INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 5, Number 1-2, Pages 57–65

AN OPTIMIZED CALCULATION METHOD FOR 3D-FDTD PARALLELIZATION

JIANG XIE, ZHONGHUA ZHOU, BING HE, HUIRAN ZHANG, AND DONGBO DAI

Abstract. The finite-difference time-domain (FDTD) has been commonly used for electromagnetic field simulation. However, parallelization of FDTD usually consumes large amount of memories and very long time, especially when the target domain is in 3D space. And the communication between calculating cores can be the bottleneck of the parallelize performance. In this paper we propose an optimized method named Crossing Calculation Method to accelerate FDTD simulation. Crossing Calculation Method can overlap the communication time which includes data exchanging time and synchronization time between neighborhood calculating nodes. The result shows that Crossing Calculation Method has linear speedup along with the increasing of calculating cores.

Key words. Parallel Computing, Finite-Difference Time-Domain, Crossing Calculation Method and Communication Time Overlapping.

1. Introduction

The finite-difference time-domain (FDTD) method [1] is a popular numerical computation method in electromagnetic field distribution analysis. Researchers have done a lot of work to complete this method. Although some people have promoted some absorbing boundary conditions theories [2, 3, 4] to reduce the simulation area, the simulation target area is still very large, especially when it comes to a three-dimensional problem. The size of grid grows cubically so that the use of memory is too large to simulate by a single computer and the computing time can be very long. It is very important to make the simulation fast. Many researchers have contributed a lot of useful work [6, 7, 8] on accelerating FDTD method.

Parallelization is usually used to make the simulation faster. The Message Passing Interface (MPI) [5] is a very common library for parallelization in CPU clusters. It is very convenient to use MPI to parallel FDTD simulation on CPU [9]. Recently GPU parallelization has become a hot issue in high performance calculation area, researches have promoted some GPU parallelization method [10, 11, 12] and at the same time some people have combined GPU and CPU [13, 14] to make the FDTD simulation much more faster.

Those methods can better improve the simulation speed of FDTD, however in the MPI based parallelization method [15, 16, 17, 18] when the calculation core number increases the speedup increases slower because the communication time which includes data exchanging time and synchronization time becomes a bottleneck. Sometimes it takes more time to communication than calculation. Figure 1 displays the speedup result of the classic parallelization method. Figure 2 shows the percentage of the communication time.

We promote an optimized calculation method named Crossing Calculation Method (CCM) to accelerate the parallelization. CCM overlaps the data exchanging time by controlling the calculation nodes calculating from different directions. The experiment results of our method in a CUP cluster with 160 Intel(R) Xeon(R) 2.53GHz

Received by the editors January 15, 2014 and, in revised form, March 21, 2014.
2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.

cores and 160 GB memory shows that CCM has linear speedup ratio along with the increasing of calculating cores.

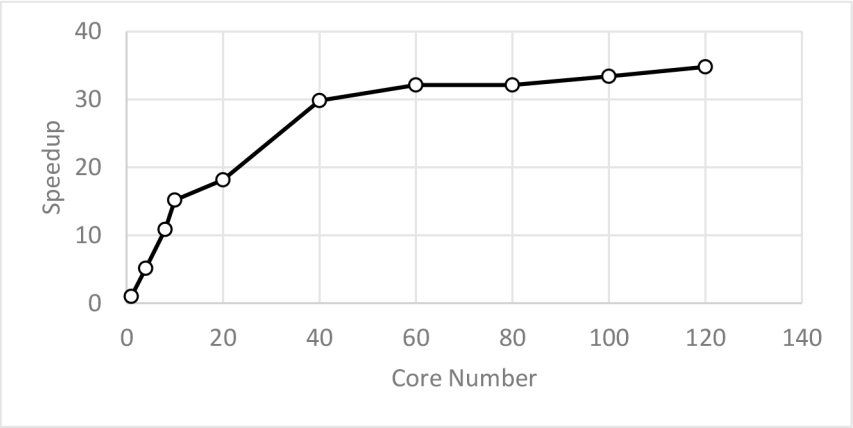

FIGURE 1. Speedup of classic FDTD parallelization method. In this figure, with the increasing of the calculation core number the increase rate of speedup declines. It mainly because the communication between cores becomes the bottleneck of the performance.

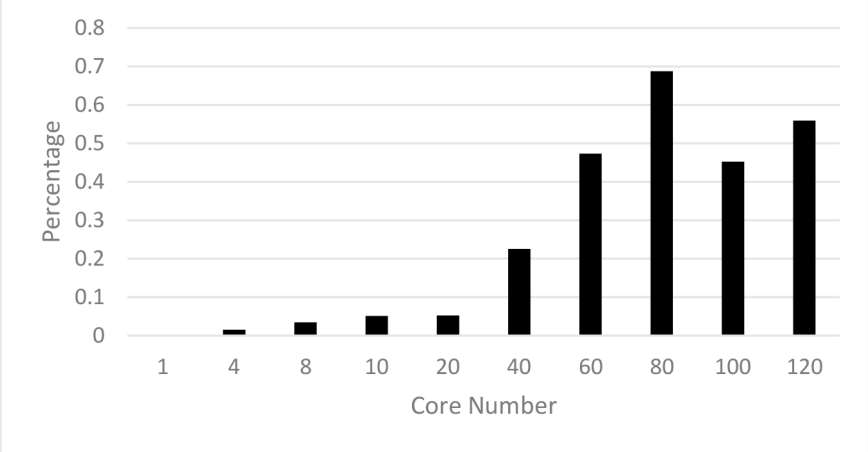

FIGURE 2. The percentage of the communication time of classic FDTD parallelization method. In this figure, the percentage of communication time increases as the core number increases and sometime it even takes more than 50%.

2. CCM

2.1. FDTD Parallelization. FDTD method is easy to parallelize because the whole calculation space region can be separated into some different small regions. The sub regions are relative independent because only the neighboring regions share some data. Figure 3 displays a basic architecture of the parallelization.

The classic FDTD parallelization procedure of each subdomain follows the pseudo code below.

```
For time=0 to max time
    From right to left
        Calculate electromagnetic field
    End
    If core index is odd
        Send the data to its neighbors
        Receive data from its neighbor
    Else
        Receive data from its neighbor
        Send the data to its neighbors
    End
    Synchronize until the others finish data exchanging
End
```

During the simulation, the areas start calculating at the same time and at the end of each time step every subdomain will share the data with their neighbors. And this will consume some time, in addition, the computer cores are not totally the same so that some extra work should be done t o make all the subdomains works in the same time step. The communication time T_{COM} includes data exchanging time T_{CX} and synchronization time T_{CX} (1). The total parallel time T_{PA} includes both communication time T_{COM} and calculating time T_{CAL} (2).

$$T_{COM} = T_{CX} + T_{CS}. \tag{1}$$

$$T_{PA} = T_{COM} + T_{CAL}. \tag{2}$$

When it comes to the serial FDTD algorithm, the total simulation time T_{SE} only includes the calculation part T_{CAL2} (3). Usually, the calculating time T_{CAL} and data exchanging time T_{CX} of each subdomain are always same due to the calculation areas have the same size and the shared areas are also same in size as it is showed in figure 3. So that T_{CAL} is approach to T_{CAL2}/N as the number of subdomains is N (4). The synchronization time T_{CS} is related to N. When N increases the synchronization between different cores becomes more complicated so that T_{CS} also increases (5). Then the speedup R of the parallel method can be generated by equation 6.

$$T_{SE} = T_{CAL2}. \tag{3}$$

$$T_{CAL} = T_{CAL2}/N. \tag{4}$$

$$T_{CS} = f(N). \tag{5}$$

$$R = T_{serial}/T_{parallel}. \tag{6}$$

With some simple mathematical derivation we can get the final result of speedup R from equation 7 in which R means speedup of the parallel method, N means the core number, T_{CX} means parallel communication time, T_{CS} means parallel synchronization time and T_{CAL} means parallel calculating time.

$$R = N/((T_{CX} + T_{CS})/T_{CAL} + 1). \tag{7}$$

To improve the speedup we can simply increase the calculation core number N, however this will also increase the denominator of equation 7. If we can reduce the communication part of equation 7 the speedup result can be better. Than we come out with our parallel method CCM.

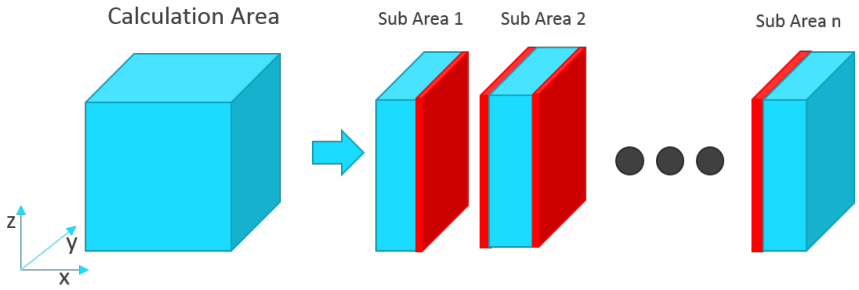

FIGURE 3. Basic architecture of FDTD parallelization method. The whole cubic simulation target is separated into several sub simulation areas and the number of areas is generally equal to the number of computer cores. Each subdomain shares the red part with its closest area. For example, subdomain 2 shares its left red part with subdomain 1 and sub shares its right red part with subdomain 3.

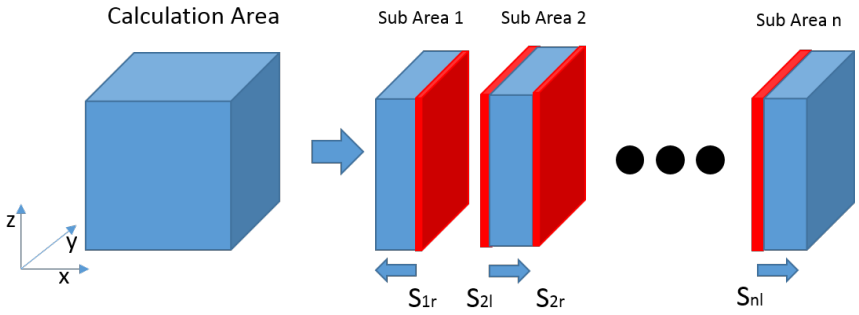

FIGURE 4. Architecture of CCM. The architecture of CCM looks the same as the basic parallel method in figure 3. However the calculation method is different. In each time step the neighbor subdomains calculate with different directions following the small arrow under each subdomain. After one time loop the node can directly continue next loop without any synchronization.

2.2. CCM Workflow. The key of CCM is that the neighbor areas calculate from different directions so that they can exchanging data while doing calculation and each subdomain does not have to keep in the same time step. The communication part of equation 7 can be reduced and the speedup will increase without using additional computer resources. Figure 4 shows the architecture of CCM.

CCM works following the pseudo code.

```
For time=0 to max time
    Begins a nonblocking receive
    If core index is odd
        From right to left
            Calculate electromagnetic field
            If finish shared area
                Send the data to its neighbors
            End
        End
    Else
        From left to right
            Calculate electromagnetic field
            If finish shared area
                Send the data to its neighbors
            End
        End
    End
    Synchronize until the others finish data exchanging
End
```

At the beginning of the time step, each subdomain will start a nonblocking receiving service with the MPI_Irecv function to receive data from its neighbors. During the simulation, each area starts calculation at the same time and share the data with their neighbors immediately after calculating the shared area. In addition, the computer cores are not totally the same so that some extra work should be done to make sure all the calculate cores are in the same time step. However, it is not the same as the classic FDTD method that every subdomain must be strictly in the same time step. For example, subdomain 1 and subdomain 2 start from different direction at the same time. Then subdomain 2 finishes the simulation earlier. In CCM it can directly go to next time step because the shared data of subdomain 1 in the right has already been calculated and transferred at the first beginning by subdomain 1. However, if the simulation of subdomain 2 is more than two steps faster than subdomain 1. Then the subdomain 2 should wait the subdomain 1.

CCM overlaps the data exchanging time so that the T_{CX} in equation 7 is 0 and the speedup R can be calculated by equation 8.

$$R = N/(T_{CS}/T_{CAL} + 1). \tag{8}$$

In equation 8, the speed up only related to the synchronization time when the number of calculation cores N is constant. Because the synchronization of CCM is not as strict as the classic parallel method, the T_{CS} of CCM is smaller. And theoretically, if the calculate cores and the size of the subdomains are totally the same, T_{CS} is always zero and the speedup R has linear increase along with the increasing of calculating cores (9).

$$R = N. \tag{9}$$

3. Experiment

In this section, we will introduce several experiments between the classic FDTD parallelization method and CCM. We assume that the simulation area is in the vacuum with a single source in the center of the area. Different area has different

number of grids. The size of each grid is $0.01m \times 0.01m \times 0.01m$. Session 3.1 shows the speedup and time consuming of each simulation target and session 3.2 shows the percentage of the communication time P_{COM} (10) and the percentage of calculating time P_{CAL} can be calculated by equation 11. P_{CAL} signifies the usage of the computer calculating resources.

$$P_{COM} = T_{COM}/T_{parallel}. \tag{10}$$

$$P_{CAL} = 1 - P_{COM}. \tag{11}$$

3.1. Speedup and Time Consuming. Figure 5 shows the speedup of the simulation of a 100×100×1000 area between the classic FDTD parallelization method and CCM. The core numbers of each point in the figure are 1, 4, 10, 40, 80, 100 and 120.Figure 6 shows the time consuming of the simulation of a 500×500×2000 area between the classic FDTD parallelization method and CCM. Because the area is too large to calculate by a single computer, we can only get the calculating time of both method, the displayed value T_D of time axis is calculated by equation 12 from the real time T_R.

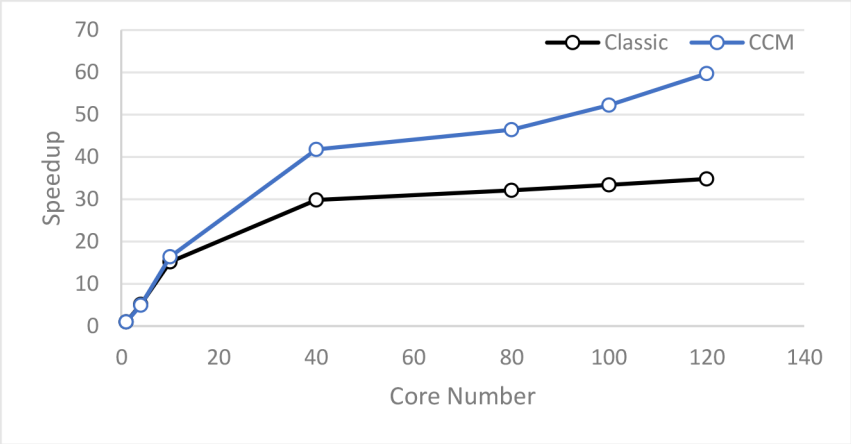

FIGURE 5. Speedup of the simulation of 100 × 100 × 1000 area. When the core number is small both methods have similar speedup and increase fast. As the core number increases the speedup of the classic method increases slowly but the speed up of CCM still increases fast and is nearly 2 times faster than the classic method.

$$T_D = 10Log10(T_R). \tag{12}$$

3.2. Communication Time Percentage. Figure 7 shows PCOM of the simulation of a 100×100×1000 area between the classic FDTD parallelization method and CCM.Figure 8 shows PCOM of the simulation of a 500×500×2000 area between the classic FDTD parallelization method and CCM. The result is the same as figure 7.

Figure 9 shows five different areas including 100×100×1000 , 100×100×2000 , 200×200×2000, 500×500×1000 and 500×500×2000 simulation result between the

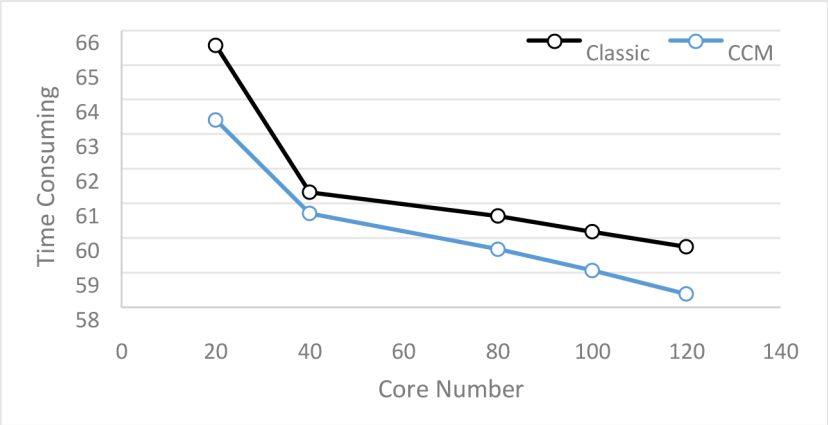

FIGURE 6. Time consuming of the simulation area with $500\times500\times2000$. As the increase of the core number CCM always has better performance than the classic parallelization method.

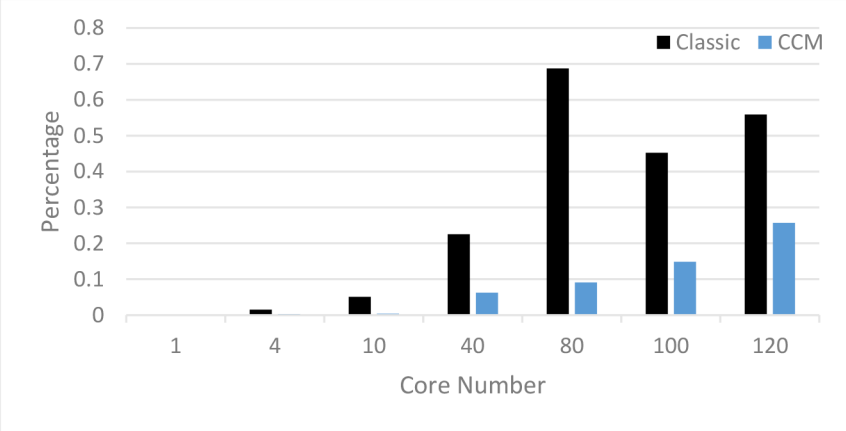

FIGURE 7. P_{COM} of the simulation area of $100\times100\times1000$. When the core number is less than 40 both methods have very little percentage and the P_{COM} of CCM is nearly 0. When the core number increases, P_{COM} of CCM is less than classic method and in some conditions P_{COM} of classic method even takes more than 50%.

classic FDTD parallelization method and CCM. The core number of each simulation is 120.

4. Conclusion

As an optimized parallelization FDTD method, CCM has some advantages than the classic method.

CCM does not need to strictly keep each subdomain in the same time step. As the share data can be exchanged while calculating, when a subdomain finishes one time step calculation it can go directly to the next time step.

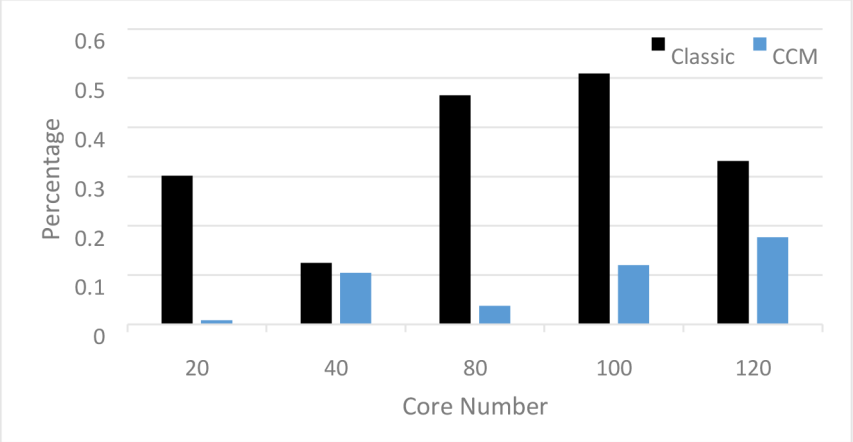

FIGURE 8. P_{COM} of the simulation area with $500 \times 500 \times 2000$.

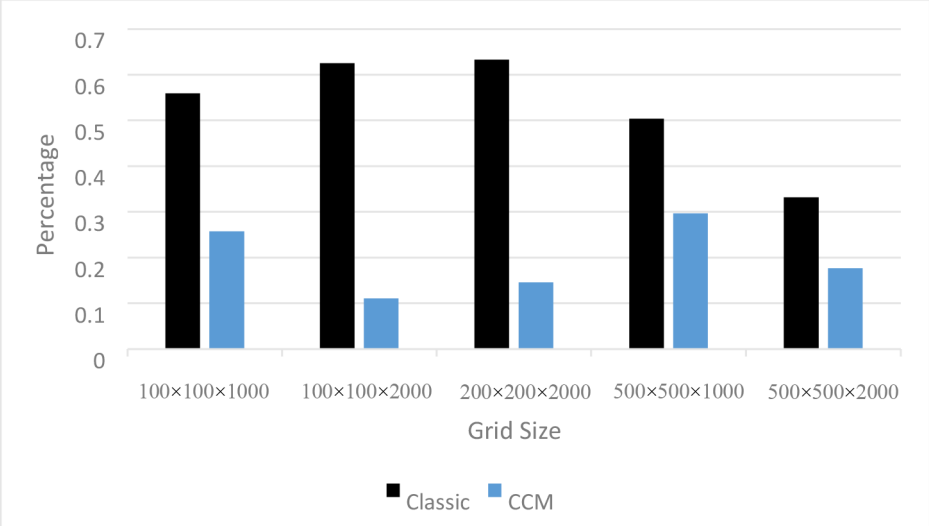

FIGURE 9. P_{COM} of the simulation with 120 cores. P_{COM} of the classic method is always very large and more than 50% and P_{COM} of CCM is always less than 30%.

CCM has higher speedup than the classic method. From experiment result we can find that CCM works much faster than the classic method as the speedup is nearly twice higher than the classic method and when the number of cores increases the speedup still has linear increase.

CCM has better performance in computing resource usage. PCOM of CCM is always smaller.

Acknowledgments

This research is partially supported by the Specialized Research Fund for the Doctoral Program of Higher Education [SRFDP 20113108120022], the Key Project

of Science and Technology Commission of Shanghai Municipality [No. 11510500300], and the Major Research Plan of NSFC [No. 91330116].

References

[1] Yee K. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. Antennas and Propagation, IEEE Transactions on, 14(3): 302-307(1966).
[2] Mur G. Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations[J]. Electromagnetic Compatibility, IEEE Transactions on, 1981 (4): 377-382(1981).
[3] Sacks Z S, Kingsland D M, Lee R, et al. A perfectly matched anisotropic absorber for use as an absorbing boundary condition[J]. Antennas and Propagation, IEEE Transactions on, 43(12): 1460-1463(1995).
[4] Gedney S D. An anisotropic perfectly matched layer-absorbing medium for the truncation of FDTD lattices[J]. Antennas and Propagation, IEEE Transactions on, 44(12): 1630-1639(1996).
[5] Gropp W, Lusk E, Doss N, et al. A high-performance, portable implementation of the MPI message passing interface standard[J]. Parallel computing, 22(6): 789-828(1996).
[6] Srinivasan K, Engin E, Swaminathan M. Fast FDTD simulation of multiscale 3D models using laguerre-MNA. Signal Propagation on Interconnects. SPI 2007. IEEE Workshop on. IEEE, 2007: 141-144(2007).
[7] Sandeep S, Gasiewski A J. Electromagnetic analysis of radiometer calibration targets using dispersive 3d FDTD. Antennas and Propagation, IEEE Transactions on, 60(6): 2821-2828 (2012).
[8] Miskiewicz M N, Bowen P T, Escuti M J. Efficient 3D FDTD analysis of arbitrary birefringent and dichroic media with obliquely incident sources. SPIE OPTO. International Society for Optics and Photonics: 82550W-82550W-10 (2012).
[9] Millington T M, Cassidy N J. Optimising GPR modelling: A practical, multi-threaded approach to 3D FDTD numerical modelling. Computers and Geosciences, 36(9): 1135-1144 (2010).
[10] Kim K H, Kim K H, Park Q H. Performance analysis and optimization of three-dimensional FDTD on GPU using roofline model. Computer Physics Communications, 182(6): 1201-1207 (2011).
[11] Kim K H, Park Q H. Overlapping computation and communication of three-dimensional FDTD on a GPU cluster. Computer Physics Communications, 183(11): 2364-2369 (2012).
[12] Shams R, Sadeghi P. On optimization of finite-difference time-domain (FDTD) computation on heterogeneous and GPU clusters. Journal of Parallel and Distributed Computing, 71(4): 584-593 (2011).
[13] Francs J, Bleda S, Neipp C, et al. Performance analysis of FDTD applied to holographic volume gratings: Multi-core CPU versus GPU computing. Computer Physics Communications (2012).
[14] Stefanski T P. Implementation of FDTD-compatible Green's function on heterogeneous CPU-GPU parallel processing system. Progress In Electromagnetics Research, 135: 297-316 (2013).
[15] D.D. Xu, H.W. Yang, The calculation of plasma reflection with parallel (FD)(TD)-T-2 method based on MPI, Optik, 124 1832-1835(2013).
[16] He Z L, Huang K, Zhang Y, et al. Study on High Performance of MPI-Based Parallel FDTD from WorkStation to Super Computer Platform[J]. International Journal of Antennas and Propagation, 2012 (2012).
[17] Stefaski T P, Chavannes N, Kuster N. Parallelization of the FDTD method based on the open computing language and the message passing interface[J]. Microwave and Optical Technology Letters, 54(3): 785-789 (2012).
[18] Qi X F, Guo L X, Tsang H. MPI-based Parallel FDTD for EM Scattering from Coated Complex Targets[J]. Session 2AP, 255 (2010).

School of Computer Engineering and Science, Shanghai University, Shanghai SH 200444, China
E-mail: jiangx@shu.edu.cn