2022 IEEE International Symposium on Information Theory (ISIT) | 978-1-6654-2159-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISIT50566.2022.9834752

Adaptive Worker Grouping for Communication-Efficient and Straggler-Tolerant Distributed SGD

Feng Zhu, Jingjing Zhang, Osvaldo Simeone, *Fellow, IEEE,* and Xin Wang, *Senior Member, IEEE*

***Abstract*—Wall-clock convergence time and communication load are key performance metrics for the distributed implementation of stochastic gradient descent (SGD) in parameter server settings. Communication-adaptive distributed Adam (CADA) has been recently proposed as a way to reduce communication load via the adaptive selection of workers. CADA is subject to performance degradation in terms of wall-clock convergence time in the presence of stragglers. This paper proposes a novel scheme named grouping-based CADA (G-CADA) that retains the advantages of CADA in reducing the communication load, while increasing the robustness to stragglers at the cost of additional storage at the workers. G-CADA partitions the workers into groups of workers that are assigned the same data shards. Groups are scheduled adaptively at each iteration, and the server only waits for the fastest worker in each selected group. We provide analysis and experimental results to elaborate the significant gains on the wall-clock time, as well as communication load and computation load, of G-CADA over other benchmark schemes.**

***Index Terms*—Adaptive selection, coding, distributed learning, stochastic gradient descent (SGD), grouping.**

I. Introduction

STOCHASTIC gradient descent (SGD)-based distributed learning has become an enabling technology for many artificial intelligence applications [1][2]. Wall-clock convergence time and communication load between workers and parameter servers (PS) are key performance indicators for distributed learning [3][4][5]. Wall-clock time performance is affected by workers that may be straggling [6], since in the standard implementation the PS needs to wait for all workers to respond at each iteration. Also, the communication overhead between the PS and the workers for the standard implementation grows linearly with the number of workers.

To address these issues, several techniques have been developed, including gradient coding (GC) and grouping [7], which leverage storage and computation redundancy to mitigate the impact of stragglers, and adaptive selection, which selects workers adaptively to reduce the communication load [8]. This paper proposes for the first time to combine grouping with adaptive selection for the distributed implementation of SGD (see Fig. 1).

Feng Zhu, Jingjing Zhang and Xin Wang are with the Department of Communication Science and Engineering, Fudan University, Shanghai 200433, China (e-mail: 20210720072@fudan.edu.cn; jingjingzhang@fudan.edu.cn; xwang11@fudan.edu.cn).

Osvaldo Simeone is with the Department of Informatics, King's College London, London WC2R 2LS, U.K. (e-mail: osvaldo.simeone@kcl.ac.uk).

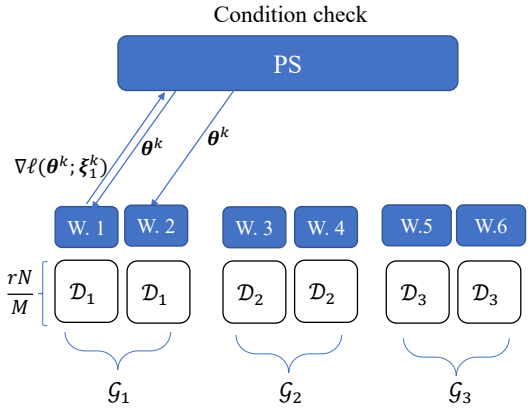

Fig. 1. Illustration of the proposed adaptive worker grouping technique with $M = 6$ workers and $M_G = 2$ workers per group, assuming $\mathcal{G}_1$ is the selected group.

A. Related Work

1) Stochastic Gradient Descent (SGD): In large-scale machine learning, SGD has become the primary algorithm to trade convergence rate for computation complexity [9]. Many algorithms aiming to reduce the variance of SGD have been designed. Notable examples include the works in [10], [11] and [12] that introduce stochastic variance reduced gradient (SVRG), stochastic average gradient (SAG) and stochastic dual coordinate ascent (SDCA), respectively. Also, adaptive SGD algorithms like AdaGrad [13], Adam [14] and AMSGrad [15] have been demonstrated to be effective and efficient improvements of SGD for deep learning tasks.

2) Gradient Coding and Grouping: GC and grouping were introduced in [7] as means to speed up iterations in distributed gradient descent (GD) at the cost of storage and computation redundancy by allowing the PS to wait only for a subset of fastest workers. This work has been extended in several directions. In [16], the authors developed algorithms to leverage partial computations at the stragglers; the communication and computation properties of GC were studied in [17], [18] and [19]; while GC was extended to distributed SGD in [20] and [21].

3) Adaptive Selection: Adaptive worker selection was introduced in [8] for distributed GD with the lazily aggregated gradient (LAG) scheme. The work [22] combined the idea of GC and LAG to develop lazily aggregated GC (LAGC), which achieves good results both in countering stragglers and in

reducing communication load. The idea of LAG was extended to SGD via the lazily aggregated stochastic gradient (LASG) in [23]; while Adam was substituted for the standard SGD in LASG to develop the communication-adaptive distributed Adam (CADA) in [24].

B. Main Contributions

This paper proposes a novel straggler-tolerant and communication-efficient scheme for SGD-based distributed learning, which combines the advantages of adaptive selection and grouping. The main idea is to adaptively schedule groups of workers at each iteration and then wait only for the fastest worker in each selected group. The proposed scheme can be interpreted as a grouping-based version of CADA [24], and is hence referred to as grouping-based CADA (G-CADA). It can also be viewed as a generalization of LAGC [22] from distributed GD to SGD. Importantly, unlike LAGC, G-CADA does not increase the computational load at the workers, since the mini-batch size does not depend on the storage redundancy. To gauge the performance of the proposed scheme, complexity analysis and numerical results are provided in terms of wall-clock time, as well as communication and computation loads.

The rest of this article is organized as follows. Section II presents the system model; the CADA scheme is reviewed in Section III; and Section IV introduces the proposed G-CADA scheme. Section V includes the analysis of different schemes in terms of different metrics; numerical results are given in Section VI and conclusions are drawn in Section VII.

II. System Model

A. Setting

The PS has available a global training dataset $\mathcal{D} = \{z_n = (\boldsymbol{x}_n, y_n)\}_{n=1}^N$, with $\boldsymbol{x}_n$ being a d-dimensional vector and y_n being a scalar label, and the objective here is to address the empirical risk minimization problem

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \mathcal{L}(\boldsymbol{\theta};\mathcal{D}) = \frac{1}{N}\sum_{z\in\mathcal{D}} \ell(\boldsymbol{\theta};z) \tag{1}$$

for some smooth loss function $\ell(\boldsymbol{\theta};z)$. To this end, SGD is applied by the PS as

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha^k F^k(\hat{\boldsymbol{\nabla}}^k\mathcal{L}), \tag{2}$$

where α^k is the stepsize, superscript k denotes the iteration index, $\hat{\boldsymbol{\nabla}}^k\mathcal{L}$ is an estimate of the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^k;\mathcal{D})$, and $F^k(\cdot)$ is some function that can be used to implement memory mechanisms such as Adam [14]. The estimate $\hat{\boldsymbol{\nabla}}^k\mathcal{L}$ is obtained by leveraging parallel computing on multiple workers as discussed next.

Prior to the start of the iterations (2), the PS distributes the dataset $\mathcal{D}$ among the set of M workers in set $\mathcal{M} \triangleq \{1,...,M\}$. Each worker $m \in \mathcal{M}$ is assigned a sub-data set $\mathcal{D}_m$ of rN/M samples, where integer $r \geq 1$ is defined as the storage redundancy factor. In particular, $r > 1$ implies that all data points are stored, and hence can be processed, at $r > 1$ workers. Following the principle of adaptive selection [8], at the beginning of each iteration k, the PS selects a subset of workers $\mathcal{M}_D^k \subseteq \mathcal{M}$, and it sends them the current global iterate $\boldsymbol{\theta}^k$. Each selected worker $m \in \mathcal{M}_D^k$ computes the local gradient $\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta}^k;\boldsymbol{\xi}_m^k) = \nabla_{\boldsymbol{\theta}}(\sum_{z\in\boldsymbol{\xi}_m^k}\ell(\boldsymbol{\theta}^k;z))$ where $\boldsymbol{\xi}_m^k$ is a mini-batch of fixed size randomly selected from the dataset $\mathcal{D}_m$ at iteration k.

Due to the fact that some workers might be straggling, only a subset $\mathcal{M}_U^k \subseteq \mathcal{M}_D^k$ of fastest workers that complete the computation upload the gradients. The PS then aggregates the uploaded gradients from the workers in subset $\mathcal{M}_U^k$ along with stale gradients from the workers in subset $\tilde{\mathcal{M}} \triangleq \mathcal{M}\backslash\mathcal{M}_U^k$, and arrives at the estimated gradient

$$\hat{\boldsymbol{\nabla}}^k\mathcal{L} = \sum_{m\in\mathcal{M}_U^k}\nabla\ell(\boldsymbol{\theta}^k;\boldsymbol{\xi}_m^k) + \sum_{m\in\tilde{\mathcal{M}}}\nabla\ell(\boldsymbol{\theta}^{k-\tau_m^k};\boldsymbol{\xi}_m^{k-\tau_m^k}), \tag{3}$$

where $\tau_m^k \geq 1$ is the number of iteration elapsed since the last update from worker m, which we refer to age of information (AoI) of worker m at iteration k. Finally, parameter $\boldsymbol{\theta}^k$ is updated through (2).

B. Performance Metrics

We explore the performance of different distributed SGD-based schemes in terms of wall-clock time, communication, and computation complexities as in [22]. To this end, for each iteration k, the computing time of each worker m, denoted by T_m^k, is assumed to be an exponential random variable with mean $\eta > 0$. Note that the mini-batch size is fixed, and hence the distribution of time T_m^k does not depend on the redundancy r, unlike in [22]. The variables $\{T_m^k\}_{m\in\mathcal{M}}$ are independently and identically distributed (i.i.d.) across all workers and iterations. Since the PS has to wait for the slowest worker in subset $\mathcal{M}_U^k$ that needs to upload the gradient, the wall-clock time complexity per iteration is given as

$$\bar{T} = \mathbb{E}\left[\max_{m\in\mathcal{M}_U^k}\{T_m^k\}\right]. \tag{4}$$

The communication load per iteration is defined as the average sum of the number of workers that download the global parameter from the PS and the number of workers uploading their fresh gradients, i.e.,

$$\bar{C} = \mathbb{E}\left[|\mathcal{M}_D^k| + |\mathcal{M}_U^k|\right]. \tag{5}$$

Finally, the computation load per iteration is defined as the total number of mini-batch gradients computed at the workers, i.e.,

$$\bar{P} = \mathbb{E}\left[|\mathcal{M}_D^k|\cdot\mu\right], \tag{6}$$

where the constant μ denotes the size of the mini-batch in terms of number of samples.

III. CADA

In this section, we review a close variant of CADA [24] that is modified here to fit the system model described in Section II, in which worker selection is carried out at the PS (and not at the workers as in [24]).

CADA assumes no computational redundancy, i.e., $r = 1$, and it splits the general dataset $\mathcal{D}$ into M equal-sized disjoint datasets $\mathcal{D}_1,...,\mathcal{D}_M$, with $\mathcal{D}_m$ allocated to worker m. At each

iteration k, the PS includes in the subset $\mathcal{M}_D^k$ worker m that violates the following condition introduced in [23]:

$$L_m^2 \left\| \boldsymbol{\theta}^k - \boldsymbol{\theta}^{k-\tau_m^k} \right\|^2 \leq c \sum_{d=1}^{D} \left\| \boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d} \right\|^2, \quad (7)$$

where L_m is the smoothness constant of the local function $\ell(\boldsymbol{\theta}; \mathcal{D}_m) = \frac{M}{rN} \sum_{z \in \mathcal{D}_m} \ell(\boldsymbol{\theta}; \mathcal{D}_m)$ of each worker and $c > 0$ is some constant; and D is an integer $D \geq 1$. The left-hand side of (7) estimates the change in the squared norm of the gradient at worker m; and the right-hand side represents the per-worker average contribution to the global iterate over the D latest iterations. Additionally, a worker is included in subset $\mathcal{M}_D^k$ if its AoI is greater than or equal to D.

The parameter $\boldsymbol{\theta}^k$ is updated with the AMSGrad rule [15], which uses the exponentially weighted stochastic gradient $\boldsymbol{h}^{k+1}$ as the direction of update and the weighted stochastic gradient magnitude vector $\boldsymbol{v}^{k+1}$ to adaptively control the stepsize. Specifically, the updated rule is

$$\boldsymbol{h}^{k+1} = \beta_1 \boldsymbol{h}^k + (1-\beta_1) \hat{\boldsymbol{\nabla}}^k \mathcal{L} \quad (8a)$$

$$\boldsymbol{v}^{k+1} = \beta_2 \hat{\boldsymbol{v}}^k + (1-\beta_2)(\hat{\boldsymbol{\nabla}}^k \mathcal{L})^2 \quad (8b)$$

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha^k (\epsilon \boldsymbol{I} + \hat{\boldsymbol{V}}^{k+1})^{-\frac{1}{2}} \boldsymbol{h}^{k+1} \quad (8c)$$

where $\beta_1 \in (0,1)$ and $\beta_2 \in (0,1)$ are momentum weights; $\hat{\boldsymbol{v}}^{k+1} \triangleq \max(\hat{\boldsymbol{v}}^k, \boldsymbol{v}^{k+1})$ is the element-wise maximum; $\hat{\boldsymbol{V}}^{k+1}$ is a $p \times p$ diagonal matrix whose diagonal vector is $\hat{\boldsymbol{v}}^{k+1}$; $\boldsymbol{I}$ is a $p \times p$ identity matrix; $\epsilon > 0$ is a small number; and the square operation in (8b) is element-wise.

IV. Grouping-Based CADA (G-CADA)

In this section, we introduce the proposed G-CADA scheme. G-CADA leverages storage redundancy, i.e., $r > 1$, via grouping in order to improve the robustness to stragglers of CADA, while still retaining CADA's benefits in terms of communication and computational loads.

In CADA, prior to training, the M workers are divided into G groups, $\mathcal{G}_1, ..., \mathcal{G}_G$, each with the same number of workers $M_G = M/G$. The global dataset $\mathcal{D}$ is split into equal-sized disjoint datasets $\mathcal{D}_1, ..., \mathcal{D}_G$, and each worker in group $\mathcal{G}_g$ is assigned $\mathcal{D}_g$. This implies a storage redundancy factor $r = M_G$ where M_G is chosen such that $M_G \leq r$. For each group $\mathcal{G}_g$, we define the group-wise AoI τ_g^k, which is maintained by the PS.

At each iteration k, the PS selects groups, rather than individual workers as in CADA. This is done by choosing the groups $\mathcal{G}_g$ that violate the condition

$$L_g^2 \left\| \boldsymbol{\theta}^k - \boldsymbol{\theta}^{k-\tau_g^k} \right\|^2 \leq c \sum_{d=1}^{D} \left\| \boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d} \right\|^2, \quad (9)$$

where L_g is the smoothness constant of the local function $\ell(\boldsymbol{\theta}; \mathcal{D}_g) = \frac{M}{rN} \sum_{z \in \mathcal{D}_g} \ell(\boldsymbol{\theta}; \mathcal{D}_g)$ of group $\mathcal{G}_g$. The condition has a similar interpretation to (7). We also include group $\mathcal{G}_g$ if the AoI τ_g^k is larger than D.

After determining the subset $\mathcal{G}_D^k$ of selected groups, the PS sends parameter $\boldsymbol{\theta}^k$ to all the workers in the selected groups, and each worker m in $\mathcal{G}_D^k$ computes $\nabla \ell(\boldsymbol{\theta}^k; \boldsymbol{\xi}_m^k)$. The fastest worker m in each selected group $\mathcal{G}_g$ uploads the computed gradient $\nabla \ell(\boldsymbol{\theta}^k; \boldsymbol{\xi}_g^k) = \nabla \ell(\boldsymbol{\theta}^k; \boldsymbol{\xi}_m^k)$ to the PS.

Then the PS updates the AoI of the selected groups as $\tau_g^{k+1} = 1$, while for other groups it sets $\tau_g^{k+1} = \tau_g^k$. Finally, the parameter $\boldsymbol{\theta}^k$ is updated via (8a)-(8c). Similarly we define $\tilde{\mathcal{G}} \triangleq \mathcal{G} \backslash \mathcal{G}_D^k$. The aggregated gradient is hence given as $\hat{\boldsymbol{\nabla}}^k \mathcal{L} = \sum_{g \in \tilde{\mathcal{G}}} \nabla \ell(\boldsymbol{\theta}^{k-\tau_g^k}; \boldsymbol{\xi}_g^{k-\tau_g^k}) + \sum_{g \in \mathcal{G}_D^k} \nabla \ell(\boldsymbol{\theta}^k; \boldsymbol{\xi}_g^k)$. The complete procedure of G-CADA is summarized in Algorithm 1.

Algorithm 1 G-CADA

Input: number of groups $G = M/M_G$, stepsize $\alpha^k > 0$, delay counter$\{\tau_g^0\}$, constants $\{c_d\}$, max delay D, smoothness constants $\{L_g\}$

Initialize: $\boldsymbol{\theta}^0$, $k = 0$

1: **repeat**
2: the PS checks the condition (9)
3: **for** each group $\mathcal{G}_g$ in $\mathcal{G}_D^k$ or satisfies $\tau_g^k \geq D$ in parallel **do**
4: all the workers in group $\mathcal{G}_g$ download $\boldsymbol{\theta}^k$ from the PS
5: each worker m in group $\mathcal{G}_g$ computes $\nabla \ell\left(\boldsymbol{\theta}^k; \boldsymbol{\xi}_m^k\right)$
6: the fastest worker uploads the gradient $\nabla \ell\left(\boldsymbol{\theta}^k; \boldsymbol{\xi}_g^k\right)$ with $\nabla \ell\left(\boldsymbol{\theta}^k; \boldsymbol{\xi}_g^k\right) = \nabla \ell\left(\boldsymbol{\theta}^k; \boldsymbol{\xi}_m^k\right)$
7: **end for**
8: server updates $\{\boldsymbol{h}^k, \boldsymbol{v}^k\}$ and $\boldsymbol{\theta}^k$ via (8a)-(8c) and sets $\{\tau_g^{k+1} = 1\}_{g \in \mathcal{G}_D^k}$ and $\{\tau_g^{k+1} = \tau_g^k + 1\}_{g \in \tilde{\mathcal{G}}}$
9: $k = k + 1$
10: **until** convergence criterion is satisfied

As detailed in Algorithm 1, while in CADA the PS has to wait for the slowest selected worker, in G-CADA the PS only needs to wait for the fastest worker in each selected group, which can potentially reduce the wall-clock time.

V. Analysis

In this section we analyze the wall-clock time complexity, communication complexity, and computation complexity of distributed Adam (a direct distributed implementation of Adam [14]), CADA (as described in Section III) and G-CADA. Based on [23], we can conclude that all schemes have sublinear convergence rate for convex loss functions. Motivated by this, in this section, we analyze the per-iteration metrics defined in Section II.

A. Wall-Clock Time

Let $T_{a:b}$ be the ath order statistics of i.i.d. variables $\{T_i\}_{i=1}^b$, which is the ath smallest value in the set $\{T_i\}_{i=1}^b$, and $T_{a:a} := T_a$. We have the average [25]

$$\bar{T}_{a:b} = \mathbb{E}[T_{a:b}] = \eta(H_b - H_{b-a}), \quad (10)$$

where $H_a = \sum_{k=1}^a 1/k$ is the ath harmonic number.

Distributed Adam: In distributed Adam, at each iteration, the PS broadcasts the parameter to all the workers and each worker m computes the gradient with the current iterate $\nabla \ell(\boldsymbol{\theta}^k; \boldsymbol{\xi}_m^k)$

and uploads it to the PS. Since the PS waits for all the workers, the average runtime per iteration is

$$\bar{T}_{D-Adam} = \bar{T}_M. \tag{11}$$

CADA: For CADA, from [22] and [8], we can upper bound the average workers selected per iteration as

$$\bar{M} = M \sum_{d=0}^{D} \frac{h(d)}{d+1} \leq M, \tag{12}$$

where we have defined the function $h(d) = (1/M) \sum_{m \in \mathcal{M}} \mathbb{I}\left(\bar{L}_{d+1}^2 < L_m^2 < \bar{L}_d^2\right)$, where $\mathbb{I}$ is the indicator function, with $\bar{L}_d^2 = c_d/(dM^2)$ and $\bar{L}_0 = \bar{L}_{D+1} = 0$. Therefore, since the PS waits for all the selected workers, the average runtime per iteration is

$$\bar{T}_{CADA} = \bar{T}_{\bar{M}}. \tag{13}$$

G-CADA: In G-CADA, the PS waits for the fastest worker in each selected group. Therefore the runtime T_g^G for group $\mathcal{G}_g$ is the first order statistic of the random variables $\{T_i\}_{i \in \mathcal{G}_g}$. Let $\bar{T}_{a:G}^G$ be the average of the ath smallest number of the random variables $\{T_g^G\}_{g=1}^G$ and $\bar{T}_{G:G}^G = \bar{T}_G^G$. This can be evaluated as

$$\bar{T}_{a:M_G}^G = \int_0^{+\infty} \left(1 - \left(F^G(x)\right)^a\right) dx, \tag{14}$$

where the cumulative distribution function (CDF) of each variable T_g^G is $F^G(x) = \sum_{j=1}^{M_G} \binom{M_G}{j} (F(x))^j (1-F(x))^{M_G - j}$. Defining the function $h_G(d) = (1/G) \sum_{g \in [G]} \mathbb{I}(\bar{L}_{G,d+1}^2 < L_g^2 < \bar{L}_{G,d}^2)$, with $\bar{L}_{G,d}^2 = c_d/(dG^2)$ and $\bar{L}_0 = \bar{L}_{D+1} = 0$, the average groups selected per iteration can be upper bounded as

$$\bar{G} = G \sum_{d=0}^{D} \frac{h_G(d)}{d+1} \leq G. \tag{15}$$

Since the PS waits for the slowest group, the average runtime per iteration for G-CADA is

$$\bar{T}_{G-CADA} = \bar{T}_{\bar{G}}^G. \tag{16}$$

By comparing (16) with (11) and (13), we observe that wall-clock time advantage is achieved as compared to distributed Adam and CADA.

B. Communication Load

Distributed Adam: The communication load per iteration for the distributed Adam scheme is

$$\bar{C}_{D-Adam} = \mathbb{E}\left[|\mathcal{M}_D^k| + |\mathcal{M}_U^k|\right] = 2M, \tag{17}$$

since all the workers download the parameters and upload the gradients.

CADA: For CADA, the communication load per iteration is

$$\bar{C}_{CADA} = \mathbb{E}\left[|\mathcal{M}_D^k| + |\mathcal{M}_U^k|\right] \leq 2\bar{M}, \tag{18}$$

with $\bar{M}$ defined in (12), since only the selected workers download the parameter and upload the gradients.

G-CADA: The communication load per iteration for G-CADA is

$$\bar{C}_{G-CADA} = \mathbb{E}\left[|\mathcal{M}_D^k| + |\mathcal{M}_U^k|\right] \leq \bar{G}(M_G + 1), \tag{19}$$

since all the workers in the selected groups download the parameters and the fastest ones upload the gradient.

C. Computation Load

Distributed Adam: The per-iteration computation load of the distributed Adam scheme is

$$P_{D-Adam} = \mathbb{E}\left[|\mathcal{M}_D^k| \cdot \mu\right] = \mu M \tag{20}$$

since all the workers need to compute the gradient.

CADA: The computation load per iteration for CADA is

$$P_{CADA} = \mathbb{E}\left[|\mathcal{M}_D^k| \cdot \mu\right] = \mu \bar{M} \tag{21}$$

since only the selected workers need to compute the gradients.

G-CADA: For G-CADA, the computation load per iteration is

$$P_{G-CADA:} = \mathbb{E}\left[|\mathcal{M}_D^k| \cdot \mu\right] \leq \mu \bar{G} \cdot M_G \tag{22}$$

since only the workers in the selected groups need to compute the gradients.

VI. Numerical Results

In this section, we provide numerical results to compare the performance of the considered schemes in terms of training loss, communication load, and computation load with respect to wall-clock time. For comparison, we also consider distributed SGD, which applies standard constant-stepsize SGD. We consider the linear regression model with MNIST dataset and quadratic error loss function. We set a total of $M = 12$ workers, number of groups $G = 3$, $M_G = r = 4$, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\mu = 10^{-4}$ sec. The learning rate α^k in (2) for distributed SGD is 2.6, while it is set to 0.01 for all other schemes. The constant c in (7) for CADA is $c = 2$ and for G-CADA we set $c = 0.3$ in (9).

Fig. 2 illustrates the loss function, communication complexity (5) and computation complexity (6) as a function of the wall-clock time. Comparing CADA with distributed Adam we observe that adaptive selection reduces communication and computation loads. G-CADA achieves further improvement in wall-clock time and communication load with respect to CADA thanks to grouping, while maintaining the same level of computation load as CADA when measured at the same value of training loss. For instance, given a training loss level of 10^{-1}, G-CADA requires a wall-clock time of 0.068 sec, a communication load of 12,292, and a computation load of 9,219. In contrast, CADA requires a wall-clock time of 0.380 sec, a communication load of 19,292, and a computation load of 9,646.

VII. Conclusions

In this paper, we have proposed a method that trades storage redundancy for wall-clock time and communication load in SGD-based distributed learning. The novel scheme, G-CADA, integrates grouping and adaptive selection. Grouping brings robustness to stragglers, while adaptive selection renders the system communication-efficient and decreases the computation load. Numerical results have shown that G-CADA achieves significant improvements in wall-clock time and reduced communication load, at the cost of storage redundancy.

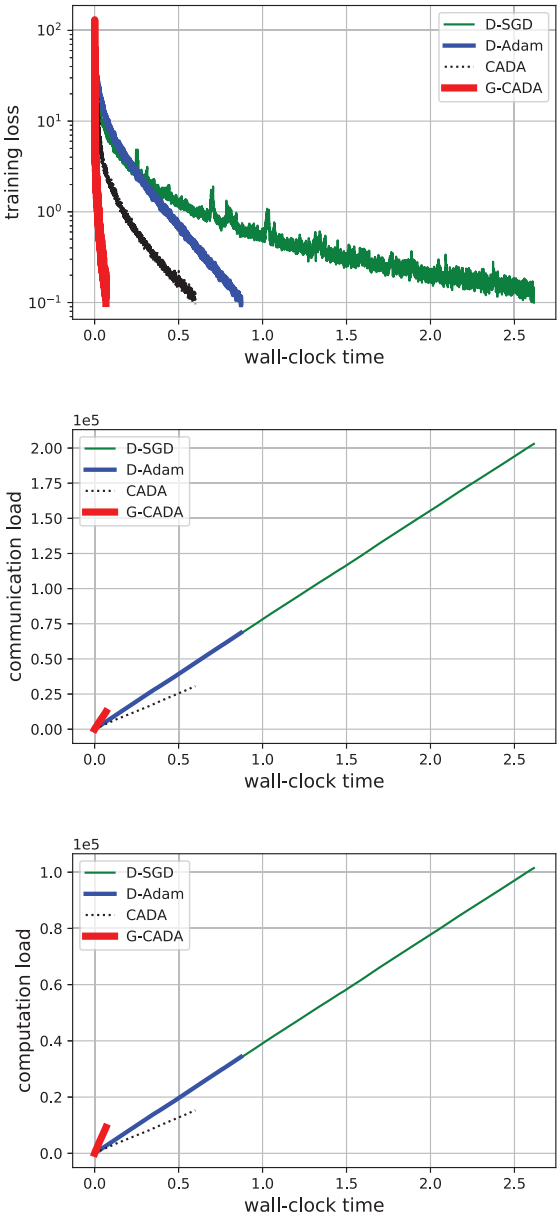

Fig. 2. Training loss, communication load (5) and computation load (6) against wall-clock time with exponential distribution for the computing times where $M = 12, G = 3, M_G = 4$ and $\mu = 10^{-4}$ sec.

VIII. Acknowledgement

The work of Feng Zhu and Jingjing Zhang has been supported by National Natural Science Foundation of China Grant No. 62101134. Osvaldo Simeone has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 725731). Xin Wang has been supported by the Innovation Program of Shanghai Municipal Science and Technology Commission Grant 20JC1416400, and the National Natural Science Foundation of China Grant No. 62071126.

References

[1] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Neural Information Processing Systems*, Dec. 2012, pp. 1223–1231.

[2] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. of Neural Information Processing Systems*, Dec. 2017, p. 4427–4437.

[3] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proc. Symp. Oper. Syst. Design Implement.*, 2014, pp. 583–598.

[4] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. J. Smola, "Scalable inference in latent variable models," in *Proc. ACM Int. Conf. on Web Search and Data Mining*, Feb. 2012, pp. 123–132.

[5] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ML via a stale synchronous parallel parameter server," in *Proc. Neural Information Processing Systems*, 2013, pp. 1223–1231.

[6] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. Neural Information Processing Systems*, vol. 27, Dec. 2014, pp. 19–27.

[7] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. on Machine Learning*, vol. 70, Aug. 2017, pp. 3368–3376.

[8] T. Chen, G. B. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. Neural Information Processing Systems*, 2018, p. 5055–5065.

[9] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT'2010*, 2010, pp. 177–186.

[10] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Neural Information Processing Systems*, 2013, pp. 315–323.

[11] N. Le Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proc. Neural Information Processing Systems*, 2012.

[12] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization." *Journal of Machine Learning Research*, vol. 14, no. 2, 2013.

[13] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.

[14] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computer Science*, 2014.

[15] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proc. Int. Conf. on Learning Representations*, 2018.

[16] E. Ozfatura, D. Gündüz, and S. Ulukus, "Gradient coding with clustering and multi-message communication," in *Proc. 2019 IEEE Data Science Workshop*, pp. 42–46.

[17] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded mapreduce," in *Proc. Annual Allerton Conf. on Communication, Control, and Computing*, 2015, pp. 964–971.

[18] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, 2017.

[19] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *Proc. Int. Conf. on Machine Learning*, 2018, pp. 5610–5619.

[20] H. Wang, Z. Charles, and D. Papailiopoulos, "Erasurehead: Distributed gradient descent without delays using approximate gradient coding," vol. arXiv:1901.09671, 2019. [Online]. Available: https://arxiv.org/abs/1901.09671

[21] R. Bitar, M. Wootters, and S. El Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 277–291, 2020.

[22] J. Zhang and O. Simeone, "LAGC: Lazily aggregated gradient coding for straggler-tolerant and communication-efficient distributed learning," *IEEE Trans. Neural Networks and Learning Systems*, pp. 1–13, 2020.

[23] T. Chen, Y. Sun, and W. Yin, "LASG: Lazily aggregated stochastic gradients for communication-efficient distributed learning," vol. arXiv:2002.11360, 2020. [Online]. Available: https://arxiv.org/abs/2002.11360

[24] T. Chen, Z. Guo, Y. Sun, and W. Yin, "CADA: Communication-adaptive distributed Adam," in *Proc. Int. Conf. on Artificial Intelligence and Statistics*, 2021, pp. 613–621.

[25] A. Mallick, M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *Proc. ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–40, 2019.