2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

GR-ADMM: A Communication Efficient Algorithm Based on ADMM

Xin Huang
School of Computer Engineering and Science
Shanghai University
Shanghai, China
Email: huangxin03@shu.edu.cn

Guozheng Wang
School of Computer Engineering and Science
Shanghai University
Shanghai, China
Email: gzh.wang@outlook.com

Yongmei Lei
School of Computer Engineering and Science
Shanghai University
Shanghai, China
Email: lei@shu.edu.cn

***Abstract*—In recent work, the decentralized algorithm has received more attention. In the centralized network, the worker nodes need to communicate with the central nodes, which results in the growth of communication traffic with the network expansion. Based on the purpose of reducing the communication costs in the distributed system, we proposed a decentralized algorithm based on ADMM - Grouping Ring All-Reduce ADMM (GR-ADMM) in this paper. First, GR-ADMM adopts decentralized architecture to avoid the problem of communication bottleneck in the central network. Second, to ensure the scalability of the distributed system, GR-ADMM introduces the Ring All-Reduce to the ADMM. Ring All-Reduce architecture has the advantage of its constant communication overhead. However, its performance is bounded by the stragglers (i.e., slow nodes). Third, GR-ADMM adopts the grouping strategy to alleviate the problem of stragglers. Experiments show that our algorithm has better convergence performance than QSGD and GADMM, especially in massive clusters. Compared with GADMM's, the overall communication cost of GR-ADMM is reduced by 72%.**

Keywords—Communication-efficient decentralized algorithm, ADMM，GR-ADMM, Ring All-Reduce, grouping strategy

I. INTRODUCTION

Distributed machine learning (DML) has attracted increasing attention in various fields, such as image recognition, machine translation, and so on. With the rapid growth of mobile devices, centralized machine learning computing via cloud computing incurs considerable communication overhead and privacy concerns [1]. Thus, the consensus is that future machine learning tasks will start from the network edge, namely devices [2]. Distributed optimization aims at solving the consensus problem

$$x^* = \underset{x}{argmin} \sum_{i=1}^{N} f_i(x) , \quad (1)$$

which is defined over a bidirectionally connected network with N nodes. Each node i has a local objective function $f_i(x)$: $R^N \to R$. All the nodes share the common optimization variable $x \in R^N$ and the nodes aim to find an optimal argument $x^* \in R^N$ cooperatively. In this process, the performance of distributed algorithms is usually characterized by the computation cost and communication cost. For an extensive scale network, the communication cost takes the dominant position compared to the computation cost [3]. Therefore, much work focuses on reducing communication overheads in the distributed system.

The standard distribution strategy in machine learning is data parallelism [4]: each node maintains its copy of the model and nodes jointly optimize the same model on different parts of the training data with parameters exchanged over the network. Network communication is involved in this process, usually using the Parameter Server [5] architecture or collective routines (e.g., Ring All-Reduce [6]).

The Parameter Server (PS) architecture provides a logically separate device to store global parameters. Typically, data parallelism with PS contains several steps [7]: (1) each worker node computes its local parameters and sends them to PS; (2) PS sums the parameters sent by worker nodes; (3) each worker node gets parameters from PS.

In the Ring All-reduce architecture, all the nodes are organized as a logistic ring. Each node in the cluster only communicates with its peers. Ring All-Reduce architecture does not need any central node to hold the global parameters. Thus, it is a decentralized network by nature.

With the parameter server nodes, the PS architecture can easily manage all worker nodes, and it enables more flexible parameter synchronization. However, the central nodes need to hold models for all worker nodes and communicate with all worker nodes. As a result, the central nodes may become the bottleneck of the network expansion [8]. To solve the problem, Parameter Hub [9] introduces a high-performance multi-tenant, rack-scale PS design, still, it requires special hardware to support its system, which does not exist in a typical distributed environment.

Ring All-Reduce architecture has constant communication traffic for different sizes of clusters [10], and it has shown good performance on the distributed training, but it is a fully synchronous communication architecture. Thus, the stragglers may slow down the entire system. ADMMLIB [11] introduces asynchronous communication to Ring All-Reduce architecture, but it needs a coordinate node to manage synchronization conditions.

In our work, we focus on the problem of communication bottlenecks and stragglers in the distributed system. Based on the ADMM algorithm, we proposed the GR-ADMM, our contributions are as follows:

- We apply the ADMM algorithm in the decentralized network. The worker nodes are divided into several groups, and only the nodes in the same group can communicate with each other.
- To ensure high communication efficiency, we realize the communication based on the Ring All-Reduce in the decentralized network. And with the grouping strategy, the effect of stragglers is decreased.

978-1-6654-3574-1/21/$31.00 ©2021 IEEE

DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00041

- We evaluate the performance of the proposed GR-ADMM algorithm with large data sets. Experiments show that GR-ADMM has better convergence speed and low communication costs.

The rest of this paper is organized as follows. In Section II, we introduce the related work, and in Section III, we describe the GR-ADMM design. Section IV presents the performance of GR-ADMM, and the last section gives some concluding remarks.

II. Related Work

A. Alternating Direction Method of Multipliers

Alternating Direction Method of Multipliers (ADMM) [12] has been widely studied due to its outstanding convergence guarantees. The objective function of the original problem can be decomposed into several solvable sub-problems by the ADMM algorithm.

Distributed ADMM was first proposed by Boyd and is implemented in the centralized network, like AD-ADMM [13]. AD-ADMM uses the master node to store the latest parameters sent by the worker nodes and compute the global parameters. To reduce the communication waiting time, AD-ADMM was realized under the stale synchronous parallel (SSP [14]) protocol.

However, the master node in the central network will be the bottleneck of network expansion [2] due to the following two reasons. First, large-scale machine learning tends to learn large models, which puts tremendous pressure on the memory of the master node. Second, under the central architecture, the master node must communicate with all worker nodes, causing network congestion.

The decentralized ADMM algorithm relaxes the algorithm's consistency constraints. Problem (1) can be expressed as

$$\min_{x_i,u_{ij}} \sum_{i=1}^{N} f_i(x_i)$$
$$s.t.\ \ x_i = u_{ij}, x_j = u_{ij}, \forall (i,j) \in E, \tag{2}$$

Where E represents the set of edges of the distributed network and u_{ij} is an auxiliary variable, which maintains the consistency constraint of node i and node j.

Decentralized ADMM does not need central nodes to aggregate local variables of all nodes. Thus, the problem of communication bottleneck is avoided.

B. Decentralized Optimization Algorithm

Decentralized consensus optimization has attracted much more attention in recent years. For the decentralized network, each worker node only communicates with its neighboring nodes, which makes the failure of one node have little impact on the distributed system [15].

D-ADMM [16] is an early attempt for ADMM applying in the decentralized network, which shows that decentralized algorithm requires fewer communications to achieve a given accuracy level. [17] theoretically proved that the decentralized ADMM converges at a globally linear rate if the objective function is strongly convex.

The distributed ADMM algorithm can accomplish convergence through local data and parameters transmitted over the network. In order to relieve the communication pressure of the whole distributed system, much work commits to reduce the communication traffic of the system. GADMM [3] divides all worker nodes into two parts, and only half of the worker nodes are competing for communication resources at any time. GADMM reduces the communication costs by fixing the communication sequence, but it may suffer from the problem of the stragglers. SCCD-ADMM [15] concentrates on reducing the total cost of the system. Unlike the normal decentralized algorithm, SCCD-ADMM selects part of the neighboring nodes to communicate by evaluating the communication cost and computation cost.

For decentralized Stochastic Gradient Descent (SGD), two main algorithms were proposed so far, which are Decentralized Parallel SGD (D-PSGD) [2] and Asynchronous Decentralized Parallel SGD (AD-PSGD) [8]. D-PSGD relies on a fixed communication topology and it may encounter the problem of slow nodes. AD-PSGD introduces a random communication mechanism over the D-PSGD, but it may slow the convergence rate.

Since there is no central node, the decentralized algorithm does not have the problem of communication hotspot. Thus, much work commits to reduce communication overhead to increase the extensibility of the distributed system.

III. Proposed Algorithm: GR-ADMM

To solve the problem of communication hotspots, we apply the ADMM algorithm to the decentralized network. Unlike the normal decentralized algorithms, we use Ring All-Reduce architecture to realize communication between nodes.

In this section, we use the undirected graph $G = (V, E)$ denotes the set of nodes and edges, where $V \in \{1, \cdots, n\}$ and $E \in V * V$. Each node in the graph can only communicate with its neighbors, N_i denotes the set of neighboring workers of node i, and $|N_i|$ is the number of the neighboring nodes.

A. Problem Formulation

Point-to-Point communication is usually adopted in the decentralized network. However, the communication costs of normal Point-to-Point communication will increase a lot as the number of nodes in the cluster increase.

Currently, Ring All-Reduce has a good performance on the distributed SGD [10], but the decentralized ADMM has not benefited from the all-reduce communication. With Ring All-Reduce architecture, the communication traffic will not increase significantly as the distributed network expands.

[17] proposed that problem (2) can be solved with the following iterations:

$update\ x_i^{k+1}$ by solving:

$$\nabla f_i\left(x_i^{k+1}\right) + \alpha_i^k + 2\rho|N_i|x_i^{k+1} - \rho(|N_i|x_i^k + \sum_{j\in N_i} x_j^k) = 0 \tag{3}$$

$update\ \alpha_i^{k+1}$:

$$\alpha_i^{k+1} = \alpha_i^k + \rho(|N_i|x_i^{k+1} - \sum_{j\in N_i} x_j^{k+1}),$$

where ρ denotes the penalty term parameter, $x_i^k \in R^N$ is the local variables and $\alpha_i^k \in R^N$ is the dual variables at iteration k.

The algorithm of GR-ADMM: It is easy to find that the iterative formula (3) can apply to the normal Point-to-Point

communication, but it is not suitable for all-reduce communication, since each node has different neighboring nodes sets N_i. We reformulate the iterative formula (3) as formula (4).

$update\ x_i^{k+1}$ by solving:

$$x_i^{k+1} = \underset{x_i}{argmin} f_i(x_i) + x_i^T \alpha_i^k - \rho x_i^T \left[(|N_i| - 1)x_i^k + \sum_{j\in(N_i+i)} x_j^k\right] + \rho|N_i|x_i^2 \quad (4)$$

$update\ \alpha_i^{k+1}$:

$$\alpha_i^{k+1} = \alpha_i^k + \rho[(|N_i| + 1)x_i^{k+1} - \sum_{j\in(N_i+i)} x_j^{k+1}],$$

Within the Ring All-Reduce architecture, the value of $i + N_i$ is the same for each node i. Thus, this form of ADMM algorithm can be easily implemented with all-reduce architecture since each node i exchanges the local parameters x_i to get the same parameters $\sum_{j\in(i+N_i)} x_j$.

In order to utilize the high communication efficiency of the ring-based architecture, the paper proposes a new algorithm based on ADMM. In the next section, the paper will introduce the model design of GR-ADMM.

B. The Model Design of Grouping Ring All-Reduce ADMM

Normal decentralized algorithms choose to communicate with their neighboring nodes [2][16][17] or subsets of their neighbors [8][15], which may cause high communication traffic ($|N_i|B$, where B is the bytes of the parameter) and introduce few "hot spots" [17] . Unlike normal point-to-point communication, the nodes in Ring All-Reduce are equal, thus there are no hot spots in the network. And the communication cost ($\frac{2|N_i|B}{|N_i|+1}$) is constant for different sizes of cluster.

Thus, the paper chooses Ring All-Reduce for high communication efficiency. However, Ring All-Reduce is recognized as a fully synchronous communication model. In practice, the distributed system may suffer from the problem of stragglers. Thus, the paper comes up with an idea that communication can be carried out in a smaller group. Based on the idea, the paper proposes a new communication model named Grouping Ring All-Reduce ADMM (GR-ADMM).

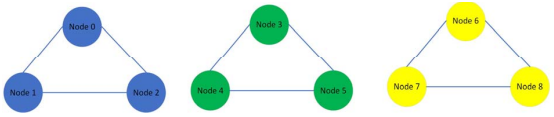

Fig. 1. An example of the grouping strategy of GR-ADMM, the nodes with the same color means they are in the same group and the lines between nodes indicates the nodes can communicate with each other.

Basic idea: The basic idea of GR-ADMM is to divide all nodes into several groups through an appropriate grouping strategy. For example, in Fig.1, we suppose that nine working nodes are divided into three groups, and nodes in the same group are colored in the same color. The nodes in the same group communicate using the Ring All-Reduce communication pattern. However, nodes in the same group can communicate easily, but nodes in the different groups cannot exchange their parameters. As a result, the algorithm will not converge.

To realize the communication between inter-group, GR-ADMM adopts the grouping strategy.

Grouping Strategy: Some work of centralized algorithms like [11] also adopts the method of grouping, and they realize the parameter exchange of nodes in different groups through inter-group communication. However, GR-ADMM cannot take this approach, since all nodes are equal in the decentralized network and there is no central node to coordinate the communication between groups. Thus, GR-ADMM adopts the grouping strategy instead of inter-group communication.

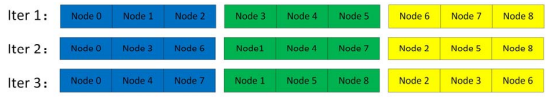

Fig. 2. An example of the grouping strategy of GR-ADMM at different iteration. To realize inter-group communication, GR-ADMM uses different grouping strategies at different iteration.

As is shown in Fig.2, the grouping strategy must be determined before communication starts. Unlike Fig.1, the grouping strategy in Fig.2 is different at different iteration. The procedure is shown in Algorithm 1.

Algorithm 1: Grouping Ring All-Reduce ADMM

For all nodes $i \in V$ [in parallel] do

Initialize:$x_i^{(0)} = 0, \alpha_i^{(0)}, k = 0, Group_i$

End do

While $k < maxiteration$ do

$k = k + 1$

For all nodes $i \in V$ [in parallel] do

Get $Group_i^{(k)}$ at iteration k

Get $\sum_{j\in(N_i+i)} x_j^k$ in the same $Group_i^{(k)}$

Update $x_i^{k+1} = \underset{x_i}{argmin} f_i(x_i) + x_i^T\{\alpha_i^k - \rho[(|N_i| - 1)x_i^k + \sum_{j\in(N_i+i)} x_j^k]\} + \rho|N_i|x_i^2$

Get $\sum_{j\in(N_i+i)} x_j^{k+1}$ in the same $Group_i^{(k)}$

Update $\alpha_i^{k+1} = \alpha_i^k + \rho[(|N_i| + 1)x_i^{k+1} - \sum_{j\in(N_i+i)} x_j^{k+1}]$

End do

End do

C. The Static Grouping Strategy and Dynamic Grouping Strategy

Section B proposes that GR-ADMM choose a grouping strategy rather than inter-group communication. Notice that $Group_i$ is determined by the user, so it could be changed for different algorithms or datasets to achieve a faster convergence speed, but it should follow the requirement: the parameters of one node should be disseminated to other nodes in several iterations.

In this section, the paper proposes two grouping strategies: static grouping strategy and dynamic grouping strategy.

Static Grouping Strategy: Grouping will introduce few additional computation overheads. Thus, as Algorithm 1 shows, it is wise to determine the grouping strategy before the algorithm starts. Communication is a time-consuming operator, so conflict should be avoided in the grouping strategy. Grouping with some fixed patterns is much more suitable in GR-ADMM. Next, the paper will introduce the static grouping strategy based on the above ideas.

As Fig.3 shows, GR-ADMM modifies the group based on half of the nodes in the group (rules: the front half of the nodes, the back half of the nodes, the odd number of the nodes, the even number of the nodes). Fig.3 shows how Group 0 changes the group. In practice, for each iteration, each group will choose the nodes according to the random rule and exchange with one other group randomly. And GR-ADMM can reuse serval iterations' grouping strategy, Thus, the worker nodes need not store all grouping strategies. Fig.3 shows how to form the grouping strategies in Group 0 if the number of nodes is even. If the number of nodes is odd, GR-ADMM will use virtual nodes to supplement, the process of grouping is still as Fig.3 shows, and GR-ADMM will exclude virtual nodes when communicating begins.

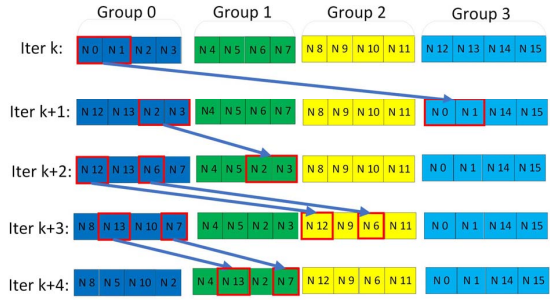

Fig. 3. Static Grouping Strategy in Group 0. At each iteration, half of the nodes in each Group will select another group randomly to exchange. And the figure shows how Group 0 determines the group strategy of Group 0.

Dynamic Grouping Strategy: By introducing the static grouping strategy, the impact of slow nodes can be controlled in a set of working nodes. Although the problem of stragglers can be relieved by grouping, the slow nodes will still cause excessive waiting time. To further reduce the communication waiting time, the paper attempts to solve this problem by modifying the grouping strategy before each iteration starts.

The purpose of optimizing the grouping strategy is to make the slow nodes' iteration catch up with the fast nodes' and make the slow nodes will not slow down the system. Therefore, in this process, slow nodes need to be defined. The time consumption of a single node includes computation time (t_{compu}) and communication time (t_{comm}), which can be regarded as the total time of an iteration (t_{iter}). By recording t_{iter}, we can determine the slow nodes, which have a larger value of t_{iter}.

Inspired by parameter server, the paper introduces a coordinator node to the system to record the value of t_{iter} and readjust the grouping strategy. The procedure is shown in Algorithm 2.

Algorithm 2: Grouping Ring All-Reduce ADMM with **the Coordinator node**

Computation nodes:

For all computation nodes $i \in V$ [in parallel] do

Initialize:$x_i^{(0)} = 0, \alpha_i^{(0)}, k_i = 0$

End do

While $k_i < maxiteration$ do

$k_i = k_i + 1$

For all computation nodes $i \in V$ [in parallel] do

Send k to the Coordinator node

Get $Group_i^{(k_i)}$ from Coordinator node

Get $\sum_{j\in(N_i+i)} x_j^{k_i}$ in the same $Group_i^{(k_i)}$

Update $x_i^{k_i+1} = \underset{x_i}{argmin} f_i(x_i)$
$+x_i^T\{\alpha_i^{k_i} - \rho[(|N_i|-1)x_i^{k_i} + \sum_{j\in(N_i+i)} x_j^{k_i}]\}$
$+\rho|N_i|x_i^2$

Get $\sum_{j\in(N_i+i)} x_j^{k_i+1}$ in the same $Group_i^{(k_i)}$

Update $\alpha_i^{k_i+1} = \alpha_i^{k_i} + \rho[(|N_i|+1)x_i^{k_i+1}$
$-\sum_{j\in(N_i+i)} x_j^{k_i+1}]$

End do

Coordinator node:

Initialize:$k = 0$, $Group^k$ for each iteration k,
$t_{i\_iter} = 0$ for all computation nodes i

While $k < maxiteration$ do

Get k_i from Computation node i

Update t_{i_iter} for computation nodes i

If $k_i > k$ do

$k = k_i$

Modify the grouping strategy at iteration k according to t_{i_iter}

End do

Send $Group_i^{(k)}$ at iteration k to all nodes

End do

As is shown in Fig.4, the coordinator node will keep the origin grouping strategy (static grouping strategy) and t_{iter}. The t_{iter} is stored in the format of key-value, in which key is the number of worker nodes and value is the time difference between two iterations. For example, at iteration $i+1$, $(0, T1 - T0)$ will be stored for node 0. By recording t_{iter}, the coordinator node can find the stragglers and modify the grouping strategy. In practice, we can choose one worker node to be the coordinate node. Unlike the central node in the master-

slave architecture, the coordinator node will not be the bottleneck of the system since the computation complexity ($O(nlogn)$, where n is the number of worker nodes) is fairly low and the data traffic (a few kilobytes) is small.

By introducing a coordinator node, it may cause additional computation and communication overhead. It is necessary to optimize this process. Grouping conflict will bring extra communication waiting time. To avoid this problem, the coordinator node will modify the grouping strategy only when the first worker node of the new iteration asks for the grouping strategy, and then the grouping strategy at this iteration is determined. Thus, the coordinator can send the grouping strategy to other nodes without requests from other worker nodes. In this way, the small amount of computation and communication of the coordinator node is overlapped with the computation of worker nodes.

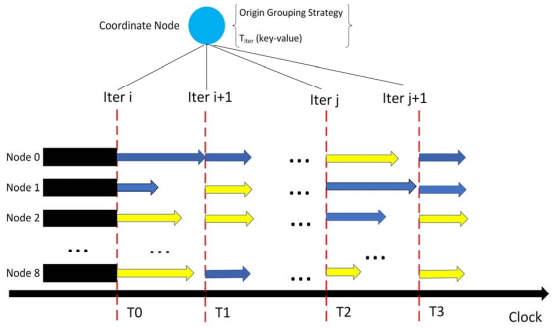

Fig. 4. Modifying the grouping strategies in GR-ADMM, arrows with the same colors means they communicate in the same group

Fig.4 shows how the coordinator node modifies the grouping strategy. Suppose node 1 in Fig.4 is the straggler in the distributed system. When the nodes request the set of the communication nodes, the coordinator node will assign the fast nodes to the node if the node has a larger or smaller value of t_{iter}, since, in this way, the fast nodes will not be slowed down by slow nodes (see T1 at Fig.4, when the node 0 requests for communication, the coordinate node will modify the grouping strategy according to the $t_{iter\ i}$ and the node1 will not slow down the system) and the slow nodes will have a chance to catch up with the fast nodes (see T3 at Fig.4, the coordinate node will modify the grouping strategy according to the $t_{iter\ j}$ and node 1 will catch up to the iteration).

GR-ADMM applies the ADMM algorithm in the decentralized network to avoid the problem of hot spots. And with Ring All-Reduce architecture, GR-ADMM can achieve high communication efficiency and constant communication traffic. To realize the communication between inter-group and avoid the problem of stragglers, grouping strategies are introduced in the GR-ADMM. In the next section, the paper will demonstrate the advantages of GR-ADMM by experiments.

IV. Experiment

In this section, we evaluate the performance of the GR-ADMM in logistic regression problems in the HPC cluster of Shanghai University. GR-ADMM is implemented in the C++ language and we use the MPICH library for distributed communication.

We compare GR-ADMM with two benchmark algorithms, (i) Grouping Alternating Direction Method of Multipliers (GADMM) [3] and (ii) Quantized SGD (QSGD) [21] in terms of the convergence and the communication cost of the system.

In the experiment, we solve the logistic regression problem:

$$f(w) = \min_{w} \sum_{i=1}^{N} log[\,1 + exp(-y_i x_i^T w)] + \\ +\lambda ||w||_2^2 \ , \tag{8}$$

where $w \in R^n$ is the model parameter, $x_i \in R^n$ is the sample, $y_i \in \{-1,1\}$ is the label of the sample and $\lambda \geq 0$ is the scalar regularization parameter. We use the trust region Newton method [23] to solve the sub-problem in ADMM.

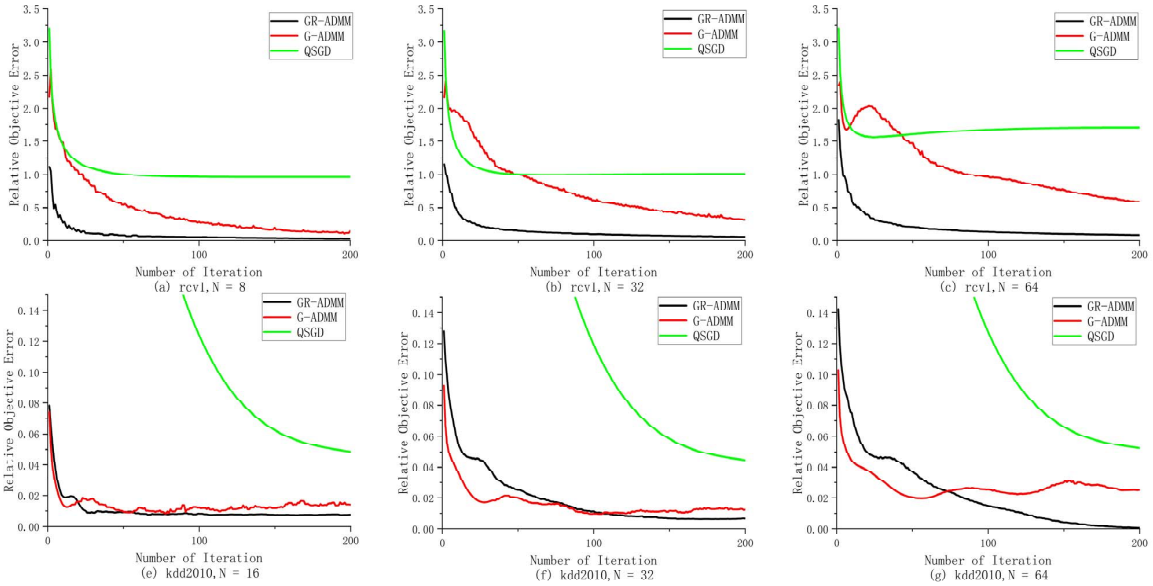

Fig. 5. Relative Objective Error Curve of GR-DMM, GADMM and QSGD

We consider the performance of GR-ADMM on the two datasets: rcv1[1] and kdd2010 raw version (bridge to algebra)[2]. The rcv1 dataset has 677,399 samples and 47,236 features, and the kdd2010 dataset has 19,264,097 samples and 1,163,024 features. Compared to rcv1, the kdd2010 dataset is much sparse and it converges slower.

For GR-ADMM and GADMM, the penalty term parameter ρ is set to 3e-3 for the rcv1 and 1 for the kdd2010. For QSGD, the paper adopts decreasing learning rate. In this section, we use N denotes the number of nodes in the cluster and $|N_i|$ denotes the number of neighboring nodes of the node i.

A. Convergence Curve

In this section, we test the convergence performance of GADMM, QSGD, and GR-ADMM. The experiments are set up with two datasets. For GR-ADMM, we set the worker nodes of each group to 4. The convergence performance of each algorithm is measured by (9) at iteration k.

The relative objective error of node i:

$$\left|\frac{f_i(w_i^{(k)})-f_i(w^*)}{f_i(w^*)}\right| \tag{9}$$

We conduct experiments on 16, 32, and 64 nodes. Fig.5 shows the convergence curve for the rcv1 dataset and the kdd2010 dataset.

For both the rcv1 and the kdd2010 dataset, Fig.5 indicates that GADMM and GR-ADMM have a faster convergence speed than QSGD, which shows ADMM has the advantage of reaching a higher accuracy in few iterations.

[17] theoretically analyze the factors that affect the convergence of decentralized ADMM, including topology-related properties of the network, the condition number of the objective function, and the algorithm parameter

GADMM divided all nodes into two groups and one worker node only communicates with up to two nodes. According to the factors affecting the convergence, we compare GADMM and GR-ADMM. In this section, these two algorithms all solve the same problem, so the condition number of the objective function is the same. And we choose the same penalty term parameter for them to fix the algorithm parameter. Thus, topology-related properties will eventually affect the convergence performance. Topology-related properties of the network are related to the following factors.

1) Condition Number of the Network: [17] define the connectivity ratio of the network (p) as the actual number of edges divided by $\frac{N(N-1)}{2}$. Thus, compare to GADMM ($\frac{2}{N}$), GR-ADMM ($\frac{2}{N-1}$) has larger p ,which lead faster convergence.

2) Network Diameter: Network Diameter (D) is defined as the longest distance between any pair of agents in the network. D is related to how many iterations the information from one worker to all the other workers. The value of D in GADMM is N, while D in GR-ADMM is $\frac{N}{N_i}$. A larger D causes a worse condition number of the network and thus slower convergence. With the grouping strategy in GR-ADMM, the information of one node can transmit to others with fewer iteration, resulting in faster convergence.

3) Geometric Average Degree: Define the geometric average degree $d_s = \sqrt{d_{min} d_{max}}$ reflects the nodes' number of neighbors in a geometric average sense, where d_{min} and d_{max} is the largest and smallest degrees of the agents in the network. Thus, the value of d_s in GADMM is $\sqrt{2}$ and GR-ADMM's is 2. The larger d_s in GR-ADMM implies better connectedness and thus a smaller condition number of the network as well as faster convergence.

4) Imbalance of Bipartite Network: The value of imbalance is defined by $|l_A| - |l_B|$, which can vary between 0 and $N-1$, and $|l_A|$ represents the number of agents in one group, $|l_B|$ is the number of agents in another group. GADMM divides all nodes into two groups, and the number of nodes is the same in different groups. GR-ADMM is not a normal bipartite network, but the number of nodes is also the same in different groups. We can infer that the value of $|l_A| - |l_B|$ is all zero for GADMM and GR-ADMM. Thus we ignore the factor of imbalance.

As Fig.5 shows, compared to GADMM, GR-ADMM has better convergence performance on the rcv1 dataset. And for kdd2010, GADMM has a lower relative objective error at the

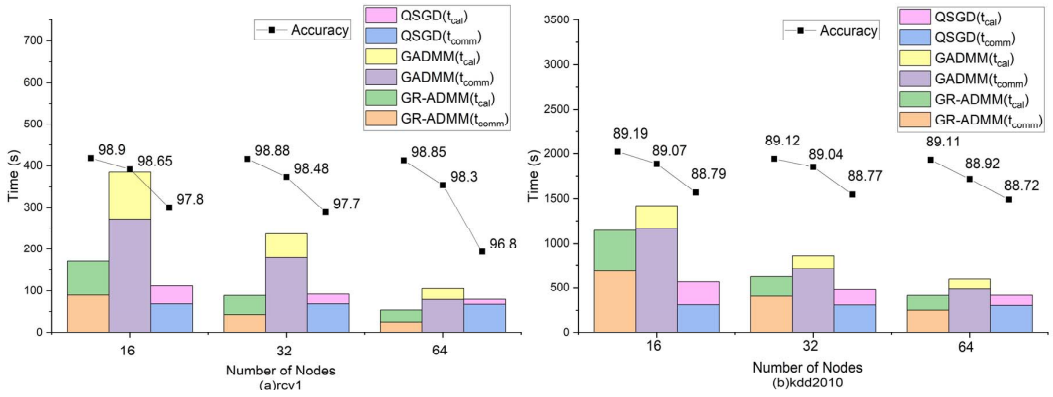

Fig. 6. Performance comparisons among GR-ADMM, GADMM and QSGD

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#rcv1.binary
[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#kdd2010%20raw%20version%20(bridge%20to%20algebra)

beginning, while GR-ADMM will have a better performance after several iterations. In summary, GR-ADMM shows convergence advantages both in theory and experiments.

B. Total Cost of the System

In Section III.B, the system time is divided into computation time (t_{compu}) and communication time (t_{comm}). Communication time includes synchronization waiting time and data transmission time. In this section, we evaluate the performance of the distributed systems according to the computation time and communication time. We compare these three systems by running them to 200 iterations. Fig.6 shows the performance comparison on the rcv1 and kdd2010. From Fig.6, we can find that computation time reduces as the level of parallelism increases since the amount of calculation is evenly amortized by more nodes.

For these three distributed systems, the communication traffic at each iteration is fixed, since the number of communication nodes of each iteration for each node is fixed. Therefore, we can infer that, for different sizes of clusters, data transmission time will not change and the synchronous waiting time is the main factor for the change of communication time. Fig.6 indicates that the communication time is decreasing with the increase of parallelism. Because the decrease in computation time leads to a decrease in network waiting time. From Fig.5 and Fig.6, the experiments show that GR-ADMM has high accuracy and lower objective error after 200 iterations, which reflects that GR-ADMM has outstanding global convergence. In Fig.6, we can find GR-ADMM has a lower time overhead, especially in communication time, which shows a greater performance of Ring All-Reduce than normal point-to-point communication.

In Fig.6, QSGD takes less time when testing with 16 nodes, that is because SGD has low computation costs but it converges slow. And with the network expands, the calculation load decreases, and SGD loses its advantages. For rcv1, GR-ADMM can reduce communication time by about 72% compared to GADMM and 27% compared to QSGD. For kdd2010, the communication time is reduced by 23% compared to GADMM. When compared to QSGD, GR-ADMM shows similar performance on time, but GR-ADMM has higher accuracy. Our analysis is that the solving method of the sub-problem takes too much time for high accuracy. And with the network expands, the computation costs decrease, the GR-ADMM begins to show better performance.

C. Performance of the Coordinate node

To solve the problem of stragglers, GR-ADMM takes the strategy to modify the grouping strategy. In practice, it is not easy to control the appearance of the slow nodes. Inspired by [24], we try to create the stragglers artificially to test the effectiveness of this method. We simulate the stragglers by randomly selecting working nodes and prolonging their computing time.

Compared to RCV1, the size and dimension of the dataset of kdd2010 are larger, and the algorithm will spend more time on computation. As is shown in Fig.7, it does not affect the computation time that modifying the grouping strategy, while it will have a significant influence on reducing the communication time, especially for complex computing tasks. We analyze the reason is that GR-ADMM avoids too long synchronization waiting time by modifying the grouping, which can be seen that communication time decreases a lot (nearly half) when testing with the kdd2010 dataset, and with the network expansion, the effect of modifying the grouping strategy decreases.

D. Ring All-Reduce in GR-ADMM

To analyze the advantage of Ring All-Reduce in GR-ADMM, we also realize the normal Point-to-Point communication in GR-ADMM. With the increase of the number of nodes in one group, the complexity of the sub-problem will increase. To reduce the impact of computation time on communication efficiency, we set the number of iterations of the sub-problem to 1. We run the system on 64 worker nodes to 100 iterations, and TABLE I shows the performance comparisons of the two communication methods on kdd2010.

In TABLE I, as the number of worker nodes in each group increasing, the communication time increases a lot when we choose Point-to-Point communication, while it only increases a little for Ring All-Reduce communication. We use D to represent the dimension of the parameters and N to denote the number of worker nodes in one group of GR-ADMM. Thus, communication traffic in one group can be represented with (11).

$$communication\ traffic = \begin{cases} D \times (|N_i| + 1) \times |N_i| & Point - to - Point \\ D \times 2 \times (|N_i| - 1)/|N_i| & Ring\ All - Reduce \end{cases} \quad (11)$$

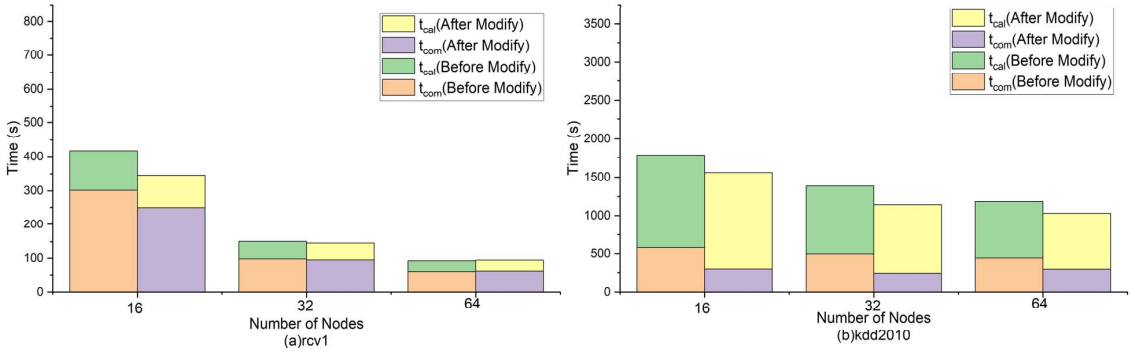

Fig. 7. Performance of Dynamic Grouping Strategy

It can be found that GR-ADMM with Ring All-Reduce communication pattern has constant communication cost and experiment results also show this advantage.

TABLE I. PERFORMANCE COMPARISONS BETWEEN POINT-TO-POINT AND RING ALL-REDUCE OF GR-ADMM ON KDD2010

Communication Method	*Number of Workers in each group*	*Total Runtime (s)*	*Communication Time (s)*	*Computation Time (s)*
Ring All-Reduce	16	218.76	47.3	171.46
Point-to-Point	16	223.05	52.7	170.35
Ring All-Reduce	32	256.34	48.61	207.73
Point-to-Point	32	317.95	110.01	207.94
Ring All-Reduce	64	273.67	50.63	223.04
Point-to-Point	64	806.8	598.53	208.27

V. CONCLUSION

Aiming at solving the problem of communication bottlenecks and stragglers in the distributed system, the paper proposes GR-ADMM, which is a decentralized algorithm based on the ADMM. With Ring All-Reduce architecture, GR-ADMM shows high communication efficiency and great extensibility. And GR-ADMM chooses the grouping strategy rather than inter-group communication, which makes the information in one node can transmit to the other in fewer iterations, resulting in faster convergence. Experiments show that our system has a faster convergence speed and less communication time than QSGD and GADMM. However, for high accuracy, the computation tasks take too much time in GR-ADMM. In future work, we will focus on the work to get a balance on the low objective error and high communication effectiveness. GR-ADMM also shows potential value in mobile communication. Another important work is to apply GR-ADMM on wireless communication and some nonconvex problems.

ACKNOWLEDGMENT

The research is support by the National Natural Foundation of China under grant NO. U1811461.

REFERENCES

[1] J. Sun, T. Chen, G. B. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," *arXiv*, no. 2, pp. 1–20, 2019.

[2] X. Lian, C. Zhang, H. Zhang, C. J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. 1, pp. 5331–5341, 2017.

[3] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "GADMM: Fast and communication efficient framework for distributed machine learning," arXiv, vol. 21, pp. 1–39, 2019.

[4] X. Jia *et al.*, "Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes," *arXiv*, 2018.

[5] M. Li et al., "Scaling distributed machine learning with the parameter server," Proc. 11th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2014, pp. 583–598, 2014.

[6] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," J. Parallel Distrib. Comput., vol. 69, no. 2, pp. 117–124, 2009, doi: 10.1016/j.jpdc.2008.09.002.

[7] Y. Peng *et al.*, "A generic communication scheduler for distributed DNN training acceleration," *SOSP 2019 - Proc. 27th ACM Symp. Oper. Syst. Princ.*, pp. 16–29, 2019, doi: 10.1145/3341301.3359642.

[8] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 7, pp. 4745–4767, 2018.

[9] L. Luo, J. Nelson, L. Ceze, A. Phanishayee, and A. Krishnamurthy, "Parameter hub: A rack-scale parameter server for distributed deep neural network training," *SoCC 2018 - Proc. 2018 ACM Symp. Cloud Comput.*, pp. 41–54, 2018.

[10] P. Sun, W. Feng, R. Han, S. Yan, and Y. Wen, "Optimizing Network Performance for Distributed DNN Training on GPU Clusters: ImageNet/AlexNet Training in 1.5 Minutes," pp. 1–13, 2019, [Online]. Available: http://arxiv.org/abs/1902.06855.

[11] J. Xie and Y. Lei, "ADMMLIB: A Library of Communication-Efficient AD-ADMM for Distributed Machine Learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11783 LNCS, pp. 322–326, doi: 10.1007/978-3-030-30709-7_27.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," Found. Trends Mach. Learn., vol. 3, no. 1, pp. 1–122, 2010, doi: 10.1561/2200000016.

[13] R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization," 31st Int. Conf. Mach. Learn. ICML 2014, vol. 5, no. 2, pp. 3689–3697, 2014.

[14] Q. Ho et al., "More effective distributed ML via a stale synchronous parallel parameter server," Adv. Neural Inf. Process. Syst., no. 1, pp. 1–9, 2013.

[15] Z. Tian, Z. Zhang, J. Yan, and J. Wang, "Distributed ADMM with Synergetic Communication and Computation," 2020 Int. Conf. Comput. Netw. Commun. ICNC 2020, pp. 956–962, 2020, doi: 10.1109/ICNC47757.2020.904965

[16] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, "D-ADMM : A Communication-Efficient Distributed Algorithm For Separable Optimization," vol. 2008, pp. 1–6, 2011.

[17] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," IEEE Trans. Signal Process., vol. 62, no. 7, pp. 1750–1761, 2014, doi: 10.1109/TSP.2014.2304432.

[18] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. NeurIPS 2018, pp. 7652–7662, 2018.

[19] Y. Lin, Y. Wang, S. Han, W. J. Dally, and H. Mao, "DEEP Gradient compression: Reducing the communication bandwidth for distributed training," *arXiv*, no. Nips, 2017.

[20] Tang, Z., Shaohuai Shi and Xiaowen Chu. "Communication-Efficient Decentralized Learning with Sparsification and Adaptive Peer Selection." *ArXiv* abs/2002.09692 (2020): n. pag.

[21] D. Alistarh and J. Z. Li, "QSGD : Communication-Efficient SGD via Gradient Quantization and Encoding," no. 1, pp. 1–12, 2017.

[22] S. Zhu, M. Hong, and B. Chen, "Quantized consensus ADMM for multi-agent distributed optimization," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2016-May, pp. 4134–4138, 2016, doi: 10.1109/ICASSP.2016.7472455.

[23] C. J. Lin, R. C. Weng, and S. S. Keerthi, "Trust region Newton methods for large-scale logistic regression," ACM Int. Conf. Proceeding Ser., vol. 227, no. May, pp. 561–568, 2007, doi: 10.1145/1273496.127356

[24] Q. Luo, J. Lin, Y. Zhuo, and X. Qian, "Hop: Heterogeneity-aware Decentralized Training," *Int. Conf. Archit. Support Program. Lang. Oper. Syst. - ASPLOS*, pp. 893–907, 2019, doi: 10.1145/3297858.3304009