



HSAC-ALADMM: an asynchronous lazy ADMM algorithm based on hierarchical sparse allreduce communication

Dongxia Wang¹ · Yongmei Lei¹ · Jinyang Xie¹ · Guozheng Wang¹

Accepted: 23 December 2020 / Published online: 14 January 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

Abstract

The distributed alternating direction method of multipliers (ADMM) is an effective algorithm for solving large-scale optimization problems. However, its high communication cost limits its scalability. An asynchronous lazy ADMM algorithm based on hierarchical sparse allreduce communication mode (HSAC-ALADMM) is proposed to reduce the communication cost of the distributed ADMM: firstly, this paper proposes a lazily aggregate parameters strategy to filter the transmission parameters of the distributed ADMM, which reduces the payload of the node per iteration. Secondly, a hierarchical sparse allreduce communication mode is tailored for sparse data to aggregate the filtered transmission parameters effectively. Finally, a Calculator-Communicator-Manager framework is designed to implement the proposed algorithm, which combines the asynchronous communication protocol and the allreduce communication mode effectively. It separates the calculation and communication by multithreading, thus improving the efficiency of system calculation and communication. Experimental results for the L1-regularized logistic regression problem with public datasets show that the HSAC-ALADMM algorithm is faster than existing asynchronous ADMM algorithms. Compared with existing sparse allreduce algorithms, the hierarchical sparse allreduce algorithm proposed in this paper makes better use of the characteristics of sparse data to reduce system time in multi-core cluster.

Keywords ADMM · Scalability · Lazily aggregate parameters · Hierarchical sparse allreduce communication · Calculator-Communicator-Manager framework

Mathematics Subject Classification 68W15 · 68W10 · 68T09

This work is supported by the National Natural Science Foundation of China under Grant No. U1811461.

✉ Yongmei Lei
lei@shu.edu.cn

Extended author information available on the last page of the article

1 Introduction

Machine learning has been widely used in risk prediction [31], disease diagnosis [24], text categorization [12] and other applications, and most machine learning programs can be viewed as optimization-centric programs. As the scale of data increases, the requirements for the computing power and memory capacity of the system become higher and higher. Therefore, allocating large-scale machine learning tasks on multiple computing nodes is becoming common to achieve faster training time [6, 19, 23]. The alternating direction method of multipliers (ADMM) is an effective method to solve optimization problems. It can transform the large global problem into several local sub-problems and can derive the solution of the global problem by coordinating the solutions of the sub-problems. Therefore, it can be naturally implemented in distributed systems. Moreover, the ADMM can quickly obtain a moderately accurate solution, so it is widely used in solving regression, classification and other supervised machine learning applications [3, 16]. When solving large-scale linear regression and linear classification problems by the ADMM, the original problem is usually transformed into a consensus or sharing problem [4, 15, 30], the sub-problems are solved by N workers in parallel and then local parameters of all workers are aggregated, finally the global solution is obtained by the aggregated parameters. As the number of workers increases, the communication becomes a bottleneck due to local parameters of all workers need to be aggregated per iteration. Many techniques are proposed to reduce the communication cost, such as quantization [9, 20], structured sparsification [27] and asynchronous communication [5, 30]. However, how to effectively combine these techniques when implementing distributed ADMM algorithms is still a challenge.

First, the communication cost of the system is determined by the characteristics of the algorithm. The total number of communication rounds is determined by the convergence speed of the algorithm, and the transmission parameters between workers, which determines the communication payload size per each link, are also determined by the algorithm. Furthermore, when implementing the algorithm in a distributed environment, bridging model, topology and communication strategy of the algorithm are also the main factors that affect the communication cost. Compared with the synchronous distributed ADMM algorithm, the asynchronous distributed ADMM algorithm [5, 30] based on the stale synchronous parallel (SSP) [10] bridging model can reduce the expensive synchronization cost. In terms of topology selection, the parameter server approach using master–slave mode and the allreduce approach with peer-to-peer mode are two main topologies to realize the distributed ADMM algorithm. The former approach provides flexible synchronization mechanisms, which can easily implement the asynchronous ADMM algorithm. However, it needs to use a special master to aggregate the local parameters of all workers, which wastes computing resources. In addition, the master must store the local parameters of all workers, which makes the master memory overhead large. In the distributed ADMM based on the allreduce approach, the communication payload among workers is balanced, and each worker only needs to store its own local parameters, but how

to effectively combine the SSP bridging model with the allreduce communication mode is still a challenge. Furthermore, how to combine the sparsity methods with the allreduce communication mode is also a concern of this paper.

In this paper, the distributed ADMM algorithm is improved according to the characteristics of L1-regularized optimization problems; the new algorithm reduced the payload size by filtering the transmission parameters without affecting the convergence. In addition, a hierarchical allreduce mode tailored for sparse data is designed by combining the characteristics of sparse data and the underlying communication mechanism. Furthermore, this paper proposes a Calculator-Communicator-Manager framework, which uses multithreading technology to effectively combine the SSP bridging model with the allreduce communication mode. This paper combines the algorithm, bridging model, topology and communication algorithm to reduce the communication cost of the distributed ADMM algorithm; the main contributions of this paper are as follows.

- (1) A distributed ADMM algorithm based on the lazily aggregate parameters strategy (LADMM) is proposed for the L1-regularized optimization problem in this paper. In the LADMM algorithm, the increment of the local variables and dual variables is used to update the global variable. In each iteration, each worker does not send all increment parameters, but only when the value of one feature of the increment parameter is greater than the threshold. And the other increment parameters are accumulated. The LADMM algorithm reduces the communication payload between workers by reducing the number of transmission parameters.
- (2) Since the number of transmission parameters may be different per iteration in the LADMM algorithm, a hierarchical sparse allreduce algorithm is tailored for aggregating the transmission parameters. The algorithm first aggregates the parameters of workers on different nodes and then aggregates the parameters of workers on the same node, due to the number of sparse parameters may increase during the aggregation process. The data type of the transmission parameters is selected automatically according to the number of transmission parameters.
- (3) The Calculator-Communicator-Manager framework is designed to implement the asynchronous LADMM algorithm based on the hierarchical sparse allreduce communication mode (HSAC-ALADMM). The framework uses multithreading to implement the asynchronous allreduce approach: each worker is assigned two threads, one is responsible for updating variables and the other is responsible for communicating with other workers. It improves the computing and communication efficiency of the system by separating the computing and communication.
- (4) The HSAC-ALADMM algorithm is used to solve the L1-regularized logistic regression problem. Experimental results show that the HSAC-ALADMM algorithm, the asynchronous ADMM algorithm based on the master-slave mode (AD-ADMM [5]) and the asynchronous ADMM algorithm based on ring-allreduce communication mode (ADMMLIB [29]) have similar convergence. The HSAC-ALADMM completes the task 5.5× faster than the AD-ADMM and 1.6× faster than the ADMMLIB. Compared with the sparse allreduce algorithm proposed

by Renggli et al. [22], the hierarchical sparse allreduce algorithm proposed in this paper has higher communication efficiency when multiple workers are assigned to each node.

The rest of the paper is organized as follows. The related work of this paper is introduced in Sect. 2, and the distributed ADMM algorithm for the global consensus problem is reviewed in Sect. 3. The HSAC-ALADMM algorithm and its parallel implementation are introduced in Sect. 4. The experimental results are presented in Sect. 5. Finally, the conclusion and future work of this paper are given in Sect. 6.

2 Related work

In practice, parallel algorithm, bridging model, topology and communication strategy are not independent. Designing communication-efficient ADMM algorithm needs to consider the factors such as bridging model and topology [11, 28, 29]. In the ADMM algorithm, the aggregation of parameters can be regarded as performing allreduce operation on local parameters. The performance of the allreduce algorithm is related to the characteristics of the data set [22] and the topology [21]. In this section, the related work of communication-efficient ADMM algorithms and allreduce algorithms for sparse data are reviewed.

2.1 Communication-efficient ADMM algorithms

Hong et al. [14] proposed an incremental ADMM algorithm to reduce the communication cost by reducing the number of transmitted parameters. Zhang et al. [30], Chang et al. [5], respectively, implemented asynchronous ADMM algorithm based on the partially asynchronous model and reduced the system time by reducing the waiting time among nodes. However, Zhang [30], Chang [5] and Hong [14] all implemented the asynchronous distributed ADMM algorithm based on the master–slave mode. In this case, the communication traffic of the master is large and the scalability of the system is poor. Wang et al. [28] proposed a hierarchical parameter server communication mode to implement asynchronous ADMM algorithm, which alleviates the problem of high communication traffic of the master to some extent. Compared with the distributed ADMM algorithm based on master–slave mode, the distributed ADMM algorithm based on peer-to-peer mode can balance the communication traffic better. Xie et al. [29] implemented an asynchronous distributed ADMM algorithm based on the hierarchical ring-based allreduce algorithm. In this algorithm, the allreduce algorithm based on the peer-to-peer mode is used to aggregate the parameters among workers. The communication cost of the algorithm is proportional to the number of features and has little correlation with the number of workers. In this paper, a lazily aggregated parameters strategy is proposed to reduce the number of transmission parameters of workers per iteration of the distributed ADMM algorithm, and a hierarchical sparse allreduce algorithm is implemented to

aggregate the transmission parameters. Compared with the algorithm proposed by Xie et al. [29], it can further reduce the system communication cost.

2.2 Allreduce algorithms for sparse data

In the distributed ADMM algorithm, the local parameters of all workers need to be aggregated to update the global variables, so the allreduce operation can be used to aggregate local parameters of all workers. Allreduce is one of the most used operations in collective communication, and the optimization of allreduce algorithm has become a research hot spot in recent years. Thakur et al. [26] implemented multiple allreduce algorithms depending on the message size. Hasanov et al. [13] implemented a topology-oblivious hierarchical allreduce algorithm. Dong et al. [8] implemented a topology-aware hierarchical allreduce algorithm for the deep neural networks. However, the above communication algorithms are all for dense data, so the communication payload size between nodes is usually proportional to the number of features. With the wide application of communication-reduction techniques such as compression and quantization [2, 18], many researchers focus on allreduce algorithms for sparse data. Zhao et al. [32] implemented sparse allreduce operation on general cluster and applied it to distributed machine learning and image processing. However, Zhao et al. assumed knowledge of the data distribution in advance. Renggli et al. [22] implemented a class of sparse communication algorithms for distributed machine learning, and they realized sparse allreduce operation by the SSAR_Split_allgather algorithm in the bandwidth dominated case. But the SSAR_Split_allgather algorithm does not consider the differences in the communication between intra-node and inter-node. Nguyen et al. [21] designed a topology-aware allreduce algorithm for sparse data, the algorithm first performs reduce-scannter operation inside nodes, then performs allreduce operation between nodes, and finally performs allgather operation inside nodes. Different from the algorithm proposed by Nguyen et al. [21], the hierarchical sparse allreduce algorithm proposed in this paper firstly aggregates parameters of workers on different nodes and then aggregates parameters of workers on the same node. As the number of sparse parameters may increase during the aggregation process, the method proposed in this paper can reduce payload size between nodes.

3 Preliminaries—the global consensus ADMM

This paper focuses on the supervised machine learning problems such as linear classification and linear regression, which can be abstracted into the optimization problem as shown in (1).

$$\min_x l(x) + r(x) \quad (1)$$

where $x \in R^M$ represents the model parameter, M is the number of features of samples, $l(x)$ is the loss function and $r(x)$ is the regularization term. The most commonly used regularization terms in machine learning are L2-regularization and

L1-regularization. The models with L2-regularization (such as ridge regression and L2-regularized logistic regression) can avoid overfitting and improve the stability. The models with L1-regularization (such as lasso and L1-regularized logistic regression) can output sparse model parameters, avoid overfitting and have strong robustness. This paper focuses on L1-regularized problems, that is, $r(x) = \gamma \|x\|_1$. When the loss function $l(x)$ is separable, the problem (1) can be converted into the global consensus problem [4] as shown in (2).

$$\begin{aligned} \min \quad & \sum_{i=0}^{N-1} l_i(x_i) + r(z) \\ \text{s.t.} \quad & x_i = z, \quad i = 0, 1, \dots, N-1 \end{aligned} \quad (2)$$

where $x_i \in R^M$ represents the local variable, $z \in R^M$ represents the global variable. The augmented lagrangian (\mathcal{L}_ρ) for (2) can be deduced as shown in (3). Then, the update rules of the ADMM for problem (2), which are derived by minimizing $\mathcal{L}_\rho(\{x_i\}, z)$, are shown in (4)–(6).

$$\mathcal{L}_\rho(x_i, z, y_i) = \sum_{i=0}^{N-1} (l_i(x_i) + (\rho/2) \|x_i + y_i/\rho - z\|_2^2) + \gamma \|z\|_1 \quad (3)$$

$$y_i^{k+1} := y_i^k + \rho(x_i^k - z^k) \quad (4)$$

$$x_i^{k+1} := \underset{x_i}{\operatorname{argmin}} (l_i(x_i) + (\rho/2) \|x_i + y_i^{k+1}/\rho - z^k\|_2^2) \quad (5)$$

$$z^{k+1} := S_{\gamma/\rho N} \left(\frac{1}{N} \sum_{i=0}^{N-1} (x_i^{k+1} + y_i^{k+1}/\rho) \right) \quad (6)$$

where $y_i \in R^M$ is the dual variable, ρ is the penalty parameter, S is the soft threshold and its definition is shown in (7).

$$S_{\gamma/\rho N}(a) = \begin{cases} a - \gamma/\rho N, & a > \gamma/\rho N, \\ 0, & |a| \leq \gamma/\rho N, \\ a + \gamma/\rho N, & a < -\gamma/\rho N. \end{cases} \quad (7)$$

According to (4)–(6), the local variables x_i and the dual variables y_i can be solved in parallel. A simple popular solution is to update x_i and y_i in parallel by N workers. Then, the local variables and dual variables of all workers are aggregated. Lastly, the global variable z is updated with the aggregated parameters. Since x_i and y_i of all workers need to be aggregated before updating the global variables, the system time per iteration is determined by the slowest worker. The asynchronous distributed ADMM algorithm (AD-ADMM) based on partially asynchronous model can reduce the synchronization overhead. In the AD-ADMM, a master is responsible

for updating the global variable and N workers are responsible for updating the local variable and dual variable, respectively. Two key conditions, partial barriers and bounded delay, are proposed to ensure the convergence of the algorithm. That means in each iteration the master only needs to receive variables from a partial set of workers to update the global variable, and the master should wait for the workers who have been inactive for τ iterations. Since the master needs to communicate with all workers, its communication traffic is large, and it needs to save the local variables and dual variables received from all workers, which will cost a large amount of memory overhead. In order to better balance the communication traffic and memory overhead, this paper will implement the asynchronous distributed ADMM algorithm based on peer-to-peer mode and reduce the communication cost from two aspects, designing communication-efficient ADMM algorithm and communication strategy.

4 The asynchronous lazy ADMM algorithm and its parallel implementation

This paper first proposes a lazily aggregate parameters strategy to reduce the number of the transmission parameters by analyzing the characteristics of the ADMM algorithm and then implements a hierarchical sparse allreduce algorithm, based on which the asynchronous lazy ADMM algorithm is implemented. For simplicity, the notations used in the rest of this paper are shown in Table 1.

4.1 The distributed ADMM algorithm based on lazily aggregate parameters

In the global consensus ADMM algorithm, it can be seen from (6) that the global variable z is determined by the sum of the $(x_i + y_i/\rho)$ of all workers, so we take $(x_i + y_i/\rho)$ as a whole and define it as ω_i (the definition is shown in (8)). Then, the parameter ω_i , instead of x_i and y_i , is used to update z . In this case, (6) can be modified to (9).

$$\omega_i^{k+1} := x_i^{k+1} + y_i^{k+1}/\rho \quad (8)$$

$$z^{k+1} := S_{\gamma/\rho N} \left(\frac{1}{N} \sum_{i=0}^{N-1} \omega_i^{k+1} \right) \quad (9)$$

To reconstruct the parallel algorithm, (8) and (9) are modified as shown in (10)–(12).

$$\Delta \omega_i^{k+1} := \begin{cases} x_i^{k+1} + y_i^{k+1}/\rho, & k = 0, \\ (x_i^{k+1} + y_i^{k+1}/\rho) - (x_i^k + y_i^k/\rho), & k \geq 1 \end{cases} \quad (10)$$

$$\omega_i^{k+1} := \omega_i^k + \sum_{i=0}^{N-1} \Delta \omega_i^{k+1} \quad (11)$$

Table 1 Notations

Variable	Description
N	Number of workers
G	Number of worker nodes
P	Number of processes on each node
M	Number of features of samples
N_i	The i th worker
m	Number of transmitted parameters
$\Delta\omega_i$	Increment parameters of N_i
$\Delta\omega_{ij}$	The j th feature of $\Delta\omega_i$
ψ	The collection of transmission parameters of all workers
ψ_i	Transmission parameters of N_i
$\psi(\text{key})$	The collection of all the “key” of ψ
$\psi(\text{value})$	The collection of all the “value” of ψ
Th	The threshold to filter increment parameters
B_{ij}	The j th block of ψ_i
B_i	The i th block of ψ
F_v	The number of bytes to store the value of parameter
F_k	The number of bytes to store the index of parameter
β_{inter}	The data transfer rate between process on different nodes
β_{intra}	The data transfer rate between process on the same node
τ	The maximum delay period in the asynchronous ADMM
κ	The minimum barrier in the asynchronous ADMM

$$z^{k+1} := S_{\gamma/\rho N} \left(\frac{1}{N} \omega^{k+1} \right) \quad (12)$$

Obviously, (10)–(12) are equivalent to (8) and (9). It can be deduced from (10)–(12) that the communication traffic is proportional to the number of features of increment parameters in the global consensus ADMM algorithm. However, there are a large number of values of 0 in the increment parameters of each worker for large-scale and high-dimension sparse data sets. These parameters have no effect on the update of z , but occupy a large amount of transmission bandwidth. Therefore, they can be abandoned when transmitting parameters, so as to reduce the communication cost. Furthermore, it can be deduced from (7) and (11) that when $|\omega_j|$ is less than $\gamma/\rho N$, the j th feature of the global variable z_j is set to 0, that is to say, when $\Delta\omega_{ij}$ is very small, it has little influence on the global variable. Based on the above characteristics, this paper proposes a distributed ADMM algorithm based on lazily aggregate parameters (LADMM): In each iteration, only the increment parameters larger than a predetermined threshold are transmitted; otherwise, the increment parameters are accumulated. When $\Delta\omega_{ij}$ is larger than the threshold, it is added to the output list ψ_i in the form of key-value pairs, then the update rules of $\Delta\omega_{ij}$ and ψ_i are shown in (13)–(15).

$$\Delta\omega_{ij}^{k+1} := \begin{cases} x_{ij}^{k+1} + y_{ij}^{k+1}/\rho, k = 0, \\ \Delta\omega_{ij}^k + (x_{ij}^{k+1} + y_{ij}^{k+1}/\rho) - (x_{ij}^k + y_{ij}^k/\rho), k \geq 1 \end{cases} \quad (13)$$

$$\psi_i^{k+1} := \{ \langle j, \Delta\omega_{ij}^{k+1} \rangle \mid |\Delta\omega_{ij}^{k+1}| > Th \} \quad (14)$$

$$\Delta\omega_{ij}^{k+1} := 0, \text{ if } |\Delta\omega_{ij}^{k+1}| > Th, j = 1, 2, \dots, M \quad (15)$$

In the LADMM algorithm, the transmission parameters ψ_i of all workers are aggregated to update z . Since ψ_i contains the “key-value” information, when the worker received parameters from other workers, it is necessary to extract the “value” from the parameters, and finally update z with the aggregated values. The update rules are shown in (16)–(18).

$$\psi = \cup_{i=0}^{N-1} \psi_i \quad (16)$$

$$\omega_j^{k+1} = \begin{cases} \omega_j^k + \psi_j(\text{value}), \forall j \in \psi(\text{key}), \\ \omega_j^k, \forall j \notin \psi(\text{key}) \end{cases} \quad (17)$$

$$z_j^{k+1} := S_{\gamma/\rho N} \left(\frac{1}{N} \omega_j^{k+1} \right), j = 1, 2, \dots, M \quad (18)$$

4.2 Data representation and messages aggregation

The variables are represented in different data types and the messages are aggregated by different rules according to the data type in this paper.

The Representation of Different Data In this paper, the variables are classified into dense data and sparse data, which are represented by dense vector and sparse vector, respectively. If the number of elements of the variable is variable, such as ψ_i , we call this type of variable sparse data. And the sparse data is stored in two associative arrays: a “key” array ($\psi_i(\text{key})$) and a “value” array ($\psi_i(\text{value})$). If the j th dimension is in the “key” array of ψ_i , that is $j \in \psi_i(\text{key})$, then the j th dimension is said to be related to ψ_i . Other variables, such as x_i , y_i , ω_i , $\Delta\omega_i$ and z , have the same number of features as the sample, these variables are called dense data and only need to be stored in a “value” array.

The Selection of Data Type In the LADMM algorithm, the increment parameters and corresponding indexes need to be transmitted when transmitting sparse data, that is to say, the number of bytes occupied by m sparse parameters is $m(F_v + F_k)$, whereas the number of bytes occupied by dense parameters is MF_v . If $m(F_v + F_k) \geq MF_v$, the dense parameters is selected to transmit; otherwise, the sparse parameters is selected to transmit.

Messages Aggregation There are two different situations when aggregating messages: If the parameters saved and received by the worker are both dense data, then

the model parameters are directly accumulated. Otherwise, it is necessary to sum two sparse vectors. Suppose ψ_i and ψ_l represent the transmission parameters of N_i and N_l respectively. There are two different situations when merge ψ_i and ψ_l . If the “key” in $\psi_l(\text{key})$ is related to ψ_i , the corresponding “values” are accumulated. Otherwise, the index-value pairs in the ψ_l are added to the ψ_i directly. The merging rule of ψ_i and ψ_l is shown in (19).

$$\begin{aligned} \psi_i \cup \psi_l = \{ < j, \psi_{i,j}(\text{value}) > | \psi_{i,j}(\text{value}) = \psi_{i,j}(\text{value}) + \psi_{l,j}(\text{value}), \\ \forall j \in \psi_i(\text{key}) \cap \psi_l(\text{key}) \} \cup \{ < j, \psi_{i,j}(\text{value}) > | \forall j \in \psi_l(\text{key}) \cap \psi_i^C(\text{key}) \} \end{aligned} \quad (19)$$

where $\psi_{i,j}(\text{value})$ represents the value of the element corresponding to the element whose key value is j in ψ_i , that is $\Delta\omega_{ij}$. And $\psi_i^C(\text{key})$ represents the complement of $\psi_i(\text{key})$. When summing sparse data and dense data, dense data can be treated as special sparse data with all indexes.

4.3 Hierarchical sparse allreduce communication mode

Hierarchical hardware design has become common in high-performance systems, in which shared memory nodes with several multi-core CPUs are connected via a network infrastructure. This paper designs the communication algorithm according to the communication mechanism of the multi-core cluster and the characteristics of the transmission parameters in the LADMM algorithm. The algorithm is implemented by the message passing interface (MPI) programming model, and the communication cost is evaluated by the latency–bandwidth communication model [22], which defines the time required for a communication between two processes as $T(C) = \alpha + C\beta$. Here α is the latency time per message, β is the transfer time per byte, and C is the total number of bytes transmitted. In this paper, a working process is treated as a worker.

(1) Hierarchical Allreduce Algorithm for Sparse Data

Since this paper mainly focuses on the problem of high communication cost of high-dimension data sets, this section mainly studies allreduce algorithms in the bandwidth-dominated case [22]. In this case, Renggli et al. [22] proposed a SSAR_Split_allgather algorithm to implement the sparse allreduce operation, and the algorithm includes two steps: the first step is the split phase, in which the transmission parameters of each process are divided into N partitions on average according to the features (e.g. ψ_i is divided into $\{B_{i0}, B_{i1}, \dots, B_{iN-1}\}$) and the corresponding partitions are aggregated by each process, respectively. The second step is the sparse allgather phase, in which each process broadcast the parameters aggregated in the first step to other processes. The sparse Recursive-doubling algorithm [22] is used to implement the allgather operation in the second step. Figure 1 shows the process of aggregating the transmission parameter by the SSAR_Split_allgather algorithm when N is 4. In the first step, the communication cost will reach the lower bound if all the transmission parameters of each worker are distributed in the partition where the worker is responsible for aggregation. On the contrary, the communication cost will reach the upper bound if all the transmission parameters of each worker are not distributed in the corresponding

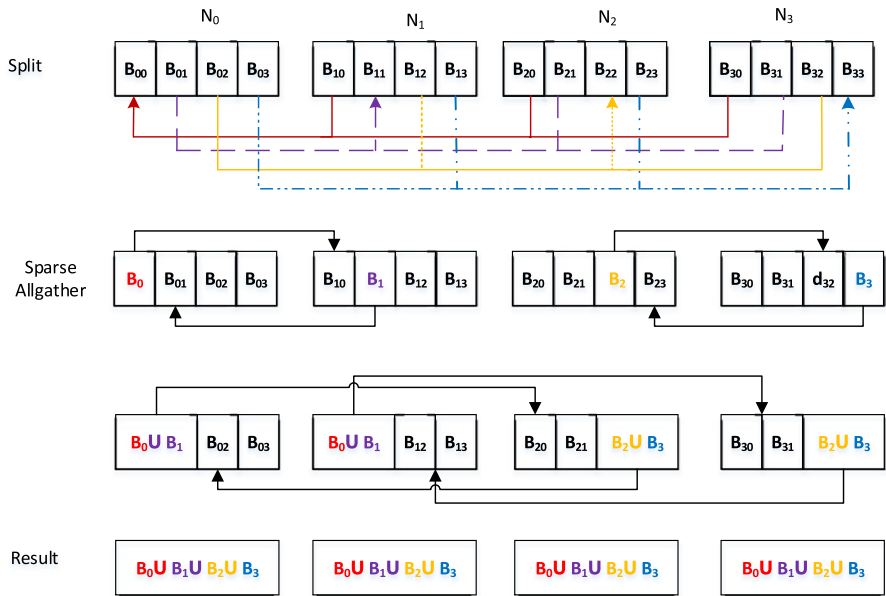


Fig. 1 The schematic diagram of the communication process of workers by the SSAR_Split_allgather algorithm: $N = 4$

partition. Therefore, the communication time in the split phase ($T_{ssa-split}$) is as shown in (20). In the second step, the communication cost will reach the lower bound if the transmission parameters of each worker are evenly distributed and completely overlap, and it will reach the upper bound if the transmission parameters are concentrated in a certain block and do not overlap at all. Therefore, the communication time in the sparse allgather phase (T_{ssa-al}) is as shown in (21).

$$\begin{aligned} (N-1)\alpha &\leq T_{ssa-split} \\ &\leq (N-1)\alpha + (F_k + F_v)\beta_{intra}Pm/N + (F_k + F_v)\beta_{inter}(N-P)m/N \end{aligned} \quad (20)$$

$$\begin{aligned} \alpha \log_2 N + ((G-1)m\beta_{intra}/N + (P-1)Gm\beta_{inter}/N)(F_k + F_v) \\ \leq T_{ssa-al} \leq \alpha \log_2 N + ((G-1)m\beta_{intra} + (P-1)Gm\beta_{inter})(F_k + F_v) \end{aligned} \quad (21)$$

As can be seen from Fig. 1 that, in the second phase of the SSAR_Split_allgather algorithm, since the indexes of the data blocks sent and received between workers are completely non-overlapping, the number of transmission parameters is constantly increasing in the communication process. In a multi-core cluster, the communication speed of inter-process on different nodes is often slower than that of inter-process on the same node. Based on this characteristic, a hierarchical sparse allreduce (HSA) algorithm is designed: Similar to the SSAR_Split_allgather algorithm, the HSA algorithm includes two phases, split and allgather. The

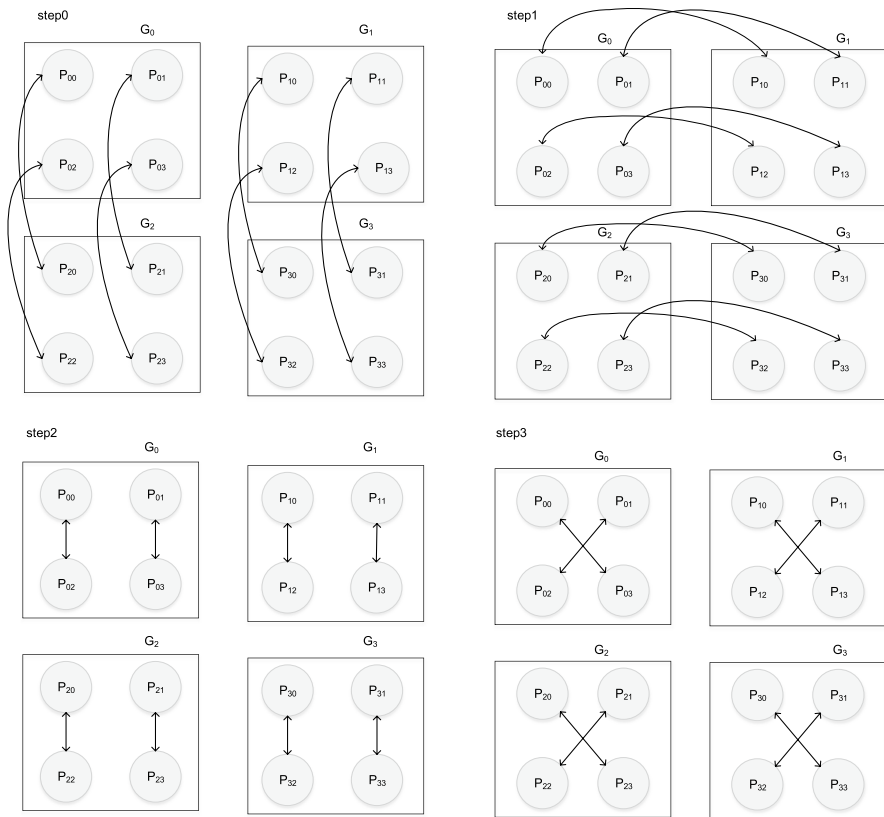


Fig. 2 The schematic diagram of the communication process of workers in the allgather phase of the HSA algorithm: G and P are set to 4, P_{ij} represents the j th process in the G_i

split phase is the same as the first phase of the SSAR_Split_allgather algorithm. First, the transmission parameter ψ is split into N partitions according to the key value, and then, N_i is responsible for aggregating the transmission parameters of the i th partition of all workers. And the communication time of HSA in the split phase is equal to $T_{ssa-split}$. In the allgather phase, we first aggregate the parameters of workers on different nodes and then aggregate the parameters of workers on the same node so as to reduce the communication traffic between nodes. Figure 2 shows the communication process of workers in the allgather phase of the HSA algorithm when both G and P are set to 4, and the communication time in this phase (T_{hsa-al}) is shown in (22). If $\beta_{inter} < \beta_{intra}$ and $P \geq 2$, it can be deduced from (21) and (22) that $T_{hsa-al} < T_{ssa-al}$.

$$\begin{aligned} & \alpha \log_2 N + ((G-1)m\beta_{inter}/N + (P-1)Gm\beta_{intra}/N)(F_k + F_v) \\ & \leq T_{hsa-al} \leq \alpha \log_2 N + ((G-1)m\beta_{inter} + (P-1)Gm\beta_{intra})(F_k + F_v) \end{aligned} \quad (22)$$

(2) Automatic Selection of Transmission Parameters and Transmission of Message

In the LADMM algorithm, the data type of transmission parameter is selected according to the number of filtered transmission parameters. Since the number of transmission parameters changes dynamically during the communication process, the sender needs to determine whether to transmit dense data or sparse data each time before sending data. In order to select the transmission data automatically, each worker saves both dense and sparse transmission parameters. When $m(F_v + F_k) \geq MF_v$, the dense parameters are transmitted; otherwise, the sparse parameters are transmitted. The receiver also judges the data type of received data according to the number of received parameters.

Non-blocking communication is used between processes, the sender sends messages by `MPI_Isend`, and the receiver receives message by `MPI_Irecv`. The datatype of message is `MPI_CHAR`. For the sparse data, the parameters are stored in two associative arrays in sequence, the elements of the “key” array are spliced to the tail of the “value” array, and all the elements are copied to the send buffer together. For the dense data, all the elements of transmission parameters are copied to the send buffer. The receiver obtains the length of the message by `MPI_Get_count` and then judges whether the received data is sparse or dense.

4.4 Implementation of asynchronous LADMM algorithm based on hierarchical sparse allreduce communication mode

In Sect. 4.3, the allreduce communication mode requires all workers to communicate synchronously. In this section, a Calculator-Communicator-Manager (CCM) framework is designed based on the hierarchical sparse allreduce communication mode to implement the asynchronous LADMM algorithm. The CCM framework is shown in Fig. 3: The framework includes a Manager and N workers. Each worker is allocated two threads, a Calculator and a Communicator. The Calculator is responsible for updating the variables x_i , y_i , $\Delta\omega_i$, ω and z . The Communicator is responsible for communicating with other workers and aggregating the transmission parameters of all workers. And the Manager is responsible for coordinating the communication process among Communicators.

The asynchronous LADMM algorithm based on hierarchical sparse allreduce communication mode (HSAC-ALADMM) includes five steps: (1) Each Calculator updates x_i , y_i and $\Delta\omega_i$ in parallel, notifies the Communicator after the updating is completed, then the Calculator is blocked until the Communicator wakes it up. (2) The Communicator will request allreduce operation to the Manager after it receives the notification from the Calculator. (3) After receiving request from a Communicator, the Manager judges whether the two conditions proposed in Sect. 3, namely “partial barriers” and “bounded delay”, are satisfied. If they are met, the Manager will notify all Communicators to perform allreduce operation and send A_k to all Communicators. Otherwise, it will wait to receive request from other Communicators until the two conditions are met. (4) The Communicator will check whether itself is in A_k after it receives the message from the Manager.

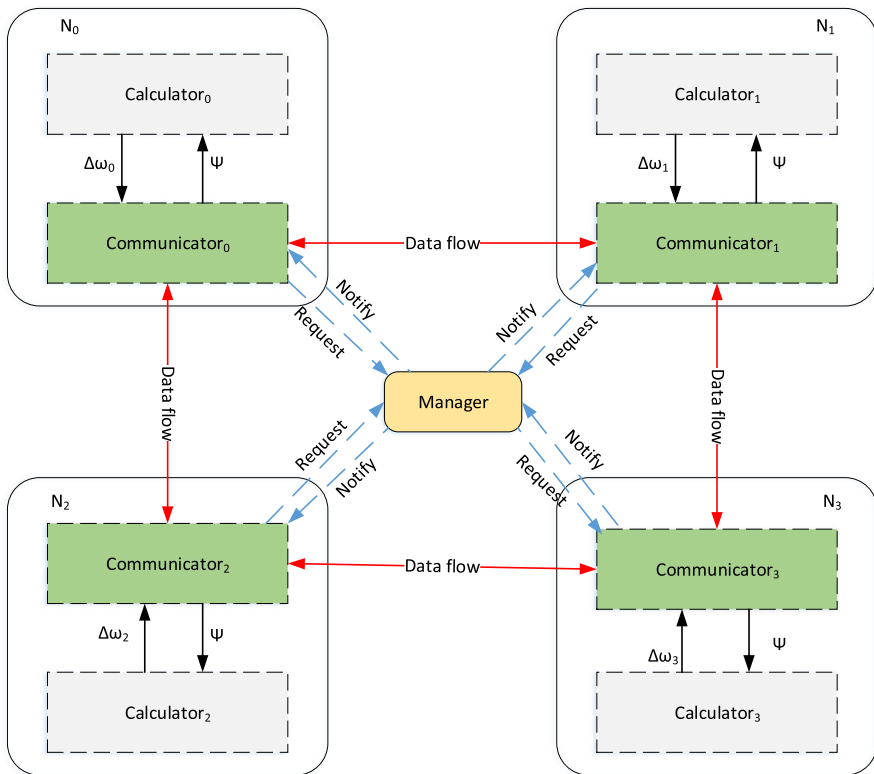


Fig. 3 The Calculator-Communicator-Manager framework

If not, ψ_i will be set to empty, otherwise, the Communicator will get the latest $\Delta\omega_i$ from the Calculator to update ψ_i . Then, all the Communicators allreduce ψ_i by the HSA algorithm. Finally, the Communicators in A_k wake up their Calculators and send ψ to their Calculators. (5) The Calculator will update ω and z when it receives ψ from the Communicator. These five steps will loop until the stop condition is satisfied. The whole algorithm is described in Algorithm 1-3 and the whole source code in this paper is available at <https://github.com/wangdongxia/ALADMM.git>.

Figure 4 describes the flowchart of the HSAC-ALADMM algorithm at one iteration. Figure 4 and Algorithm 2 show that, if the worker is not in the list that requests allreduce operation, ψ_i of this worker will be set to empty before do allreduce operation. That is to say, we just use the increment parameters of the workers that requested the allreduce operation to update the global variable.

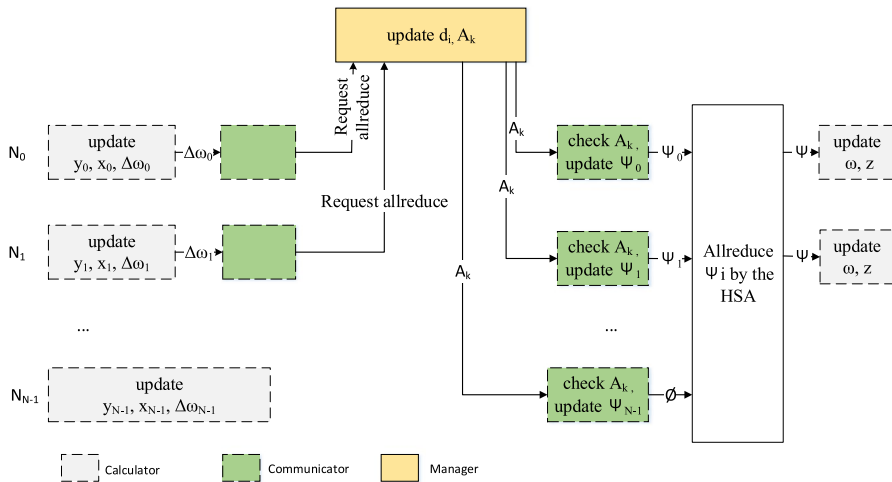


Fig. 4 The flowchart of HSAC-ALADMM algorithm at one iteration

Algorithm 1: The HSAC-ALADMM algorithm: the process of the Manager

Initialize: $k = 0, 0 < \kappa \leq N, \tau, d_i = 0 (i = 0, 1, \dots, N - 1)$.
repeat
 wait until receiving allreduce request from Communicators $i, i \in A_k$ such that $|A_k| \geq \kappa$ and $d_i < \tau$ for all $i \in \{0, 1, \dots, N - 1\}$.
 notify all Communicators to do allreduce operation and send A_k to all Communicators.
 wait until receiving report from all Communicators.
 for $\forall i \in A_k$ **do**
 $d_i \leftarrow 0$.
 for $\forall i \in A_k^c$ **do**
 $d_i \leftarrow d_i + 1$.
 $k \leftarrow k + 1$.
until *terminal*;

Algorithm 2: The HSAC-ALADMM algorithm: the process of the Communicator of N_i

Initialize: $c_i = 0, \psi_i^0 = \phi$.
repeat
 if receive notice and $\Delta\omega_i$ from Calculator, **then**
 send allreduce request to Manager.
 if receive allreduce command and A_k from Manager **then**
 $c_i \leftarrow c_i + 1$.
 if A_k contains N_i **then**
 update $\psi_i^{c_i}$ and $\Delta\omega_i$ using (14),(15).
 else
 $\psi_i^{c_i} = \phi$.
 allreduce $\psi_i^{c_i}$ by the hierarchical sparse allreduce algorithm.
 send $\psi_i^{c_i}$ to Calculator.
 send report to Manager.
until *terminal*;

Algorithm 3: The HSAC-ALADMM algorithm: the process of the Calculator of N_i

Initialize: $x_i^0, y_i^0, \omega_i^0, \Delta\omega_i^0, z^0$ and set $k_i = 0$.
repeat
 update $y_i^{k_i}$ and $x_i^{k_i}$ using (4) and (5).
 update $\Delta\omega_i^{k_i}$ using (13).
 send completion notice and $\Delta\omega_i^{k_i}$ to the Communicator.
 wait until receiving ψ_i from Communicator.
 update $\omega_i^{k_i}$ and z^{k_i} using (17) and (18).
 $k_i \leftarrow k_i + 1$.
until the stop condition is satisfied;

5 Experiments and discussion

Logistic regression is the most commonly used statistical model for predicting binary classification problems [7]. The L1-regularized logistic regression has received extensive attention in machine learning [12], medicine [1, 25] and other fields, due to the model with L1 regularization can avoid overfitting. In this section, the HSAC-ALADMM algorithm is used to solve the L1-regularized logistic regression problem, which can be described as shown in (23).

$$\min_x f(x) = \min_x \left(\frac{1}{n} \sum_{i=1}^n \log(1 + e^{-b_i x^T D_i}) + \gamma \|x\|_1 \right) \quad (23)$$

where $x \in R^M$ is the model parameter, n is the number of samples, $D_i \in R^M$ is the i th sample, $b_i \in \{-1, 1\}$ represents the label of the i th sample and γ is the scalar regularization parameter.

We test convergence, system time and scalability of the HSAC-ALADMM algorithm in this section and compare it with the following approaches.

AD-ADMM is an asynchronous ADMM algorithm based on master–slave mode proposed by Chang et al. [5].

ADMMLIB is an asynchronous ADMM algorithm based on ring-allreduce communication mode [29], but the transmission parameters are dense data.

SSAC-ALADMM is another implementation form of the asynchronous LADMM algorithm, the only difference between the SSAC-ALADMM and the HSAC-ALADMM is the communication algorithm adopted by the Communicators. The former adopts the SSAR_Split_allgather [22] communication algorithm, whereas the latter adopts the HSA communication algorithm.

The experimental platform used in this paper is the high-performance cluster “Ziqiang 4000” of Shanghai University. Five computing nodes interconnected with a Gigabit Ethernet are used in the experiments. Each node has two Intel E5-2690 CPU(2.9 GHz/8-core) processors and 64 GB memory. All nodes run Linux (Ubuntu) with the 2.6.32 kernel. The MPICH 3.1.4 is installed on each node. The above algorithms are all implemented by C++11 and the sub-problem is solved by the Trust

Table 2 A summary of datasets

Dataset	Number of training samples	Number of testing samples	Number of features
KDDDB	19,264,097	748,401	1,163,024
URL	2,000,000	396,130	3,231,961

Region Newton method (TRON) [17]. The datasets used in this paper are the public dataset KDDDB¹ and URL,² which are described in Table 2.

5.1 Convergence

The convergence of the AD-ADMM, the ADMMLIB, the SSAC-ALADMM and the HSAC-ALADMM are compared by the relative error of the loss function (f_{err}). The definition of f_{err} is shown in (24). Five nodes are used in the experiments, one of which is used as the master of the AD-ADMM and the ADMMLIB or the Manager of the SSAC-ALADMM and the HSAC-ALADMM, and the other nodes are worker nodes, each worker node is assigned 16 processes, that is, there are 64 workers.

$$f_{err} = |f - f^*|/f^* \quad (24)$$

where f represents the value of the loss function in the current state, and f^* represents the minimum value of the loss function obtained from all algorithms. Figure 5 shows the convergence of these four algorithms when κ is 16.

Figure 5 shows that these four algorithms have similar convergence, because this paper filters the transmission parameters according to the characteristics of the distributed ADMM algorithm, the filtering strategy has little effect on the convergence of the algorithm. In Fig. 5, the convergence speed of the AD-ADMM is much slower than that of the other three algorithms, the main reason is that the communication traffic of the master in the AD-ADMM is large, which makes the communication time large. Figure 5 also shows that the convergence speed of the SSAC-ALADMM and the HSAC-ALADMM is faster than that of the ADMMLIB, because the asynchronous LADMM algorithm uses a lazily aggregate parameters strategy to reduce the number of transmission parameters. The reason why the convergence speed of the HSAC-ALADMM is faster than that of the SSAC-ALADMM is that the HSAC-ALADMM uses a sparse hierarchical communication mode to reduce the communication traffic of inter-nodes. We will further test the system time of these four algorithms for different κ .

¹ <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#kdd2010> raw version (bridge to algebra).

² <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#url>.

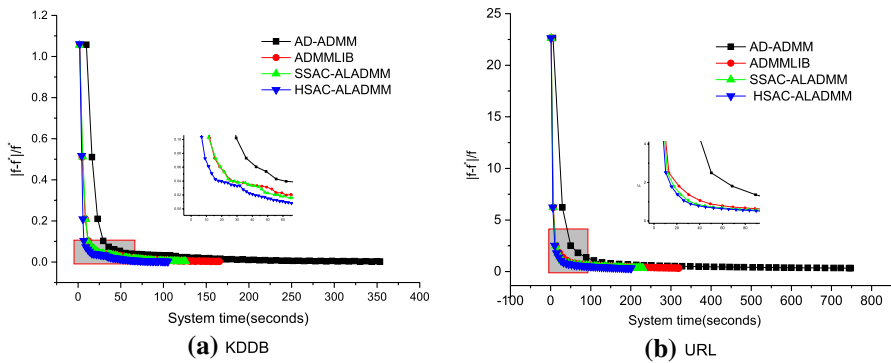


Fig. 5 Convergence of the relative error with system time

5.2 System time

The system time is defined as the sum of the computing time and the waiting time in this paper. The computing time includes the updating time of variables x , y and $\Delta\omega$. The average computing time of each worker is used as the computing time in the experiments. The waiting time includes the communication time among workers and the updating time of z and ω . Since z is updated by the master in the AD-ADMM, while z is updated by each worker in parallel in other three algorithms, the updating time of z and ω is included in the waiting time. As in Sect. 5.1, N is set to 64 in the experiments. κ is set to 64, 32 and 16, respectively, and the system time of various algorithms is shown in Table 3.

It can be seen from Table 3 that the difference of the computing time of the four algorithms is very small. However, the computing time of the AD-ADMM algorithm is slightly less than that of other algorithms. The main reason is that each worker has two threads in other three algorithms, the synchronization overhead between threads increases the computing time. On the other hand, the waiting time of the AD-ADMM algorithm is much larger than that of the other algorithms, the main reason is the ADMMLIB, the SSAC-ALADMM and the HSAC-ALADMM implement the asynchronous ADMM algorithm based on the allreduce communication mode, which balances the communication traffic between nodes.

Table 3 shows that when κ is 64, for the URL dataset, the system time of the AD-ADMM and the ADMMLIB are 1558.5 s and 476.53 s, whereas the system time of the HSAC-ALADMM is 283.58 s. This implies that the HSAC-ALADMM completes the task 5.5 \times faster than the AD-ADMM and 1.6 \times faster than the ADMMLIB. For the KDD dataset, the HSAC-ALADMM completes the task 3.4 \times faster than the AD-ADMM and 1.2 \times faster than the ADMMLIB. When κ is 16, for the URL dataset, the HSAC-ALADMM completes the task 3.7 \times faster than the AD-ADMM and 1.6 \times faster than the ADMMLIB. For the KDD dataset, the HSAC-ALADMM completes the task 3.3 \times faster than the AD-ADMM and 1.5 \times faster than the ADMMLIB. The main reason why the HSAC-ALADMM is faster than the ADMMLIB is that the HSAC-ALADMM algorithm reduces the payload between

Table 3 The system time of various algorithms for different κ

Dataset	κ	Algorithm	System time (s)	Computing time (s)	Waiting time (s)
kddb	64	AD-ADMM	697.52	65.14	632.38
		ADMMLIB	250.2	68.1	182.1
		SSAC-ALADMM	225.46	68.77	156.69
		HSAC-ALADMM	202.54	68.85	133.69
	32	AD-ADMM	419.39	54.23	365.16
		ADMMLIB	159.23	56.23	103
		SSAC-ALADMM	106.86	56.99	49.87
		HSAC-ALADMM	97.26	56.08	41.18
	16	AD-ADMM	353.45	47.5	305.95
		ADMMLIB	165.48	50.57	114.91
		SSAC-ALADMM	125.91	50.51	75.4
		HSAC-ALADMM	105.33	50.87	54.46
url	64	AD-ADMM	1558.5	122.85	1435.65
		ADMMLIB	476.53	126.69	349.84
		SSAC-ALADMM	351.87	127.78	224.09
		HSAC-ALADMM	283.58	127.26	156.32
	32	AD-ADMM	1126.16	110.52	1015.64
		ADMMLIB	488	118.41	369.59
		SSAC-ALADMM	328.4	119.71	208.69
		HSAC-ALADMM	265.76	119.86	145.9
	16	AD-ADMM	748.05	104.2	643.8
		ADMMLIB	319.52	110.72	208.8
		SSAC-ALADMM	230.47	110.85	119.62
		HSAC-ALADMM	200.85	111.72	89.13

workers by filtering the transmission parameters. Table 3 also shows that when κ is 16, for the URL dataset, the waiting time of the HSAC-ALADMM is 89.13 s and that of the SSAC-ALADMM is 119.62 s. For the KDDb dataset, the waiting time of the HSAC-ALADMM is 54.46 s and that of the SSAC-ALADMM is 75.4 s. These results also show that as the dimension of the data set increases, the advantages of the HSAC-ALADMM over the other algorithms are more obvious. The algorithm proposed in this paper has more advantages in solving large-scale classifications.

5.3 Scalability

In this section, the system time and accuracy of the four algorithms are tested. The number of workers(N) is set to 4, 8, 16, 32 and 64, respectively, and κ is set to a quarter of N . Figures 6 and 7, respectively, show the system time and the accuracy of these four algorithms. And the accuracy is defined as the ratio of the number of samples correctly classified by the algorithm to the total number of samples for a given test data set.

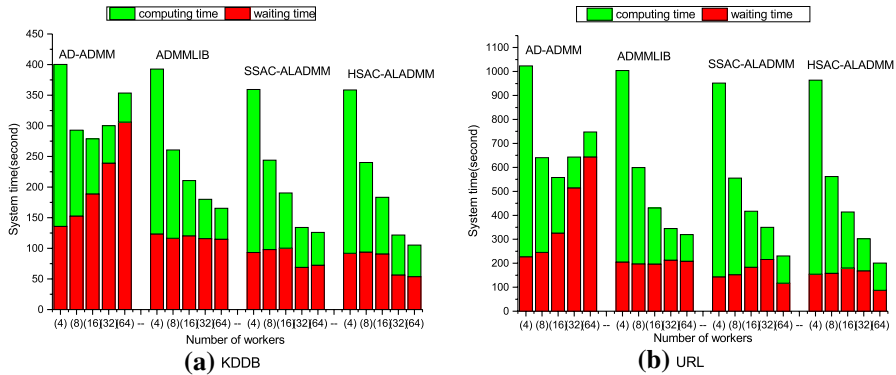


Fig. 6 System time of various algorithms for different number of workers

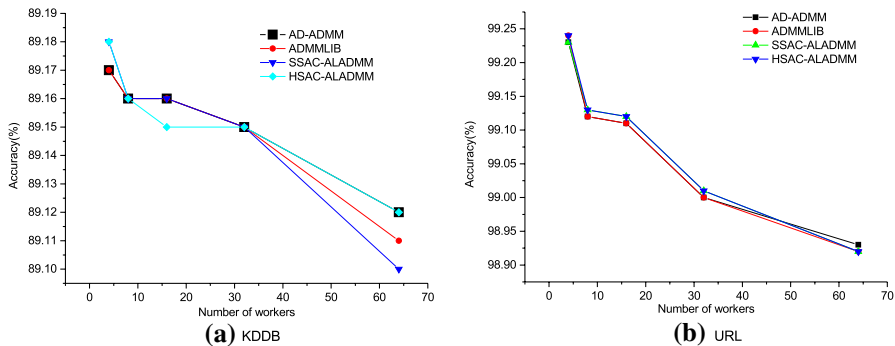


Fig. 7 The accuracy of various algorithms for different number of workers

Firstly, Fig. 6 shows that, as N increases, the waiting time of the AD-ADMM increases, while that of the ADMMLIB is basically unchanged. Because the communication cost of the master in the AD-ADMM algorithm increases with the increase of N , the communication cost of workers in the ADMMLIB algorithm is not affected by N . The communication cost of the SSAC-ALADMM algorithm and the HSAC-ALADMM algorithm is mainly affected by the distribution of data sets. As N increases, the number of samples processed by each worker decreases, and the data sparsity may increase, so the communication cost of the SSAC-ALADMM and the HSAC-ALADMM may decrease. Secondly, Fig. 6 shows that when N is set to 4, 8 and 16, there is little difference in the waiting time between the SSAC-ALADMM and the HSAC-ALADMM. When N is 32 and 64, the advantages of the HSAC-ALADMM algorithm are revealed. Compared with the SSAC-ALADMM algorithm, the HSAC-ALADMM algorithm only uses the hierarchical communication structure to reduce communication cost. When the number of processes allocated by each node is small, the difference between these two algorithms is subtle. However, as the number of processes allocated by each node and the number of features of sample increase, the advantages of the HSAC-ALADMM algorithm will

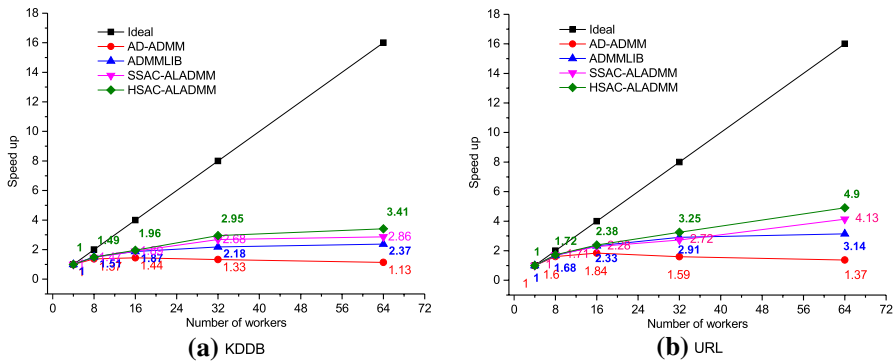


Fig. 8 The speed up of various algorithms for different number of workers

become more and more obvious. Lastly, Fig. 6 also shows that the waiting time and computing time of the HSAC-ALADMM algorithm decrease with the increase of N , indicating that the HSAC-ALADMM algorithm is very suitable for solving large-scale problems.

As can be seen from Fig. 7, the accuracy of these four distributed ADMM algorithms decreases as N increases, because the convergence speed of distributed ADMM algorithm will slow down as N increases. And the accuracy of these four algorithms differs very little (the maximum difference is 0.04%) with the same parameters, because the convergence of these four algorithms is similar.

In order to further analyze the scalability of these four algorithms, we take the system time when N is 4 as the benchmark and, respectively, test the speed up when N is 8, 16, 32 and 64. The results are shown in Fig. 8. As can be seen from Fig. 8 that, when N is 64 and the data set is URL, the speed up of the AD-ADMM is 1.37, that of the ADMLLIB is 3.14, that of the SSAC-ALADMM is 4.13, and that of the HSAC-ALADMM is 4.9. The most fundamental reason is that the HSAC-ALADMM algorithm reduces the number of transmission parameters between workers by filtering the parameters and uses the hierarchical sparse communication algorithm to improve the communication efficiency among nodes, thus reducing the communication cost of the system. Similar to Fig. 6, when N is small, the speed up of these four algorithms are not much different, but as N increases, the advantages of the HSAC-ADMM algorithm are more and more obvious.

6 Conclusion and future work

In order to reduce the communication cost and improve the scalability of the distributed ADMM algorithm, this paper implements an asynchronous lazy ADMM algorithm based on the hierarchical sparse allreduce communication mode (HSAC-ALADMM) to solve the L1-regularized optimization problems. The HSAC-ALADMM algorithm reduces the system communication cost through three

approaches: firstly, a lazily aggregate parameters strategy is proposed to reduce the total amount of transmission parameters among workers, which reduces the payload size among workers by modifying the ADMM algorithm. Secondly, according to the characteristics of sparse data, a hierarchical sparse allreduce communication mode is designed to reduce the system communication cost by improving the communication efficiency among workers. Finally, the Calculator-Communicator-Manager(CCM) framework is designed to implement the HSAC-ALADMM. The CCM framework effectively combines the advantages of the asynchronous calculation mode and the allreduce communication mode and separates the computing and communication of the algorithm through multithreading technology, thus improving the computing and communication efficiency of the system. Experimental results for the L1-regularized logistic regression problem show that the HSAC-ALADMM completes the classification task 1.6x faster than the ADMLLIB and has better system scalability without loss of accuracy. Compared with the SSAC-ALADMM algorithm, the communication cost of the HSAC-ALADMM algorithm is smaller when each node is assigned multiple processes. This means that the hierarchical sparse communication algorithm proposed in this paper has higher communication efficiency than the SSAR_Split_allgather algorithm proposed by Renggli et al. [22] in the multi-core cluster. Moreover, the waiting time of the HSAC-ALADMM algorithm proposed in this paper reduces as the number of workers increases. Therefore, it is very suitable for solving large-scale distributed optimization problems.

This paper improves the scalability of the distributed ADMM algorithm by reducing the system communication cost, but the convergence speed of the distributed ADMM algorithm will slow down as the number of workers increases. We will study how to improve the system concurrency without affecting the convergence speed of the algorithm. In addition, although the HSAC-ALADMM algorithm has been experimentally verified to have the similar convergence to the AD-ADMM algorithm, this paper does not provide a theoretical proof for it. We will theoretically analyze the convergence of the HSAC-ALADMM algorithm in the future.

Acknowledgements Thanks to the High Performance Computing Center of Shanghai University for providing computing resources and technical support. Funding was provided by the National Natural Science Foundation of China–Basic Algorithms and Programming Environment of Big Data Analysis Based on Supercomputing (Grant No. U1811461).

References


1. Algamal ZY, Lee MH (2019) A two-stage sparse logistic regression for optimal gene selection in high-dimensional microarray data classification. *Adv Data Anal Classif* 13:753–771
2. Alistarh D, Grubic D, Li J, Tomioka R, Vojnovic M (2017) Qsgd: communication-efficient sgd via gradient quantization and encoding. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) *Advances in neural information processing systems* 30. Curran Associates, Inc., pp 1709–1720
3. Balamurugan P, Posinasetty A, Shevade S (2016) Admm for training sparse structural svms with augmented l1 regularizers. In: *Siam International Conference On Data Mining*. <https://doi.org/10.1137/1.9781611974348.77>

4. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J et al (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found Trends® Mach Learn* 3(1):1–122. <https://doi.org/10.1561/22000000016>
5. Chang TH, Hong M, Liao WC, Wang X (2016) Asynchronous distributed admm for large-scale optimization-part i: algorithm and convergence analysis. *IEEE Trans Signal Process* 64(12):3118–3130. <https://doi.org/10.1109/TSP.2016.2537271>
6. Chiang W, Lee M, Lin C (2016) Parallel dual coordinate descent method for large-scale linear classification in multi-core environments. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp 1485–1494). ACM. <https://doi.org/10.1145/2939672.2939826>
7. Cucchiaro A (2012) Applied logistic regression. *Technometrics* 34(3):358–359
8. Dong J, Cao Z, Zhang T, Ye J, Wang S, Feng F, Zhao L, Liu X, Song L, Peng L et al (2020) Efllops: algorithm and system co-design for a high performance distributed training platform. In: *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp 610–622). IEEE
9. Elgabli A, Park J, Bedi AS, Issaid CB, Bennis M, Aggarwal V (2020) Q-gadmm: quantized group admm for communication efficient decentralized machine learning. *IEEE Trans Commun.* <https://doi.org/10.1109/TCOMM.2020.3026398>
10. Xing EP, Ho Q, Wei D, Xie P (2016) Strategies and principles of distributed machine learning on big data. *Engineering* 2:179–195
11. Fang L, Lei Y (2016) An asynchronous distributed admm algorithm and efficient communication model. In: *2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress* (pp 136–140). IEEE. <https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.41>
12. Genkin A, Madigan LD (2007) Large-scale Bayesian logistic regression for text categorization. *Technometrics* 49(3):291–304
13. Hasanov K, Lastovetsky A (2017) Hierarchical redesign of classic mpi reduction algorithms. *J Supercomput* 73(2):713–725. <https://doi.org/10.1007/s11227-016-1779-7>
14. Hong M (2018) A distributed, asynchronous and incremental algorithm for nonconvex optimization: an ADMM based approach. *IEEE Trans Control Netw Syst* 5(3):935–945. <https://doi.org/10.1109/TCNS.2017.2657460>
15. Lei D, Du M, Chen H, Li Z, Wu Y (2019) Distributed parallel sparse multinomial logistic regression. *IEEE Access* 7:55496–55508. <https://doi.org/10.1109/ACCESS.2019.2913280>
16. Li Y, Wang X, Fang W, Xue F, Li X (2019) A distributed admm approach for collaborative regression learning in edge computing. *Comput Mater Continua* 58(2):493–508
17. Lin C, Weng RC, Keerthi SS (2007) Trust region Newton method for large scale logistic regression. In: *The 24th International Conference on Machine Learning* (vol 358, pp 561–568). <https://doi.org/10.1145/1273496.1273567>
18. Lin Y, Han S, Mao H, Wang Y, Dally B (2018) Deep gradient compression: reducing the communication bandwidth for distributed training. In: *In ICLR 2018 International Conference on Learning Representations*
19. Liu J, Chen J, Ye J (2009) Large-scale sparse logistic regression. In: *In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining* (pp 547–556). ACM. <https://doi.org/10.1145/1557019.1557082>
20. Liu Y, Yuan K, Wu G, Tian Z, Ling Q (2019) Decentralized dynamic admm with quantized and censored communications. In: *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE
21. Nguyen TT, Wahib M, Takano R (2019) Topology-aware sparse allreduce for large-scale deep learning. In: *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)* (pp 1–8). IEEE
22. Renggli C, Ashkboos S, Aghagolzadeh M, Alistarh D, Hoefer T (2019) Sparcml: high-performance sparse communication for machine learning. In: *International Conference for High Performance Computing, Networking, Storage and Analysis*. <https://doi.org/10.1145/3295500.3356222>
23. Richtarik P, Takac M (2016) Distributed coordinate descent method for learning with big data. *J Mach Learn Res* 17(1):2657–2681
24. Safdar S, Zafar S, Zafar N, Khan NF (2018) Machine learning based decision support systems (dss) for heart disease diagnosis: a review. *Artif Intell Rev* 50(4):597–623. <https://doi.org/10.1007/s10462-017-9552-8>

25. Shevade SK, Keerthi SS (2003) A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics* 19(17):2246–2253
26. Thakur R, Rabenseifner R, Gropp W (2005) Optimization of collective communication operations in mpich. *Int J High Perform Comput Appl* 19(1):49–66. <https://doi.org/10.1177/1094342005051521>
27. Wang H, Sievert S, Liu S, Charles Z, Papailiopoulos D, Wright S (2018) Atomo: communication-efficient learning via atomic sparsification. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) *Advances in neural information processing systems* 31. Curran Associates, Inc., pp 9850–9861
28. Wang S, Lei Y (2018) Fast communication structure for asynchronous distributed admm under unbalance process arrival pattern. In: *27th International Conference on Artificial Neural Networks*. Springer
29. Xie J, Lei Y (2019) AdmmLib: a library of communication-efficient ad-admm for distributed machine learning. In: *IFIP International Conference on Network and Parallel Computing*. Springer. <https://doi.org/10.1109/TCNS.2017.2657460>
30. Zhang R, Kwok J (2014) Asynchronous distributed admm for consensus optimization. In: *Proceedings of the 31th International Conference on Machine Learning* (pp 1701–1709). jmlr. <http://respository.ust.hk/ir/Record/1783.1-66353>
31. Zhang X, Mahadevan S (2019) Ensemble machine learning models for aviation incident risk prediction. *Decis Support Syst* 116:48–63
32. Zhao H, Canny J (2014) Kylix: a sparse allreduce for commodity clusters. In: *43rd International Conference on Parallel Processing* (pp 273–282). IEEE. <https://doi.org/10.1109/ICPP.2014.36>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Dongxia Wang¹  · Yongmei Lei¹ · Jinyang Xie¹ · Guozheng Wang¹

Dongxia Wang
wangdongxia1983@126.com

Jinyang Xie
jyxie@shu.edu.cn

Guozheng Wang
329chenyang@gmail.com

¹ School of Computer Engineering and Science, Shanghai University, Shanghai, China