Hindawi
Mobile Information Systems
Volume 2017, Article ID 3824765, 12 pages
https://doi.org/10.1155/2017/3824765

Research Article

A Parallel Strategy for Convolutional Neural Network Based on Heterogeneous Cluster for Mobile Information System

Jilin Zhang,[1,2,3,4] Junfeng Xiao,[1,2] Jian Wan,[1,2,4,5] Jianhua Yang,[6] Yongjian Ren,[1,2] Huayou Si,[1,2] Li Zhou,[1,2] and Hangdi Tu[1,2]

[1]*School of Computer and Technology, Hangzhou Dianzi University, Hangzhou 310018, China*
[2]*Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou, China*
[3]*College of Electrical Engineering, Zhejiang University, Hangzhou 310058, China*
[4]*School of Information and Electronic Engineering, Zhejiang University of Science & Technology, Hangzhou 310023, China*
[5]*Zhejiang Provincial Engineering Center on Media Data Cloud Processing and Analysis, Hangzhou, Zhejiang, China*
[6]*College of Computer Science and Technology, Zhejiang University, Hangzhou 310018, China*

Correspondence should be addressed to Jian Wan; wanjian@hdu.edu.cn

Received 25 January 2017; Accepted 23 February 2017; Published 21 March 2017

Academic Editor: Jaegeol Yim

With the development of the mobile systems, we gain a lot of benefits and convenience by leveraging mobile devices; at the same time, the information gathered by smartphones, such as location and environment, is also valuable for business to provide more intelligent services for customers. More and more machine learning methods have been used in the field of mobile information systems to study user behavior and classify usage patterns, especially convolutional neural network. With the increasing of model training parameters and data scale, the traditional single machine training method cannot meet the requirements of time complexity in practical application scenarios. The current training framework often uses simple data parallel or model parallel method to speed up the training process, which is why heterogeneous computing resources have not been fully utilized. To solve these problems, our paper proposes a delay synchronization convolutional neural network parallel strategy, which leverages the heterogeneous system. The strategy is based on both synchronous parallel and asynchronous parallel approaches; the model training process can reduce the dependence on the heterogeneous architecture in the premise of ensuring the model convergence, so the convolution neural network framework is more adaptive to different heterogeneous system environments. The experimental results show that the proposed delay synchronization strategy can achieve at least three times the speedup compared to the traditional data parallelism.

1. Introduction

Mobile devices are involved in our daily life, from online shopping to social connection to working assistant. From the business perspective, the information gathered from the devices is so valuable that it can be used to learn customers' expense characteristics and improve user experience. For example, location is one of the key factors that are related to users' consumption behavior. We can recommend nearby restaurants, shopping malls, and parks based on location, and the function is implemented in many map applications like Google Maps. For merchants, understanding user behavior is very helpful to proactively provide potential services and increase customer engagements [1, 2]. To take advantage of the data analysis benefits, the first step is to figure out customers' consumption patterns. In this paper, we proposed a novel delay synchronization based machine learning strategy to improve pattern recognition and it laid a foundation for intelligent business marketing.

Due to the increasing data volume, the data processing is a huge challenge in mobile information systems. With the development of machine learning, convolution neural network has become a suitable method to deal with such large data. Convolutional neural network is a special multistage global training deep neural network model produced for two-dimensional image recognition [3], which combines the traditional artificial neural network with deep learning model. It not only has the general characteristics of the traditional artificial neural network, such as nonlinear, unlimited, nonstationary, and nonconvexity characteristics [4], but also contains more advantages, including fault-tolerant ability, self-learning ability and localized receptive fields, weight sharing, and pooling (secondary sampling). Convolutional neural network can discover the characteristics directly from a large number of image data and make a more profound description of the vast amount of information contained in the image. Convolutional neural network can achieve more than two orders of magnitude improvement compared to the human identification accuracy [5, 6]. With its powerful learning ability, convolutional neural network has been widely used in target tracking [7, 8], face detection [9–11], license plate detection [12], and handwriting recognition [13, 14]. It is an important research topic in machine learning, computer vision, mobile information system, and other scientific research fields.

The previous research has shown that the low model quality can be improved by the large-scale iterative training process, through modifying, testing, and evaluating the parameters of the model (network structure, the initial value of the range, learning methods, learning rate, etc.). However, with the increasing of the model scale and the training data, the time complexity of the training process is so large [15, 16] which restricts the development of the convolutional neural network. Several researchers [15, 17–20] proposed leveraging a distributed and parallel's data processing technology to support the rapid expansion of the model scale and data size.

Existing convolutional neural network training framework often takes the algorithm characteristics and the specific machine attributes as the main basis for the system design and optimization; however, this approach does not consider the relationship of the parallel computing model and the common characteristics in machine learning applications. For example, in the system architecture, (1) multicore and many-core technology has been widely used in parallel computers to increase the processing speed; besides, in order to reduce the cost, heterogeneous cluster has gradually replaced the traditional custom machine and becomes the mainstream architecture structure. So the traditional parallel computing is not suitable for the new era of big data parallel computing system; (2) the traditional parallel computing mode is of vertical expansion by leveraging more computing resources to enhance the performance, but terabyte or petabyte data processing and analysis require the horizontal expansion to improve the performance. Due to the different expansion requirements, the traditional methods of parallel computing are difficult to solve modern big data application issues.

In the general characteristics of convolution neural network application, the characteristics of parallel applications changed. Machine learning or deep learning application is a typical intensive computing iterative convergence application which shows the characteristics of fault-tolerance, structural dependence, nonuniform convergence, and sparse optimization. Traditional parallel computing application guarantees the accuracy of data parallelism or model parallelism by large amount of data synchronization; it cannot make full use of the characteristics of dense computing iterative convergence algorithm to improve the performance of application.

The basic idea of existing large-scale convolutional neural network parallelization is through reconstruction of convolutional neural network algorithm to give full play to the system performance advantages, in order to improve training efficiency and effect. However, such a program usually has two key issues. First, optimization methods issue means how to choose the optimization method to improve the efficiency of intensive computing iterative convergence algorithm. Second, the allocation of machine resources and data communication between nodes all require developers to manually perform single static tuning. Not only is it for too long, but it also relies on the experience of developers heavily. It is difficult to adapt to the structural changes in computing resources.

In this paper, we proposed a new delay synchronization parallel strategy for heterogeneous distributed cluster system. It is based on the traditional data parallel and model parallel method and combined with stale synchronous parallel parameter server system [21]. This strategy can shield the factors in model training process such as the communication bandwidth, memory bandwidth, memory hierarchy, memory latency, thread management, and processing mode. The training process will not be affected by the dynamic changes of the computing resources if the resources are adequate. As a result of decoupling the training algorithm and system hardware resources, the proposed strategy successfully frees developers from process calculation, resource allocation, and data communication optimization, and it effectively improves the program, especially in the heterogeneous environments.

Section 2 introduces an overview of the convolutional neural network parallel strategy. Section 3 gives the problem description, including training methods and existing problems of data parallel and model parallel. Section 4 describes the process of delay synchronous parallel strategy. Section 5 presents the experimental results and corresponding analysis. Section 6 summarizes the paper.

2. Related Work

This section mainly describes the strategies related to convolution neural network parallelization, including several early methods of parallelization and the current mainstream distributed data parallelization and model parallelization method.

2.1. Early Parallelization Methods. FPGA (field programable gate array), as a kind of computing intensive accelerator, can accelerate algorithm by mapping it to the hardware module. We usually use the method that combined the "host" with "FPGA" [22], and the host used in the control of training process of beginning and ending provides image data as input in the forward propagation. The application of FPGA based artificial neural network includes image segmentation [23], image and video processing [24], intelligent image analysis [25, 26], autonomous robot technology [27], and sensorless control [28, 29]. But because this kind of parallel method requires the programer to have the solid digital

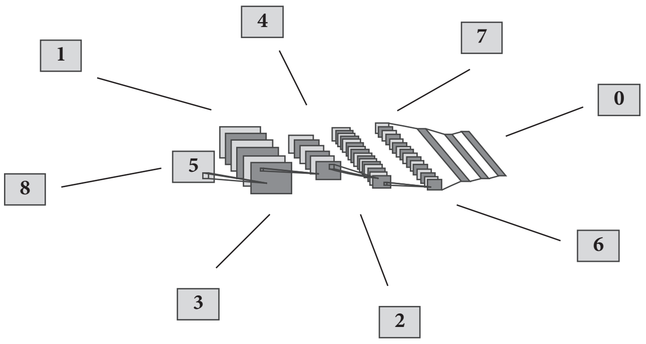

Figure 1: The schema of data parallel.

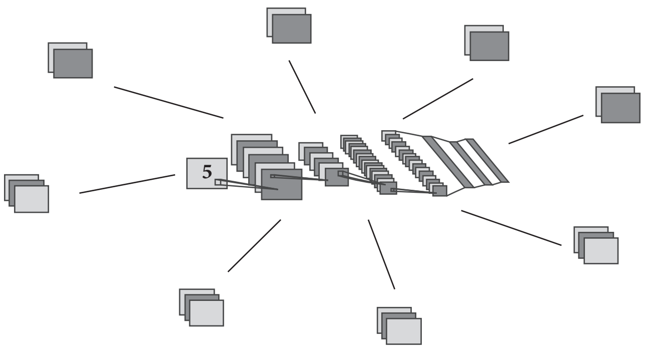

Figure 2: The schema of model parallel.

circuit knowledge and the programing complexity is high, it is barely used in practice.

With the rapid development of the GPU, the floating point operation speed is 10 times faster than the same period CPU; the researchers began to use GPU to accelerate the convolution neural network algorithm [30–32]. GPU contains a large (up to 10 G) shared memory and thousands of streaming processors, suitable for the inherent parallel structure of the convolutional neural network. GPU indeed can accelerate the performance [33, 34], but, taking into account the heterogeneous system, mapping algorithm to the hardware system cannot give full play to the resources of computing hardware.

When faced with the massive data of terabyte or petabyte, MapReduce is used to solve the problem. Convolutional neural network based on MapReduce parallel [35, 36] can also achieve a better result, but with the increasing of the number of parameters in the network model, and the difficulty of model training are increased as well. MapReduce is not suitable for high computing density iterative algorithm.

2.2. Data Parallel and Model Parallel. Data parallel and model parallel were proposed by Google distributed researcher Jeff Dean and deep learning researcher Andrew Ng in 2012 on the project called "Google Brain" [37, 38]. It referred to use of CPU cluster architecture combined with model parallel and data parallel implementation of the deep learning system DistBelief.

Data parallel means that, in the process of the model training, the training samples are divided and distributed to different computing nodes, and each computing node has a training model. After the end of each iteration, each node is doing a weight update communication to update the training model. Data parallel schema is shown in Figure 1 [37, 38].

In the model parallel, the model is divided into multiple slices, each of which is stored into a single server, all of which can be trained for a complete model. In the process of the training, each node contains a complete model network but only trained a specific part of the model. The model parallel is more suitable for the large model; it can solve the problem of limited training memory on a single machine. Adopting the model parallel distributed method can reduce the size of the occupied memory in each node. The model parallel scheme is shown in Figure 2 [37, 38].

By using the two methods in heterogeneous system, it is difficult for the convolutional neural network algorithm to select the appropriate optimization method and the optimal time, which means it cannot give full play to the advantages of the computing resources of the heterogeneous system. Moreover, when the hardware condition changes, the training algorithm cannot dynamically adapt to the computing resources, so the efficiency of the training process is not high.

In short, the existing parallel methods cannot fully adapt to the heterogeneous architecture, so as not to take advantages of heterogeneous architecture resource.

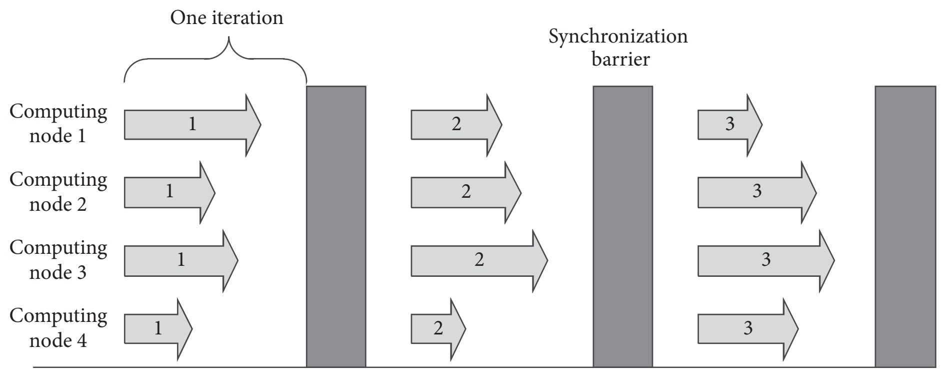

Figure 3: Training process of synchronous parallel.

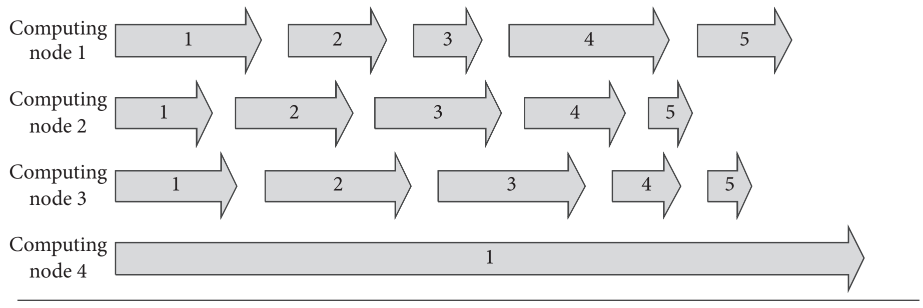

Figure 4: Training process of asynchronous parallel.

3. Problem Description

In this paper, we mainly introduce how to train the neural network in the heterogeneous and distributed environment. In this section, we first describe the training methods and existing problems of data parallel and model parallel and then further explain the factors that should be considered in the parallelization process.

3.1. Data Parallel and Model Parallel Training Methods. Both data parallel and model parallel can be categorized as synchronous parallel and asynchronous parallel, parameter server parallel and nonparametric server parallel.

Synchronous and asynchronous parallel processes are shown in Figures 3 and 4. Synchronous parallel method means that, in the process of model training, each update of the training model is carried out after the completion of an iteration of all computing nodes. And each node begins to continue the next iteration of the training after they obtain a new training model. But, in the method of asynchronous parallel, when the iteration is completed, faster computing node notices other computing nodes to update the weights after the completion of one iteration but does not wait for other nodes to be updated.

From Figures 3 and 4, we can see that the synchronous parallel calculation requires the use of synchronization barrier to force all the computing nodes to carry out a parameter update after one iteration is completed. This parallel method will lead to faster nodes waiting for the other slower nodes, which greatly affected the model training speed. So in the synchronous parallel training method, in order to obtain a better training speed, the load balance between each computing node is a stringent requirement. In practical training, the current performance of the node is affected by the external environment and other tasks in the computing nodes. Hence, the current performance of the node is random, eventually leading to the performance of the synchronous parallel method being dragged down by the slowest computing nodes. Because the model update is completed at the same time, it will take up a lot of memory and generate a data communication storm, making a higher requirement for computing nodes.

In asynchronous parallel computing method, each node directly updated model parameters immediately after the calculation completion without waiting for other nodes to complete their iteration. Asynchronous parallel method does not need to consider the performance of computing nodes; it only needs to focus on the calculation of the node itself. Asynchronous parallel method in N nodes can get almost N times the speedup. But, in the asynchronous parallel mode, the training of the model parameters is not the newest, making the training process easy to fall into local optimal solution, resulting in poor network training convergence. So the asynchronous parallel method cannot be used to model training in practice.

As Figure 5 shows, in the process of model training, the parameter server is used to complete the update process after each iteration. Parameter servers can also be responsible for

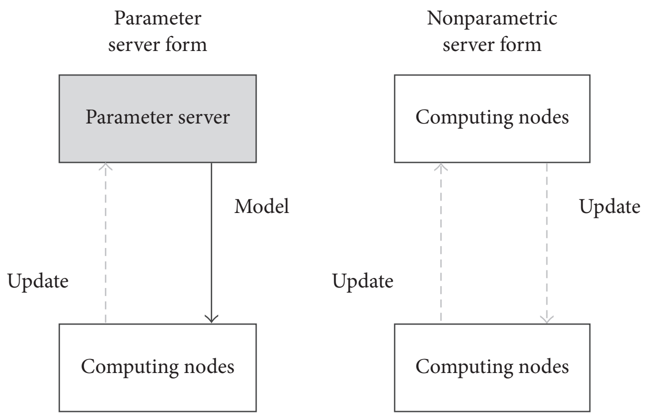

Figure 5: Comparison between parameter server parallel and nonparametric server parallel.

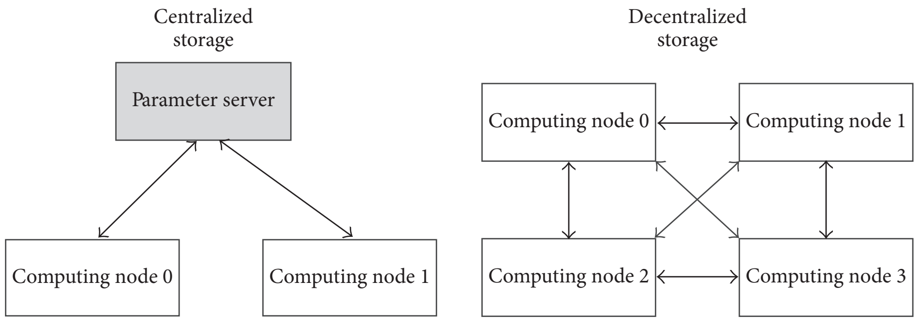

Figure 6: Communication topology comparison between centralized storage and decentralized storage.

sending training data and test data. When the parameter server is used, each computing node does not communicate with the other; they communicate with the parameter server.

3.2. Considerations for Parallel. From the previous discussion of data parallel and the model parallel training methods, we summarize some factors which should be considered in the implementation of the heterogeneous system.

3.2.1. Parallel Method. In the heterogeneous system, data parallel was chosen to train the model in general. The reasons are as follows: (1) in data parallel, each node has the same training method for the network model. But, in the model parallel, each node does not have the same training method for the network model (because each node is training different part of the model). So it is more difficult to implement the model parallel compared to the data parallel. (2) Convolutional neural network has the structure dependence; the model parameter matrix update order will affect the time of model training. Moreover, because the network has fault-tolerant ability, it will recover from error which was caused by the unreasonable task division. When the error accumulated to a certain degree, the network model may get a local optimal solution. In addition, the division of the model is lack of theoretical guidance.

3.2.2. Maximizing Effective Training Time: Delay Synchronous Parallel. In the distributed and synchronous parallel environment, the calculation nodes have to wait to synchronize the parameters after each iteration in the training process. In order to reduce the waiting time of computing nodes, the load balancing between each node is required. However, in the actual situation, the performance of the machine is often affected by a lot of external factors, such as temperature, and they is random factors. In the case of asynchronous parallel environment, the computing of each node does not interfere with that of the other. The faster nodes do not have to wait for the slower nodes, and they can directly update network model. This training mode is equivalent to shielding the impact of different hardware computing capabilities. Based on the characteristics of synchronous parallel and asynchronous parallel, the delay synchronous parallel is proposed. See Section 4 for details.

3.2.3. Parameter Storage and Communication Topology. The choice of parameter storage will affect the communication topology, and the topology of the communication will influence the weight parameters communication between each node. Depending on whether or not the parameter server is used, the storage of the parameters can be divided into centralized storage and decentralized storage. The communication topology is shown in Figure 6. Centralized storage can use the "master and slave" mode for implementation, the

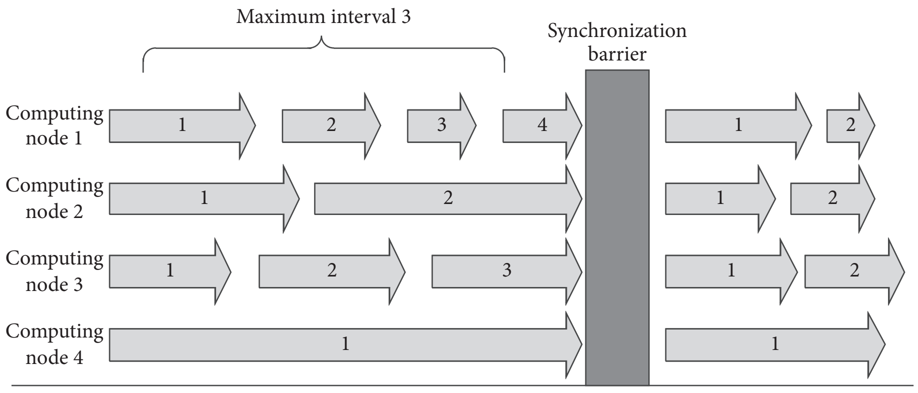

Figure 7: Training method of delay synchronization parallel.

master node acts as a parameter server and a data transmitter, and the slave node is used for training the model. Parameter server contains a complete model and sends the model to each slave node before training. The weight update will be sent to the parameter server after the slave completes one iteration. Parameter server will update the network model as soon as it gets all the weight updates from slave nodes. Then the master node will send the latest network model to slave node for training. Because the centralized storage only needed to send the weight update data to the master node from the slave node, there is no exchange of data between slave nodes, which greatly reduces the cost of communication. Decentralized storage can usually be implemented by the end-to-end topology, and each endpoint is a calculation node. Each node will send the weight update data to others after one iteration is complete; this will cause too much communication during the whole process of distributed training. In order to reduce the communication overhead in the training process, the centralized storage is used to implement the distributed training.

In the specific implementation, centralized storage is more challenging than decentralized storage. Firstly, the master node in the centralized storage has high performance requirements to coordinate the whole training process. Secondly, concurrency control between nodes should be considered in centralized storage. Finally, we need to consider the storage mode of the training data to reduce communication loss.

4. Delay Synchronization Parallel

From the section entitled "Maximizing Effective Training Time: Delay Synchronous Parallel," we know the purpose of the delay synchronization parallel method is to ensure that the model is not trapped in local optimal solution, and the effective training time of the nodes is maximized. The synchronous parallelism can ensure that the training process does not fall into local optimal solution, and the asynchronous approach can make the effective training time maximized. So the delay synchronization parallel approach combines the advantages of synchronous and asynchronous parallelism. In this section, we first describe the training method of the delay synchronization, then introduce the training characteristics of this method, and finally show the conditions the node should have to achieve for this method.

4.1. Training Method. For a network model training, assuming that there are P calculation nodes, after the end of each iteration, training faster computing nodes need to wait for the slower training nodes to finish in synchronous update method. While using delay synchronization in the parallel way, faster computing node does not need to stop to wait for the slower computing nodes, and faster computing node can directly update the network model parameters and then continue to the next iterative training. As Figure 7 shows, when the slowest computing node is slower by s (s value can be set by the user) times of iterations than the fastest node, the fastest node is forced to wait until all the computing nodes complete their one iteration, and then a training model parameter update between all computing nodes is completed.

4.2. Training Characteristics. In order to reduce the influence of communication process, we apply the server parameter to implement delay synchronous parallel method; it means parameters are stored and updated by the centralized node and all the computing nodes only need communication with the server parameter. Moreover, the update of the model also depends on the parameters server for completion.

Synchronous parallel computing requires the use of the synchronization barrier to force all the computing nodes to do a parameter update after completing an iteration. In the training method of synchronous parallel, in order to get a better training speed, the load balance between the nodes is strict. But, in the asynchronous parallel, each computing node does not wait for the others after they complete one iteration, and the completed nodes will directly update their training model through parameter server. Asynchronous parallel method does not need to consider the performance of computing node; it only needs to focus on the calculation of the computing nodes. It can be said that the asynchronous parallel way can shield a series of problems caused by the uneven performance between calculation nodes.

Delay synchronization parallel contains the characteristics of synchronous parallel and asynchronous parallel;

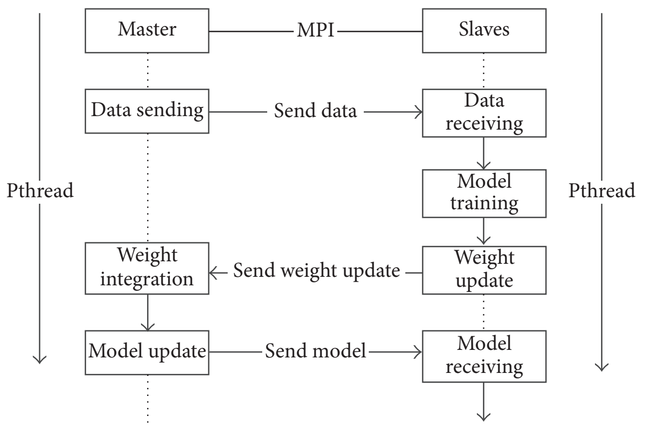

FIGURE 8: Training process of "master and slave" model.

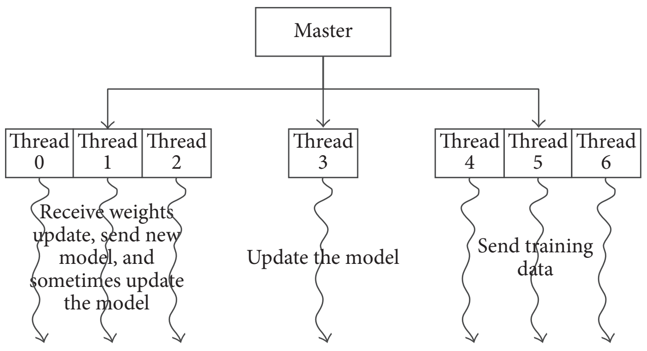

FIGURE 9: Design of the main process.

we adopt the asynchronous training approach in the early training phase, when the difference between the fastest and the slowest nodes is s iterations, using the synchronous barrier to mandate all nodes to do a model update. All nodes will continue to do asynchronous training until next s iteration emerged. In the process of asynchronous training, the training method can shield the effect caused by different node performance, and the synchronous barrier can avoid the local optimal solution. Therefore, the parallel method of delay synchronization can get the same speedup as the computation node as well as a better training result.

4.3. Implementation Conditions. In order to implement the delay synchronization, the algorithm must meet the following conditions: (1) the fastest node and the slowest node work even if the interval is less than the number of s. (2) Each computing node has a training model, with noninterference between others. (3) Third one is using a parameter server to update the model parameters and undertake date distribution function.

5. Experimental Results and Discussions

In this section, we verify the effectiveness, performance, and scalability of the delay synchronization parallel strategy and present the influence of different maximum interval s on the training results. The data set we used is the MNIST handwritten digital font data set which includes 60000 pictures' training data and 10000 pictures' test data. We use the classic LeNet-5 model for training, and the model includes one input layer, one output layer, three convolutional layers, two pooling layers and a fully connected layer. The batch size of the training model is 64, and the maximum number of iterations is 10000.

5.1. Experimental Framework. The training environment is based on the MPI master-slave model of distributed data parallel, the specific training process and the detailed design of the master and slave nodes are shown in Figures 8, 9, and 10, respectively.

The main process consists of three thread groups: the data distribution thread group, the parameter communication thread group, and the model update thread. Data distribution thread group and parameters communication thread group have the same thread number which is the number of computing processes. The data distribution thread group is mainly used for distributing the training task and data to each computing process. The parameters communication thread group is mainly used for receiving the weight update data sent by the computing processes and sending the new model for computing processes after the model is updated. When the interval between the fastest node and the slowest node

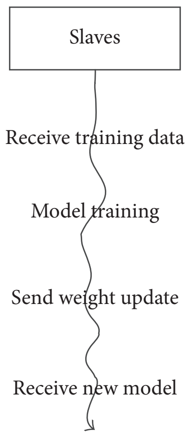

Figure 10: Design of the computing process.

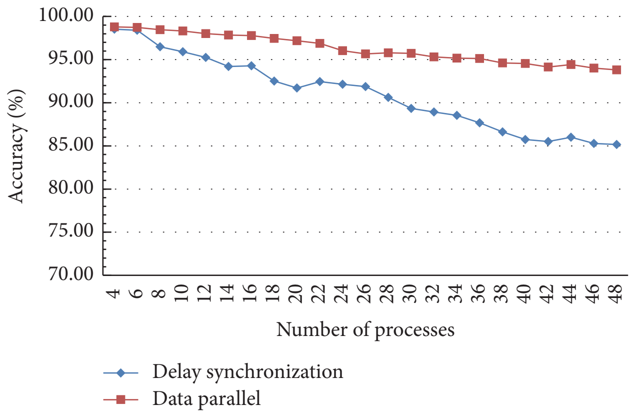

Figure 11: Delay synchronization parallel performance test: accuracy.

is less than the maximum interval number s, parameters communication thread group also used to update the model. Model is trained by the computing processes in a serialization manner.

5.2. Effectiveness and Performance. In this section, we will verify the effectiveness of the delay synchronization strategy and evaluate its performance through experiments. Hardware environment used in the experiments consists of two heterogeneous server nodes connected by Gigabit Ethernet. One node is configured with the 24 Intel Xeon E5-2620 V2 @2.10 GHz CPU and 128 G memory, and the operating system is Red Hat Enterprise Linux Server release 6.3. The other node is configured with the 32 Intel Xeon E5-2670 @2.6 GHz CPU and 32 G memory, and the operating system is Red Hat Enterprise Linux Server release 6.2. We compare the performance of traditional data parallel and delay synchronous parallel from aspects of time and accuracy. The specific experimental results are shown in Figures 11 and 12, where the maximum interval s is set to 3, and the recording time includes all the time from the distribution of the model to the time of testing test data.

From Figures 11 and 12, we can see that, in the traditional data parallel and delay synchronous parallel training methods, the accuracy rate is decreased with the increase of computing nodes, and delay synchronization parallel accuracy rate declines faster than traditional data parallel. Because of the communication cost, the training time is nonlinearly reduced with the increasing of the computing processes. When passing a certain computing process number, the time even increased. In the best case, the delay synchronization parallel strategy can get almost three times faster than the traditional data parallel. When the process number is 10 and we add other unrelated process tasks in the training environment, the training time of data parallel method increased, but the delay synchronization scheme was not affected. It can be seen that the delay synchronization parallel strategy reduces the impact of the hardware environment; that is, the training time is not easy to be dragged by a short board of computing process.

5.3. Scalability. In order to verify the scalability of the delay synchronization parallel strategy, the experiment environment consists of four heterogeneous servers connected with the Gigabit Ethernet. One node is configured with the 24 Intel Xeon E5-2620 V2 @2.10 GHz CPU and 128 G memory, and the operating system is Red Hat Enterprise Linux Server release 6.3. Another three nodes are configured with the 32

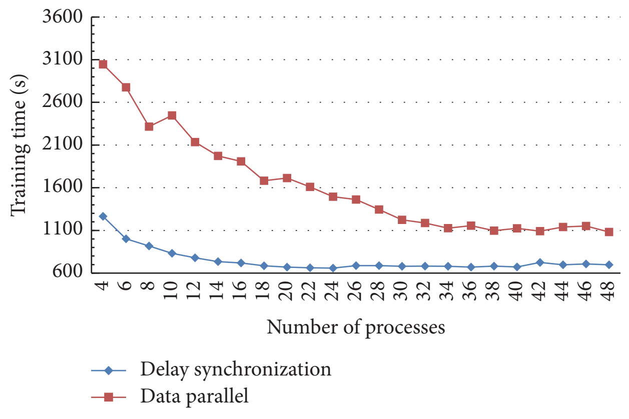

Figure 12: Delay synchronization parallel performance test: training time.

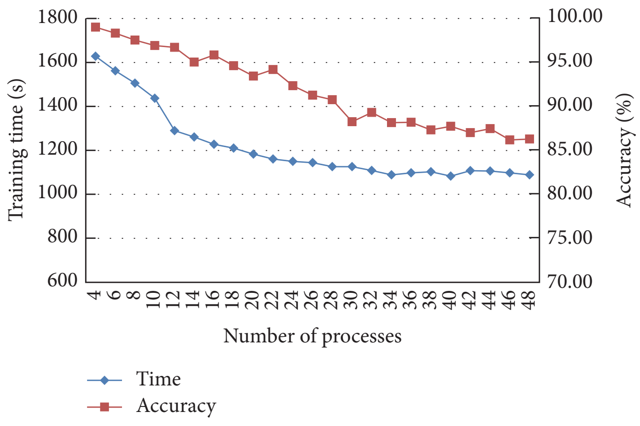

Figure 13: Verification of scalability.

Intel Xeon E5-2670 @2.6 GHz CPU and 32 G memory, and the operating system is Red Hat Enterprise Linux Server release 6.2. The evaluation metrics of the experiment are the same as the section entitled "Effectiveness and Performance," which are time and accuracy. In order to balance the iterative tasks on each node, the training processes are distributed on the four nodes, the maximum number of intervals in delay synchronization parallel training is still three, and the specific experimental results are in Figure 13.

From Figure 13, for accuracy, the training of four nodes has almost the same effect as described in the section entitled "Effectiveness and Performance," and it presents a downward trend with the increment of computing processes. From the perspective of training time, compared to experiments in section entitled "Effectiveness and Performance," the overall training time is increased as a result of more communication overhead. Same as section entitled "Effectiveness and Performance," the training time is nonlinearly decreased with the increasing of the computing processes. Time may increase after the number of computing processes passes a certain value. The experimental results show that the delay synchronization strategy has good scalability, but this kind of good scalability is inevitable to involve a certain amount of communication cost.

5.4. The Maximum Interval Influence on Model Training Process. Delay synchronous parallel strategy is a combination of the synchronous parallel strategy and asynchronous parallel strategy. When the fastest node is s (the maximum interval) iteration(s) faster than the slowest node, the strategy uses the mandatory synchronization barrier to prevent the model divergence from falling into local optimal solution. This section verifies the effects of different maximum interval s on the model training. The experimental environment is the same as described in section entitled "Effectiveness and Performance," which is the two heterogeneous servers connected by Gigabit Ethernet. The time and the effect of the training process were tested with the maximum interval of 1, 2, and 3, and the results are presented in Figures 14 and 15.

From Figures 14 and 15, we can see that, with the increasing of the maximum interval, model training time decreases, but the accuracy of the model was effected dramatically with the computing process increase. Hence, considering the influence of both time and accuracy, we prefer to select smaller interval.

Based on the experimental results, we can see that the proposed delay synchronization parallel strategy indeed has a better performance.

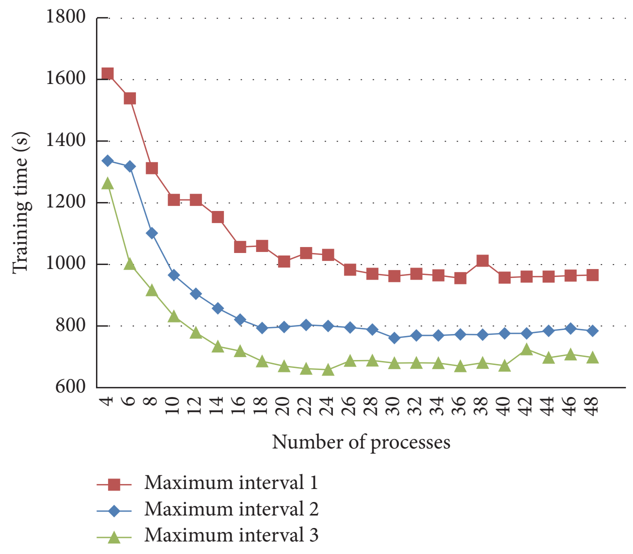

Figure 14: The maximum interval influence on model training process: training time.

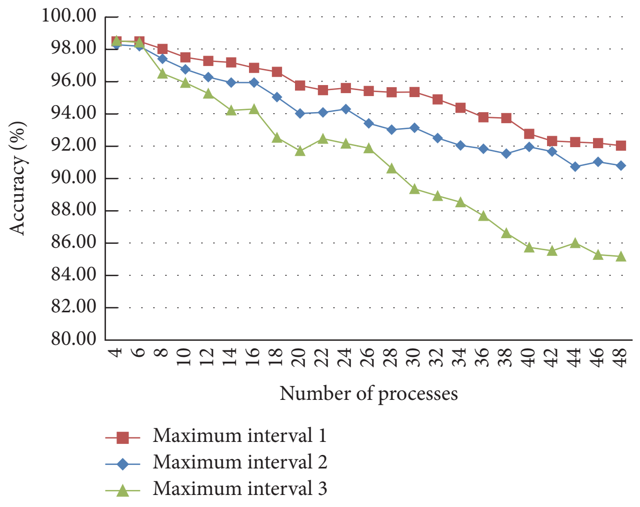

Figure 15: The maximum interval influence on model training process: accuracy.

6. Conclusion

Mobile device is an integral part of our daily life; business market can be more intelligent to automatically provide services based on users' location and context environment. To learn users' habits and patterns, machine learning strategies, such as CNN, are applied. However, the existing parallel implementation cannot fully use the parallel computing architecture resources, making heterogeneous computing resources wasted, especially in the mobile information system field. To this end, this paper proposes a convolutional neural network parallel strategy based on the heterogeneous clusters named delay synchronization parallel strategy. The strategy leverages the benefits of both synchronous parallel and asynchronous parallel approaches. It can achieve almost 3 times the speedup compared to the data parallel. The scalability of the strategy can make convolution neural network framework more adaptive to different heterogeneous system environments.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Jilin Zhang and Junfeng Xiao contributed equally to this work and should be considered co-first authors.

Acknowledgments

This work is partly supported by the National Natural Science Foundation of China under Grants no. 61672200 and no. 61572163; the National High Technology Research and Development Program of China under Grant no. 2015AA01A303; the Zhejiang Natural Science Funds under Grants no. LY16F020018 and no. LY17F020029; the Key Laboratory of Complex Systems Modeling and Simulation program of the Ministry of Education and the Chinese Postdoctoral Science Foundation no. 2013M541780 and no. 2013M540492; Hangzhou Dianzi University construction project of graduate enterprise innovation practice base no. SJJD2014005; Research project of Zhejiang Provincial Department of Education under Grant no. Y201016492.

References

[1] P. Racherla, C. Furner, and J. Babb, "Conceptualizing the implications of mobile app usage and stickiness: a research agenda," 2012.

[2] M. Gençer, G. Bilgin, Ö. Zan, and T. Voyvodaoglu, "A new framework for increasing user engagement in mobile applications using machine learning techniques," in *Proceedings of the International Conference on Design, User Experience, and Usability*, pp. 651–659, Springer, Las Vegas, Nev, USA, 2013.

[3] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, pp. 396–404, 1990.

[4] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[5] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 3642–3649, Providence, RI, USA, June 2012.

[6] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 2012.

[7] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1610–1623, 2010.

[8] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[9] S. Naji, R. Zainuddin, S. A. Kareem, and H. A. Jalab, "Detecting faces in colored images using multi-skin color models and neural network with texture analysis," *Malaysian Journal of Computer Science*, vol. 26, no. 2, pp. 101–123, 2013.

[10] N. Rajput, P. Jain, and S. Shrivastava, "Face detection using HMM-SVM method," in *Advances in Computer Science, Engineering & Applications*, pp. 835–842, Springer, Berlin, Germany, 2012.

[11] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*, pp. 3476–3483, IEEE, Portland, Ore, USA, June 2013.

[12] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. Von Deneen, and P. Shi, "An algorithm for license plate recognition applied to intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 830–845, 2011.

[13] D. V. Phạm, "Online handwriting recognition using multi convolution neural networks," in *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 310–319, Springer, Hanoi, Vietnam, 2012.

[14] S. S. Ahranjany, F. Razzazi, and M. H. Ghassemian, "A very high accuracy handwritten character recognition system for Farsi/Arabic digits using convolutional neural networks," in *Proceedings of the IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA '10)*, pp. 1585–1592, IEEE, Changsha, China, September 2010.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, December 2012.

[16] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[17] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," https://arxiv.org/abs/1404.5997.

[18] Y. Jia, E. Shelhamer, J. Donahue et al., "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM Conference on Multimedia (MM '14)*, pp. 675–678, November 2014.

[19] J. Yin, X. Lu, C. Pu, Z. Wu, and H. Chen, "JTangCSB: a cloud service bus for cloud and enterprise application integration," *IEEE Internet Computing*, vol. 19, no. 1, pp. 35–43, 2015.

[20] D. Povey, A. Ghoshal, G. Boulianne et al., "The Kaldi speech recognition toolkit," in *Proceedings of the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (No. EPFL-CONF-192584)*, IEEE Signal Processing Society, Waikoloa, Hawaii, USA, 2011.

[21] Q. Ho, J. Cipar, H. Cui et al., "More effective distributed ML via a stale synchronous parallel parameter server," in *Advances in Neural Information Processing Systems*, pp. 1223–1231, 2013.

[22] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '10)*, pp. 257–260, IEEE, Paris, France, May-June 2010.

[23] R. Hemalatha, N. Santhiyakumari, and S. Suresh, "Implementation of medical image segmentation using Virtex FPGA kit," in *Proceedings of the 4th International Conference on Signal Processing and Communication Engineering Systems (SPACES '15)*, pp. 358–362, January 2015.

[24] J. G. Pandey, A. Karmakar, and S. Gurunarayanan, "Architectures and algorithms for image and video processing using FPGA-based platform," in *Proceedings of the 18th International Symposium on VLSI Design and Test (VDAT '14)*, July 2014.

[25] E. C. Pedrino, O. Morandin Jr., E. R. R. Kato, and V. O. Roda, "Intelligent FPGA based system for shape recognition," in *Proceedings of the 7th Southern Conference on Programmable Logic (SPL '11)*, pp. 197–202, IEEE, Córdoba, Spain, April 2011.

[26] A. Zawadzki and M. Gorgoń, "Automatically controlled pan-tilt smart camera with FPGA based image analysis system dedicated to real-time tracking of a moving object," *Journal of Systems Architecture*, vol. 61, no. 10, pp. 681–692, 2015.

[27] T. Nakamura, Y. Touma, H. Hagiwara, K. Asami, and M. Komori, "Scene recognition based on gradient feature for autonomous mobile robot and its FPGA implementation," in *Proceedings of the 4th International Conference on Informatics,*

Electronics and Vision (ICIEV '15), pp. 1–4, IEEE Computer Society, Kitakyushu, Japan, June 2015.

[28] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully FPGA-based sensorless control for synchronous AC drive using an extended Kalman filter," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 10, pp. 3908–3918, 2012.

[29] S. Narjess, T. Ramzi, and M. M. Faouzi, "Implementation of sensorless control of an induction motor on FPGA using Xilinx system generator," *Journal of Theoretical & Applied Information Technology*, vol. 92, no. 2, pp. 322–334, 2016.

[30] K. Yu, "Large-scale deep learning at Baidu," in *Proceedings of the the 22nd ACM International Conference*, pp. 2211–2212, ACM, San Francisco, Calif, USA, October 2013.

[31] A. Coates, "Deep learning with COTS HPC systems," in *Proceedings of the International Conference on Machine Learning (ICML '13)*, pp. 1337–1345, Atlanta, Ga, USA, 2013.

[32] O. Yadan, K. Adams, Y. Taigman, and M. A. Ranzato, "Multi-GPU training of convnets," https://arxiv.org/abs/1312.5853.

[33] R. Uetz and S. Behnke, "Large-scale object recognition with CUDA-accelerated hierarchical neural networks," in *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS '09)*, vol. 1, pp. 536–541, IEEE, Shanghai, China, November 2009.

[34] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 1237–1242, July 2011.

[35] Z. Liu, H. Li, and G. Miao, "MapReduce-based backpropagation neural network over large scale mobile data," in *Proceedings of the 6th International Conference on Natural Computation (ICNC '10)*, vol. 4, pp. 1726–1730, IEEE, Yantai, China, August 2010.

[36] Q. Wang, J. Zhao, D. Gong, Y. Shen, M. Li, and Y. Lei, "Parallelizing convolutional neural networks for action event recognition in surveillance videos," *International Journal of Parallel Programming*, 2016.

[37] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 8595–8598, Vancouver, Canada, May 2013.

[38] J. Dean, G. Corrado, R. Monga et al., "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, pp. 1223–1231, 2012.