Distributed Machine Learning and ”data-preprocessing” System

Ming Cheng 517021910750, Shanghai Jiao Tong University

Abstract—By reading and studying 10 top-level conference papers of distributed machine learning systems in nearly 5 years and comparing them, I think they mainly explain and solve two mainstream problems: The problem of resource allocation balance and the speed of information transfer between different servers when doing distributed machine learning training. Some papers have proposed new types of distributed system architecture while others propose new resource allocation strategies, solving the above two problems from the aspect of hardware and software. In addition, I have found another problem: when the server performs a certain task, the completion progress is inconsistent due to the different speed of different servers. Therefore, when a new task comes, how to optimally assign it to the server is a problem worth considering. For this, I propose an algorithm similar to multi-objective convex programming, considering the constraints of distributed system. And we can solve the above problem by solving the algorithm.

In addition, considering the background of the modern era, data on Internet today is huge and the content is not standardized, so it is necessary to perform data pre-processing before passing the data as input to the machine learning system. Inspired by the neural network model, I designed the so called ”data-preprocessing” system. The system consists of an input layer, a standardization layer, a sampling layer and an intermediate interface. Through this system, the content of the original data is more standardized and more useful as well, which may help to improve the performance of the distributed machine learning system.

Index Terms—distributed machine learning system, resource allocation, information transmission, ”data-preprocessing” system

I. "In-depth understanding of cloud computing" research report

The reference is the 10 related papers for the study.

Among the papers, the main area involved is machine learning and deep learning training in distributed systems. The main problems can be roughly divided into two types. The first one is that when the training is performed, the number of model parameters is huge due to the complexity of the network structure. A lot of resource space is occupied when implementing the optimization algorithms such as SGD or gossiping SGD. Therefore, how to balance resource allocation among various workers is a problem to be solved. The second one is that the speed of interaction between workers affects the efficiency of training. When training in a distributed system, parameters of the model need to be transferred and updated between workers, so how to communicate between different workers quickly and efficiently is also a problem to be solved.

The selected articles have proposed different technical solutions to the above porblems.

A. Problem 1

The solution to Problem 1 comes from two perspectives: constructing a new distributed architecture or proposing a new balancing scheme. For the new architecture, for example, [1] proposed one named AKO, which does not require a parameter server and implements scalable and decentralized synchronization. For new balancing scheme, [3] proposed LB-BSP to balance the allocation of resources between workers. It works by coordinately setting the batch size of each worker so that they can finish batch processing at around same time. Similarly, the ”partial gradient exchange” scheme used to synchronize the model copy between workers is also proposed in [1]. To synchronise replicas as often as possible subject to the available network bandwidth, workers exchange partitioned gradient updates directly with each other. After several rounds, workers eventually receive all gradient partitions so that the model convergence is not affected.

B. Problem 2

The solution to Problem 2 is to propose a new interaction mechanism. An adaptive communication model was proposed in [7]. This method considers that the communication should be driven according to the difference between the averaged models in two consecutive iterations, instead of a fix number of iterations. Under this strategy, communication time between workers can be reduced by 92% but the accuracy of the model can still be maintained[7]. In addition, MLNET is also proposed as a local process on workers and parameter servers[10]. Through its traffic reduction techniques[10] and traffic management techniques[10], it successfully reduces network traffic when servers communicate with each other.

In addition to the architectures and strategies described above, there are other new architectures and schemes that have been proposed. For example, Orpheus[2] is proposed to reduce the size of information transmission, and DLion[6], a new system architecture, is designed to reduce the amount of gradient exchange.

C. New Problems Proposed

Combining these papers, we can see that for solving the same problem, they have different concerns: either proposing a new system architecture or a new scheme. Therefore, they solved the problem from the aspect of hardware structure and software optimization, respectively. Of course, the problems described and solved in these papers are essentially to improve the convergence speed of the model training without affecting the accuracy. By reading these papers, I think we can also improve the convergence speed of the model from other aspects.

As we all know, the method of distributed training can be generally divided into two types: model parallelism and data parallelism. For model parallelism, each parameter server is responsible for different parts of the network model. For example, different layers of the neural network can be assigned to different servers, and each server is connected to form a complete network model. In this way, the data going through all the servers is same. For data parallelism, however, different machines have multiple copies of the same model, and each machine is assigned only a portion of data. Finally, the results of all machines are combined to produce the final result. Of course, for a multi-GPU cluster system, we can combine two methods: model parallelism on the same machine (such as splitting the model between GPUs), and data parallelization between different machines.

However, no matter which architecture is adopted, the strategy of ”allocating a task to multiple machines and then performing the next task” is followed. The whole training process can be regarded as a combination of different tasks. Therefore, I believe that in the architecture of the above two mainstream distributed machine learning systems, the transfer process between different tasks will affect the efficiency of entire training process. If we can fix the time for each task to be completed such as setting a variable like ”maximum deadline” (each task must be completed within the maximum period), it becomes more convenient to manage the time of each task as well as facilitate the start time of follow-up tasks. However, considering the inconsistency of the speed of different machines, it is likely that the task has been completed in some machines but not in others which are slower. Therefore, when a new task arrives, how to redistribute it to each machine is a problem worth considering and solving. And this problem will affect the convergence speed of the whole model, obviously. After all, as mentioned above, the whole training process can be seen as ”a combination of multiple tasks in turn”.

D. Possible Solution to Problems Proposed

For the solution of this problem, I think that we can develop a ”task assignment algorithm”, similar to multi-objective linear programming or multi-objective convex programming algorithm. We can set the goal as shortest total time and highest performance. The pseudo code representation of the algorithm is shown as: $goal = min\{w_1 * max\{t_i\} - w_2 * accuracy - w_3 * resource\}$. In the formula, the total time is the maximum time for all machines to complete the task while the performance is the weighted sum of the accuracy and the average resource occupancy in all training times.

By adding a series of constraints (e.g. all tasks must be completed within their deadlines respectively, etc.) and solving the above expression, we can derive an optimal strategy for assigning a series of tasks to each machine.

II. My Innovation

A. Introduction & Looking for Problems

Considering background today, how to deal with extremely large amounts of data is a very important and critical issue for distributed machine learning systems. In the traditional parallel machine learning system, the purpose is more to ”put more machines, the calculation speed is faster with same data size”. And when data volume is gradually increased, content of data is also uneven and not standardized. How to pre-process the data before inputting into the system is one of the key factors affecting system performance. In today’s era, data can not be loaded by one or a few servers due to its huge volume, or even if it is loaded successfully, the access speed is still very slow because of the limitation of the bandwidth of the machine’s I/O channel. For example, in the Web 2.0 era, various Web 2.0 tools help users build their own web pages, such as blogs and Weibo. Therefore, when a search engine crawls down a web page and analyzes its content, the number of web pages increases exponentially and is also hard to estimate because of the irregularity of data. For example, in the Internet advertising system, the action that a user clicks on the advertisement will be passed as data to the machine learning system, and finally a new estimated advertisement will be generated. However, these data record the behavior of billions of users on the Internet, and the number of behaviors and types of them are different at different times. Therefore, the scale of data and the degree of standardization are the key factors affecting the performance of machine learning systems.

As mentioned above, in this big data era, the scale of data is extremely large, and the degree of standardization is low. Therefore, how to effectively preprocess data is a critical and important issue before passing data into the machine learning system.

B. My Solutions:"data-preprocessing" System

In order to solve the above problems, I think that it is full of necessity to establish a special data-preprocessing system. Here are my possible solutions to the above issue:

1) Construct the Input Layer: The input layer consists of a number of storage servers that do not have to store all available data at once, as these data are subsequently standardized and randomly sampled.

2) Construct Standardization Layers: The standardization layer and the input layer are connected one by one, and the data coming from the input layer is subjected to standardization processing in several steps. Several servers form a standardization layer to eliminate data which is not related to the task or has less impact on the quality of the results. As for the relevant ones, we can also abandon them if the error is still within tolerance.

3) Construct Sampling Layers: The connection between the standardization layer and the sampling layer is inspired by the neural network. The server in the first layer of total sampling layers is connected to all servers at the last layer of the standardization layer. The randomly extracted data having been standardizated is realized by information transmission between the sampling layer and the standardization layer server. Second, several sampling layers are fully connected, and the latter server randomly reads data from the previous server. Repeat this step to get a randomly sampled data set.

4) Construct the Interface: After the raw data is read, normalized abd randomly sampled, it can be used as input to machine learning or deep learning systems. However, the data preprocessing system may be inconsistent with the interface of the distributed machine learning system, so it is necessary to construct an interface between them. After this step, the normalized random data can be read into the distributed machine learning training system.

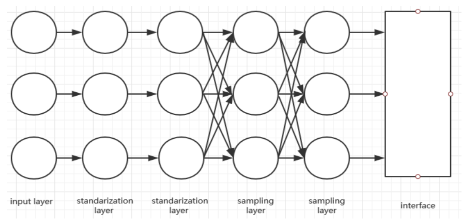

Fig. 1. structure of ”data-preprocesing” system

C. Benefits of the Solution

The above idea is inspired by the neural network model, which is to reproduce the software-like neural network in a hardware structure. I think the above-mentioned data pre-processing system has the following advantages:

1. The data structure is relatively standardized. The original data set is very large and it has a great possibility that data content is inconsistent. However, after the above processing procedures, the non-standard data will be eliminated to achieve the standardization, which is conducive to improving the accuracy of the machine learning system.

2. The data content is more useful. The original data set is very large and there exists a lot of miscellaneous data, so it is necessary to eliminate them. Through the processing of the standardization layer, the content of the original data is more compact, which is beneficial to improve the precision of the machine learning system.

3. The robustness of the machine learning system is improved. After going through a number of sampling layers which are fully connected, the data is highly random so that passing it as input data to the machine learning system can make the system perform well without being affected by a certain part of data. The bias in the final output caused by only several parts of data set will decrease.

III. Conclusion

In the first phase, I mainly expounded the problems encountered in the current distributed system for machine learning and deep learning. There are two main problems: the first one is the resource allocation and balance problem on different servers during training. Uneven distribution may result in overall training time being extended due to slower servers. The second one is the problem of information transmission between servers. Whether it is model parallelism or data parallelism, information transmission between devices is inevitable. Therefore, striving to shorten the information transmission time can greatly improve the overall training efficiency. In addition, I have come up with another question: how to assign the next task is a question worth considering during the process of training. Taking into account the training objectives and assumptions (such as setting the deadline for each task to achieve unified management of time), we can turn this problem into multi-objective planning, and then solve the optimal allocation strategy.

After the unified management of the assignment time of each task and the adoption of the optimal follow-up task assignment strategy, the model training time will be greatly shortened, which is conducive to improving efficiency.

In the second stage, I combined the current background of the era. Considering that today's data on Internet is huge and not that standardizated, it is necessary to perform data pre-processing before model training. Therefore, I proposed to build a ”data-preprocessing” system. This system consists of an input layer, several standardization layers, several sampling layers and an interface between data-preprocessing system and distributed machine learning model. The data is passed from the input layer to the standardization layer, going through which the data that has little impact on the output, or those have some effect on the output but can be ignored under precision tolerance will be rejected. Through the standardization layer, standardization of data is improved. Next, pass the normalized data to the sampling layer. Several sampling layers are fully connected to achieve multiple-random-sampling of data. After the sampling layer, the randomness of the data is strong, which is beneficial to improve the robustness of the distributed machine learning system. Finally, through the intermediate interface, the pre-processed data can be passed to the distributed system for training.

Through the processing of the ”data-preprocessing” system, the data standardization and randomness are greatly improved, which makes the distributed machine learning system more robust and accurate.

References

[1]Pijika Watcharapichat, Victoria Lopez Morales, Raul Castro Fernandez, Peter Pietzuch. Ako: Decentralised Deep Learning with Partial Gradient Exchange. SoCC '16 Proceedings of the Seventh ACM Symposium on Cloud Computing. Santa Clara, CA, USA — October 05 - 07, 2016

[2]Pengtao Xie, Jin Kyu Kim, Qirong Ho, Yaoliang Yu, Eric Xing. Orpheus: Efficient Distributed Machine Learning via System and Algorithm Co-design. SoCC '18 Proceedings of the ACM Symposium on Cloud Computing. Carlsbad, CA, USA — October 11 - 13, 2018

[3]Chen Chen, Qizhen Weng, Wei Wang, Baochun Li, Bo Li. Fast Distributed Deep Learning via Worker-adaptive Batch Sizing. SoCC '18 Proceedings of the ACM Symposium on Cloud Computing. Carlsbad, CA, USA — October 11 - 13, 2018

[4]Anas Toma, Juri Wenner, Jan Eric Lenssen, Jian-Jia Chen. Adaptive Quality Optimization of Computer Vision Tasks in Resource-Constrained Devices using Edge Computing. 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Larnaca, Cyprus, Cyprus. 14-17 May 2019

[5]Aarati Kakaraparthy, Abhay Venkatesh, Amar Phanishayee, Shivaram Venkataraman. The Case for Unifying Data Loading in Machine Learning Clusters. 11th USENIX Workshop on Hot Topics in Cloud Computing. Renton, WA, USA, July 8, 2019

[6]Rankyung Hong, Abhishek Chandra. DLion: Decentralized Distributed Deep Learning in Micro-Clouds. 11th USENIX Workshop on Hot Topics in Cloud Computing. Renton, WA, USA, July 8, 2019

[7]Li-Yung Ho, Jan-Jan Wu, Pangfeng Liu. Adaptive Communication for Distributed Deep Learning on Commodity GPU Cluster. [2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)]. Washington, DC, USA 1-4 May 2018

[8]Huangshi Tian, Minchen Yu, Wei Wang. Continuum: A Platform for Cost-Aware, Low-Latency Continual Learning. SoCC '18 Proceedings of the ACM Symposium on Cloud Computing. Carlsbad, CA, USA — October 11 - 13, 2018

[9]Liang Luo, Jacob Nelson, Luis Ceze, Amar Phanishayee, Arvind Krishnamurthy. Parameter Hub: a Rack-Scale Parameter Server for Distributed Deep Neural Network Training. SoCC '18 Proceedings of the ACM Symposium on Cloud Computing. Carlsbad, CA, USA — October 11 - 13, 2018

[10]Luo Mai, Chuntao Hong, Paolo Costa. Optimizing Network Performance in Distributed Machine Learning. Usenix Workshop on Hot Topics in Cloud Computing. Santa Clara, CA, USA — July 06 - 07, 2015