Hindawi Publishing Corporation
International Journal of Antennas and Propagation
Volume 2015, Article ID 615743, 10 pages
http://dx.doi.org/10.1155/2015/615743

*Research Article*

# An MPI-OpenMP Hybrid Parallel $\mathscr{H}$-LU Direct Solver for Electromagnetic Integral Equations

**Han Guo, Jun Hu, and Zaiping Nie**

*The School of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China*

Correspondence should be addressed to Jun Hu; hujun@uestc.edu.cn

Received 20 July 2014; Revised 12 November 2014; Accepted 15 December 2014

Academic Editor: Wei Hong

In this paper we propose a high performance parallel strategy/technique to implement the fast direct solver based on hierarchical matrices method. Our goal is to directly solve electromagnetic integral equations involving electric-large and geometrical-complex targets, which are traditionally difficult to be solved by iterative methods. The parallel method of our direct solver features both OpenMP shared memory programming and MPl message passing for running on a computer cluster. With modifications to the core direct-solving algorithm of hierarchical LU factorization, the new fast solver is scalable for parallelized implementation despite of its sequential nature. The numerical experiments demonstrate the accuracy and efficiency of the proposed parallel direct solver for analyzing electromagnetic scattering problems of complex 3D objects with nearly 4 million unknowns.

## 1. Introduction

Computational electromagnetics (CEM), which is driven by the explosive development of computing technology and vast novel fast algorithms in recent years, has become important to the modeling, design, and optimization of antenna, radar, high-frequency electronic device, and electromagnetic metamaterial. Among the three major approaches for CEM, finite-difference time-domain (FDTD) [1], finite element method (FEM) [2], and method of moments (MoM) [3], MoM has gained widely spread reputation for its good accuracy and built-in boundary conditions. Generally, MoM discretizes the electromagnetic integral equations (IEs) into linear systems and solves them via numerical algebraic methods. Although the original MoM procedure forms a dense system matrix which is computationally intensive for solving large-scale problems, a variety of fast algorithms have been introduced to accelerate MoM via reducing both of its CPU time and memory cost. Well-known fast algorithms for frequency domain IEs, such as multilevel fast multipole method (MLFMM) [4–6], multilevel matrix decomposition algorithm (MLMDA) [7–9], adaptive integral method (AIM) [10], hierarchical matrices ($\mathscr{H}$-matrices, $\mathscr{H}^2$-matrices) method [11–15], multiscale compressed block decomposition (MLCBD) [16, 17], and adaptive cross-approximation (ACA) [18, 19], have remarkably increased the capacity and ability of MoM to analyze radiation/scattering phenomena for electric-large objects.

Traditionally, iterative solvers are employed and combined with fast algorithms to solve the MoM. Despite of the availability of many efficient fast algorithms, there are still some challenges in the iterative solving process for discretized IEs. One major problem is the slow-convergence issue. Due to various reasons, such as complex geometrical shapes of the targets, fine attachments, dense discretization, and/or non-uniform meshing, the spectrum condition of the impedance matrix of discretized IEs can be severely deteriorated. Therefore, the convergence speed of iteration will be slowed down significantly so that we are not able to obtain an accurate solution within a reasonable period of time. In order to overcome this difficulty, preconditioning techniques are usually employed to accelerate the convergence. There are some popular preconditioners, including diagonal blocks inverse [20], incomplete Cholesky factorization [21], incomplete LU factorization (ILU) [21, 22], and sparse approximate inverse (SAI) [23], used widely. However, the effectiveness of preconditioning techniques is problem dependent and the convergence still cannot be guaranteed. For some extreme cases preconditioning shows little alleviation to the original problem.

By contrast, direct solving like Gaussian elimination or LU factorization [24] spends fixed number of operations, whose impedance/system is only related to the size of impedance/system matrix, namely, the number of unknowns $N$, to obtain the accurate solution of MoM. Although these standard direct solvers are not favored over iterative solvers because of their costly computational complexity $O(N^3)$, a number of fast direct solvers (FDS) are introduced recently [14–19, 25], which inherit the merit of direct solving for avoiding slow-convergence issue, and have significantly reduced the required CPU time and memory.

So far, there are some existing literatures discussing the parallelization of direct solvers for electromagnetic integral equations [26, 27]. It is notable that the implementation of parallelizing FDS is not trivial. Since the algorithms of major FDS share a similar recursive factorization procedure in a multilevel fashion, their implementations are intrinsically sequential in nature. However, they can be parallelized with good scalability by elaborate dissection and distribution. In this paper, we proposed an MPI-OpenMP hybrid parallel strategy to implement the $\mathcal{H}$-matrices based direct solver with hierarchical LU ($\mathcal{H}$-LU) factorization [15, 28]. This parallel direct solver is designed for running on a computer cluster with multiple computing nodes with multicore processors for each node. OpenMP shared memory programming [29] is utilized for the parallelization on multicore processors for each node, while MPI message passing [30] is employed for the distribution computing among all the nodes. The proposed parallel FDS shows good scalability and parallelization efficiency, and numerical experiments demonstrate that it can solve electrical-large IE problems involving nearly 4 million unknowns within dozens of hours.

The rest of this paper is organized as follows. Section 2 reviews the discretized IE we aim to solve and basic MoM formulas. Section 3 outlines the construction of $\mathcal{H}$-matrices based IE system, including spatial grouping of basis functions and the partition of the impedance matrix, as well as the direct solving procedure of $\mathcal{H}$-LU factorization. Section 4 elaborates the parallelizing strategy for the construction and LU factorization of $\mathcal{H}$-matrices based IE system, with the discussion of its theoretical parallelization efficiency. Section 5 gives some results of numerical tests to show the efficiency, capacity, and accuracy of our solver. Finally, the summary and conclusions are given in Section 6.

## 2. Electromagnetic Integral Equation and Method of Moments

In this section, we give a brief review of surface IEs and their discretized forms [3] that our parallel FDS deals with. In the case of 3D electromagnetic field problem, electric field integral equation (EFIE) is written as

$$-\hat{\mathbf{t}} \cdot ik\eta \int_{s'} ds' \mathbf{J}\left(\mathbf{r}'\right) \cdot \left(\mathbf{I} - \frac{\nabla\nabla'}{k^2}\right) g\left(\mathbf{r}, \mathbf{r}'\right) = \hat{\mathbf{t}} \cdot \mathbf{E}^{\text{inc}}(\mathbf{r}), \quad (1)$$

in which $k$ is the wavenumber, $\eta$ is the wave impedance, $\hat{\mathbf{t}}$ is the unit tangential component on the surface of the object, and $g(\mathbf{r}, \mathbf{r}')$ is the free space Green function. $\mathbf{E}^{\text{inc}}(\mathbf{r})$ is the incident electric field and $\mathbf{J}(\mathbf{r}')$ is the excited surface current to be determined. Correspondingly, the 3D magnetic field integral equation (MFIE) is written as

$$\frac{-\mathbf{J}(\mathbf{r})}{2} + \hat{\mathbf{n}} \times \text{P.V.} \int_{s} ds' \mathbf{J}\left(\mathbf{r}'\right) \times \nabla' g\left(\mathbf{r}, \mathbf{r}'\right) = -\hat{\mathbf{n}} \times \mathbf{H}^{\text{inc}}(\mathbf{r}), \quad (2)$$

in which $\hat{\mathbf{n}}$ is the normal component on the surface of the object, $\mathbf{H}^{\text{inc}}(\mathbf{r})$ is the incident magnetic field, and P.V. stands for the Cauchy principal value integration. For time harmonic electromagnetic plane wave, $\mathbf{E}^{\text{inc}}$ and $\mathbf{H}^{\text{inc}}$ have the relationship

$$\mathbf{H}^{\text{inc}} = \frac{1}{\eta}\hat{\mathbf{k}} \times \mathbf{E}^{\text{inc}}, \quad (3)$$

where $\hat{\mathbf{k}}$ is the unit vector of wave propagating direction. In order to cancel the internal resonance that occurred in the solution of individual EFIE or MFIE, the two equations are usually linearly combined. Specifically, we have the combined field integral equation (CFIE) written as

$$\text{CFIE} = \alpha \cdot \text{EFIE} + (1-\alpha)\,\eta \cdot \text{MFIE}. \quad (4)$$

To solve this IE numerically, first we need to mesh the surfaces of the object with basic patches. For electromagnetic IEs, triangular and quadrilateral patches are two types of frequently used patches, which are associated with Rao-Wilton-Gliso (RWG) basis function [31] and roof-top basis function [32], respectively. By using these basis functions, the undetermined $\mathbf{J}(\mathbf{r})$ in (1) and (2) can be discretized by the combination of basis functions $\mathbf{f}_i(\mathbf{r})$, $(i = 1, 2, \ldots, N)$ as

$$\mathbf{J}(\mathbf{r}) = \sum_{i=1}^{N} I_i \mathbf{f}_i(\mathbf{r}), \quad (5)$$

in which $I_i$ is the unknown coefficients and $N$ is the total number of unknowns. By using Galerkin test, we can form the $N \times N$ linear systems for (1) and (2) as

$$\begin{aligned} \mathbf{Z}^{\text{EFIE}} \cdot \mathbf{I} &= \mathbf{V}^{\text{EFIE}}, \\ \mathbf{Z}^{\text{MFIE}} \cdot \mathbf{I} &= \mathbf{V}^{\text{MFIE}}, \end{aligned} \quad (6)$$

in which the system matrices $\mathbf{Z}^{\text{EFIE}}$ and $\mathbf{Z}^{\text{MFIE}}$ are also called impedance matrices in IEs context. The entries of the system matrices and right-hand side vectors can be calculated numerically through the following formulas:

$$\begin{aligned} \mathbf{Z}_{mn}^{\text{EFIE}} = &-ik\eta \int_{s}\int_{s'} g\left(\mathbf{r}, \mathbf{r}'\right)\left[\mathbf{f}_m(\mathbf{r}) \cdot \mathbf{f}_n\left(\mathbf{r}'\right)\right] ds' ds \\ &+ \frac{i\eta}{k}\int_{s}\int_{s'} g\left(\mathbf{r}, \mathbf{r}'\right)\left[\nabla_s \cdot \mathbf{f}_m(\mathbf{r})\, \nabla_{s'} \cdot \mathbf{f}_n\left(\mathbf{r}'\right)\right] ds' ds, \end{aligned}$$

$$\mathbf{Z}_{mn}^{\mathrm{MFIE}} = \frac{1}{2}\int_s ds\mathbf{f}_m(\mathbf{r})\cdot\mathbf{f}_n(\mathbf{r}) + \hat{\mathbf{n}}\cdot\int_s ds\mathbf{f}_m(\mathbf{r})\times\int_{s'} ds'\mathbf{f}_n(\mathbf{r}')\times\nabla' g(\mathbf{r},\mathbf{r}'),$$
$$\mathbf{V}_m^{\mathrm{EFIE}} = \int_s ds\mathbf{f}_m(\mathbf{r})\cdot\mathbf{E}^{\mathrm{inc}}(\mathbf{r}),$$
$$\mathbf{V}_m^{\mathrm{MFIE}} = \int_s ds\mathbf{f}_m(\mathbf{r})\cdot\hat{\mathbf{n}}\times\mathbf{H}^{\mathrm{inc}}(\mathbf{r}). \tag{7}$$

Consequently, the discretized CFIE system matrix and excitation vector are

$$\mathbf{Z}_{mn}^{\mathrm{CFIE}} = \alpha\mathbf{Z}_{mn}^{\mathrm{EFIE}} + (1-\alpha)\eta\cdot\mathbf{Z}_{mn}^{\mathrm{MFIE}},$$
$$\mathbf{V}_m^{\mathrm{CFIE}} = \alpha\mathbf{V}_m^{\mathrm{EFIE}} + (1-\alpha)\eta\cdot\mathbf{V}_m^{\mathrm{MFIE}}. \tag{8}$$

By solving the linear system

$$\mathbf{Z}^{\mathrm{CFIE}}\cdot\mathbf{I} = \mathbf{V}^{\mathrm{CFIE}}, \tag{9}$$

we can get the excited surface current $\mathbf{J}$ and corresponding radiation/scattering field.

## 3. Fast Direct Solver and Hierarchical LU Factorization

Generally, we can solve the linear system (9) iteratively or directly. For iterative solvers, the Krylov subspace methods are frequently used, such as CG, BiCG, GMRES, and QMR [33]. Because iterative methods often suffer from slow-convergence issues for complex applications, direct methods have become popular recently. Several researchers proposed a variety of FDS [14–19, 25], which aim at avoiding convergence problems for iterative solvers and utilize similar hierarchical inverse/factorization procedure and low rank compression techniques to reduce the computational and storage complexity.

$\mathscr{H}$-matrices method [11, 12, 15] is a widely known mathematical framework for accelerating both finite element method (FEM) and IE method. In the scenario of electromagnetic IE, $\mathscr{H}$-matrices method is regarded as the algebraic counterpart of MLFMM [4–6]. The most advantage of $\mathscr{H}$-matrices technique over MLFMM is that it can directly solve the IE system by utilizing its pure algebraic property. In this section, we briefly review the primary procedures of IE-based $\mathscr{H}$-matrices method and the algorithm of hierarchical LU factorization which will be parallelized.

In order to implement $\mathscr{H}$-matrices method, firstly the impedance matrix $\mathbf{Z}^{\mathrm{CFIE}}$ needs to be formed block-wisely and compressed by low rank decomposition scheme. We subdivide the surface of the object hierarchically and, according to their interaction types, form the impedance/forward system matrix in a multilevel pattern. This procedure is very similar to the setting-up step in MLFMM. By contrast to the octree structure of MLFMM, the structure in $\mathscr{H}$-matrices is a binary tree. The subdivision is stopped when the dimension of smallest subregions covers a few wavelengths. The subblocks in the impedance matrix representing the far-field/weak interactions are compressed by low rank (LR) decomposition schemes. Here we use the adaptive cross-approximation (ACA) algorithm [18, 19] as our LR decomposition scheme, which only requires partial information of the block to be compressed and is thus very efficient.

After the $\mathscr{H}$-matrices based impedance matrix is generated, the hierarchical LU ($\mathscr{H}$-LU) factorization [12, 26, 28] is employed on it and finally we get an LU decomposed system matrix LU ($\mathbf{Z}^{\mathrm{CFIE}}$). During and after the factorization process, subblocks in LR format are kept all the time. Having the LU ($\mathbf{Z}^{\mathrm{CFIE}}$), we can easily solve the IE with given excitations by a recursive backward substitution procedure. Figure 1 illustrates the overall procedure of the $\mathscr{H}$-matrices based FDS introduced above.

From $\mathbf{Z}^{\mathrm{CFIE}}$ to LU ($\mathbf{Z}^{\mathrm{CFIE}}$), the hierarchical LU factorization is employed. The factorization algorithm is elaborated below. Consider the block LU factorization of the level-1 partitioned $\mathbf{Z}^{\mathrm{CFIE}}$ matrix; namely,

$$\mathbf{Z}^{\mathrm{CFIE}} = \begin{bmatrix}\mathbf{Z}_{11} & \mathbf{Z}_{12}\\ \mathbf{Z}_{21} & \mathbf{Z}_{22}\end{bmatrix} = \begin{bmatrix}\mathbf{L}_{11} & \\ \mathbf{L}_{21} & \mathbf{L}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{U}_{11} & \mathbf{U}_{12}\\ & \mathbf{U}_{22}\end{bmatrix}. \tag{10}$$

We carry out the following procedures in a recursive way: (i) get $\mathbf{L}_{11}$ and $\mathbf{U}_{11}$ by LU factorization $\mathbf{Z}_{11} = \mathbf{L}_{11}\mathbf{U}_{11}$; (ii) get $\mathbf{U}_{12}$ by solving lower triangular system $\mathbf{Z}_{12} = \mathbf{L}_{11}\mathbf{U}_{12}$; (iii) get $\mathbf{L}_{21}$ by solving upper triangular system $\mathbf{Z}_{21} = \mathbf{L}_{21}\mathbf{U}_{11}$; (iv) update $\mathbf{Z}_{22}$: $\mathbf{Z}_{22} = \mathbf{Z}_{22} - \mathbf{L}_{21}\mathbf{U}_{12}$; and (v) get $\mathbf{L}_{22}$ and $\mathbf{U}_{22}$ by LU factorization $\mathbf{Z}_{22} = \mathbf{L}_{22}\mathbf{U}_{22}$. Recursive implementations are participated among all procedures (i)–(v). Procedures (ii) and (iii) (and similar operations under the hierarchical execution of (i) and (v)) are performed via partitioned forward/backward substitution for hierarchical lower/upper triangular matrices [12, 28]. Procedure (iv) is a vital operation that contributes most of the computing time cost. This subblocks addition-multiplication procedure occurs not only in one or several certain levels, but pervades in all of the recursive executions. Figure 1 illustrates the overall procedure of $\mathscr{H}$-matrices based FDS. Implementation details about $\mathscr{H}$-matrices compression and recursive LU factorization can be found in [12, 14, 15]. In addition, a similar recursive direct solving process for noncompression IE system matrix is discussed in [26].

Based on the result of $\mathscr{H}$-LU factorization, we can easily get the solution for any given excitation, namely, right-hand side (RHS) vector by hierarchical backward substitution. The CPU time cost for backward substitution is trivial compared to that of $\mathscr{H}$-LU factorization.

## 4. Parallelization of $\mathscr{H}$-Matrices Based IE Fast Direct Solver

The parallelization of $\mathscr{H}$-matrices based IE FDS contains two parts: (a) the parallelization (of the process) of generating the forward/impedance matrix with $\mathscr{H}$-matrices form; (b) the parallelization of $\mathscr{H}$-LU factorization and backward substitution. In this section, we will elaborate the approaches for implementing these two parts separately.

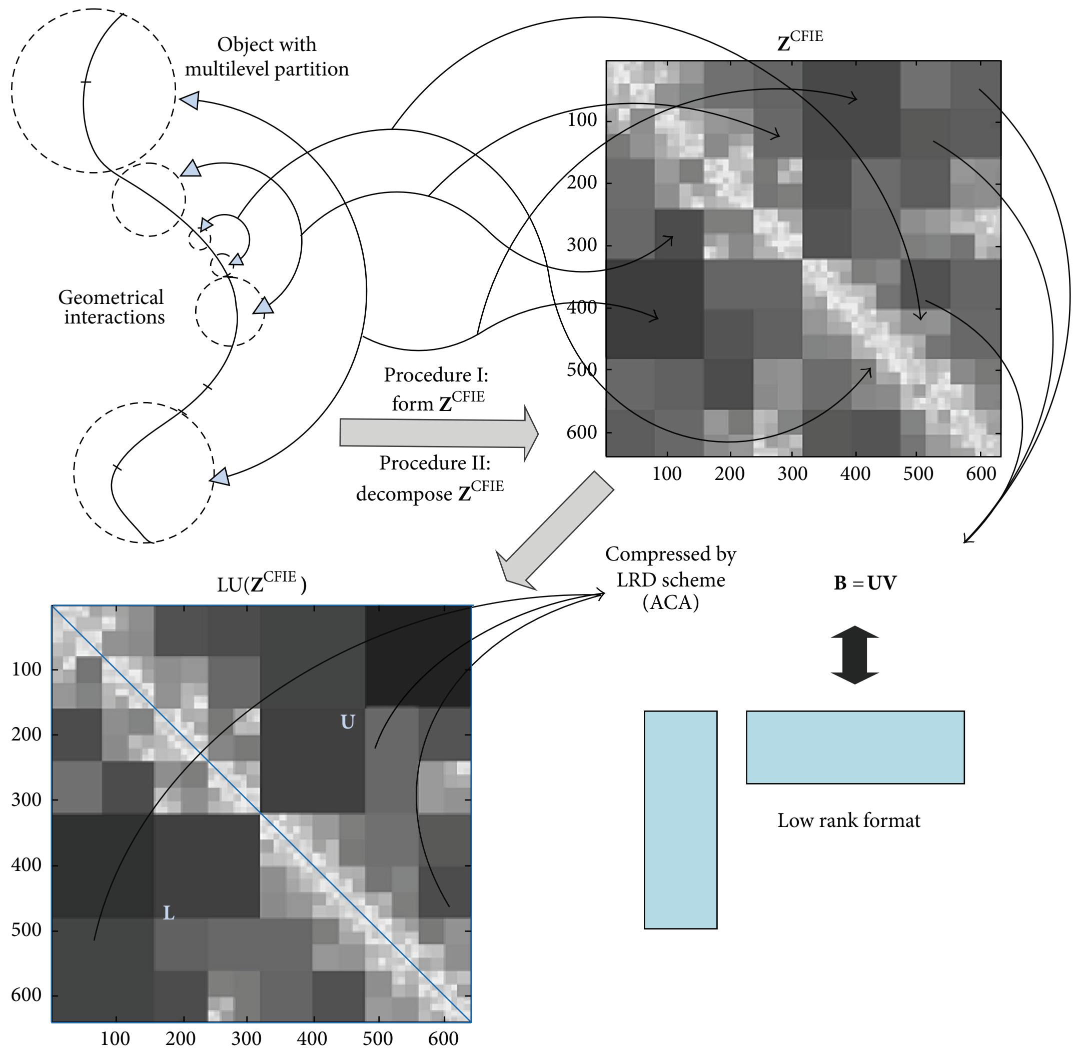
Object with multilevel partition
Geometrical interactions
$\mathbf{Z}^{CFIE}$
Procedure I: form $\mathbf{Z}^{CFIE}$
Procedure II: decompose $\mathbf{Z}^{CFIE}$
Compressed by LRD scheme (ACA)
$\mathbf{B} = \mathbf{UV}$
Low rank format
LU($\mathbf{Z}^{CFIE}$)
U
L
100
200
300
400
500
600

Figure 1: Overall procedure of $\mathscr{H}$-matrices based FDS.

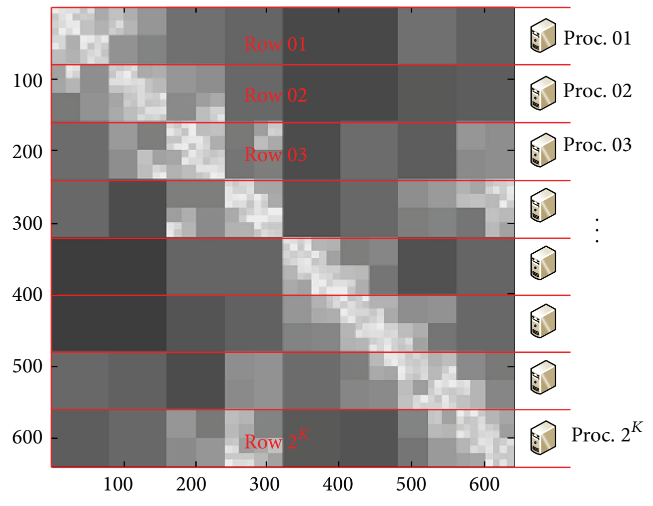

Figure 2: Data distribution strategy for forward/impedance matrix and also for LU factorized matrix.

*4.1. Parallelization of Generating Forward/Impedance Matrix.* Since our FDS is executed on a computer cluster, the computing data must be stored distributively in an appropriate way, especially for massive computing tasks. The main data of $\mathscr{H}$-matrices based IE FDS are the LR compressed forward/impedance and LU factorized matrices. In principle, for forward/impedance matrix, we divide it into multiple rows, and each computing node stores one of these row-blocks. This memory distribution strategy is shown in Figure 2. The width of each row is not arbitrary but in correspondence to the hierarchical partition of the matrix in certain level $K$. Therefore, no LR formatted block will be split by the division lines that cross the entire row. In practice, the width of each row is approximately equal and the total number of rows is the power of 2, namely, $2^K$. $K$ is an integer which is less than the depth of the binary tree structure of $\mathscr{H}$-matrices, which is denoted by $L$. Each row-block is computed and filled locally by each MPI node it is assigned to. Since every subblock is an independent unit, the filling process for these row-blocks can be parallelized for all MPI nodes without any communication or memory overlap. Furthermore, for each MPI node with multicore processor, the filling process of its corresponding blocks can be accelerated by utilizing the OpenMP memory share parallel computing. In the case of MoM, OpenMP is very efficient to parallelize the calculation of impedance elements, since this kind of calculation only depends on the invariant geometrical information and electromagnetic parameters, which are universally shared among all nodes.

*4.2. Parallelization of $\mathcal{H}$-LU Factorization.* The $\mathcal{H}$-LU factorization is executed immediately after the forward/impedance matrix is generated. Compared with the forward/impedance matrix filling process, the parallelization of $\mathcal{H}$-LU factorization is not straightforward because its recursive procedure is sequential in nature. In order to put $\mathcal{H}$-LU factorization into a distributed parallel framework, here we use an incremental strategy to parallelize this process gradually.

During the $\mathcal{H}$-LU factorization, the algorithm is always working/computing on a certain block which is in LR form or full form or the aggregation of them, so that we can set a criterion based on the size of the block for determining whether current procedure should be parallelized. Specifically, assuming that the (total) number of unknowns is $N$, we have about $2^K$ nodes for computing; then if the maximal dimension of ongoing processing block is larger than $N/2^K$, MPI parallel computing with multiple nodes is employed; otherwise the process will be dealt with by single node. It should be noted that OpenMP memory share parallel is always employed for every node with multicore processor participating in computing. In the $\mathcal{H}$-LU factorization process, OpenMP is in charge of intranode parallelization of standard LU factorization, triangular system solving for full subblocks, and SVD or QR decomposition for LR subblocks addition and multiplication, and so forth, which can be implemented through highly optimized computing package like LAPACK [34].

According to the recursive procedure of $\mathcal{H}$-LU factorization, at the beginning, the algorithm will penetrate into the finest level of $\mathcal{H}$-matrices structure, namely, level $L$, to do the LU factorization for the top-left most block with the dimension of approximately $(N/2^L) \times (N/2^L)$. Since $L \geq K$, this process is handled by one node. After that the algorithm will recur to level $L-1$ to do the LU factorization for the extended top-left block of approximately $(N/2^{L-1})\times(N/2^{L-1})$ upon the previous factorization. If $L-1 \geq K$ remains true, this process is still handled by one node. This recursion is kept on when it returns to level $l$ that $l < K$, for which MPI parallel with multiple nodes will be employed.

For the sake of convenience, we use $\mathcal{H}$-LU($l$) to denote the factorization on certain block in the level $l$ of $\mathcal{H}$-matrices structure. Recall the $\mathcal{H}$-LU factorization procedure in Section 3. Procedure (i) is $\mathcal{H}$-LU itself, so its parallelization is realized by other procedures on the finer levels. For the procedure (ii) and procedure (iii) we solve the triangular linear system as

$$\mathbf{LX} = \mathbf{A}, \tag{11}$$

$$\mathbf{YU} = \mathbf{B}, \tag{12}$$

in which $\mathbf{L}$, $\mathbf{U}$, $\mathbf{A}$, and $\mathbf{B}$ are the known matrices and $\mathbf{L}$, $\mathbf{U}$ are lower and upper triangular matrices, respectively. $\mathbf{X}$ and $\mathbf{Y}$ need to be determined. There two systems can actually be solved recursively through finer triangular systems. For instance, by rewriting (11) as

$$\begin{bmatrix} \mathbf{L}_{11} & \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{X}_{21} & \mathbf{X}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \tag{13}$$

we carry out the following procedures: (i) get $\mathbf{X}_{11}$ and $\mathbf{X}_{12}$ by solving lower triangular systems $\mathbf{A}_{11} = \mathbf{L}_{11}\mathbf{X}_{11}$ and $\mathbf{A}_{12} = \mathbf{L}_{11}\mathbf{X}_{12}$, respectively; (ii) get $\mathbf{X}_{21}$ and $\mathbf{X}_{22}$ by solving lower triangular system $\mathbf{A}_{21} - \mathbf{L}_{21}\mathbf{X}_{11} = \mathbf{L}_{22}\mathbf{X}_{21}$ and $\mathbf{A}_{22} - \mathbf{L}_{21}\mathbf{X}_{12} = \mathbf{L}_{22}\mathbf{X}_{22}$, respectively. From the above procedures we can clearly see that getting columns $[\mathbf{X}_{11}, \mathbf{X}_{21}]^T$ and $[\mathbf{X}_{12}, \mathbf{X}_{22}]^T$ represents two independent processes, which can be parallelized. Similarly, by refining $\mathbf{Y}$ as

$$\begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{bmatrix}, \tag{14}$$

getting rows $[\mathbf{Y}_{11}, \mathbf{Y}_{12}]$ and $[\mathbf{Y}_{21}, \mathbf{Y}_{22}]$ represents two independent processes. Therefore, if the size of $\mathbf{X}$ or $\mathbf{Y}$, namely, $\mathbf{U}_{12}$ or $\mathbf{L}_{21}$ in the $\mathcal{H}$-LU context is larger than $N/2^K$, their solving processes will be dealt with by multiple nodes.

After solving processes of triangular systems is done, the updating for $\mathbf{Z}_{22}$, namely, the procedure (iv) of $\mathcal{H}$-LU, is taking place. We here use UPDATE($l$) to denote any of these procedures on level $l$. Rewriting it as

$$\widehat{\mathbf{Z}} = \mathbf{Z} - \mathbf{AB}, \tag{15}$$

apparently, this is a typical matrix addition and multiplication procedure, which can be parallelized through the addition and multiplication of subblocks when it is necessary:

$$\begin{aligned} &\begin{bmatrix} \widehat{\mathbf{Z}}_{11} & \widehat{\mathbf{Z}}_{12} \\ \widehat{\mathbf{Z}}_{21} & \widehat{\mathbf{Z}}_{22} \end{bmatrix} \\ &\quad = \begin{bmatrix} \mathbf{Z}_{11} & \mathbf{Z}_{12} \\ \mathbf{Z}_{21} & \mathbf{Z}_{22} \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} \\ &\quad = \begin{bmatrix} \mathbf{Z}_{11} - \mathbf{A}_{11}\mathbf{B}_{11} - \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{Z}_{12} - \mathbf{A}_{11}\mathbf{B}_{12} - \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{Z}_{21} - \mathbf{A}_{21}\mathbf{B}_{11} - \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{Z}_{22} - \mathbf{A}_{21}\mathbf{B}_{12} - \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}. \end{aligned} \tag{16}$$

In the final form of (16), the matrix computation for each of the four subblocks is an independent process and can be dealt with by different nodes simultaneously. When $\mathbf{Z}_{22}$ is updated, the LU factorization is applied to $\mathbf{Z}_{22}$ for getting $\mathbf{L}_{22}$ and $\mathbf{U}_{22}$. This procedure is identical to the $\mathcal{H}$-LU factorization for $\mathbf{Z}_{11}$, so any parallel scheme employed for factorizing $\mathbf{Z}_{11}$ is also applicable for factorizing $\mathbf{Z}_{22}$.

The overall strategy for parallelizing $\mathcal{H}$-LU factorization is illustrated in Figure 3. Before the LU factorization on level $K$—$\mathcal{H}$-LU($K$) is finished, only one node $P_1$ takes the computing job. Afterwards, two nodes $P_1$ and $P_2$ are used to get $\mathbf{U}_{12}$ and $\mathbf{L}_{21}$ simultaneously for $\mathcal{H}$-LU($K-1$). Then $\mathbf{Z}_{22}$ is updated and factorized for getting $\mathbf{U}_{22}$ and $\mathbf{L}_{22}$ by the process $\mathcal{H}$-LU($K$). For triangular solving process on the $K-2$ level,

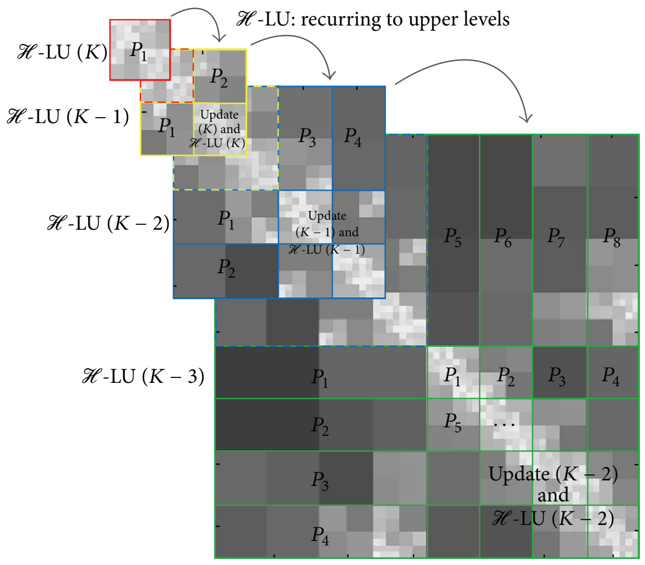
ℋ-LU: recurring to upper levels
ℋ-LU (K)
ℋ-LU (K − 1)
ℋ-LU (K − 2)
ℋ-LU (K − 3)
P1
P2
Update (K) and ℋ-LU (K)
P3
P4
Update (K − 1) and ℋ-LU (K − 1)
P5
P6
P7
P8
...
Update (K − 2) and ℋ-LU (K − 2)

Figure 3: Parallel strategy/approach for $\mathscr{H}$-LU factorization. $P_i$ ($i = 1, 2, 3, \ldots$) denotes the $i$th node.

4 nodes can be used concurrently to get $\mathbf{U}_{12}$ and $\mathbf{L}_{21}$ based on the parallelizable processes of solving the upper and lower triangular systems as we discussed above. Besides, updating $\mathbf{Z}_{22}$ can be parallelized by 4 nodes for 4 square blocks as enclosed by blue line in Figure 3. This parallelization can be applied to coarser levels in general. As for factorization $\mathscr{H}$-LU($l$) ($l < K$), $2^{K-l}$ nodes can be used for parallelizing the process of solving triangular systems and $4^{K-l}$ nodes can be used for parallelizing the procedure of updating $\mathbf{Z}_{22}$. During and after $\mathscr{H}$-LU factorization, the system matrix is kept in the form of distributive format as the same as that for forward/impedance matrix shown in Figure 2. Under the perspective of data storage, those row-blocks are updated by $\mathscr{H}$-LU factorization gradually. It should be noted that, for the nodes participating in parallel computing, the blocks they are dealing with may not relate to the row-blocks they stored. Therefore, necessary communication for blocks transfer occurs among multiple nodes during the factorization.

*4.3. Theoretical Efficiency of MPI Parallelization.* To analyze the parallelization efficiency under an ideal circumstance, we assume that the load balance is perfectly tuned and the computer cluster is equipped with high-speed network; thus there is no latency for remote response and the internode communication time is negligible. Suppose there are $P$ nodes and each node has a constant number of OpenMP threads. The parallelization efficiency is simply defined as

$$\eta_P = \frac{T_1}{PT_P}, \tag{17}$$

in which $T_i$ is the processing time with $i$ nodes.

Apparently, for the process of generating forward/impedance matrix, $\eta_P$ could be 100% if the number of operations for filling each row-block under $P$-nodes parallelization $\mathscr{O}_P^{\text{rowfill}}$ is $1/P$ of the number of operations for filling the whole matrix $\mathscr{O}_1^{\text{fill}}$ by single node, since $T_1 \propto \mathscr{O}_1^{\text{fill}}$ and $T_P \propto \mathscr{O}_P^{\text{rowfill}} = \mathscr{O}^{\text{fill}}/P$ are true if the CPU time for taking single operation is invariant. This condition can be easily satisfied to a high degree when $P$ is not too large. When $P$ is large enough, to divide the system matrix into $P$ row-blocks may force the depth of $\mathscr{H}$-matrices structure inadequately that consequently increases the overall complexity of $\mathscr{H}$-matrices method. In other words,

$$\mathscr{O}_P^{\text{rowfill}} > \frac{\mathscr{O}_1^{\text{fill}}}{P}, \quad (\text{for large } P). \tag{18}$$

Under this situation, the parallelization becomes inefficient.

For the process of $\mathscr{H}$-LU factorization, by assuming that $P = 2^K$, the total number of operations under $P$-nodes parallelization $\mathscr{O}_P^{\text{LU}}$ is

$$\begin{aligned} \mathscr{O}_P^{\text{LU}} \approx{} & P \cdot \mathscr{O}[\mathscr{H}\text{-LU}(K)] \\ & + P(P-1) \cdot \mathscr{O}[\mathscr{H}\text{-TriSolve}(K)] \\ & + \frac{P^3}{6} \mathscr{O}[\mathscr{H}\text{-AddProduct}(K)], \end{aligned} \tag{19}$$

where $\mathscr{H}$-LU($K$), $\mathscr{H}$-TriSolve($K$), and $\mathscr{H}$-AddProduct($K$) denote the processes for LU factorization, triangular system solving, and matrices addition and multiplication in $\mathscr{H}$-format in the $K$th level of $\mathscr{H}$-matrices structure, respectively. Their computational complexity is $O(n^\alpha)$ ($2 \le \alpha < 3$), in which $n = N/P$. However, because of parallelization, some operations are done simultaneously. Hence, the nonoverlap number of operations $\mathscr{O}_P^T$ that represents the actual wall time of processing is

$$\begin{aligned} \mathscr{O}_P^T \approx{} & P \cdot \mathscr{O}[\mathscr{H}\text{-LU}(K)] \\ & + \frac{PK}{2} \cdot \mathscr{O}[\mathscr{H}\text{-TriSolve}(K)] \\ & + \frac{P^2}{4} \mathscr{O}[\mathscr{H}\text{-AddProduct}(K)]. \end{aligned} \tag{20}$$

Additionally, the number of operations without parallelization for $\mathscr{H}$-LU factorization $\mathscr{O}_1^{\text{LU}}$ is definitely no larger than $\mathscr{O}_P^{\text{LU}}$ for the potential increased overall complexity by parallelization. Therefore, the parallelization efficiency is

$$\eta_P = \frac{\mathscr{O}_1^{\text{LU}}}{P \cdot \mathscr{O}_P^T} \le \frac{\mathscr{O}_P^{\text{LU}}}{P \cdot \mathscr{O}_P^T}. \tag{21}$$

When $P$ goes larger, the result is approximately

$$\eta_P \le \frac{(P^3/6)\, \mathscr{O}[\mathscr{H}\text{-AddProduct}(K)]}{P \cdot (P^2/4)\, \mathscr{O}[\mathscr{H}\text{-AddProduct}(K)]} \approx \frac{2}{3}. \tag{22}$$

According to our analysis, we should say that, for a given size of IE problems, there is an optimal $P$ for best implementation. If $P$ is too small, the solving process cannot be fully parallelized. On the other hand, larger $P$ will make the width of the row-blocks narrower that eliminate big LR subblocks and cause $\mathcal{H}$-matrices structure to be flatter, which will consequently impair the low complexity feature of $\mathcal{H}$-matrices method.

Similarly, the number of OpenMP threads $M$ has an optimal value too. Too small $M$ will restrict OpenMP parallelization, and too large $M$ is also a waste because the intranode parallel computing only deals with algebraic arithmetic for subblocks of relatively small size; excessive threads will not improve its acceleration and lower the efficiency.

## 5. Numerical Results

The cluster we test our code on has 32 physical nodes in total; each node has 64 GBytes of memory and four quad-core Intel Xeon E5-2670 processors with 2.60 GHz clock rate. For all tested IE cases, the discrete mesh size is $0.1\lambda$, in which $\lambda$ is the wavelength.

*5.1. Parallelization Efficiency.* First, we test the parallelization scalability for solving scattering problem of PEC spheres with different electric sizes. The radii of these spheres are $R = 2.5\lambda$, $R = 5.0\lambda$, and $R = 10.0\lambda$, respectively. The total numbers of unknowns for these three cases are 27,234, 108,726, and 435,336, respectively. Figure 4 shows the efficiency for the total time solved by 1, 2, 4, 8, 16, 32, 64, and 128 MPI nodes, with 4 OpenMP threads for each node. The definition of parallelization efficiency is the same as formula (17); thus in this test we only consider the efficiency of pure MPI parallelization. Unsurprisingly, for parallelization with large number of nodes, bigger cases have higher efficiencies because of better load balance and less distortion to the $\mathcal{H}$-matrices structure. Besides, since the $\mathcal{H}$-LU factorization dominates the solving time consumption, these tested efficiencies of bigger cases are close to the maximum efficiency of theoretical one $\eta_P^{\mathrm{LU}} = 2/3$, which demonstrates the good parallelizing quality of our code.

Next, we investigate the hybrid parallelization efficiency for different MPI nodes ($P$)/OpenMP threads ($M$) proportion. We employ all 512 cores in cluster to solve the 3 PEC sphere cases above in four scenarios, from pure MPI parallelization (1 OpenMP thread for each MPI node, large $P/M$) to heavy OpenMP parallelization embedded in MPI (16 OpenMP threads for each MPI node, small $P/M$). Here we slightly change the definition of parallelization efficiency as

$$\eta_{(P,M)} = \frac{T_{(1,1)}}{P \cdot M \cdot T_{(P,M)}}, \qquad (23)$$

in which $T_{(i,j)}$ denotes the total solving time parallelized by $i$ MPI nodes with $j$ OpenMP threads for each node. The results are presented in Figure 5. We can clearly see that both pure MPI parallelization and heavy OpenMP hybrid parallelization have poorer efficiency compared to that of moderate OpenMP hybrid parallelization.

Ideally, larger $P/M$ should give better efficiency for bigger cases. But, in reality, since larger $P/M$ can result in more frequent MPI communication, more time is required for communication and synchronization, and so forth. Besides, large $P$ also impairs the low complexity feature of $\mathcal{H}$-matrices method regardless of the size of problems. Therefore, pure MPI parallelization does not show superiority over hybrid parallelization in our test. Referring to the optimal number of OpenMP threads per node, it is mainly determined by the size of smallest blocks in $\mathcal{H}$-matrices which is preset, since OpenMP efficiency strongly relates to the parallelization of full or LR blocks addition and multiplication at low levels. For all cases of this test, the size of smallest blocks is between 50 and 200, so their optimal number of OpenMP threads per node turns out to be 4. However, we can expect the optimal number of OpenMP threads to be larger (smaller) if the size of the smallest blocks is larger (smaller). For the case of solving PEC sphere of $R = 10.0\lambda$ with 128

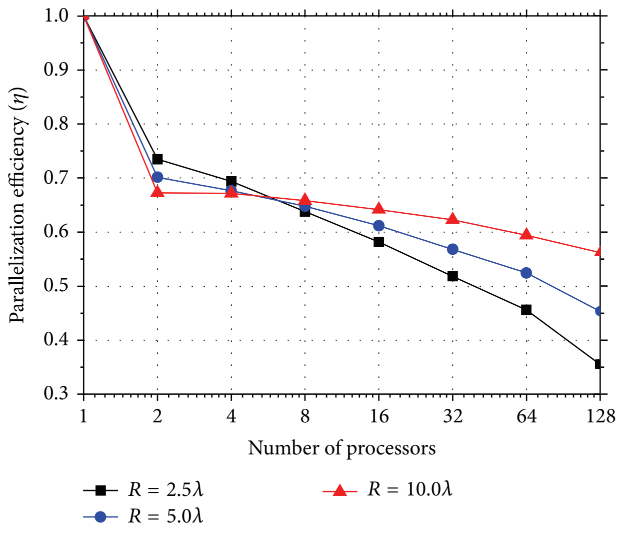

FIGURE 4: Parallelization efficiency for solving scattering problems of PEC sphere of different electric sizes.

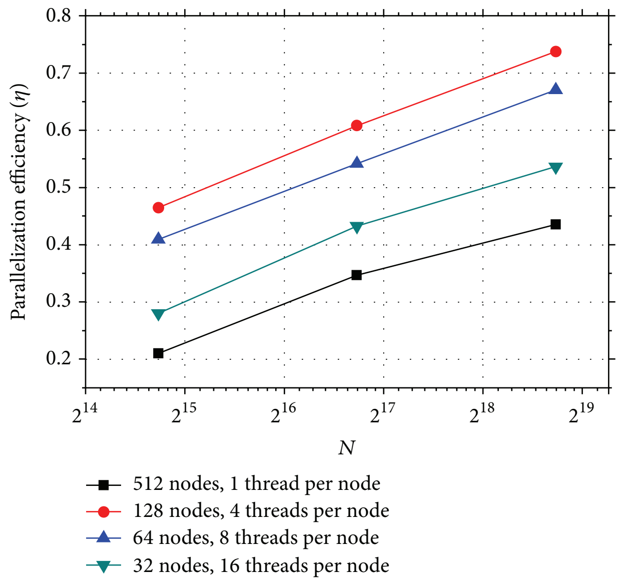

FIGURE 5: Parallelization efficiency of different MPI nodes/OpenMP threads allocation.

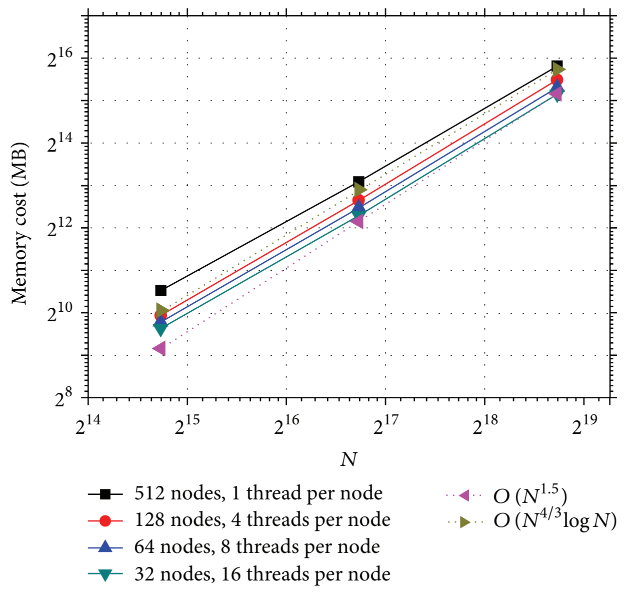

FIGURE 6: Memory cost for different MPI nodes/OpenMP threads proportion.

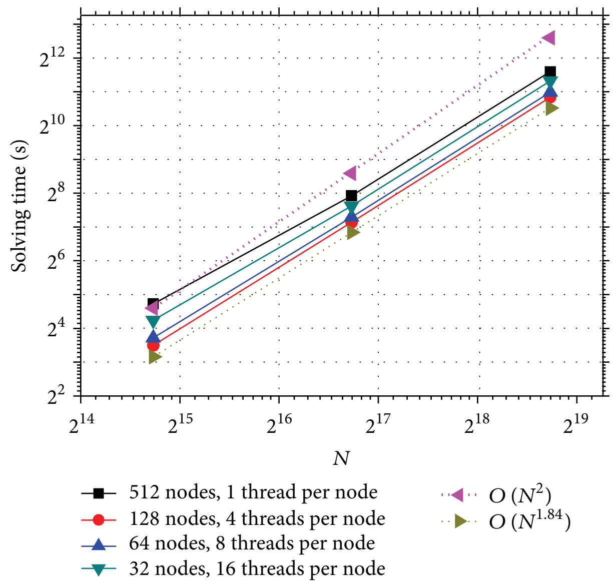

FIGURE 7: Solving time for different MPI nodes/OpenMP threads proportion.

nodes, 4 threads per node, the parallelization efficiency in this test is 0.737, which is higher than that of 0.562 in previous test. This is because, by our new definition of efficiency, the OpenMP parallelization is taken into account. These results also indicate the superiority of MPI-OpenMP hybrid parallelization over pure MPI parallelization.

Figures 6 and 7 show the memory cost and solving time for different problem sizes under different MPI nodes/ OpenMP threads proportion. The memory cost complexity generally fits the $O(N^{1.5})$ trend, which has also been demonstrated in sequential FDS [15, 17, 19]. Since the parallelization efficiency gradually rises along with the increase of problem size as shown above, the solving time complexity is about $O(N^{1.84})$ instead of $O(N^2)$ exhibited in sequential scenario [15, 17, 19].

*5.2. PEC Sphere.* In this part, we test the accuracy and efficiency of our proposed parallel FDS and compare its results to analysis solutions. A scattering problem of one PEC sphere with radius $R = 30.0\lambda$ is solved. After discretization, the total number of unknowns of this case is 3,918,612. For hybrid parallelizing, 128 MPI nodes are employed, with 4 OpenMP threads for each node. The building time for forward system matrix is 4,318.7 seconds, $\mathscr{H}$-LU factorization time is 157,230.2 seconds, and solving time for each excitation (RHS) is only 36.6 seconds. The total memory cost is 987.1 GBytes. The numerical bistatic RCS results agree with the analytical Mie series very well, as shown in Figure 8.

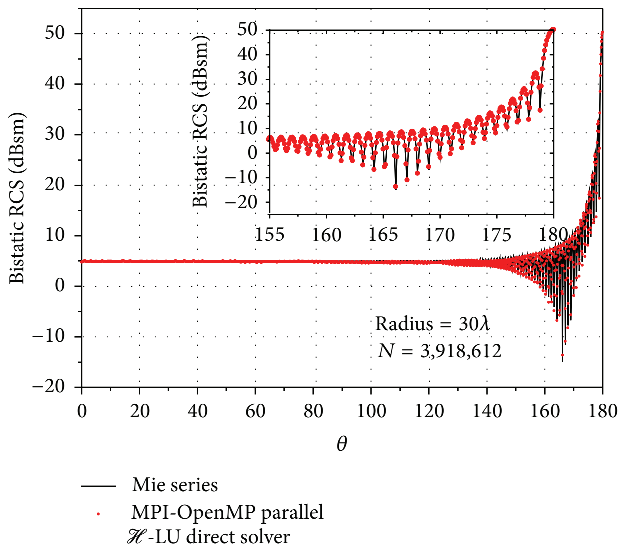

FIGURE 8: Bistatic RCS of PEC sphere with $R = 30.0\lambda$ solved by proposed hybrid parallel FDS.

*5.3. Airplane Model.* Finally, we test our solver with a more realistic case. The monostatic RCS of an airplane model with the dimension of $60.84\lambda \times 35.00\lambda \times 16.25\lambda$ is calculated. After discretization, the total number of unknowns is 3,654,894. The building time for forward system matrix is 3548.5 seconds; $\mathscr{H}$-LU factorization time is 110,453.4 seconds; and the peak memory cost is 724.3 GBytes. With the LU form of the system matrix, we directly calculate the monostatic RCS for 10,000 total vertical incident angles (RHS) by backward substitution. The average time cost for calculating each incident angle is 31.2 seconds. The RCS result is compared with that obtained by FMM iterative solver of 720 incident angles. Although FMM iterative solver only costs 112.4 GBytes for memory, the average iteration time of solving back scattering for each angle is 938.7 seconds with SAI preconditioner [23], which is not practical for solving thousands of RHS. From Figure 9 we can see that these two results agree with each other well.

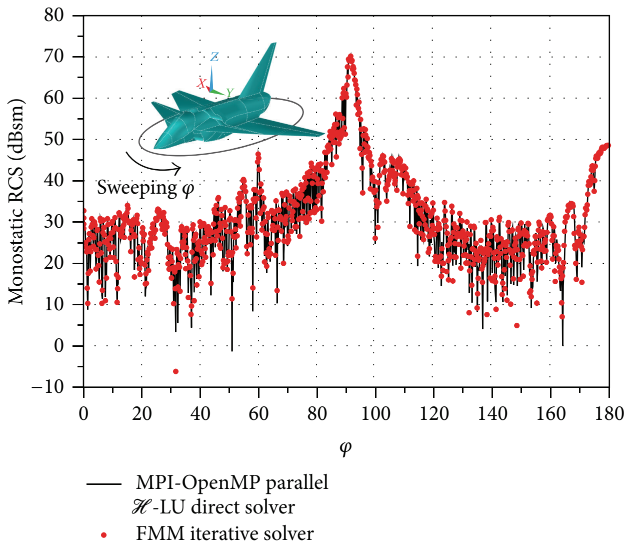

Figure 9: Monostatic RCS of airplane model solved by proposed hybrid parallel FDS and FMM iterative solver.

## 6. Conclusion

In this paper, we present an $\mathscr{H}$-matrices based fast direct solver accelerated by both MPI multinodes and OpenMP multithreads parallel techniques. The macrostructural implementation of $\mathscr{H}$-matrices method and $\mathscr{H}$-LU factorization is parallelized by MPI, while the microalgebraic computation for matrix blocks and vector segments is parallelized by OpenMP. Despite the sequential nature of $\mathscr{H}$-matrices direct solving procedure, this proposed hybrid parallel strategy shows good parallelization efficiency. Numerical results also demonstrate excellent accuracy and superiority in solving massive excitations (RHS) problem for this parallel direct solver.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work is supported by the China Scholarship Council (CSC), the Fundamental Research Funds for the Central Universities of China (E022050205), the NSFC under Grant 61271033, the Programme of Introducing Talents of Discipline to Universities under Grant b07046, and the National Excellent Youth Foundation (NSFC no. 61425010).

## References

[1] A. Taflve, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Artech House, Norwood, Mass, USA, 1995.

[2] J.-M. Jin, *The Finite Element Method in Electromagnetics*, Wiley, New York, NY, USA, 2002.

[3] R. F. Harrington, *Field Computation by Moment Methods*, MacMillan, New York, NY, USA, 1968.

[4] R. Coifman, V. Rokhlin, and S. Wandzura, "The fast multiple method for the wave equation: a pedestrian prescription," *IEEE Antennas and Propagation Magazine*, vol. 35, no. 3, pp. 7–12, 1993.

[5] J. Song, C. C. Lu, and W. C. Chew, "Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 10, pp. 1488–1493, 1997.

[6] J. Hu, Z. P. Nie, J. Wang, G. X. Zou, and J. Hu, "Multilevel fast multipole algorithm for solving scattering from 3-D electrically large object," *Chinese Journal of Radio Science*, vol. 19, no. 5, pp. 509–524, 2004.

[7] E. Michielssen and A. Boag, "A multilevel matrix decomposition algorithm for analyzing scattering from large structures," *IEEE Transactions on Antennas and Propagation*, vol. 44, no. 8, pp. 1086–1093, 1996.

[8] J. M. Rius, J. Parrón, E. Úbeda, and J. R. Mosig, "Multilevel matrix decomposition algorithm for analysis of electrically large electromagnetic problems in 3-D," *Microwave and Optical Technology Letters*, vol. 22, no. 3, pp. 177–182, 1999.

[9] H. Guo, J. Hu, and E. Michielssen, "On MLMDA/butterfly compressibility of inverse integral operators," *IEEE Antennas and Wireless Propagation Letters*, vol. 12, pp. 31–34, 2013.

[10] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, "AIM: adaptive integral method for solving large-scale electromagnetic scattering and radiation problems," *Radio Science*, vol. 31, no. 5, pp. 1225–1251, 1996.

[11] W. Hackbusch and B. Khoromskij, "A sparse matrix arithmetic based on $H$-matrices. Part I: Introduction to $H$-matrices," *Computing*, vol. 62, pp. 89–108, 1999.

[12] S. Börm, L. Grasedyck, and W. Hackbusch, *Hierarchical Matrices*, Lecture Note 21, Max Planck Institute for Mathematics in the Sciences, 2003.

[13] W. Chai and D. Jiao, "An **calH**$^2$-matrix-based integral-equation solver of reduced complexity and controlled accuracy for solving electrodynamic problems," *IEEE Transactions on Antennas and Propagation*, vol. 57, no. 10, pp. 3147–3159, 2009.

[14] W. Chai and D. Jiao, "$H$- and $H^2$-matrix-based fast integral-equation solvers for large-scale electromagnetic analysis," *IET Microwaves, Antennas & Propagation*, vol. 4, no. 10, pp. 1583–1596, 2010.

[15] H. Guo, J. Hu, H. Shao, and Z. Nie, "Hierarchical matrices method and its application in electromagnetic integral equations," *International Journal of Antennas and Propagation*, vol. 2012, Article ID 756259, 9 pages, 2012.

[16] A. Heldring, J. M. Rius, J. M. Tamayo, and J. Parrón, "Compressed block-decomposition algorithm for fast capacitance extraction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 265–271, 2008.

[17] A. Heldring, J. M. Rius, J. M. Tamayo, J. Parrón, and E. Ubeda, "Multiscale compressed block decomposition for fast direct solution of method of moments linear system," *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 2, pp. 526–536, 2011.

[18] K. Zhao, M. N. Vouvakis, and J.-F. Lee, "The adaptive cross approximation algorithm for accelerated method of moments computations of EMC problems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 47, no. 4, pp. 763–773, 2005.

[19] J. Shaeffer, "Direct solve of electrically large integral equations for problem sizes to 1 M unknowns," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 8, pp. 2306–2313, 2008.

[20] J. Song, "Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 10, pp. 1488–1493, 1997.

[21] M. Benzi, "Preconditioning techniques for large linear systems: a survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[22] J. Lee, J. Zhang, and C.-C. Lu, "Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems," *Journal of Computational Physics*, vol. 185, no. 1, pp. 158–175, 2003.

[23] J. Lee, J. Zhang, and C.-C. Lu, "Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 9, pp. 2277–2287, 2004.

[24] G. H. Golub and C. F. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Md, USA, 1996.

[25] R. J. Adams, Y. Xu, X. Xu, J.-s. Choi, S. D. Gedney, and F. X. Canning, "Modular fast direct electromagnetic analysis using local-global solution modes," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 8, pp. 2427–2441, 2008.

[26] Y. Zhang and T. K. Sarkar, *Parallel Solution of Integral Equation-Based EM Problems in the Frequency Domain*, John Wiley & Sons, Hoboken, NJ, USA, 2009.

[27] J.-Y. Peng, H.-X. Zhou, K.-L. Zheng et al., "A shared memory-based parallel out-of-core LU solver for matrix equations with application in EM problems," in *Proceedings of the 1st International Conference on Computational Problem-Solving (ICCP '10)*, pp. 149–152, December 2010.

[28] M. Bebendorf, "Hierarchical LU decomposition-based preconditioners for BEM," *Computing*, vol. 74, no. 3, pp. 225–247, 2005.

[29] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Computational Science & Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[30] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, vol. 1, The MIT Press, 1999.

[31] S. M. Rao, D. R. Wilton, and A. W. Glisson, "Electromagnetic scattering by surfaces of ar-bitrary shape," *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 3, pp. 409–418, 1982.

[32] A. W. Glisson and D. R. Wilton, "Simple and efficient numerical methods for problems of electromagnetic radiation and scattering from surfaces," *IEEE Transactions on Antennas and Propagation*, vol. 28, no. 5, pp. 593–603, 1980.

[33] R. Barrett, M. Berry, T. F. Chan et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, vol. 43, SIAM, Philadelphia, Pa, USA, 1994.

[34] E. Anderson, Z. Bai, C. Bischof et al., *LAPACK Users' Guide*, vol. 9, SAIM, 1999.