Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 7309431, 9 pages
https://doi.org/10.1155/2019/7309431

Research Article

Hybrid Parallel FDTD Calculation Method Based on MPI for Electrically Large Objects

Qingwu Shi,[1,2] Bin Zou,[1] Lamei Zhang,[1] and Desheng Liu[2]

[1] *Department of Information Engineering, School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China*
[2] *College of Information Science & Electronic Technique, Jiamusi University, Jiamusi 15407, China*

Correspondence should be addressed to Bin Zou; zoubin@hit.edu.cn and Lamei Zhang; lmzhang@hit.edu.cn

Received 5 December 2018; Revised 22 April 2019; Accepted 14 May 2019; Published 23 June 2019

Guest Editor: Qiang Ye

At present, the Internet of Things (IoT) has attracted more and more researchers' attention. Electromagnetic scattering calculation usually has the characteristics of large-scale calculation, high space-time complexity, and high precision requirement. For the background and objectives of complex environment, it is difficult for a single computer to achieve large-scale electromagnetic scattering calculation and to obtain corresponding large data. Therefore, we use Finite-Difference Time-Domain (FDTD) combined with Internet of Things, cloud computing, and other technologies to solve the above problems. In this paper, we focus on the FDTD method and use it to simulate electromagnetic scattering of electrically large objects. FDTD method has natural parallelism. A computing network cluster based on MPI is constructed. POSIX (Portable Operating System Interface of UNIX) multithreading technology is conducive to enhancing the computing power of multicore CPU and to realize multiprocessor multithreading hybrid parallel FDTD. For two-dimension CPU and memory resources, the Dominant Resource Fairness (DRF) algorithm is used to achieve load balancing scheduling, which guarantees the computing performance. The experimental results show that the hybrid parallel FDTD algorithm combined with load balancing scheduling can solve the problem of low computational efficiency and improve the success rate of task execution.

1. Introduction

The rapid growth of the IoT causes a sharp growth of data. Enormous amounts of networking sensors are continuously collecting and transmitting data to be stored and processed in the cloud [1, 2]. Such data can be remote sensing data, geographical data, and astronomical data. Radar is one type of the important remote sensing data, which plays a significant role in the society. Radar raw echo data must meet specific conditions on radar system design, imaging processing algorithm, geometric distortion correction, and other occasions. If these data are obtained through flight radar carrier, the cost is too high. Therefore, it is an important solution to obtain the required echo data through signal simulation [3]. The main method of obtaining echo data depends on the development of computational electromagnetics.

The core problem of electromagnetic computing is to achieve high precision and high efficiency calculation and to ensure accuracy while reducing costs. Traditional distributed computing and parallel computing methods use accumulated CPU resources in local area networks to achieve high-performance collaborative computing. However, the electromagnetic computing tasks that need to be solved are becoming more and more complex. Exploring elastic computing based on Internet of Things and cloud computing technology has become an economical, feasible, and efficient solution [4, 5]. After evaluating the characteristics of computational tasks with predictive learning model, it can schedule computational resources and improve computational efficiency and success by using adaptive load balancing method [6–9].

The calculation accuracy and stability conditions restrict the discrete size of FDTD in time and space. When calculating electrically large objects, especially, the serial FDTD method cannot meet the requirements in engineering because of the long time and excessive memory requirements [10, 11]. Therefore, expanding the calculation scale and reducing the calculation time becomes the key to scattering calculation and simulation.

Aiming to solve the above problems, our novelties in this paper are as follows:

(1) POSIX multithreading technology is utilized to improve the computing power of multicore CPU.

(2) The DRF algorithm is used to achieve load balancing scheduling; this can guarantee the computing performance.

(3) We design a novel calculation and exchange strategy in the same time to save computing time.

Based on the characteristics of FDTD suitable for parallel computing, a parallel FDTD method based on MPI (MPI: Message Passing Interface) and POSIX (POSIX: Portable Operating System Interface of UNIX) is proposed. Taking the resource called and load balancing allocation in the COW (COW: Cluster of Workstation) into consideration, the DRF (DRF: Domain Resource Fairness) algorithm is adopted to achieve load balancing scheduling for electromagnetic computing tasks in terms of CPU and memory resources.

2. The Basic Principle of Parallel FDTD Algorithm

2.1. Maxwell's Equations. The FDTD algorithm was proposed by Yee in 1966. The starting point of FDTD algorithm is that the curl equation in Maxwell equation is differentiated and the electromagnetic field is discretized in space and time. Therefore, when deriving the FDTD equation, only two curl equations [12] need to be considered, namely,

$$\nabla \times \boldsymbol{E} = \frac{\partial \boldsymbol{D}}{\partial t} + \boldsymbol{J} \tag{1}$$

$$\nabla \times \boldsymbol{E} = \frac{\partial \boldsymbol{B}}{\partial t} - \boldsymbol{J}_m \tag{2}$$

In Cartesian coordinates, (1) and (2) can be written as

$$\begin{aligned} \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} &= \varepsilon \frac{\partial E_x}{\partial t} + \sigma E_x \\ \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} &= \varepsilon \frac{\partial E_y}{\partial t} + \sigma E_y \\ \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} &= \varepsilon \frac{\partial E_z}{\partial t} + \sigma E_z \end{aligned} \tag{3}$$

And

$$\begin{aligned} \frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} &= -\mu \frac{\partial E_x}{\partial t} - \sigma_m H_x \\ \frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} &= -\mu \frac{\partial E_y}{\partial t} - \sigma_m H_y \\ \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} &= -\mu \frac{\partial E_z}{\partial t} - \sigma_m H_z \end{aligned} \tag{4}$$

The spatial arrangement of the discrete electric and magnetic fields in the FDTD is shown in Figure 1.

2.2. FDTD Iteration Formula. In Cartesian coordinates, (3) and (4) are deduced. Taking (3) as an example, the three-dimensional FDTD iteration formulas are obtained.

$$\begin{aligned} E_x^{n+1}\left(i+\frac{1}{2}, j, k\right) &= CA(m) \bullet E_x^{n}\left(i+\frac{1}{2}, j, k\right) \\ &+ CB(m) \bullet \left[\frac{H_z^{n+1/2}(i+1/2, j+1/2, k) - H_z^{n+1/2}(i+1/2, j-1/2, k)}{\Delta y} - \frac{H_y^{n+1/2}(i+1/2, j, k+1/2) - H_y^{n+1/2}(i+1/2, j, k-1/2)}{\Delta z}\right] \end{aligned} \tag{5}$$

In (5), the coefficients are related to the medium parameters and the discrete parameters. It can be seen from Figure 1 and (5) that each magnetic field component is surrounded by four electric field components. Similarly, each electric field component is surrounded by four magnetic field components, and the electric field and the magnetic field are alternately sampled in time series, sampling interval. It is half a time step away from each other so that it can be solved iteratively in time [12].

2.3. Parallelism of FDTD Method. The electric/magnetic field values of each grid are only correlated with the magnetic /electric field values of the adjacent grid. Therefore, a complete computational domain can be divided into several subdomains to exchange data of tangential fields at the boundary of adjacent subdomains. There is no correlation between the grid points in the subdomain and other subdomains, and each subdomain can execute independently.

Firstly, the electric field component is calculated. As shown in Figure 2, at the N time step, the four magnetic fields needed to calculate E_z in subregion 1 are within the region 1, so the calculation of E_z can be performed correctly without any additional operation. However, for E_z located on the boundary of region 1, the four magnetic fields H_y needed are located in subregion 2. So, at the $N-1/2$ time step, H_y must be passed from subregion 2 to subregion 1 to sure that E_z on the boundary of region 1 can be iterated correctly.

Table 1: MPI Function and Description.

Function Name	Description
MPI_Init( )	Start MPI computing environment
MPI_Comm_size( )	Number of parallel processes
MPI_Comm_rank()	Self-process identification number
MPI_Send( )	Send a message to the numbered process
MPI_Recv( )	Receive messages from a numbered process
MPI_Finalize ( )	End MPI running environment

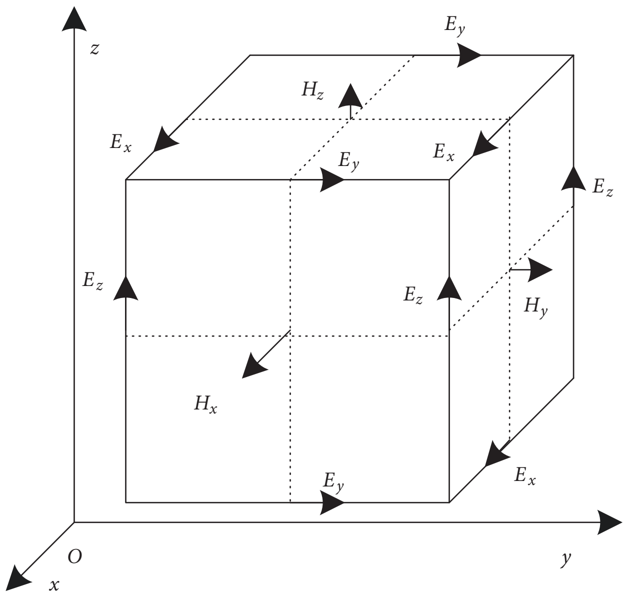


Figure 1: Discrete Yee Cellular of FDTD.

Similarly, at the $N+1/2$ time step to calculate the magnetic field component H_y on the boundary of subregion 2, it needs the electric field component E_z on the boundary of subregion 1 of the N time step. So, when E_z is computed, E_z must be passed from subregion 1 to subregion 2 to ensure the correct iteration of H_y. It is through the alternating transmission of electric and magnetic fields in time that the correct iteration of field components on the boundary of each subregion is ensured, and the parallelization of the algorithm is realized.

The field value exchange process based on domain decomposition parallel algorithm has the typical characteristics of message transmission, so it is very convenient to implement network parallel operation in the cluster of microcomputers by means of MPI communication function.

3. MPI Function Libraries and Local Area Network Configuration

3.1. Common Library Functions and Communication Modes of MPI. MPI is one of the standards for message passing parallel programming. MPI provides a library that can interface with C language [13]. In parallel computing, there are six common used functions, including starting and ending the MPI environment, identifying process, and sending and receiving messages. In the MPI program, each node has a unique process identification number. The master node allocates computing tasks according to the process identification number. Different processes, that is, nodes, perform different tasks in parallel, and each node uses the MPI_Comm_rank() function. It obtains its own process identification number and runs this process. At the same time, exchanged data between each process is required to realize parallel computing of the program [14]. The detailed explanation of function is shown in Table 1.

MPI provides two communication modes: blocking communication mode and nonblocking communication mode.

Blocking Send. The process is allowed to continue executing the next statement only after it is determined that the message has been sent to the message buffer.

Block Receive. Unless the message is received accurately in the message buffer, the process terminates the pending state and continues to execute the next instruction.

For the blocking mode, if the frequency of communication operation calls and the time of calls are too high or too long, the computing unit will be waiting for interruption long time, and the computing resources will not be fully utilized, thus reducing the calculation efficiency.

Nonblocking Send. It sends the primitive to inform the system that the message has been in the message buffers and then return. The sending process can continue to perform subsequent work without waiting for the system to actually send the message.

Nonblocking Receive. The primitive will be received and returned regardless of whether there is a notification of sending the primitive in the message buffer.

In nonblocking message transmission, while a message is sent or received accurately, the system will use interruption signal to inform the sender or receiver. Before that, the system can periodically query, suspend the process, or execute other instructions, achieving overlap between computation and communication, thereby, improving the computational efficiency of the system. However, using nonblocking messaging will make program debugging difficult.

3.2. Local Area Network Configuration and Structure. MPI-based parallel computing needs to build a computing network. The typical parallel computing structure of cluster is

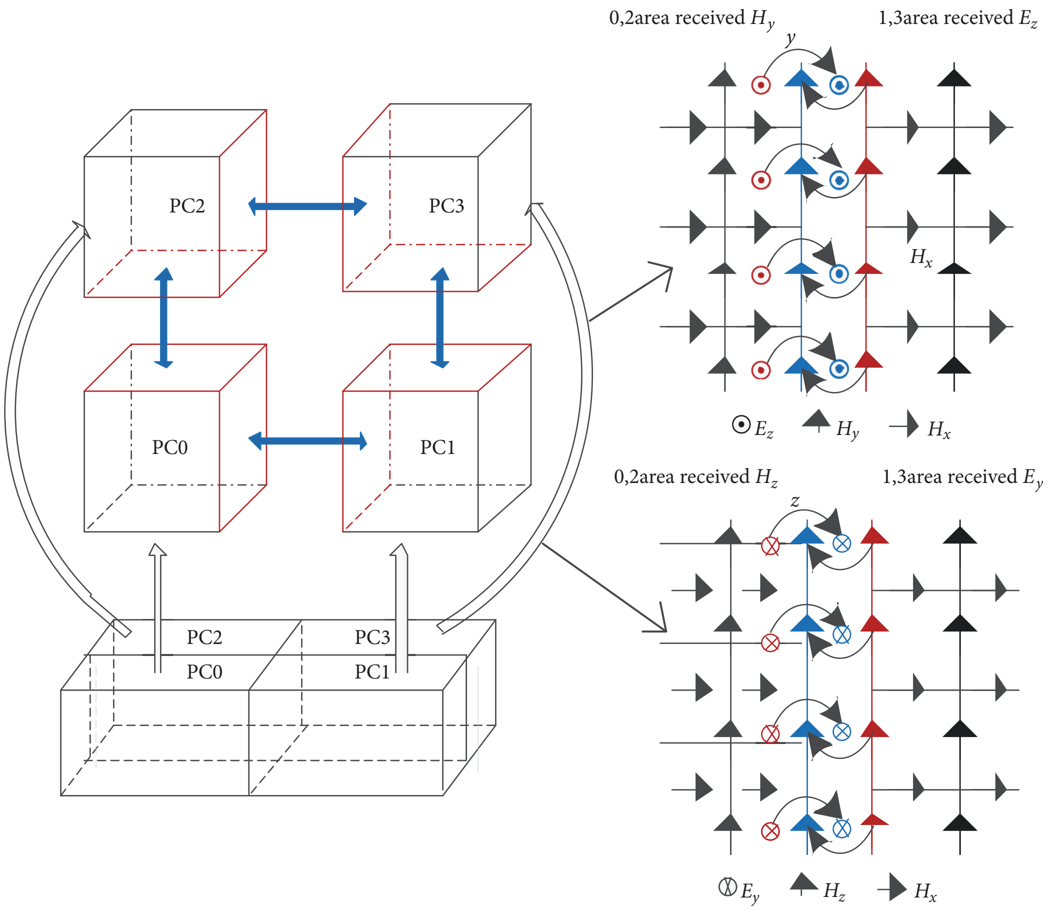


Figure 2: Parallel FDTD computational sketch of two-dimensional domain decomposition.

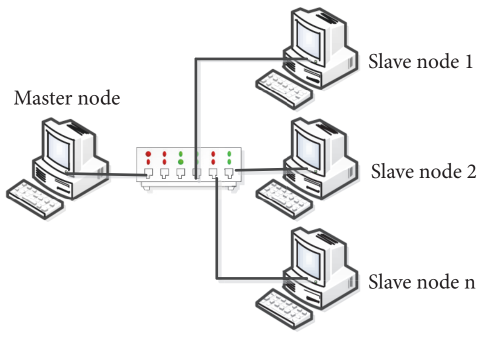


Figure 3: Network topology for parallel computing.

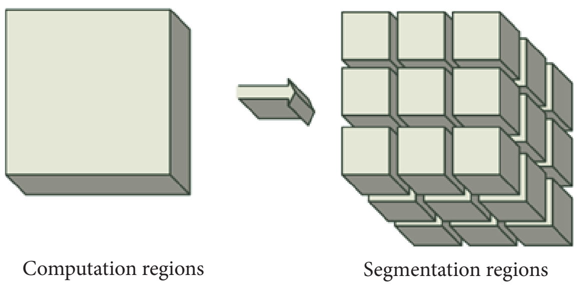


Figure 4: Divided according to the three-dimensional mode.

usually used to build MPI computing network, as shown in Figure 3.

4. Several Key Problems in Parallel FDTD Programming

4.1. Computational Region Segmentation of Parallel FDTD. Before parallel FDTD computation, it is necessary to divide the computational region of electromagnetic field into several computational subregions according to certain topological rules [15]. Each computing node in the cluster calculates one or several subregions. The electromagnetic data on the boundary surface is exchanged amongst nodes. Finally the master node collects these electromagnetic fields and saves them in the data file. Generally, spatial topological structures can be divided into one-dimensional, two-dimensional, and three-dimensional structures. In this experiment, the computational area is divided into three-dimensional, as shown in Figure 4.

4.2. Load Balancing amongst Nodes. Load balancing is the main indicator to measure the performance of a cluster

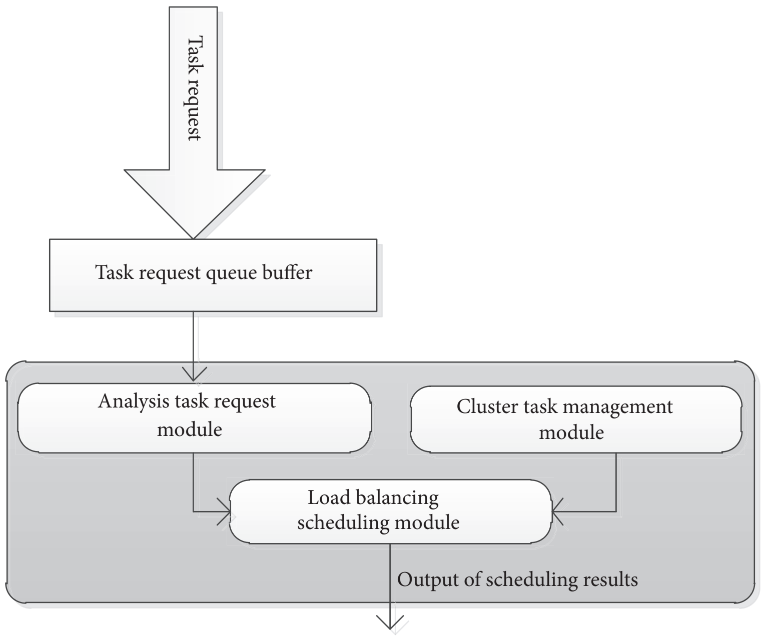

Task request
Task request queue buffer
Analysis task request module
Cluster task management module
Load balancing scheduling module
Output of scheduling results

FIGURE 5: Resource scheduling principle diagram.

system, which refers to the allocation of computing tasks amongst nodes, so that each process can complete computing tasks at the same time, thus reducing the maximum running time of the process. For resource scheduling for large-scale electromagnetic computing tasks, achieving load balancing is more important. Load balancing amongst resources ensure that resources are not overloaded, thereby, improving resource performance, improving the successful execution rate of electromagnetic computing tasks, and improving the performance of high-reliability resource scheduling mechanisms. Load balancing can be implemented in parallel programming, which effectively reduces the running time of parallel programs and improves the speedup and program performance [16].

This paper mainly focuses on the two dimensions of CPU and memory resources. The load balancing DRF (Dominant Resource Fairness) algorithm is used to design the scheduling mechanism to achieve load balancing scheduling for electromagnetic computing tasks. The resource scheduling block diagram is shown in Figure 5.

According to the above principles and Figure 4, the cluster task management module collects the status information of electromagnetic computing platform resources, extracts information such as CPU utilization rate and memory utilization rate of each cluster resource, and reports it to the electromagnetic computing platform control centre which provides the corresponding resource acquisition interface. The task request processing module receives allocation request from the electromagnetic computing platform control centre. The dispatching request sent by the control centre parses and operates different types of computing tasks, generates corresponding sequence of task requests, and reports to the load balancing scheduling module. Based on the above, we can get the flow chart of the whole process as shown in Figure 6.

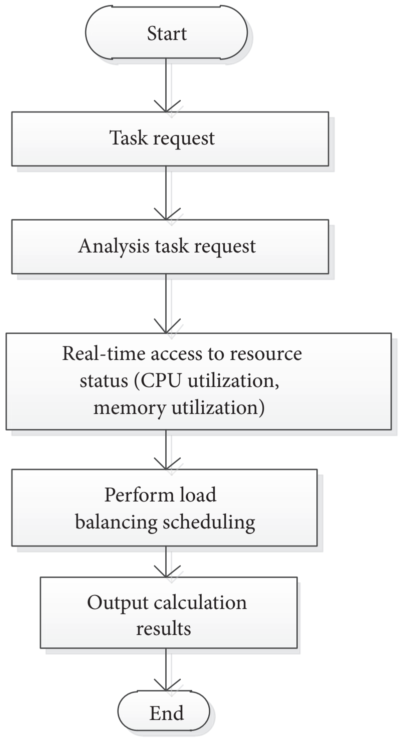

Start
Task request
Analysis task request
Real-time access to resource status (CPU utilization, memory utilization)
Perform load balancing scheduling
Output calculation results
End

FIGURE 6: Resource scheduler flow chart.

4.2.1. Building System Model. The experiment is conducted on Homogeneous Network Environments. In theory, it can be assumed that the total amount of computing tasks is T_{total}. The total number of nodes is N and each node's RAM is M in the cluster. The total resource is N processors and $N * M$

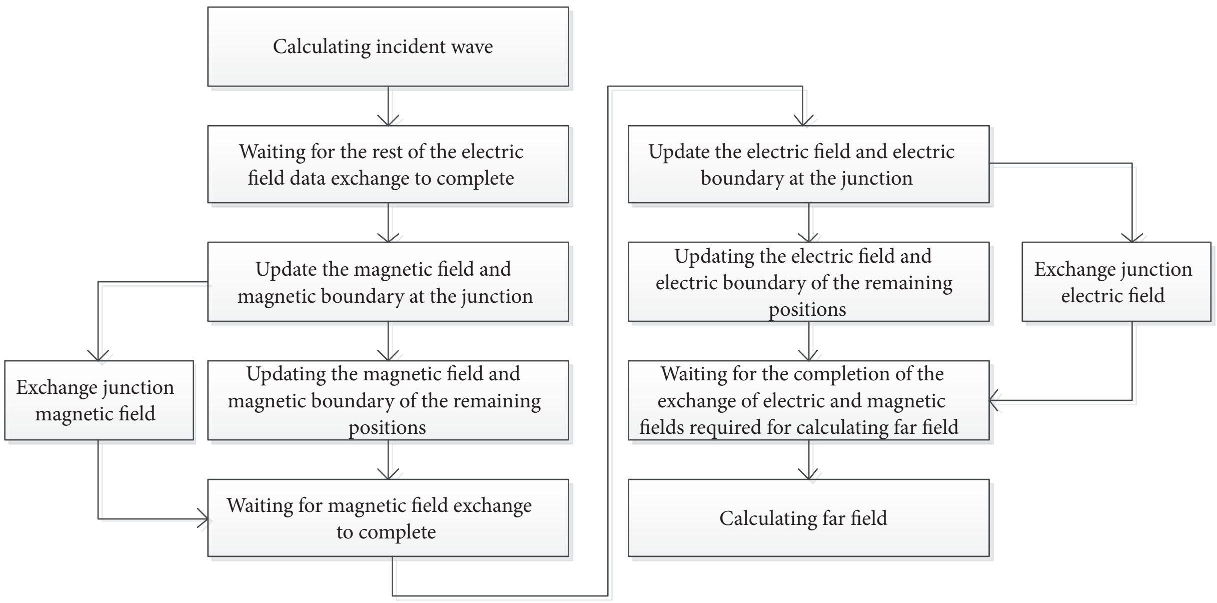

Calculating incident wave
Waiting for the rest of the electric field data exchange to complete
Update the magnetic field and magnetic boundary at the junction
Exchange junction magnetic field
Updating the magnetic field and magnetic boundary of the remaining positions
Waiting for magnetic field exchange to complete
Update the electric field and electric boundary at the junction
Updating the electric field and electric boundary of the remaining positions
Exchange junction electric field
Waiting for the completion of the exchange of electric and magnetic fields required for calculating far field
Calculating far field

Figure 7: Flow charts of programs computing and exchanging data simultaneously.

RAM number. Suppose they are all involved in computing. Then the amount of tasks assigned to each computing node is T_{N_i}:

$$T_{N_i} = \frac{T_{total}}{N} \tag{6}$$

In the heterogeneous network, resource of each node is different. Assuming that there are k nodes performing computational tasks, the amount of resources allocated to K nodes is $T_1, T_2, \ldots, T_k$, respectively. The total amount of computing tasks is T_{total}. The coefficients of the tasks allocated by each node are $\alpha_1, \alpha_2, \ldots, \alpha_K$. The processor number of each node is $(c_1, c_2, \cdots, c_k)$ and the RAM number of each node is $(m_1, m_2, \cdots, m_k)$. $S = \sum_{i=1}^{k} c_i * m_i$. The DRF algorithm can be used to give a fair strategy considering multiresource load balancing. The optimization equation is as follows:

$$\begin{gathered} \max \quad (T_1, T_2, \ldots, T_k) \\ \sum_{i=1}^{k} T_i \leq T_{total} \\ \sum_{i}^{k} \alpha_i T_i \leq S \\ \sum_{i=1}^{k} \alpha_i = 1 \end{gathered} \tag{7}$$

where max denotes the maximum resource allocation. The first inequality constraint is that it cannot exceed the total number of the tasks. The second inequality constraint is that it cannot exceed the total resource number. Equality constraint represents the main share of the balanced resources in two dimensions.

4.3. Hybrid Programming of Multithreads in Nodes. In order to make full use of the multicore of the CPU, multicore parallel computing is implemented by using shared memory inside a single node. In the shared memory programming mode, the CPU can access the data in the shared memory, and the cores of the CPU do not need to transfer data to each other. POSIX is a portable multithreaded library. POSIX-based Pthreads multithreading technology can implement internal parallelism of a single node. This standard provides a way to create and terminate new threads. The created thread executes a given function [17]. At the same time, the standard also provides three synchronization mechanisms for semaphores, condition variables, and mutexes, which make it easy to synchronize amongst threads.

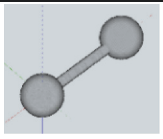
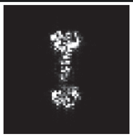
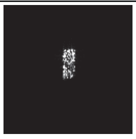
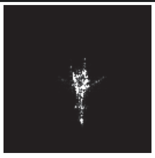
When designing the program, the FDTD algorithm only needs to exchange the electromagnetic data of a grid at the junction of nodes. If the blocking exchange mode is adopted, the time required for data exchange will become an additional overhead of the program. This reduces the efficiency of program execution. In order to improve the efficiency of program operation, the method of simultaneous calculation and data exchange is adopted in the design of program. The flow chart is shown in Figure 7.

In the hybrid parallel FDTD program, the two steps of "updating the magnetic field and magnetic boundary of the remaining positions" and "updating the electric field and electric boundary of the remaining positions" in Figure 6 is the maximum calculation. Because the electromagnetic field needed to be exchanged at the junction has only one grid layer, the computation amount is only 1% or even smaller than that of the whole electrically large object. Therefore, in order to avoid creating and destroying threads frequently, which results in a large amount of time overhead, the program does not destroy the threads of these two steps in the whole iteration process. When calculating these two steps, the

Table 2: Simulation platform and simulation parameters.

Simulation Platform	Simulation Parameters
HP-Z600 graphics workstations	incident wave: L-band
CPU: Inter (R) Xeon (R) CPU E5640@2.67GHz (2 processor)	centre frequency:1.3GHz
RAM:8GB	bandwidth: 1GHz
OS: Windows 7 Professional Edition	pulse width:1μs
	grid size:0.023m

Table 3: Model information and the SAR imaging results of models.

model	3-D information	(L∗W∗H)(cm)	Grid number	Visible grid number	SAR Imaging results
dumbbell		140∗40	376608	20899	
car		440∗159∗138	387791	22103	
Helicopter		500∗510∗320	662550	36991	

program uses the synchronization mechanism of Pthreads to perform multithreaded parallel computation. In the other steps, only one thread occupies the CPU, while the other threads wait for conditional variables.

4.4. Performance Evaluation of Parallel Computing. The performance of parallel computing is usually measured by speedup and efficiency. Suppose a serial program runs for T_s time. The running time of a parallel program on p processors is T_p. The definition of the speedup is [18]

$$S_p = \frac{T_s}{T_p} \tag{8}$$

The definition of efficiency is

$$E_p = \frac{S_p}{p} \tag{9}$$

Here, p is the number of processors in all nodes.

5. Experimental Simulation and Data Analysis

5.1. Experimental Hardware Platform. In order to verify the correctness and acceleration performance of the method, the simulation experiment uses several HP-Z600 graphics workstations to form a cluster through a Gigabit switch. Three models are calculated: dumbbell, car, and helicopter. The cluster configuration is shown in Table 2.

The computed model information and the SAR imaging results are shown in Table 3.

In the simulation process, one computer serial FDTD calculation is used as the comparison reference. Three computers and four computers are used for hybrid parallel computing. The data of the calculation are recorded, as shown in Table 4. ∗ denotes that it cannot finish the task.

5.2. Result Analysis. Assuming that the speedup of serial computing is 1 and through analysing Table 4, the conclusions can be as follows.

(1) When using one computer to calculate serial FDTD, the calculation time and speed of FDTD are only related to the main frequency of CPU and its calculation ability. At this time, the Windows operating system will retain some CPU resources, so the CPU utilization cannot reach 100%, and the actual CPU utilization can reach 90%.

(2) Under the same excitation source, the larger the size of the object and the more grids it gets, the more memory it needs. For the same purpose, parallel FDTD computing requires much less memory for each node in the cluster than serial FDTD computing, which reflects the superiority of parallel computing. Therefore, parallel computing is an effective method to solve the scattering problem of electrically large objects, which requires a lot of computing resources.

(3) The computational efficiency of the program (assuming that the efficiency of serial is 1) is related to the number of nodes in the system network. Experiments show that with the increase of the number of computing nodes, the computational efficiency will decrease correspondingly. The main reason why this phenomenon is appearing is that the CPU utilization of the intermediate nodes is usually low

Table 4: FDTD calculation and comparison of different models.

Calculation method	Objects	Time steps	time (s)	CPU usage rate (%)	one using memory (MB)	Speedup	Efficiency
	dumbbell		7245	90	1250	1	1
one, serial	Car		8902	90	1532	1	1
	helicopter		*	*	*	*	*
	dumbbell		3832	75~88	350~430	1.891	0.6303
three, parallel	car	26069	4705	78~90	430~530	1.892	0.6307
	helicopter		39842	68~72	280~300	1.785	0.5951
	dumbbell		2912	68~78	260~350	2.488	0.6220
four, parallel	car		3646	65~82	330~430	2.442	0.6105
	helicopter		25751	65~78	230~260	2.052	0.5131

after the increase of the number of computing nodes, and it also increases the additional consumption of time when the system exchanges data.

(4) In parallel FDTD computing, there is no limit on the number of nodes in the network, but the experiment shows that with the increase of the number of nodes, the computing time is shortened and the speedup is increased, which exactly reflects the advantages of parallel FDTD computing. In addition, it should be emphasized that the relationship between shortening the computing time and increasing the number of nodes is nonlinear. With the increasing of the number of nodes, the time needed for data exchanges in the network takes up more CPU computing time, and it can also explain the reason why the efficiency of the system decreases.

6. Conclusion

This paper mainly studies the hybrid parallel FDTD algorithm to solve the electromagnetic scattering calculation of electrically large objects. In the design of the program and algorithm, the rules of calculating region partition and load balancing algorithm are applied comprehensively. The program achieves parallel computing in cluster and multi-threads computing in single node. The experimental results and analysis conclude that hybrid parallel FDTD algorithm with load balancing can improve the computing speed and greatly shorten the computing time. Meanwhile, the parallel algorithm can also provide ideas and solutions for solving the shortage of computing resources. Compared with the serial FDTD algorithm, it has great advantages and engineering application value.

Data Availability

The data will be perfected, so it will be released in the future.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61401124, 61871158), in part by Scientific Research Foundation for the Returned Overseas Scholars of Heilongjiang Province (LC2018029), in part by Foundation for Returnees of Heilongjiang Province of China (NO: LC2017027), and in part by Jiamusi University Science and Technology Innovation Team Construction Project (Project Number: CXTDPY-2016-3).

References

[1] J. Gao, P. Li, and Z. Chen, "A canonical polyadic deep convolutional computation model for big data feature learning in Internet of Things," *Future Generation Computer Systems*, vol. 99, pp. 508–516, 2019.

[2] J. Gao, J. Li, and Y. Li, "Approximate event detection over multi-modal sensing data," *Journal of Combinatorial Optimization*, vol. 32, no. 4, pp. 1002–1016, 2016.

[3] Y. Shi, L. Li, and Y. Yu, "Simulation technique of the synthetic aperture radar," *Journal of Telemetry, Tracking and Command*, vol. 28, 2007.

[4] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A double deep q-learning model for energy-efficient edge scheduling," *IEEE Transactions on Services Computing*, 2018.

[5] Q. Zhang, L. T. Yang, Z. Chen, P. Li, and F. Bu, "An adaptive droupout deep computation model for industrial IoT big data learning with crowdsourcing to cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2330–2337, 2018.

[6] P. Li, Z. Chen, L. T. Yang et al., "Deep convolutional computation model for feature learning on big data in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 790–798, 2018.

[7] P. Li, Z. Chen, L. T. Yang, J. Gao, Q. Zhang, and J. Deen, "An incremental deep convolutional computation model for feature learning on industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1341–1349, 2018.

[8] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3170–3178, 2018.

[9] P. Li, Z. Chen, L. T. Yang, J. Gao, Q. Zhang, and M. J. Deen, "An improved stacked auto-encoder for network traffic flow classification," *IEEE Network*, vol. 32, no. 6, pp. 22–27, 2018.

[10] S. Jiang, Z. Lv, Y. Zhang et al., "Analysis of parallel performance of MPI based parallel FDTD on supercomputer," in *Proceedings of the IET International Radar Conference*, pp. 1–4, Xi'an, China, 2013.

[11] W. Yu, X. Yang, Y. Liu et al., "New development of parallel conformal FDTD method in computational electromagnetics engineering," *IEEE Antennas and Propagation Magazine*, vol. 53, no. 3, pp. 15–41, 2011.

[12] D. Ge and Y. Yan, *Finite-Difference Time-Domain Method for Electromagnetic Waves*, Xidian University Press, Xi'an, China, 3rd edition, 2011.

[13] Z. Duo, *High Performance Computing Parallel Programming Technology-MPI Parallel Programming*, Tsinghua University Press, Beijing, China, 2001.

[14] Z. Lu, J. Zhang, J. Shi et al., "Dynamic load balancing strategies in MPI parallel environment," *Computer Technology and Development*, vol. 20, no. 5, pp. 133–135, 2010.

[15] H. Li, *Research of Parallel Algorithm Based on Lan*, Harbin Institute of Technology, Harbin, China, 2011.

[16] L. Kezhong and L. Xiaohui, "Implementing load balance in MPI parallel program," *Microcomputer Information*, vol. 23, no. 5-3, pp. 226-227, 2007.

[17] G. Chen, *Parallel Computing-Structural Algorithmic Programming*, Higher Education Press, Beijing, China, 3rd edition, 2011.

[18] X. Pan, *Research on The Electromagnetic Scattering Calculation of Typical Targets in Time Domain and The Fast Algorithm*, Harbin Institute of Technology, Harbin, China, 2014.