Adaptive Load Balancing for Parameter Servers in Distributed Machine Learning over Heterogeneous Networks

CAI Weibo[1], YANG Shulin[1], SUN Gang[1], ZHANG Qiming[2], YU Hongfang[1]

(1. University of Electronic Science and Technology of China, Chengdu 611731, China;
2. ZTE Corporation, Shenzhen 518057, China)

DOI: 10.12142/ZTECOM.202301009

https://kns.cnki.net/kcms/detail/34.1294.TN.20230301.1800.001.html, published online March 2, 2023

Manuscript received: 2022-04-11

Abstract: In distributed machine learning (DML) based on the parameter server (PS) architecture, unbalanced communication load distribution of PSs will lead to a significant slowdown of model synchronization in heterogeneous networks due to low utilization of bandwidth. To address this problem, a network-aware adaptive PS load distribution scheme is proposed, which accelerates model synchronization by proactively adjusting the communication load on PSs according to network states. We evaluate the proposed scheme on MXNet, known as a real-world distributed training platform, and results show that our scheme achieves up to 2.68 times speed-up of model training in the dynamic and heterogeneous network environment.

Keywords: distributed machine learning; network awareness; parameter server; load distribution; heterogeneous network

Citation (IEEE Format): W. B. Cai, S. L. Yang, G. Sun, et al., "Adaptive load balancing for parameter servers in distributed machine learning over heterogeneous networks," *ZTE Communications*, vol. 21, no. 1, pp. 72–80, Mar. 2023. doi: 10.12142/ZTECOM.202301009.

1 Introduction

Machine learning is widely used in many fields such as image classification[1], speech recognition[2], and natural language processing[3]. With the continuous increase in training data and the model size, the huge time cost of single-machine training is unacceptable to users. Therefore, distributed machine learning (DML) based on multi-machine parallelism has drawn more and more attention. Usually, distributed training is carried out within a single cluster, since it is considered that networks with limited bandwidth and complex and changeable states across clusters will seriously slow down the communication process of DML. However, due to the limitations of data privacy protection[4], data aggregation among clusters for model training is not allowed in some cases. In addition, with the proposal of Computing First Network[5–6], DML model training based on the integrated computing power of the whole network gradually shows great application prospects. Based on the consideration mentioned above, DML in heterogeneous networks across clusters has great research value.

This research was partially supported by the computing power networks and new communication primitives project under Grant No. HC-CN-2020120001, the National Natural Science Foundation of China under Grant No. 62102066, and Open Research Projects of Zhejiang Lab under Grant No. 2022QA0AB02.

There are mainly two communication architectures for DML: one is a centerless architecture, represented by AllReduce[7–8], and the other is a centered architecture, represented by a parameter server (PS) architecture[9–11]. In the PS architecture, there are usually two types of nodes in the DML system: the worker responsible for model training and the server for model aggregation and parameter update. During a typical training iteration of data parallelism and synchronous update mode, workers send model gradients uniformly after completing the training based on the local model and data, and the server receives the model gradient from workers. Thereafter, the model aggregation operation is performed to generate a global model, and the global model is sent to workers. Workers immediately replace the local model after receiving the global model from the server and start a new training iteration.

In this process, since the data from all workers need to be aggregated on the server, servers with limited bandwidth resources could become the bottleneck of transmission, which is also an inherent problem of the PS architecture[12]. In order to tackle this problem, a traditional solution[13] is to increase the number of servers and let multiple servers share the heavy communication load. Since the load distribution of each server usually follows the principle of fairness, this scheme has an ideal effect on homogeneous networks. However, in networks with heterogeneous bandwidth resources, since the system is agnostic about networks, it is impossible to match the communication load undertaken by each server with its communication capability. This leads to a consequence that the servers with low communication capacity slow down the communication time during the entire iteration process due to excessive load.

To efficiently handle the problem, this paper proposes an adaptive load balancing scheme for network-aware-PS-based DML over heterogeneous networks. The scheme senses the throughput of each link in networks in real time through a designed network awareness mechanism, reasonably evaluates the communication capability of each server based on this, and then selects appropriate servers to undertake the appropriate model aggregation tasks according to their communication capabilities. Finally, each server is assigned with communication load that matches its communication capability. The main contributions of this paper are as follows:

• We achieve an effective estimation of the link throughput by the low-cost and high-precision statistics method of the data transmission time with a simple and ingenious design, so as to learn the global network state information;

• We conduct an in-depth theoretical analysis of fine-grained data transmission and find a method to solve the optimal granularity of data slices.

• We design a simple and effective aggregation node selection method and a specific data slice assignment method, which can achieve efficient slice assignment.

2 Related Work

Multiple servers are typically used to alleviate heavy traffic on a single server in the PS architecture. But the specific implementation of the traditional PS architecture is network unawareness (such as MXNet[14], TensorFlow[15], and Petuum[16]), making it impossible to distribute the communication load more reasonably according to the actual communication capabilities of each server. Therefore, it is generally assumed that their communication capabilities are basically the same and are distributed according to the principle of fairness[17]. This usually results in poor performance in heterogeneous networks.

The authors in Ref. [18] have proposed an elastic PS load distribution scheme, which mainly analyzes the performance of servers by calculating the transmission time of the parameters using the linear regression method, and finally distributes communication load accordingly. Considering that the load distribution is in a complex network environment, the primary problem is the awareness of the network state. However, the authors do not provide a statistical method of parameter transmission time to implement network awareness, which makes the engineering solution to this kind of problem practically impossible. In addition, this scheme fails to deeply consider the optimal granularity of fine-grained transmission, and only uses empirical values, which cannot make the transmission reach the optimal state.

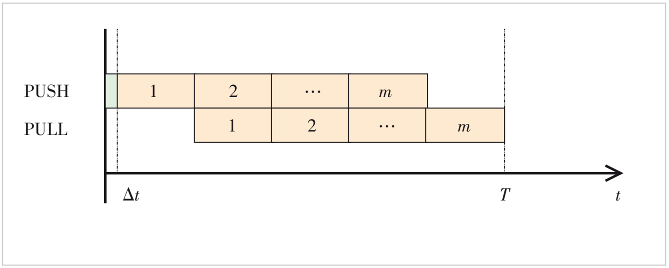
3 Proposed Approach

Based on the understanding of the related work about the PS load distribution of DML and the in-depth thinking of the problem, our approach is proposed as follows. First of all, the data are segmented according to the established slice granularity. The system in real time senses the network state through the cleverly designed network awareness mechanism, then evaluates the network communication capabilities of each node accordingly, and selects a part of the nodes as aggregation nodes. Finally, the complete distribution of fine-grained data is realized according to the PS load distribution and slice assignment algorithms.

3.1 Slice Granularity

During the model aggregation for DML, the process of workers sending data to the server to aggregate (PUSH) and the process of workers receiving the aggregated data returned from the server (PULL) are usually carried out synchronously, as shown in Fig. 1. The system performs the PULL process of data Slice 1 after all workers have completed the PUSH process of data Slice 1 (the time of data aggregation can be ignored), and the PUSH process of data Slice 2 is performed synchronously, thus overlapping PUSH and PULL. Theoretically, the smaller the data slice is, the better the overlapping of PUSH and PULL, ultimately making the aggregation quicker to complete. However, in practice, because there is a certain overhead in the data segmentation process, and there is also a certain additional network overhead in the transmission process of data slices, the granularity of slicing cannot be infinitely small.

Taking as many factors as possible into account, we analyze and solve this problem from a theoretical point of view. Considering the situation under a simple homogeneous network, in a complete data aggregation process under a single server, for a distributed system with a fixed data size M in every worker, the network bandwidth is W, and the number of nodes is N, where the slice granularity x that determines the times of the



▲ Figure 1. Illustration of data transmission, where the green block is the additional synchronization delay, and the orange block is the transmission time of each slice

data is sent separately by $m = M/x$ (the number of slices). In addition to the inherent transmission delay under the bandwidth limit, there are other network delays of data transmission during each data transmission. Hence, we compensate for the latency factor β. However, our study finds that the segmentation cost per slice is less than 1 ms, which can be ignored. We also find that $\beta \propto \frac{1}{W}$, thus let $\beta = \frac{1}{W}\alpha$. In addition, considering that the start time of the transmission of each node in practice is difficult to synchronize absolutely, there is an additional synchronization delay Δt in the total data transmission time. Eq. (1) shows the relationship between granularity x and the total time of data synchronization T.

$$T = \left(\frac{x}{W/N} + \frac{\alpha}{W}\right) \cdot \left(\frac{M}{x} + 1\right) + \Delta t, \tag{1}$$

where Δt denotes the delay of synchronization. We can expand the above equation to obtain:

$$T = \frac{NM}{W} + \frac{N}{W}x + \frac{M\alpha}{W} \cdot \frac{1}{x} + \frac{\alpha}{W} + \Delta t. \tag{2}$$

We simplify the above equation to the $y = x + \frac{a}{x}$ form and have:

$$\frac{W}{N}T = x + \frac{M\alpha}{N} \cdot \frac{1}{x} + M + \frac{\alpha}{N} + \frac{W}{N}\Delta t. \tag{3}$$

It can be found that when the left part of the equation takes the minimum value, the value of x is:

$$x = \sqrt{\frac{M\alpha}{N}}. \tag{4}$$

When M and N are determined, $\alpha = 1.2 \times 10^5$ can be obtained through actual testing. Obviously, at this point, the value x is only related to the data size M and the number of nodes N. It illustrates that during the distributed training of machine learning, when the training scale and the number of model parameters are determined, the value x is determined.

In a heterogeneous network, system performance is limited by the node with the smallest communication bandwidth (bottleneck node). If the bottleneck node is related to the server, W is calculated according to the bandwidth of the parameter server. If the bottleneck node is a worker, W can get the maximum value of T according to the bandwidth of the worker. However, in any case, the results are not related to W, so Eq. (4) is still of reference value for heterogeneous networks.

3.2 Network Awareness

In this scheme, the network state information that needs to be measured is only link throughput (available bandwidth). To avoid the large injection of probe traffic in the conventional network measurement technology[19–20] to occupy scarce network bandwidth resources, this scheme directly takes the model parameter data as the probe traffic. The granularity size_probe and the number probe_num of probe packets should be the minimum values that help the scheme to achieve an accurate measurement (the training iteration time remains stable in a stable heterogeneous network within a certain period of time), and they need to be determined in specific engineering implementation. Probe packets are segmented by each worker using the probability partition_rate to select the probe granularity size_probe to segment local data. In Eq. (5), where the coefficient γ is fixed at 0.6 in the experiment, the value of the probability partition_rate is necessary to ensure that the number of probe packets sent by the worker to each server is not less than probe_num, so as to realize the complete measurement of links between the worker and all servers.

$$\text{partition\_rate} = \frac{2N}{w/n}\gamma. \tag{5}$$

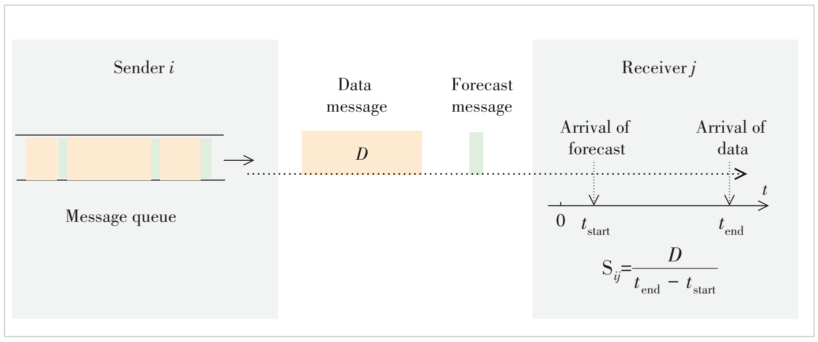
From the perspective of measurement implementation, the measurement of link throughput only needs to know the data size of the probe packet and the completion time of the probe packet transmission. Since the probe packet receiving node (receiver) has received the probe packet, the data size of the probe packet is known, but its transmission completion time is not easy to know. To calculate the transmission completion time, the start time and end time of transmission have to be figured out. When the probe packet is submitted to the upper layer, the receiver only knows the time at which the application layer received it, which is the end time of the probe packet transmission. But the receiver does not know the start time of the probe packet transmission. To obtain the start time of the probe packet transmission, the receiver can consider starting from the lower transport layer protocol and analyze the start time of the probe packet transmission in more detail, such as analyzing the Acknowledge Character (ACK) when the transmission is based on the Transmission Control Protocol (TCP). But in complex heterogeneous networks where different nodes may be deployed on different types of devices and use different network protocols, the scheme of obtaining the transmission start time of the probe packet based on the analysis of the underlying communication protocol is obviously not sufficiently pervasive.

In fact, without considering the underlying protocol analysis, it is also possible to obtain the start time of probe packet transmission. Although the application layer of the receiver does not directly know the start time of the probe packet transmission, the sending node (sender) knows. Therefore, it is only necessary to tell the receiver the start time of the probe packet transmission through the sender.

$$s_{ij} = \frac{\text{prob\_size}}{t_{\text{end}} - t_{\text{start}}}. \qquad (6)$$

Specifically, before the probe packet needs to be sent, the sender i sends the forecast message to the receiver j. After receiving the forecast message, the receiver can assume that the end time of the forecast message transmission is the start time t_{start} of the following probe (packet) message transmission. Until the following probe message arrives, and the receiver obtains the end time of probe message transmission t_{end} and the data size of probe messages size_probe. Finally, according to Eq. (6), the average rate s_{ij} of the probe message transmission from node i to node j can be calculated. The process of link throughput measurement is shown in Fig. 2. We use s_{ij} as an estimate of the throughput of the link through which the probe message is transmitted, and then use the estimated throughput as a reference for the evaluation of the communication capabilities of the node associated with the link. In this process, although additional traffic (the forecast message) is also injected into the network, it is not probe traffic. It is just the signaling message which is responsible for state forecast, and the data size is very small. Thus, the overhead of transmission over the network is almost negligible.

From the overall perspective of the network awareness mechanism, the specific measurement of network awareness is distributed at each node. If the links are required for transmission, they all need to be measured. To further enhance the reliability and stability of the measurement, we not only use special probe messages but also take data messages as probe messages to measure networks. Although it leads to some overhead, considering that the final value of throughput between nodes is the average value of the throughput record, the design can further improve the measurement effect. These measurements are obtained by the receiver, and then summarized to the central scheduling node (scheduler) which is responsible for the evaluation of the communication capacity of nodes and the distribution of communication load. When each node reports the link throughput information, the scheduler will update its recorded throughput value, evaluate capacity, and make decisions under the new network state timely, so that the system has a strong adaptive ability.



▲Figure 2. Link throughput measurement

3.3 Load Distribution

Load distribution is decided by a scheduler, which mainly involves the distribution of communication load on each server and the assignment of data slices. For the distribution of communication load, system deployment needs to be considered first. As bandwidth resources are scarce in heterogeneous networks, more physical nodes are needed in networks and the utilization of link bandwidth between nodes will be lowered if servers and workers are placed separately. To avoid these problems, we attach a server to each worker to get higher network resource utilization. In such a deployment, each node not only receives and distributes aggregated data as a server but also sends and receives aggregated data as a worker. It is important to note that in such a deployment, the node acting as a worker does not need to actually send the communication load to itself acting as a server. As all nodes as servers need to bear the corresponding proportion of the communication load, and the part of the load undertaken by themselves does not need to be actually sent, it is equivalent to reducing the data transmission of a worker.

Specifically, when the number of nodes is N, the local data size of each node is M, and the communication load of server $i\,(i \in V)$ is assumed to be m_i, the communication load L_i of node i is:

$$L_i = M - m_i + (N - 1)m_i. \qquad (7)$$

Considering that the throughput information received by the scheduler is presented as s_{ij} from node i to node j, the actual throughput S_i of node i can be calculated by Eq. (8):

$$S_i = \sum\nolimits_{j \in V \setminus i} s_{ij}. \qquad (8)$$

Based on this, we can calculate the transmission time t_i for node i to complete communication load L_i under throughput S_i by Eq. (9):

$$t_i = \frac{L_i}{S_i} = \frac{M + (N - 2)m_i}{S_i}. \qquad (9)$$

In the model aggregation stage, the data transmission of each node is carried out simultaneously, so the total transmission completion time in the training iteration is the maximum of the transmission completion time of each node $\max_{i \in V} t_i$. The purpose of reasonable communication load distribution is to minimize $\max_{i \in V} t_i$. In other words, the current problem model can be determined as:

$$\min \max_{i \in V} \frac{M + (N-2)m_i}{S_i}$$
$$\text{s.t. } M = \sum_{i \in V} m_i. \quad (10)$$

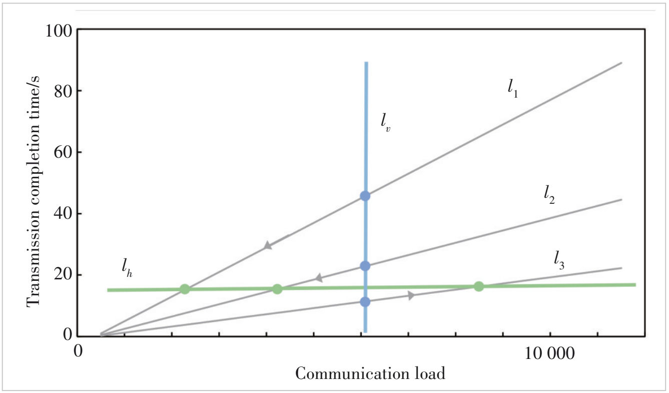
In Eq. (10), M, N and S_i are constants, and only m_i is variable. The objective function requires to minimize the maximum value of t_i. Under the strong constraint that the sum of all m_i is fixed, considering that adjusting the load of one node will inevitably affect the load of other nodes, it is intuitively difficult to determine the optimal value of m_i. However, we can write Eq. (10) as:

$$t_i = \frac{M}{S_i} + \frac{N-2}{S_i} m_i. \quad (11)$$

Eq. (11) is the linear function of t_i on m_i. For the training system with $V = \{1,2,3\}$, we draw the function curve of t_i on m_i of each node as shown in Fig. 3.

The problem of Eq. (10) can be approximately transformed to determine a point (m_i,t_i) on each line l_i in Fig. 3, and to minimize the maximum value in t_i on the premise that the sum of the abscissa of these points m_i is a constant value M. If the position of (m_i,t_i) is initialized randomly for each line and then moved gradually to minimize $\max_{i \in V} t_i$, the minimum value of $\max_{i \in V} t_i$ can be achieved if and only if all points are on the same horizontal line l_h. Otherwise, there must be a line $l_{h'}$, above and below which there are at least one point respectively. Thus, we can still get all the points closer to each other by moving the point above $l_{h'}$ down its line and moving the point below $l_{h'}$ up its line, until they are on the same horizontal line.

We distribute the communication load of each node according to the principle of equalitarianism in advance. Positions of (m_i,t_i) are initialized at the intersections of line $l_v = \frac{M}{N}$ and each line l_i. Then each point (m_i,t_i) is moved by means of iterative forced equalization of $\max_{i \in V} t_i$ and $\min_{i \in V} t_i$. Specifically, in a moving iteration, it is assumed that $i = \max$, when $t_{\max} = \max_{i \in V} t_i$, and $i = \min$, when $t_{\min} = \min_{i \in V} t_i$. When $t_{\max} = t_{\min}$, the x-coordinates $m'_{\max}$ and $m'_{\min}$ of the moved points $(m_{\max},t_{\max})$ and $(m_{\min},t_{\min})$ have the relationship as shown in Eqs. (12) and (13).



▲Figure 3. Geometrization of the load distribution problem

$$\frac{M + (N-2)m'_{\max}}{S_{\max}} = \frac{M + (N-2)m'_{\min}}{S_{\min}}. \quad (12)$$

$$m'_{\max} + m'_{\min} = m_{\max} + m_{\min}. \quad (13)$$

Therefore,

$$m'_{\max} = m_{\max} + m_{\min} - m'_{\min}$$
$$x'_{\min} = \frac{(N-2)(m_{\max} + m_{\min})S_{\min} - M(S_{\max} - S_{\min})}{(N-2)(S_{\min} - S_{\max})}. \quad (14)$$

Now, $(m_{\max}, t_{\max})$ and $(m_{\min}, t_{\min})$ move to the same ordinate position and the next iteration can be started until $\max_{i \in V} y_i = \min_{i \in V} y_i$. Algorithm 1 shows the detailed steps of the process.

Algorithm 1: Load distribution

Input: The local data size of each node M, the number of nodes N, the throughput S_i of node i, and the similarity threshold similarity_threshold of t_i, where $i \in V$.

Output: The load distribution m_i of node i.

```
1) Initialization: $m_i = \frac{M}{N}$, $t_{\max} = -\infty$, $t_{\min} = \infty$
2) for $i$ in $V$ do
3) $t = \frac{M + (N-2)m_i}{S_i}$
4) if $t_{\max} < t$ do
5) $t_{\max} = t$
6) $\text{node}_{\max} = i$
7) if $t_{\min} > t$ do
8) $t_{\min} = t$
9) $\text{node}_{\min} = i$
10) while $t_{\max} - t_{\min} \geqslant$ similarity_threshold do
11) $m_{\text{sum}} = m_{\text{node}_{\max}} + m_{\text{node}_{\min}}$
12) $m_{\text{node}_{\min}} = \frac{(N-2)(m_{\max} + m_{\min})S_{\min} - M(S_{\max} - S_{\min})}{(N-2)(S_{\min} - S_{\max})}$
13) $m_{\text{node}_{\max}} = m_{\text{sum}} - m_{\text{node}_{\min}}$
14) $t_{\max} = -\infty$, $t_{\min} = \infty$
15) for $i$ in $V$ do
16) $t = \frac{M + (N-2)m_i}{S_i}$
17) if $t_{\max} < t$ do
18) $t_{\max} = t$
19) $\text{node}_{\max} = i$
20) if $t_{\min} > t$ do
21) $t_{\min} = t$
22) $\text{node}_{\min} = i$
```

The first line of Algorithm 1 distributes the communication load of each node according to the principle of fairness in advance. Lines 2 – 9 determine the maximum and minimum transmission time of the nodes in the current communication load distribution, as well as the corresponding node. At Line 10, we judge whether the moving iteration needs to be stopped. In order to reduce the number of iterations, we define the difference between the maximum and minimum values of node transmission time as approximately equal if the difference is no more than similarity_threshold (the experience value is 1 s in our experiment). Lines 11 – 13 adjust the communication load of the nodes with the maximum and minimum transmission time. Lines 14 – 22 determine the maximum and minimum values of the transmission time of nodes after adjusting the communication load distribution, which is used for judgment in Line 10. Based on the above process, Algorithm 1 has a $\Theta\left(N + \frac{N}{2} \times N\right) = \Theta(N^2)$ time complexity when t_i has a uniform initial distribution on the timeline and similarity_threshold isn't too small.

In the specific process of slice assignment, data are transmitted as the slice, just like the basic granularity, thus the final work of load distribution is the assignment of data slices. Algorithm 2 shows the data slice assignment.

Algorithm 2: Slice assignment

Input: The load distribution m_i of node i, the data size paras_j of slice j, the number of slices num_slice, the granularity of probe slice size_probe, and the number of probe slices that each node sends to other nodes, where $i \in V$, $j \in (0, \text{num\_slice})$.

Output: The assignment result assign_j of slice j, where $j \in (0, \text{num\_slice})$.

```
1) Initialization: Initialize index variables index = 0.
2) for $i$ in $V$
3) for $h$ in $(0, \text{num\_probe})$ do
4) while index < num_slice and $\text{paras}_{\text{index}} \neq$ size_probe do
5) index = index + 1
6) if index > index_end do
7) break
8) $\text{assign}_{\text{index}} = i$
9) $m_i = m_i - \text{paras}_{\text{index}}$
10) index = index + 1
11) for index in $(0, \text{num\_slice})$ do
12) if $\text{assign}_{\text{index}}$ == NULL do
13) $\text{max\_}m = -\infty$
14) receiver = 0
15) for $i$ in $V$ do
16) if $\text{max\_}m < m_i - \text{paras}_{\text{index}}$ do
17) $\text{max\_}m = m_i - \text{paras}_{\text{index}}$
18) receiver = $i$
19) $\text{assign}_{\text{index}}$ = receiver
20) $m_{\text{receiver}} = \text{max\_}m$
```

At Lines 2 – 10 in Algorithm 2, the number of probe slices that the servers are distributed with is defined as num_probe, which is generated by segmentation probability partition_rate during data segmentation, mainly to maintain the awareness of the network state of idle nodes that are not distributed any slices. Lines 11 – 20 are used to achieve the assignment of the remaining slices. Specifically, for each slice, we traverse all current aggregation nodes and select the node with the largest remaining load as the receiving node of this slice. In this way, the receiving node with the best network state can be arranged for each slice as much as possible, and the excess load that the node needs to bear when the slice granularity is larger than the remaining load of nodes can be reduced as much as possible. Based on the above process, Algorithm 2 has a $O(\min(N \times \text{num\_probe}, \text{num\_slice}) + N \times \text{num\_slice})$ time complexity, which shows the execution time of the algorithm is mainly related to the number of nodes and data slices.

The scheme provides a standard execution process in order to make the system adaptive. In each iteration, specifically, at the beginning of the communication process, each node first reports to the scheduler the link throughput information measured in the communication process of the previous iteration, then waits for the scheduler to make the latest distribution strategy according to the link throughput information, and sends it to each node. After receiving the latest strategy information, each node updates its local strategy, transmits data according to the new strategy, and records the link throughput information measured during transmission. Based on such an interactive process, the training system can realize adaptability almost in real time.

4 Experiment

4.1 Environment and Deployment

We simulate a 12-node cluster with Intel(R) Xeon(R) E5-2678 v3 CPUs and NVIDIA 2080TI GPUs and use MXNet as a DML training platform. We have implemented our scheme by modifying the source code of MXNet and deployed the server and the worker in a 1:1 ratio, which means placing one server and one worker on each physical node in the cluster. The bandwidth limit between nodes is below the typical Wide Area Network (WAN) bandwidth of 220 M/bits with a TC-Tool[21]. The specific value of bandwidth is randomly determined and randomly adjusted periodically (300 s) to simulate the dynamic heterogeneous network environment. In addition, the hyperparameter configuration of the training system is shown in Table 1.

4.2 Experiment Design

We set up two related schemes to compare with our scheme (Aware). One scheme is Average[17], which is based on the equal distribution principle and network agnosticism,

▼**Table 1. Key hyperparameters**

Parameter	Value
Dataset	Fashion MNIST
Mini-batch	32
Optimizer	Adam
Learning rate	0.001

and the other is the elastic parameter distribution scheme named Elastic[18]. Since the network awareness mechanism of Elastic is unknown, we directly test Elastic based on our network awareness mechanism in experiments. For these three schemes, we test their performance on AleNet (228 MB), ResNet50 (93 MB), and MobileNet (21 MB) models respectively.

4.3 Performance Metrics

In our experiments, we use the training speed, namely the number of images per minute trained by the system, as the main performance evaluation metric. The higher the speed, the better the performance of the scheme. Eq. (16) shows the definition of speed, where num_iters is the number of iterations in *t*_iters time.

$$\text{speed} = \frac{\text{num\_iters}*\text{MiniBatch}*N}{t\_\text{iters}} . \tag{16}$$

In addition, single-round iteration time (SRIT) and average single-round iteration time (ASRIT) are used in the verifications of network awareness validity, verifications of segmentation granularity rationality, and cost analysis. SRIT is the time to complete a model training iteration, which is directly measured in tests. The shorter SRIT is, the better performance the scheme has.

5 Results and Analysis

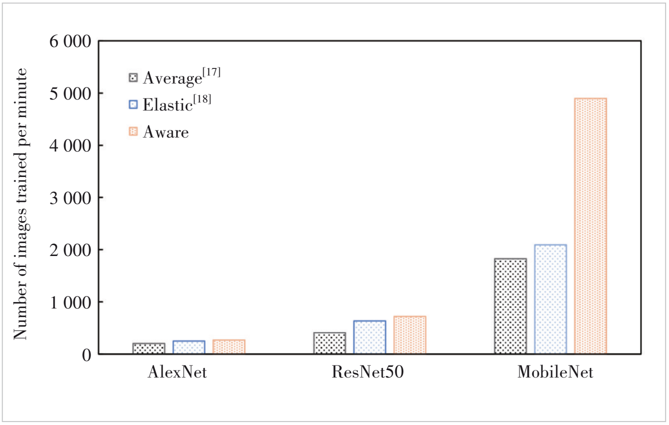
5.1 Training Speed

Fig. 4 shows the training speed of the compared schemes in different models. As we can see that network-aware Elastic and Aware schemes significantly improves performance: 1.14 times and 2.68 times for MobileNet, 1.56 times and 1.76 times for ResNet50, and 1.23 times and 1.32 times for AlexNet, compared with the Average scheme which is agnostic to network states. This shows that the PS adaptive load balancing is feasible and effective based on the network awareness. Compared with Elastic, Aware has achieved better performance improvement, 2.34 times for MobileNet, 1.13x for ResNet50, and 1.08 times for Alexnet, especially on the MobileNet model, which achieved over 2 times acceleration. This suggests that the load distribution strategy of Aware is indeed better than that of Elastic.

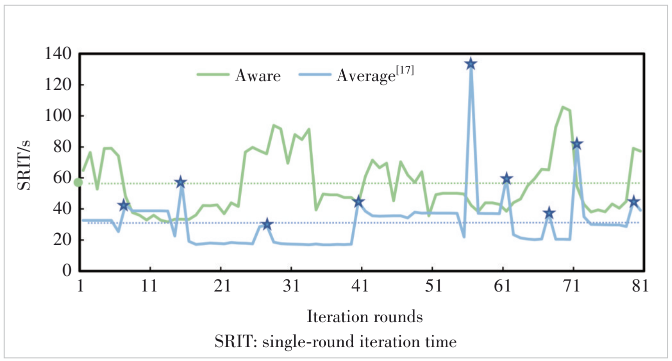
In addition, by comparing the speed gain on different models, it can be found that the gain achieved by Aware is more obvious on the smaller model (MobileNet). This is because the network load of the small model is small, the iteration time of model training is short, and the optimization effect of Aware is more significant in the same experimental network, which is finally shown as a significant increase in the training speed. On the larger model (AlexNet), Aware has almost no gain compared with Elastic. The reason is that there is no obvious room for optimization of the data aggregation process in the experiment network with limited bandwidth under the excessively large communication load.

5.2 Effectiveness Verification of Network Awareness

Fig. 5 shows the changes of SRIT of Aware and Average schemes with iteration rounds in the same dynamic network. The system parameters num_probe and size_probe are set to the best values of 2 and 10 000, respectively, which are determined by actual tests in the experiment. Due to space limitation of the paper, the details are omitted. In the figure, the curve of Average which is agnostic about the network is above the curve of Aware, which indicates that the optimization effect of Aware scheme is significant and lasting. Additionally, the curve of Aware exhibits periodic shock wave characteristics, which can be attributed to its poor performance in response to abrupt changes in network states at the crest and the end of the strategy. However, with the release of a new round



▲**Figure 4. Training speed of different schemes on different models**



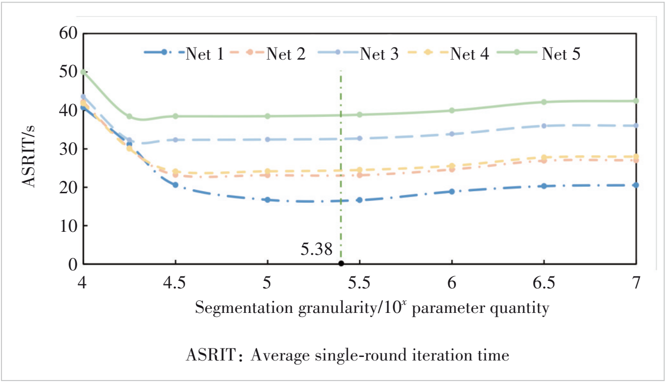
▲**Figure 5. SRIT comparison of Aware and Average schemes in dynamic networks** of strategy based on the latest network state, the performance of Aware improves rapidly. That also verifies the effectiveness and reliability of the network awareness mechanism of our scheme.

5.3 Reasonableness Verification of Segmentation Granularity

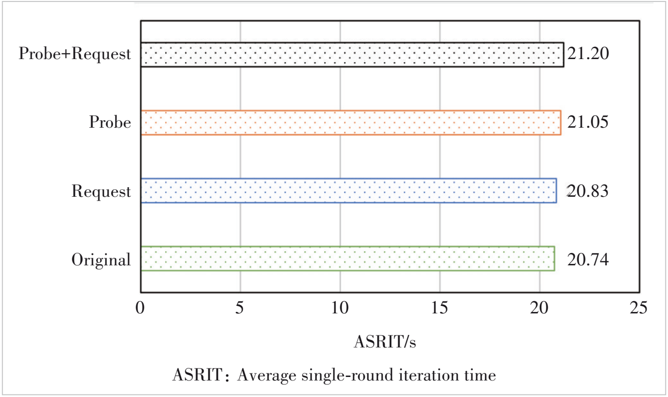
In order to verify the rationality of the theoretical analysis conclusion of slice granularity, we take the Resnet50 model as an example to test the change of ASRIT with the slice granularity under multiple network states. As shown in Fig. 6, in different network states, ASRIT remains almost unchanged within the logarithmic range of 5 – 5.5 (quantity of 10^5 – 3.16×10^5 parameters) of the slice granularity, while our theoretical value of 5.38 is exactly within this range. This indicates that our theoretical value of slice granularity can indeed achieve almost the lowest ASRIT in different network states.

5.4 Overhead Analysis

The overhead of the Aware scheme is likely to be concentrated in frequent forecast messages and synchronization of strategy requests with each round. As for the former, there should be no significant overhead because the preview message only contains extremely short header fields with a fixed length. As for the latter, because the experiments are based on the synchronous training mode and the synchronization of each round has already existed, there should be no obvious overhead. In order to verify this analysis, in a stable (static and isomorphic) network environment, we have tested ASRIT of the Average scheme under four conditions: requiring probe and strategy request synchronization (Probe + Request), only requiring probe (Probe), only requiring strategy request synchronization (Request) and neither requiring probe nor strategy request synchronization (Original). The ASRIT over dozens of iterations is shown in Fig. 7. Adding probe or strategy request synchronization does incur some overhead, but even with Probe + Request having the largest overhead, only 0.44 s (2.12%) overhead is added to Original, which is negligible compared with the huge gain shown in Fig. 5.



▲Figure 6. ASRIT of Aware scheme in different network states



▲Figure 7. ASRIT of Average scheme in different conditions

6 Conclusions

In this paper, we study the problem of PS load distribution in DML in heterogeneous networks. The state-of-the-art schemes cannot match the communication load with the communication capacity of PSs to achieve load balancing due to the lack of network awareness. The existing schemes with network awareness have not given specific network measurement methods, which makes them difficult to be realized in practice. This paper proposes a well-designed network awareness mechanism, which can realize low cost and high precision network measurement. In addition, the slice granularity determination and slice assignment of fine-grained transmission is studied. We have implemented the scheme in MXNet, and completed the function verification and performance measurement based on the experiment cluster. The results show that the proposed scheme can significantly accelerate DML.

References

[1] YU J, TAN M, ZHANG H Y, et al. Hierarchical deep click feature prediction for fine-grained image recognition [J]. IEEE transactions on pattern analysis and machine intelligence, 2022, 44(2): 563 – 578. DOI: 10.1109/TPAMI.2019.2932058

[2] KRIMAN S, BELIAEV S, GINSBURG B, et al. Quartznet: deep automatic speech recognition with 1D time-channel separable convolutions [C]//2020 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2020: 6124 – 6128. DOI: 10.1109/ICASSP40776.2020.9053889

[3] AHMAD F, ABBASI A, LI J J, et al. A deep learning architecture for psychometric natural language processing [J]. ACM transactions on information systems, 2020, 38(1): 1 – 29. DOI: 10.1145/3365211

[4] HONG R, CHANDRA A. DLion: Decentralized distributed deep learning in micro-clouds [C]//Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing. ACM, 2021: 227 – 238. DOI: 10.1145/3431379.3460643

[5] TIAN L, YANG M Z, WANG S G. An overview of compute first networking [J]. International journal of web and grid services, 2021, 17(2): 81 – 97. DOI: 10.1504/ijwgs.2021.114566

[6] KRÓL M, MASTORAKIS S, ORAN D, et al. Compute first networking: distributed computing meets ICN [C]//The 6th ACM Conference on Information-Centric Networking. ACM, 2019: 67 – 77. DOI: 10.1145/3357150.3357395

[7] AWAN A A, HAMIDOUCHE K, HASHMI J M, et al. Scaffe: co-designing MPI runtimes and caffe for scalable deep learning on modern GPU clusters [J]. ACM sigplan notices, 2017, 52(8): 193 – 205
[8] WANG S, LI D, GENG J K, et al. Impact of network topology on the performance of DML: Theoretical analysis and practical factors [C]//IEEE Conference on Computer Communications. IEEE, 2019: 1729 – 1737. DOI: 10.1109/INFOCOM.2019.8737595
[9] LI M, ZHOU L, YANG Z, et al. Parameter server for distributed machine learning [J]. Big learning NIPS workshop, 2013, 6: 2 – 12
[10] LI M, ANDERSEN D G, PARK J W, et al. Scaling distributed machine learning with the parameter server [C]//The 11th USENIX conference on Operating Systems Design and Implementation. ACM, 2014: 583 – 598. DOI: 10.5555/2685048.2685095
[11] LI M, ANDERSEN D G, SMOLA A, et al. Communication efficient distributed machine learning with the parameter server [C]//The 27th International Conference on Neural Information Processing Systems. ACM, 2014: 19 – 27
[12] ZHANG S, CHOROMANSKA A, LECUN Y. Deep learning with elastic averaging SGD [C]//The 28th International Conference on Neural Information Processing Systems. ACM, 2015: 685 – 693
[13] DEAN J, CORRADO G S, MONGA R, et al. Large scale distributed deep networks [J]. Advances in neural information processing systems, 2012, 1: 1223 – 1231
[14] CHEN T Q, LI M, LI Y T, et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems [EB/OL]. [2022-10-10]. https://arxiv.org/abs/1512.01274
[15] ABADI M, AGARWAL A, BARHAM P, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems [EB/OL]. [2022-10-10]. https://arxiv.org/abs/1603.04467
[16] XING E P, HO Q, DAI W, et al. Petuum: a new platform for distributed machine learning on big data [J]. IEEE transactions on big data, 2015, 1(2): 49 – 67. DOI: 10.1109/tbdata.2015.2472014
[17] MXNET. Distributed training in MXNet. [EB/OL]. [2022-10-10]. https://mxnet.apache.org/versions/1.7.0/api/faq/distributed_training
[18] CHEN Y R, PENG Y H, BAO Y X, et al. Elastic parameter server load distribution in deep learning clusters [C]//Proceedings of the 11th ACM Symposium on Cloud Computing. ACM, 2020: 507 – 521. DOI: 10.1145/3419111.3421307
[19] MOHAN V, REDDY Y J, KALPANA K. Active and passive network measurements: a survey [J]. International journal of computer science and information technologies, 2011, 2(4): 1372 – 1385
[20] GOEL U, WITTIE M P, CLAFFY K C, et al. Survey of end-to-end mobile network measurement testbeds, tools, and services [J]. IEEE communications surveys & tutorials, 2016, 18(1): 105 – 123. DOI: 10.1109/COMST.2015.2485979
[21] TC(8). Linux tc. [EB/OL]. [2022-10-10]. https://linux.die.net/man/8/tc

Biographies

CAI Weibo is pursuing his master's degree in communication and information system at University of Electronic Science and Technology of China. His research focuses on distributed machine learning.

YANG Shulin is pursuing his master's degree in communication and information system at University of Electronic Science and Technology of China. His research focuses on distributed machine learning.

SUN Gang (gangsun@uestc.edu.cn) is a professor of computer science at University of Electronic Science and Technology of China. His research interests include machine learning, cloud computing, high performance computing, parallel and distributed systems, ubiquitous/pervasive computing and intelligence and cyber security.

ZHANG Qiming is a senior system architect of ZTE Corporation. He received his bachelor's degree from Zhejiang University, China in 1992. His research interests include MEC and heterogeneous computing.

YU Hongfang is a professor of University of Electronic Science and Technology of China. Her research interests include network virtualization, cloud computing and next generation network.