamma^2\eta_t^2\|\sum_{l=0}^{K_{t-1}^i-1}\nabla F(\omega_{t-2,l}^i)\|_2^2\Big]\\
&+\frac{\eta_t^2L\sigma^2}{2P^2}\sum_{i=1}^{P}K_t^i+\frac{\eta_t^2L}{2P^2}\|\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&=-\frac{\bar{K}\eta_t}{2}\|\nabla F(\omega_t)\|_2^2+\Big\{\frac{3L^2\gamma^2\eta_t^3\sigma^2}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}K_{t-1}^i+\\
&+\frac{\eta_t^2L\sigma^2}{2P^2}\sum_{i=1}^{P}K_t^i+\frac{3\eta_t^3L^2\sigma^2}{4P}\sum_{i=1}^{P}(K_t^i-1)K_t^i\\
&+\frac{3\eta_t\eta_{t-1}^2L^2\sigma^2}{2P^3}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\sum_{j=1}^{P}K_{t-1}^j\Big\}_{T_3}\\
&+\Big\{-\frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&+\frac{\eta_t^2L}{2P^2}\|\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&+\frac{3\eta_t^3L^2}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\sum_{j=0}^{k-1}\nabla F(\omega_{t-1,j}^i)\|_2^2\\
&+\frac{3\eta_t\eta_{t-1}^2L^2}{2P^3}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\sum_{j=1}^{P}\sum_{l=0}^{K_{t-1}^i-1}\nabla F(\omega_{t-2,l}^j)\|_2^2\\
&+\frac{3\eta_t^3L^2\gamma^2}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\sum_{l=0}^{K_{t-1}^i-1}\nabla F(\omega_{t-2,l}^i)\|_2^2\Big\}_{T_4}. \qquad (36)
\end{aligned}
$$

We further simplify the bound of $\mathbb{E}F(\omega_{t+1})-F(\omega_t)$. According to Assumption 2-(1) and Assumption 2-(2), we have

$$
\begin{aligned}
T_3&=\frac{3L^2\gamma^2\eta_t^3\sigma^2}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}K_{t-1}^i+\frac{\eta_t^2L\sigma^2}{2P^2}\sum_{i=1}^{P}K_t^i\\
&+\frac{3\eta_t^3L^2\sigma^2}{4P}\sum_{i=1}^{P}(K_t^i-1)K_t^i+\frac{3\eta_t\eta_{t-1}^2L^2\sigma^2}{2P^3}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\sum_{j=1}^{P}K_{t-1}^j\\
&\le\frac{3L^2\gamma^2\eta_t^3\sigma^2}{4P}\sum_{i=1}^{P}((K_t^i)^2+(K_{t-1}^i)^2)++\frac{\eta_t^2L\sigma^2}{2P}\bar{K}\\
&+\frac{3\eta_t^3L^2\sigma^2}{4P}\sum_{i=1}^{P}((K_t^i)^2-K_t^i)+\frac{3\eta_t\eta_{t-1}^2L^2\sigma^2}{2P^2}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\bar{K}\\
&\le\frac{3\eta_t^3L^2\gamma^2\sigma^2}{2}(\bar{K}^2+M)+\frac{3\eta_t^3L^2\sigma^2M}{4}+\frac{3\eta_t^3L^2\sigma^2\bar{K}^2}{4}\\
&-\frac{3\eta_t^3L^2\sigma^2\bar{K}}{4}+\frac{3\eta_t\eta_{t-1}^2L^2\sigma^2\bar{K}^2}{2P}+\frac{\eta_t^2L\sigma^2\bar{K}}{2P}. \qquad (37)
\end{aligned}
$$

Similarly, we show bound of T_4

$$
\begin{aligned}
T_4&\le-\frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&+\frac{\eta_t^2L}{2P}\sum_{i=1}^{P}K_t^i\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&+\frac{3\eta_t^3L^2}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}k\sum_{j=0}^{k-1}\|\nabla F(\omega_{t-1,j}^i)\|_2^2\\
&+\frac{3\eta_t\eta_{t-1}^2L^2}{2P^2}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\sum_{j=1}^{P}K_{t-1}^j\sum_{l=0}^{K_{t-1}^j-1}\|\nabla F(\omega_{t-2,l}^j)\|_2^2\\
&+\frac{3\eta_t^3L^2\gamma^2}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}K_{t-1}^i\sum_{l=0}^{K_{t-1}^i-1}\|\nabla F(\omega_{t-2,l}^i)\|_2^2\\
&\le-\frac{\eta_t}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&+\frac{\eta_t^2L}{2P}\sum_{i=1}^{P}K_t^i\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&+\frac{3\eta_t^3L^2}{4P}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}(K_t^i-1)K_t^i\|\nabla F(\omega_{t-1,k}^i)\|_2^2\\
&+\frac{3\eta_t\eta_{t-1}^2L^2\bar{K}}{2P}\sum_{i=1}^{P}\sum_{k=0}^{K_{t-1}^i-1}K_{t-1}^i\|\nabla F(\omega_{t-2,k}^i)\|_2^2\\
&+\frac{3\eta_t^3L^2\gamma^2}{2P}\sum_{i=1}^{P}K_t^iK_{t-1}^i\sum_{l=0}^{K_{t-1}^i-1}\|\nabla F(\omega_{t-2,l}^i)\|_2^2\\
&=\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\Big[-\frac{\eta_t}{2P}+\frac{\eta_t^2LK_t^i}{2P}\\
&+\frac{3\eta_t^3L^2(K_t^i-1)K_t^i}{4P}\Big]+\sum_{i=1}^{P}\sum_{k=0}^{K_{t-1}^i-1}\|\nabla F(\omega_{t-2,k}^i)\|_2^2\\
&\Big[\frac{3\eta_t\eta_{t-1}^2L^2\bar{K}}{2P}K_{t-1}^i+\frac{3\eta_t^3L^2\gamma^2}{2P}K_t^iK_{t-1}^i\Big], \qquad (38)
\end{aligned}
$$

where the first iequality is due to CauchyCSchwarz inequality and the second iequality is due to Assumption 2-(1) and Assumption 2-(2). When considering all N iterations, we have

$$
\begin{aligned}
\sum_{t=1}^{N}T_4&\le\sum_{t=1}^{N}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\Big[-\frac{\eta_t}{2P}+\frac{\eta_t^2LK_t^i}{2P}\\
&+\frac{3\eta_t^3L^2(K_t^i-1)K_t^i}{4P}\Big]+\sum_{t=1}^{N}\sum_{i=1}^{P}\sum_{k=0}^{K_{t-1}^i-1}\|\nabla F(\omega_{t-2,k}^i)\|_2^2\\
&\Big[\frac{3\eta_t\eta_{t-1}^2L^2\bar{K}}{2P}K_{t-1}^i+\frac{3\eta_t^3L^2\gamma^2}{2P}K_t^iK_{t-1}^i\Big]\\
&\le\sum_{t=1}^{N}\sum_{i=1}^{P}\sum_{k=0}^{K_t^i-1}\|\nabla F(\omega_{t-1,k}^i)\|_2^2\Big\{\Big[-\frac{\eta_t}{2P}+\frac{\eta_t^2LK_t^i}{2P}
\end{aligned}
$$

$$+\frac{3\eta_t^3L^2(K_t^i-1)K_t^i}{4P}+\frac{3\eta_{t+1}\eta_t^2L^2\bar{K}K_t^i}{2P}+\frac{3\eta_{t+1}^3L^2\gamma^2}{2P}K_{t+1}^iK_t^i\Big]\Big\}_{T_5}. \tag{39}$$

Since $K_{max}=\max\left\{K_t^i\middle|i=1,\ldots,P,\quad t=1,\ldots,N\right\}$ and $K_t^i\geq 1$, we have

$$T_5\leq\frac{\eta_t}{2P}[-1+\eta_tLK_{max}+3\eta_t^2L^2(K_{max}-1)K_{max}/2+3\eta_{t+1}\eta_tL^2\bar{K}K_{max}+3\eta_{t+1}^3L^2\gamma^2K_{max}^2/\eta_t] \tag{40}$$

Obviously, if

$$LK_{max}(\eta_t+3\eta_t^2LK_{max}/2+3\eta_{t+1}\eta_tL\bar{K}+3\eta_{t+1}^3L\gamma^2K_{max}/\eta_t-3\eta_t^2L/2)\leq 1, \tag{41}$$

then

$$T_5\leq 0,\qquad \sum_{t=1}^{N}T_4\leq 0. \tag{42}$$

So, when summing up (36) from $t=1$ to $t=N$, and together with (37) and (38), we could get the following bound

$$\begin{aligned}&\mathbb{E}F(\omega_{N+1})-F(\omega_1)\\&\leq-\frac{\bar{K}}{2}\sum_{t=1}^{N}\eta_t\|\nabla F(\omega_t)\|_2^2+\sum_{t=1}^{N}T_3+\sum_{t=1}^{N}T_4\\&\leq-\frac{\bar{K}}{2}\sum_{t=1}^{N}\eta_t\|\nabla F(\omega_t)\|_2^2+\sum_{t=1}^{N}\Big[\frac{3\eta_t^3L^2\gamma^2\sigma^2}{2}(\bar{K}^2+M)\\&+\frac{3\eta_t^3L^2\sigma^2M}{4}+\frac{3\eta_t^3L^2\sigma^2\bar{K}^2}{4}-\frac{3\eta_t^3L^2\sigma^2\bar{K}}{4}\\&+\frac{3\eta_t\eta_{t-1}^2L^2\sigma^2\bar{K}^2}{2P}+\frac{\eta_t^2L\sigma^2\bar{K}}{2P}\Big].\end{aligned} \tag{43}$$

By denoting $\omega_\star$ as the global optimal point that achieves the F_{inf} in Assumption 2, we have

$$F(\omega_\star)-F(\omega_1)\leq\mathbb{E}F(\omega_{N+1})-F(\omega_1). \tag{44}$$

Moving sum of squared gradients to left, we have

$$\begin{aligned}&\sum_{t=1}^{N}\eta_t\mathbb{E}\|\nabla F(\omega_t)\|_2^2\leq\frac{2(F(\omega_1)-F(\omega_\star))}{\bar{K}}\\&+\sum_{t=1}^{N}\frac{\eta_t}{\bar{K}}[3\eta_t^2L^2\gamma^2\sigma^2(\bar{K}^2+M)+\frac{3\eta_t^2L^2\sigma^2M}{2}\\&+\frac{3\eta_t^2L^2\sigma^2\bar{K}^2}{2}-\frac{3\eta_t^2L^2\sigma^2\bar{K}}{2}+\frac{3\eta_{t-1}^2L^2\sigma^2\bar{K}^2}{P}+\frac{\eta_tL\sigma^2\bar{K}}{P}].\end{aligned} \tag{45}$$

Considering a fixed learning rate $\eta_t=\bar{\eta}$, the condition in (41) is equivalent to

$$\bar{\eta}LK_{max}(1+3\bar{\eta}LK_{max}/2+3\bar{\eta}L\bar{K}+3\bar{\eta}L\gamma^2K_{max}-3\bar{\eta}L/2)\leq 1, \tag{46}$$

and then the bound of average squared gradient is

$$\frac{1}{N}\sum_{t=1}^{N}\mathbb{E}\|\nabla F(\omega_t)\|_2^2\leq\frac{2(F(\omega_1)-F(\omega_\star))}{N\bar{\eta}\bar{K}}+\frac{\bar{\eta}L\sigma^2}{\bar{K}}\Big[\frac{3\bar{\eta}LM}{2}+\frac{3\bar{\eta}L\bar{K}^2}{2}+3\bar{\eta}L\gamma^2(\bar{K}^2+M)-\frac{3\bar{\eta}L\bar{K}}{2}+\frac{3\bar{\eta}L\bar{K}^2}{P}+\frac{\bar{K}}{P}\Big],$$

which completes the proof. □

Appendix C
Proof of Corollary 2

Proof. If

$$\bar{\eta}\Big(\frac{3LM}{2}+\frac{3L\bar{K}^2}{2}+3L\gamma^2(\bar{K}^2+M)-\frac{3L\bar{K}}{2}+\frac{3L\bar{K}^2}{P}\Big)\leq\frac{\bar{K}}{P}, \tag{47}$$

then the bound in (12) can be written as

$$\frac{1}{N}\sum_{t=1}^{N}\mathbb{E}\|\nabla F(\omega_t)\|_2^2\leq\frac{2\left[F(\omega_1)-F(\omega_\star)\right]}{N\bar{\eta}\bar{K}}+\frac{2\bar{\eta}L\sigma^2}{P}. \tag{48}$$

Define $f(\bar{\eta})=\frac{2[F(\omega_1)-F(\omega_\star)]}{N\bar{\eta}\bar{K}}+\frac{2\bar{\eta}L\sigma^2}{P}$, and let $f'(\bar{\eta})=0$. We have

$$\bar{\eta}=\sqrt{\frac{2(F(\omega_1)-F(\omega_\star))P}{N\bar{K}L\sigma^2}}, \tag{49}$$

and

$$f(\bar{\eta})\geq 4\sqrt{\frac{(F(\omega_1)-F(\omega_\star))\sigma^2}{\bar{K}P}}*\frac{1}{\sqrt{N}}. \tag{50}$$

Bring the value of $\bar{\eta}$ in (49) back into (47), we have

$$\sqrt{\frac{2(F(\omega_1)-F(\omega_\star))P}{N\bar{K}L\sigma^2}}\Big(\frac{3LM}{2}+\frac{3L\bar{K}^2}{2}+3L\gamma^2(\bar{K}^2+M)-\frac{3L\bar{K}}{2}+\frac{3L\bar{K}^2}{P}\Big)\leq\frac{\bar{K}}{P}. \tag{51}$$

Setting $A_1=\frac{3LM}{2}+\frac{3L\bar{K}^2}{2}+3L\gamma^2(\bar{K}^2+M)-\frac{3L\bar{K}}{2}+\frac{3L\bar{K}^2}{P}$, (51) yields

$$N\geq A_2, \tag{52}$$

where $A_2=\frac{2(F(\omega_1)-F(\omega_\star))A_1^2P^3}{\bar{K}^3L\sigma^2}$. Besides, by setting $A_3=3K_{max}/2+3\bar{K}+3\gamma^2K_{max}-3/2$, the bound of learning rate $\bar{\eta}$ in (49) becomes

$$\bar{\eta}LK_{max}(1+\bar{\eta}LA_3)\leq 1. \tag{53}$$

Together with the value of $\bar{\eta}$ in (49), we have

$$N\geq A_4, \tag{54}$$

where $A_4=\frac{K_{max}P(F(\omega_1)-F(\omega_\star))\left(\sqrt{LK_{max}}+\sqrt{LK_{max}+4A_3}\right)^2}{2\bar{K}\sigma^2}$. Considering (52) and (54) together, we obtain that, if

$$N\geq Max(A_2,A_4), \tag{55}$$

then

$$\frac{1}{N}\sum_{t=1}^{N}\mathbb{E}\|\nabla F(\omega_t)\|_2^2\leq 4\sqrt{\frac{(F(\omega_1)-F(\omega_\star))\sigma^2}{\bar{K}P}}*\frac{1}{\sqrt{N}}. \tag{56}$$

□

REFERENCES

[1] H. Wang, S. Guo, and R. Li, "OSP: overlapping computation and communication in parameter server for fast machine learning," in *Proceedings of the 48th International Conference on Parallel Processing, ICPP*, 2019.

[2] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2013.

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2014.

[4] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[5] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B. Su, "Scaling distributed machine learning with the parameter server," in *Proceedings of USENIX Symposium on Operating Systems Design and Implementation, OSDI*, 2014.

[6] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proceedings of Annual Conference on Neural Information Processing Systems 2014, NeurIPS*, 2014.

[7] A. Harlap, A. Tumanov, A. Chung, G. R. Ganger, and P. B. Gibbons, "Proteus: agile ML elasticity through tiered reliability in dynamic resource markets," in *Proceedings of the Twelfth European Conference on Computer Systems, EuroSys*, 2017.

[8] H. Zhang, Z. Zheng, S. Xu, W. Dai, Q. Ho, X. Liang, Z. Hu, J. Wei, P. Xie, and E. P. Xing, "Poseidon: An efficient communication architecture for distributed deep learning on GPU clusters," in *Proceedings of USENIX Annual Technical Conference, ATC*, 2017.

[9] J. Chen, R. Monga, S. Bengio, and R. Józefowicz, "Revisiting distributed synchronous SGD," *CoRR*, 2016.

[10] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of Annual Conference on Neural Information Processing Systems,NeurIPS*, 2012.

[11] J. Cipar, Q. Ho, J. K. Kim, S. Lee, G. R. Ganger, G. Gibson, K. Keeton, and E. P. Xing, "Solving the straggler problem with bounded staleness," in *14th Workshop on Hot Topics in Operating Systems, HotOS*, 2013.

[12] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. R. Ganger, and E. P. Xing, "More effective distributed ML via a stale synchronous parallel parameter server," in *Proceedings of Annual Conference on Neural Information Processing Systems, NeurIPS*, 2013.

[13] S. Wang, A. Pi, and X. Zhou, "Scalable distributed DL training: Batching communication and computation," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*.

[14] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z. Ma, and T. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *Proceedings of the 34th International Conference on Machine Learning, ICML*, 2017.

[15] E. Azarkhish, D. Rossi, I. Loi, and L. Benini, "Neurostream: Scalable and energy efficient deep learning with smart memory cubes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 2, pp. 420–434, 2018.

[16] W. Choi, K. Duraisamy, R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu, "On-chip communication network for efficient training of deep convolutional networks on heterogeneous manycore systems," *IEEE Trans. Computers*, vol. 67, no. 5, pp. 672–686, 2018.

[17] F. Schuiki, M. Schaffner, F. K. Gürkaynak, and L. Benini, "A scalable near-memory architecture for training deep neural networks on large in-memory datasets," *IEEE Trans. Computers*, vol. 68, no. 4, pp. 484–497, 2019.

[18] X. Chen, D. Z. Chen, Y. Han, and X. S. Hu, "modnn: Memory optimal deep neural network training on graphics processing units," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 646–661, 2019.

[19] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 2017.

[20] L. Fu, S. Ma, L. Kong, S. Liang, and X. Wang, "FINE: A framework for distributed learning on incomplete observations for heterogeneous crowdsensing networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1092–1109, 2018.

[21] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*.

[22] S. U. Stich, "Local SGD converges fast and communicates little," in *7th International Conference on Learning Representations, ICLR 2019*.

[23] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," *Workshop on International Conference on Machine Learning, ICML Workshop*, 2019.

[24] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[25] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Proceedings of Machine Learning and Systems 2019, MLSys 2019*.

[26] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and E. Dutkiewicz, "Optimal online data partitioning for geo-distributed machine learning in edge of wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2393–2406, 2019.

[27] Y. You, I. Gitman, and B. Ginsburg, "Scaling SGD batch size to 32k for imagenet training," *CoRR*, 2017.

[28] Y. You, Z. Zhang, C. Hsieh, J. Demmel, and K. Keutzer, "Imagenet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing, ICPP*, 2018.

[29] T. Akiba, S. Suzuki, and K. Fukuda, "Extremely large minibatch SGD: training resnet-50 on imagenet in 15 minutes," *CoRR*, 2017.

[30] D. Yin, A. Pananjady, M. Lam, D. S. Papailiopoulos, K. Ramchandran, and P. Bartlett, "Gradient diversity: a key ingredient for scalable distributed learning," in *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2018.

[31] X. Zhao, A. An, J. Liu, and B. X. Chen, "Dynamic stale synchronous parallel distributed training for deep learning," in *39th IEEE International Conference on Distributed Computing Systems, ICDCS*, 2019.

[32] S. Wang, W. Chen, A. Pi, and X. Zhou, "Aggressive synchronization with partial processing for iterative ML jobs on clusters," in *Proceedings of the 19th International Middleware Conference, Middleware*, 2018.

[33] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani, "Robust and communication-efficient collaborative learning," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, 2019.

[34] C. Chen, W. Wang, and B. Li, "Round-robin synchronization: Mitigating communication bottlenecks in parameter servers," in *2019 IEEE Conference on Computer Communications, INFOCOM*, 2019.

[35] J. H. Lee and H. Kim, "Stalelearn: Learning acceleration with asynchronous synchronization between model replicas on PIM," *IEEE Trans. Computers*, vol. 67, no. 6, pp. 861–873, 2018.

[36] A. Sharif-Nassab, S. Salehkaleybar, and S. J. Golestani, "Order optimal one-shot distributed learning," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, 2019.

[37] Y. Li, J. Park, M. Alian, Y. Yuan, Z. Qu, P. Pan, R. Wang, A. G. Schwing, H. Esmaeilzadeh, and N. S. Kim, "A network-centric hardware/algorithm co-design to accelerate distributed training of deep neural networks," in *Proceedings of International Symposium on Microarchitecture, MICRO*, 2018.

[38] Q. Zhou, K. Wang, P. Li, D. Zeng, S. Guo, B. Ye, and M. Guo, "Fast coflow scheduling via traffic compression and stage pipelining in datacenter networks," *IEEE Trans. Computers*, vol. 68, no. 12, pp. 1755–1771, 2019.

[39] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2184–2199, 2019.

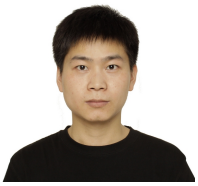
[40] A. Sapio, M. Canini, C. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. R. K. Ports, and P. Richtárik, "Scaling distributed machine learning with in-network aggregation," *CoRR*, 2019.

[41] M. Assran, N. Loizou, N. Ballas, and M. G. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*.

[42] J. Wang, H. Liang, and G. Joshi, "Overlap local-sgd: An algorithmic approach to hide communication delays in distributed SGD," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020*.

[43] S. Sra, A. W. Yu, M. Li, and A. J. Smola, "Adadelay: Delay adaptive distributed stochastic convex optimization," *CoRR*, 2015.

[44] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
[45] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proceedings of Annual Conference on Neural Information Processing Systems, NeurIPS*, 2015.
[46] J. Jiang, B. Cui, C. Zhang, and L. Yu, "Heterogeneity-aware distributed parameter servers," in *Proceedings of International Conference on Management of Data, SIGMOD*, 2017.
[47] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*, 2018.
[48] S. Shen, L. Xu, J. Liu, X. Liang, and Y. Cheng, "Faster distributed deep net training: Computation and communication decoupled stochastic gradient descent," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 2019.
[49] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.
[50] PyTorch, "Source code for MnistCNN," https://github.com/pytorch/examples/tree/server/mnist.
[51] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.
[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2012.
[54] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.
[55] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
[56] H. Inan, K. Khosravi, and R. Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," in *5th International Conference on Learning Representations, ICLR*, 2017.
[57] Aliyun, https://cn.aliyun.com/.
[58] H. Wang, "Source code for LOSP," https://github.com/AragornThorongil/JSAC2020/, 2020.



Haozhao Wang is currently a Ph.D. candidate in the School of Computer Science and Technology at Huazhong University of Science and Technology and a research assistant in the Department of Computing at The Hong Kong Polytechnic University. His research interests include Distributed Machine Learning and Federated Learning.

Zhihao Qu received his B.S. and Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2009, and 2018, respectively. He is currently an assistant researcher in the School of Computer and Information at Hohai University and in the Department of Computing at The Hong Kong Polytechnic University. His research interests are mainly in the areas of wireless networks, edge computing, and distributed machine learning. He is a member of IEEE.

Song Guo is a Full Professor in the Department of Computing at The Hong Kong Polytechnic University. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. His research interests are mainly in the areas of big data, edge AI, mobile computing, and distributed systems. With many impactful papers published in top venues in these areas, he has been recognized as a Highly Cited Researcher (Web of Science) and received over 12 Best Paper Awards from IEEE/ACM conferences, journals and technical committees. Prof. Guo is the Editor-in-Chief of IEEE Open Journal of the Computer Society. He has served on IEEE Communications Society Board of Governors, IEEE Computer Society Fellow Evaluation Committee, and editorial board of a number of prestigious international journals like IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing, IEEE Internet of Things Journal, etc. He has also served as chair of organizing and technical committees of many international conferences. Prof. Guo is an IEEE Fellow and an ACM Distinguished Member.

Ningqi Wang received the BE degree in computer science from Tongji University, Shanghai, China, in 2020. He is currently enrolled at INI, Carnegie Mellon University, USA, for his master's degree. His research interests include distributed system and networking.

Ruixuan Li is a professor in the School of Computer Science and Technology at Huazhong University of Science and Technology. He received the B.S., M.S. and Ph.D. in computer science from Huazhong University of Science and Technology, China in 1997, 2000 and 2004 respectively. He was a visiting researcher in Department of Electrical and Computer Engineering at University of Toronto from 2009 to 2010. His research interests include cloud and edge computing, big data management, and distributed system security. He is a member of IEEE and ACM.

Weihua Zhuang (M'93–SM'01–F'08) has been with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, since 1993, where she is a Professor and a Tier I Canada Research Chair in Wireless Communication Networks. Her current research focuses on resource allocation and QoS provisioning in wireless networks, and on smart grid. She is a co-recipient of several best paper awards from IEEE conferences. Dr. Zhuang was the Editor-in-Chief of IEEE Transactions on Vehicular Technology (2007-2013), and the Technical Program Chair/Co-Chair of the IEEE VTC Fall 2017/2016. She is a Fellow of the IEEE, a Fellow of the Canadian Academy of Engineering, a Fellow of the Engineering Institute of Canada, and an elected member in the Board of Governors and VP Publications of the IEEE Vehicular Technology Society.