INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 9, Number 2, Pages 257–269

© 2012 Institute for Scientific
Computing and Information

# PARALLEL DATA PARTITIONING STRATEGY IN SOLVING LARGE SCALE ELECTROMAGNETIC SCATTERING PROBLEMS

YUE HU, WEIQIN TONG, XINGANG WANG, AND XIAOLI ZHI

**Abstract.** The multilevel fast multipole algorithm (MLFMA) has shown great efficiency in solving large scale electromagnetic scattering problems. However, when unknowns become up to tens of millions, it is not trivial to keep high performance because of the complicated structure and calculation of MLFMA. In order to get rid of the bottleneck caused by load balancing, a parallel data partitioning strategy is proposed based on the hierarchical structure of an oct-tree of MLFMA. We present our data partitioning strategy in the light of different layers' properties including the processing of three kinds of layers in the tree and a fine-grained decomposition. We also put forward a solution of a coexisting data correlating problem, using a transition layer. Meanwhile, with the purpose of minimizing communication time in distributed memory system, a redundant technique is applied in the distributed layer. Parallel efficiency analysis demonstrates that the computational cost in parallelization of MLFMA can be asymptotically cut, and a high parallel efficiency can be obtained in our implementation.

**Key words.** Multilevel Fast Multipole Algorithm (MLFMA), Parallel Data Partitioning Strategy, Hierarchical Structure, Data Correlating Problem, and Redundant Technique.

## 1. Introduction

To achieve the fast computing characteristic of large scale electromagnetic problems, the Multilevel Fast Multipole Method (MLFMA) is applied, as well as Message Passing Interface (MPI) for network communications among processors. MLFMA was optimized by Song and Chew [1] in 1995, which has been widely used in recent years. Song and Chew implemented the MLFMA with $O(NlogN)$ complexity, where $N$ is the number of unknowns, and the memory requirement using translation, interpolation, anterpolation (adjoint interpolation), and a grid-tree data structure.

For the actual demand, we hope to develop a program to solute a full-sized aircraft problem that could run concurrently from single workstations to network-linked clusters. For the sake of a full-sized airplane, unknowns could be up to tens of millions. Although MLFMA has shown its high performance in reducing the computational complexity and the memory complexity of Matrix Vector Multiplications (MVMs) from $O(N^2)$ to $O(NlogN)$ , when $N$ extends to millions, several encumbrances have to be faced. And simple parallelization strategies usually fail to provide efficient solutions, owing to massive communications, poor load-balancing and necessary duplications. Advanced parallelization techniques have been proposed to improve the parallelization of MLFMA by using preconditioning strategies [2], extensively investigate the parallelization of MLFMA, identify the bottlenecks and provide remedial procedures [3], and even a novel method called nondirective stable plane wave multilevel fast multipole algorithm is developed to evaluate the low-frequency interactions which cannot be managed by MLFMA [4]. Especially

Received by the editors November 9, 2009 and, in revised form, June 21, 2010.
2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.
This research was supported by Aviation Industry Information Center of China (No.J50103), and the Graduate Innovation Fund of Shanghai University, Shanghai Leading Academic Discipline Project (No.SHUCX101062).

after answered the question that whether 10 million is big [5], Velamparambil and Chew analyzed the communication pattern, computational behavior and studied the scalability of a distributed memory implementation of MLFMA called ScaleME [6].

Recently, we developed a hierarchical partitioning strategy to fit for the multilevel structure of MLFMA. With this method an enhanced load-balancing is obtained, parallelization of MLFMA is improved significantly, and it has become possible for us to solve a three-dimensional full aircraft discretized up to 10 million unknowns with optimal parallel efficiency. In this paper, we present the details of a parallel MLFMA data partitioning implementation including investigating the parallelization procedure by focusing on different parts of an oct-tree and identifying a fine-grained data decomposition. Our approach involves the partitioning strategies to distribute tasks equally among processors and minimize the interprocessor communications.

The rest of the paper is organized as follows. In section 2, the MLFMA equations we use are briefly described, as well as a data collecting scheme and the layout of the oct-tree of MLFMA in our implementation. Section 3 narrates a fine-grained data decomposition, followed by the detail partitioning strategies of each layer of the oct-tree in section 4. The results are analyzed in section 5, and section 6 introduces our conclusion and future work.

## 2. Background

**2.1. Multilevel Fast Multipole Algorithm (MLFMA).** For the solution of the electromagnetic scattering problems involving three-dimensional conducting bodies with arbitrary shapes, Multilevel Fast Multipole Algorithm, which is detailed in [7]-[10], performs efficiently together with the Fast Multipole Method (FMM) [11] and a large problem can be solved iteratively, where the required Matrix-Vector Multiplications (MVMs) are involved. The application of boundary conditions for the electric filed and the magnetic field on the surface of an object leads to the Electric Field Integral Equation (EFIE) and the Magnetic Field Integral Equation (MFIE), respectively. For closed surfaces, EFIE and MFIE can be combined to obtain the Combined Field Integral Equation (CFIE). These three equations are briefly described as follows and considered as the point of departure in our work.

- EFIE

$$\widehat{n} \times L(\overrightarrow{J}) = \widehat{n} \times \overrightarrow{E}'. \tag{1}$$

where

$$L(\overrightarrow{J}) = -\overrightarrow{E}^s = jk\eta \int_s [\overrightarrow{J}((\overrightarrow{r})')G + \frac{1}{k^2} \bigtriangledown' \cdot \overrightarrow{J}((\overrightarrow{r})') \bigtriangledown G]ds'.$$

- MFIE

$$\frac{1}{2}J(r) + \overrightarrow{n} \times K(\overrightarrow{J}) = \overrightarrow{n} \times \overrightarrow{H}^i. \tag{2}$$

where

$$K(\overrightarrow{J}) = \int_s \overrightarrow{J}(\overrightarrow{r}) \times \bigtriangledown G ds'.$$

- CFIE

$$-\alpha\overrightarrow{n}\times\overrightarrow{n}\times L(\overrightarrow{J})+(1-\alpha)\eta[\frac{1}{2}J(r)+\overrightarrow{n}\times K(\overrightarrow{J})] \tag{3}$$
$$=-\alpha\overrightarrow{n}\times\overrightarrow{n}\times E^i(\overrightarrow{r})+(1-\alpha)\eta\overrightarrow{n}\times H^i(\overrightarrow{r}).$$

Here $\overrightarrow{n}$ , $E^i$ and $H^i$ denote any unit tangent vector on $s$, the incoming electric field vector and the incoming magnetic field. For the solution of problems involving closed surfaces, CFIE is preferable since it is free of the internal-resonance problem and provides better-conditional matrix equations than EFIE and MFIE. We attribute this favorable quality of CFIF to its linear combination of EFIE and MFIE and equation (3) can be reformulated as (4).

$$CFIE=\alpha EFIE+(1-\alpha)\eta MFIE. \tag{4}$$

The combination parameter $\alpha$ ranges from 0 to 1.

Furthermore, MLFMA splits the Matrix-Vector Multiplications (MVMs) required by the iterative solvers as (5).

$$\overrightarrow{Z}\times x=\overrightarrow{Z}_N\times x+\overrightarrow{Z}_F\times x. \tag{5}$$

In formula (5), the first term $\overrightarrow{Z}_N$ is the contributions from the near-neighbor field, which is calculated directly and stored in memory, while the second term $\overrightarrow{Z}_F$ is the interactions of the far-neighbor field under FMM, which are computed intricately in a group-by-group manner. So the computing field can be divided into two parts, i.e. the near-neighbor field and the far-neighbor field. There is the load balancing of the near-neighbor field [12], and in what follows, the load balancing and data partitioning of the far-neighbor field would be narrated. In MLFMA, the interactions of the far-neighbor field are calculated in a multilevel strategy using an oct-tree structure constructed from geometry data, which is derived from three-dimensional conducting bodies. As shown in Fig.1, the entire object is first nested into a large cube [1], which can just enclose it and its edge length is $2n$ times of the half wave length, where $n=0,1,2...N$. The cube then be divided into eight smaller cubes, and each sub-cube would be recursively subdivided into smaller ones until the edge length of the finest cube is about half of the wavelength.

In order to organize these cubs, a distributed oct-tree is constructed using Morton Ordering, as depicted in Fig.2, the nodes and leaves in the tree are labeled by the Morton Key numbers. Because Morton Ordering perfectly implies the topology structure of MLFMA and can minimize the efficiency loss during pointer conversion, owning to its convenience in representing a distributed tree in distributed memory architecture.

**2.2. Parallel Oct-Tree Layout.** The oct-tree works as the basic architecture and is built in a bottom-to-up approach initially to denote at which level a certain cube is. Several approaches have been discussed in the literatures [13, 14] for the tree's construction, but we construct it in a more flexible way, for more information in Section 4.

MLFMA has a property that at lower levels there are more Morton Keys with fewer samples, whereas fewer Morton Keys but more samples at higher levels. [15] puts forward a hierarchical partitioning strategy based on partitioning both clusters and field samples among processors at all levels of the multilevel tree structure, that is much different to our scheme. In our approach, Morton Keys are partitioned in the lower levels (distributed layer) and samples in the higher levels (shared layer).To

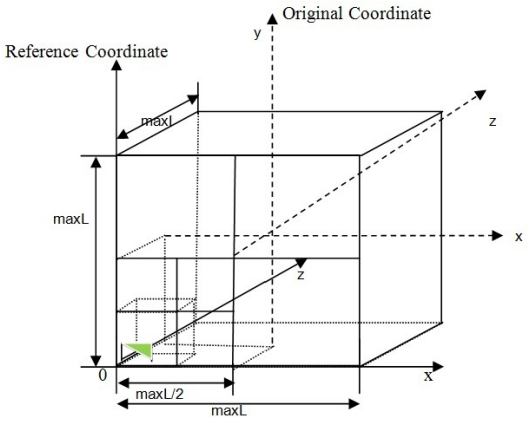


FIGURE 1. Hierarchical group division of MLFMA

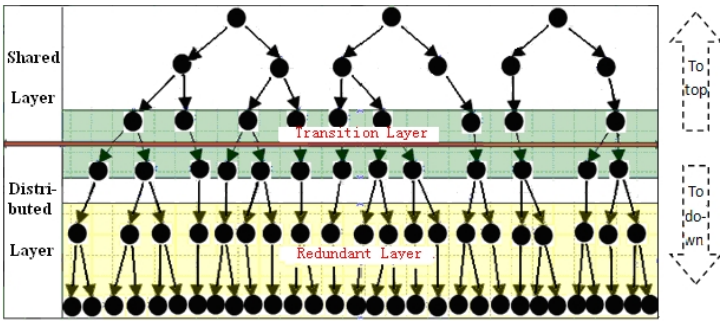


FIGURE 2. The layout of the oct-tree

decompose the Morton Keys and samples among participating processors, a fine-grained decomposing strategy is considered in section 3.

## 3. Fine-grained Decomposing Strategy

The objects to be partitioned are named as computing units in this paper, such as the Morton Keys and samples, etc. No.0 processor is the master node who figures out the total computing units and arranges each processor's task equals to the quotient initially. If some computing units remained, the master node would adjusts certain processors' task by adding one, from the processor who has the biggest *id* value until there are no computing units. Finally, a global communication is used to tell each processor's calculating section, processors then fulfill their computing queues' loading.

The dividing process can be obtained by lucid math computing, the code in our implementation using C++ programming language is:

```
num_my_work = total_number / mpi_num_procs;
  remainder = total_number % mpi_num_procs;
  begin_my_work = mpi_my_id * num_my_work;
  if((mpi_my_id-mpi_num_procs) > -remainder)
  {
    num_my_work++;
```

```
        begin_my_work += mpi_my_id - mpi_num_procs + re-
        mainder;
    }
```

Using this approach, the maximum load balancing gap between all the processors is only one computing unit, and the master processor is passed over to minimize its workload. Moreover, the data are monotonous in every processor and so as these data among processors, which means that the data in the processors whose *id* numbers are smaller must be bigger or smaller than those in the processors whose *id* value are bigger. That is vital for the following computing.

## 4. Data Partitioning Strategy of the Far-neighbor Field's Oct-tree

Due to the fact that solving large scale scattering problems is time-consuming, the critical task of parallelization of MLFMA is the load balancing and economic usage of memory. We propose an automatic load-balancing method based on a compressed oct-tree using parallel domain subdivision algorithm to effectively avoid the load balance problem between different processors. In this section, the details of the oct-tree partitioning implementation of MLFMA are based on Fig.2.

- *A. Distributed Layer*

One method we used to use is detailed in [16], but there were some deficiencies.

† First, because of the limitation of load balancing strategy in the distributed layer, there were two global communications between processors in pre-constructing and during constructing this layer, respectively;

† Second, in order to compensate this deficiency, three global communications were added after the distributed layer's construction.

For this problem, our new approach requires all processors to construct the global tree concurrently for an easy way to establish the reciprocal distribution of each local tree and minimize communications. The distributed layer is concurrently divided in a top-to-bottom approach just on the contrary way of [16]. At the top level of this layer, the clustering algorithm is applied to reduce storage requirement and communications, which means that the near-neighbor Morton Keys are loaded in the same node. Then children of all the Morton Keys' at the lower levels are recursively assigned to the same node. Thus, only one broadcast is needed by the master node to tell other processors the finest level's Morton Keys for their constructing of the global tree and the distributed layer.

Moreover, a redundant layer can be employed in the distributed layer to store the far-neighbor grids, so the communications to fetch the far-neighbor grids' aggregations can be further cut down in the redundant layer.

- *B. Redundant Layer*

In the distributed layer, because of the data partition in view of load balancing, some of the far neighbors of Morton Keys may be dispersed in other processors, and some communications are unavoidably needed. The communication establishing scheme is carefully considered.

Let $\mathrm{L}(l)$ denotes the data set at $lth$ level. We define $F_{i,j}(P)$ as Morton Key $i's\ jth$ far neighbors, who is belonging to $\mathrm{L}(l)$ at processor $P$, and the set $\{\{P_i, P_j\}, ..., \{P_k, P_l\}\}$ denotes the communication pairs.

**Theorem 1.** *The communication pair $\{P_i, P_j\}$ can be confirmed iff the following condition is met,*

TABLE 1. The total computing time and communication time in experiments

| Total Time(s) | Communication Time(s) | Proportion |
|---|---|---|
| 28.902 | 5.918734 | 28.3% |
| 31.481 | 18.136484 | 57.6% |
| 261.721 | 111.754554 | 42.7% |
| 524.630 | 353.109776 | 67.3% |

$$\exists k \in F_i, l(P_j) \cap k \in P_j \cap k \in L(l) \cap i \in L(l) \text{for all} i \in P_j. \tag{6}$$

So the set $\{\{P_0, P_i\}, ..., \{P_0, P_j\}\}$ indicates the processors with whom processor $P_0$ needs to communicate.

However, as listed in Table 1, from our experiments in computing different problems, communication in distributed memory system plays an important part in parallel computing. And when the total computing time increases, a rapid increase in the proportion of communication time to the total computing time can be noted. To exploit the parallel efficiency potentiality, a compromise between communication and calculation plus storage is of great value. In view of this problem, we make use of the redundant technique to optimize, and levels using this technique are uniformly called **redundant layer** .

In order to make best use of this skill, the redundant layer starts from the finest level as depicted in Fig.2, where the initial geometry data are read assuming the data at each level of this layer have been well divided. In this layer, all the data are selected by the communication set $\{\{P_i, P_m\}, ..., \{P_i, P_n\}\}$ , where $P_i$ is the local processor and $m < n$ .

We note that as the number of levels in redundant layer increases, memory occupancy in each processor becomes asymptotically more. In a certain parallel computing system, if there is little memory available, we hope to reduce the redundant levels, otherwise, increase them to minimize communications. For a better adapting to different systems, this layer is implemented in a flexible way that its top level can range from the lowest to the highest level of the distributed layer, even there can be no redundant layer at all. Before running our program, the only thing the users need to do is inputting the number $R_N$ of redundant levels, where $R_N \geq 0$ . If $R_N$ is bigger than the maximum value $D_N$ , which is the number of distributed levels, the top level of the redundant layer would be automatically arranged at the highest level of the distributed layer. In this case, the whole distributed layer would use the redundant technique.

- *C. Shared Layer*

The decomposition of shared layer is from bottom to up as illustrated in Fig.2. In this layer, the parallelization is used to decompose the sample matrix. Samples on the unit sphere generated by the composite Gauss-Trapezoidal Rule are viewed as a 2-D array. Usually, a Gaussian Quadrature Rule is used along the $\theta$ direction and a Trapezoidal Quadrature Rule along the $\phi$ direction. Let $N_{\theta,l}$ be the order of the Gaussian Quadrature, $N_{\phi,l}$ the order of the Trapezoidal Rule, and $D_l$ the cube size at level $l$, their relationships are shown in equation (7) and (8).

$$N_{\theta,l} \approx kD_l. \tag{7}$$

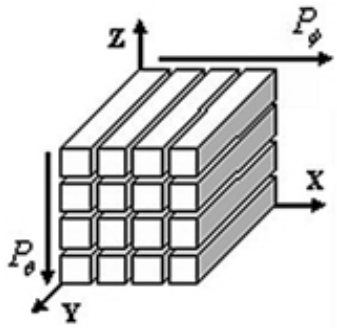


FIGURE 3. Block communication division in $\theta$ and $\phi$ directions

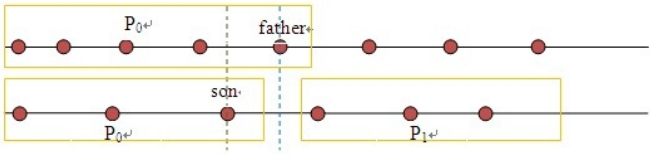


FIGURE 4. Block communication division in $\theta$ direction

$$N_{\phi,l} = 2N_{\theta,l}. \tag{8}$$

It is figured that the best partitioning scheme for this layer is to minimize the communication volume and decrease the number of processors with whom the local processor needs to communicate. For such purpose, we propose block communication [18] making use of the interpolation theory in this section. The implementation process is as follows.

**Step 1— The Arrangement of Processors:** Processors should be arranged in both $\theta$ and $\phi$ directions, which is referred to equation (8). As presented in Fig.3, the total number of processors and the number of processors in $\theta$ and $\phi$ directions are denoted by $P_N$ , $P_\theta$ and $P_\phi$ , respectively, which satisfy the following requirements,

$$P_\phi = 2n, n = 1, 2, .., N. \tag{9}$$

$$P_\phi \times P_\theta = P_N. \tag{10}$$

$$P_\phi : P_\theta = 2 : 1. \tag{11}$$

Since $P_\phi$ is an even number, so is $P_N$ . But once $P_N$ is confirmed, the third criterion maybe cannot be satisfied except for $P_N = 2 \times N^2$, where $N \geq 1$. In this case, in order to make use of resources, the third rule can be ignored, but the former two must be fulfilled.

**Step 2— The Arrangement of Processors:** In the shared layer, samples are partitioned in two directions.

◇ In the $\theta$ direction: Owning to the fact of random distribution for samples in this direction, we utilize the relationship between

fathers and children in the interpolation method to minimize the data size during communication and optimize the block communication. First, samples are equally divided according to the fine-grained decomposition at the top level, i.e. the third level, since we can directly compute without too much complexity at level three (the level number begins from one). Second, the lower levels are recursively partitioned according to their fathers. That is, the children whose values are smaller than the last sample *Father*'s value at the upper level, are assigned to the same processor with which *Father* in.

In this way, an optimal load balancing is obtained. Moreover, each processor is assigned task, which means that the abnormal termination in this direction by zero-load in some processors during the global communications can be avoided. Take Fig.4 as an example, assume *father* has been assigned in processor $P_0$, and the *son* in its lower level is the last one, whose valve is smaller than the *father*. The *son* would be arranged in processor $P_0$ either, as well as those before it in the *son's* level.

$\diamond$ In the $\phi$ direction: Samples are equidistantly located along this direction and the computing region is [0, 2 $\pi$ ], so we equally divide this region into parts at each level of this layer, according to the fine-grained decomposition. These subregions are then successively assigned to the processors, which are arranged in this direction. Thus samples in every processor are almost the same and the interpolation communications in this direction can be guaranteed.

- *D. Transition Layer*

The processing of the far-neighbor field involves three steps called the upward pass or the aggregation phase, the translation phase and the downward pass or the disaggregation phase [17]. The distributed layer and the shared layer run through this process, and they need to collaborate with each other.

In order to reduce complexity and form a clear boundary, we introduce a **transition layer** as shown in Fig.2. This layer bears the interface of the distributed layer and the shared layer. In this layer, Morton Keys and samples are redistributed and some work should be done to avoid.

(1) The communications between ghost boxes from distributed layer to transition layer in the aggregation phase;
(2) The interpolating communications from transition layer to shared layer in the aggregation phase;
(3) The communications between ghost boxes from transition layer to distributed layer in the disaggregation phase;
(4) The communications of transfer factors from transition layer to distributed layer in the disaggregation phase.

To implement it as clear and flexible as possible, this layer is set at different levels. As illustrated in Fig.5, in the aggregation phase, it is at the top level of the distributed layer, and in the disaggregation phase, at the bottom level of the shared layer.

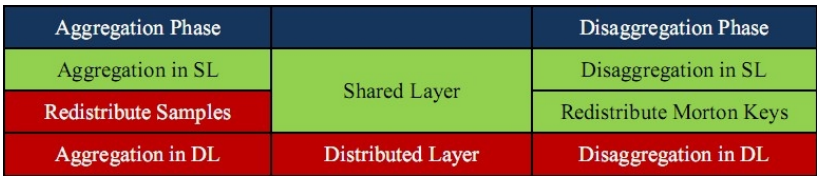


FIGURE 5. The transition layer is at different levels during the aggregation phase and disaggregation phase

TABLE 2. The detail information of two testing problems in experiments

| Testing Model | Frequency | NU | NL | NCA | SL |
|---|---|---|---|---|---|
| Diamond937 | 9.375 | 887,514 | 9 | 11 | 5 |
| Diamond577 | 5.7752 | 240,417 | 8 | 51 | 3-7 |

## 5. Experimental Results and Discussions

In this section, numerical results are listed for two electromagnetic scattering objects to demonstrate the validity of the parallel strategy proposed in this paper. The geometries considered include two diamonds, whose outlines are very similar to aircrafts. The detail testing information of the two objects are described in Table 2, where the notations NU, NL, NCA and SL represent the number of unknowns, number of levels, number of computing angles and initial level of the shared layer, respectively.

The tests were conducted at Shanghai Super Computing Center and Aviation Industry Development Research Center of China. The inner super computer Dawning 5000A at Shanghai Super Computer Center has two nodes, each node contains 16 AMD Barcelona (2.0GHz) processors and 64G memory. The super computer at Aviation Industry Development Research Center of China is a cluster. For the sake of secret, parameters are not detailed here.

In this section, we would show the experimental results from three aspects, i.e. the effect of the redundant layer, the parallel efficiency and the memory requirement varying with different initial levels of the shared layer on four processors.

The parallelization efficiency is defined as

$$\eta_m = \frac{T_n/T_m}{m/n} \times 100\% = \frac{T_n}{\frac{m}{n} \times T_m} \times 100\%, m > n. \tag{12}$$

where $T_n$ and $T_m$ are the processing time with $n$ and $m$ processors, respectively. This equation is used owning to the impossibility to compute the diamond model with a single processor, and our parallel computing ultimate target is that the processing time can be reduced to $100 \times n/m$ percent of $T_n$ , which indicates that m/n can be the parallel standard. Meanwhile, once $n$ equals to 1, equation (12) is just the canonical formula $T_1/(p \times T_p)$ , and in this paper we choose $n$ as 4.

- The Effect of the Redundant Layer

In order to evaluate the redundant layer's efficiency, two kinds of computing systems are chosen. One is 16 shared-memory processors at Shanghai Super Computing Center. Another is a cluster using 32 processors at Aviation Industry Development Research Center of China. The shared layer starts from level five. And the experimental results are respectively presented in Fig.6 and Fig.7, where a

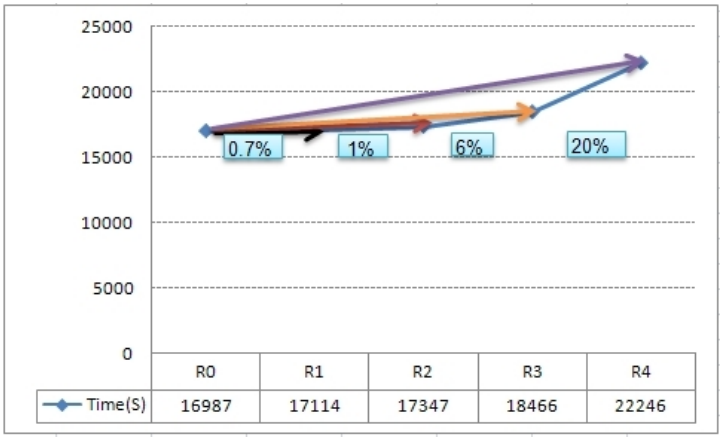


FIGURE 6. The processing time and acceleration according to the size of the redundant layer, which are tested at Shanghai Super-Computing Center


FIGURE 7. The processing time and deceleration according to the size of the redundant layer, which are tested at Aviation Industry Development Research Center

notation $R_x$ is used to denote the number of levels in redundant layer. The notation $R_0$ means there is no redundant level in the whole tree.

Fig.6 illustrates that the computing time is increasing along with the redundant levels' increasing, i.e. the accelerating rate becomes bigger and bigger, which is from 0.7% to 20% . We consider this phenomenon to have a relationship with the computing system. For the sake of the program testing benchmark, the condition in shared memory system is different from that in distributed memory system. When some calculating task is added to take place of some communications, the time reduced is always less than that added in shared memory system, because the communication time is much less than that in distributed memory system.

As demonstrated in Fig.7, the condition in distributed memory system is absolutely opposite, the redundant layer displays its efficiency in this system.

- The Parallel Efficiency

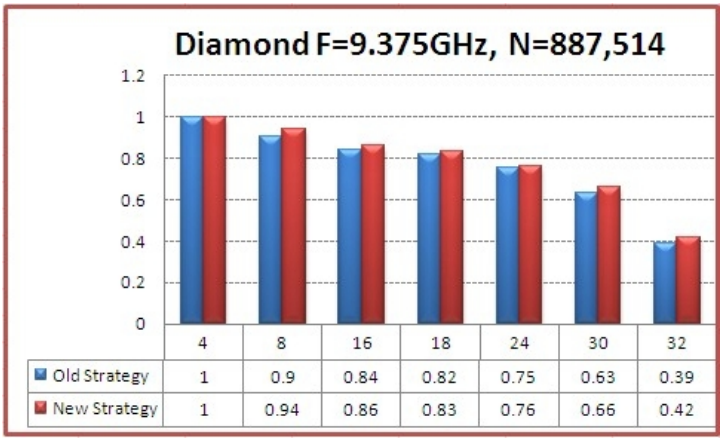


FIGURE 8. Parallel efficiency for a complex diamond

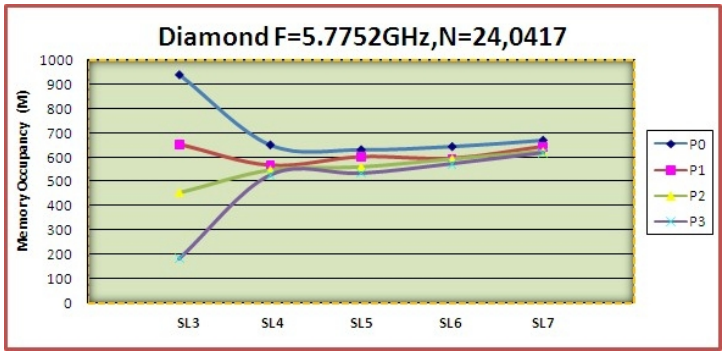


FIGURE 9. Memory occupancy in four processors with different *SL*

To assess the parallel efficiency, we contrast our new strategy with the old one. These tests were conducted at Shanghai Super Computing Center and the shared layer also started from the fifth level. Since the condition discussed above, the redundant layer was not used.

Fig.8 shows that the new strategy has greatly reduced the calculating time, and a high parallel efficiency above 80% when the processors are less than 20 has been obtained in our implementation. However, the bottom level of the shared layer is not trivial to determine for the best performance according to different problems. Moreover, distributed memory and shared memory coexist in lots of parallel computing architectures, the number of levels in redundant layer is also need an available strategy to consult. Once these two problems are solved, the parallel efficiency would get closer to the best.

- The Memory Occupancy

Since the memory requirement is huge in computing large scale scattering problems, the memory strategy would greatly influence the parallel efficiency. Moreover, the initial level of the shared layer *SL* can cause uncertainty or confusion about the best computing time. So, in this section, we make some effects to check our memory strategy under different *SL*. The experiments are done at Aviation Industry Development Research Center of China, and the results are represented in Fig.9.

From Fig.9, we can see that at the case of $SL = 3$, the gap in the memory requirements among processors is very big, which could cause expensive waiting time in the course of parallel computing. However, the memory requirement gap is rapidly reduced when $SL = 4$, so as $SL > 4$. Whereas, the total memory requirement appears increase as $SL$ growing, and the best condition is $SL = 5$. This conclusion has been confirmed by our plentiful tests that $SL$ can be set at three for small problems or five for large problems. But, we desire a systematic analysis to arrive a theoretical conclusion to compute different shapes and scales of problems.

## 6. Conclusion Remarks

In this paper, we have discussed our parallel data partitioning strategies in fine-granular scale and large scale aspects. To narrate the large scale partitioning scheme, a detail hierarchical data partitioning strategy of the far-neighbor field, using an oct-tree, is presented. The new parallel partitioning algorithm proposes a series of measures to cut down much communication time in the distributed layer and shared layer. On one hand, as shown by the experimental results, an optimal parallel efficiency has been obtained. On the other hand, the experimental results also point out that whether to use the redundant layer or how many levels are involved in the redundant layer depends on the computing environment and problem's size. The redundant layer can give full play to its advantage to compute large scale problems in the distributed memory system.

Until recently, we have solved a series of sized electromagnetic problems using this new data partitioning strategy. The scattering objects were sphere, cube, ring, diamond as well as tablet, wimble and cone-spheroid ranging from 0.15GHz to 5.7GHz, etc. And our goal is to solute a large scale dense matrix equation with more than 10 million unknowns effectively. We regard this as our milestone target of a massively parallel implementation of MLFMA.

The further work will focus on the memory and computing optimization to improve its performance, a detail performance analysis of the aggregation, translation and disaggregation at different levels will be given.

## Acknowledgments

The authors would like to show warm thankfulness to Hong-xia Zhang, Liang-cheng Zhu and Li-xing Zhang for their helpful comments on earlier drafts of this paper, and their helpful suggestions.

## References

[1] J. M. Song and W. C. Chew, Multilevel fast multipole algorithm for solving combined field integral equations of electromagnetic scattering, Microwave and Optical Technology Letters, **10**, 1995, 14-19.

[2] Tahr Malas and Levnt Grel, Incomplete LU preconditioning with the multilevel fast multipole algorithm for electromagnetic scattering, Journal on Scientific Computing, **29**, 2008,121-140.

[3] Ö. Ergl and L. Grel, Efficient Parallelization of the Multilevel Fast Multipole Algorithm for the Solution of Large-Scale Scattering Problems, IEEE Transactions on Antennas and Propagation, **56**, 2008, 2335-2345.

[4] Ignace Bogaert, Joris Peeters and Femke Olyslager, A Nondirective Plane Wave MLFMA Stable at Low Frequencies, IEEE Transactions on Antennas and Propagation, **56**, 2008, 3752 - 3767.

[5] Sanjay Velamparambil, Weng Cho Chew and Jiming Song, 10 Million Unknowns: Is It That Big?, IEEE Transactions on Antennas and Propagation, **45**, 2003, 43-58.

[6] Sanjay Velamparambil and Weng Cho Chew, Analysis and Performance of a Distributed Memory Multilevel Fast Multipole Algorithm, IEEE Transactions on Antennas and Propagation, **53**, 2005, 2719 - 2727.
[7] C. C. Lu and W. C. Chew, A multilevel algorithm for solving boundary integral equations of wave scattering, Microwave and Optical Technology Letters, **10**, 1994, 466-470.
[8] B. Dembart and E. Yip, A 3-D fast multipole method for electromagnetic with multiple levels, Electromagn.,CA, 1995, 621-628.
[9] M. A. Epton and B. Dembart, Multipole translation theory for the threedimensional Laplace and Helmholtz equations, SIAM J. Sci. Comput., **16**, 1995, 865-897.
[10] J. M. Song and W. C. Chew, Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering, Microwave and Optical Technology Letters, **10**, 1995, 14-19.
[11] A. Brandt, Multilevel computations of integral transforms and particle interactions with oscillatory kernels, Computer Phys. Comm., **65**, 1991, 24-38.
[12] Wang XinGang, Tang HuaNing, Zhi XiaoLi, Ni WeiLi and Tong WeiQin, Load Balancing and Data Locality of the Near-Field Interaction in the Parallelization of MLFMA, Proceedings of 2008 China-Japan Joint Microwave Conference, **1**, 2008, 347-349.
[13] A.Y. Grama, V.Kumar, and A.Sameh, Scalable parallel formulations of the bames-hut method for n-body simulations, Supercomputing '94 Proceedings, 1994, 439-448.
[14] M.S. Warren and J.K. Salmon, A parallel hashed oct-tree N-body algorithm, Supercomputing '93 Proceedings, 1993, 12-21.
[15] Özgr Ergl and Levent Grel, A Hierarchical Partitioning Strategy for an Efficient Parallelization of the Multilevel Fast Multipole Algorithm, IEEE Transactions on Antennas and Propagation, **57**, 2009, 1740-1750.
[16] Hailin Guo, Xiaoyan Xue, Xingang Wang, Weiqin Tong and Weili Ni, An Implementation of Parallel MLFMA on a Cluster of Computers with Distributed Memory, International Conference for Young Computer Scientists, 2008, 1379-1383.
[17] J. M. Song, C. C. Lu, and W. C. Chew, MLFMA for electromagnetic scattering from large complex objects, IEEE Transactions on Antennas and Propagation, **45**, 1997, 1488-1493.
[18] Wang Xingang, Cheng Bin and Tong Weiqin, A New Parallel Strategy for MLFMA Based on the Partitioned Blocks, International Conference on Information Science and Engineering, 2009, 43-46.

School of Computer Engineering and Science, Shanghai University, Shanghai, 200072, China
*E-mail*: `hiyuehu@gmail.com`

School of Computer Engineering and Science, Shanghai University, Shanghai, 200072, China
*E-mail*: `wqtong@staff.shu.edu.cn, wxg@shu.edu.cn and xlzhi@mail.shu.edu.cn`