Hindawi
International Journal of Antennas and Propagation
Volume 2017, Article ID 9173062, 12 pages
https://doi.org/10.1155/2017/9173062

Research Article

High Performance Computing of Complex Electromagnetic Algorithms Based on GPU/CPU Heterogeneous Platform and Its Applications to EM Scattering and Multilayered Medium Structure

Zhe Song,[1] Xing Mu,[2] and Hou-Xing Zhou[1]

[1]*The State Key Laboratory of Millimeter Waves, Southeast University, Nanjing, China*
[2]*iOpenLink Corp., Nanjing, China*

Correspondence should be addressed to Zhe Song; zhe.song@seu.edu.cn

Received 30 April 2017; Accepted 6 June 2017; Published 30 July 2017

Academic Editor: Han Guo

The fast and accurate numerical analysis for large-scale objects and complex structures is essential to electromagnetic simulation and design. Comparing to the exploration in EM algorithms from mathematical point of view, the computer programming realization is coordinately significant while keeping up with the development of hardware architectures. Unlike the previous parallel algorithms or those implemented by means of parallel programming on multicore CPU with OpenMP or on a cluster of computers with MPI, the new type of large-scale parallel processor based on graphics processing unit (GPU) has shown impressive ability in various scenarios of supercomputing, while its application in computational electromagnetics is especially expected. This paper introduces our recent work on high performance computing based on GPU/CPU heterogeneous platform and its application to EM scattering problems and planar multilayered medium structure, including a novel realization of OpenMP-CUDA-MLFMM, a developed ACA method and a deeply optimized CG-FFT method. With fruitful numerical examples and their obvious enhancement in efficiencies, it is convincing to keep on deeply investigating and understanding the computer hardware and their operating mechanism in the future.

1. Introduction

Demand boosting in high performance computing algorithms has been one of the most significant topics in computational electromagnetics (CEM). With the well-known merit of much fewer unknowns than finite-difference time-domain (FDTD) and finite element method (FEM), the method of moments (MoM) has been widely used in EM scattering problems of large-scale complex objects and full-wave analysis in multilayered structures during the past decades [1]. With the development in computer science and computational mathematics, many innovative algorithms have been established to accelerate the solving procedure of matrix equation. Several methods have been widely used in CEM, for example, multilevel fast multipole method (MLFMM) [2], adaptive integral method (AIM) [3, 4], fast Fourier transform based method (p-FFT, IE-FFT) [5–7], adaptive cross approximation method (ACA) [8–10], and conjugate gradient fast Fourier transform method (CG-FFT) [11]. Some of them have been successfully implemented in commercial software. Naturally, parallelization is the obvious way to make a further enhancement to efficiency [12]. In early stage, the implementation of parallel computing was mainly based on CPU platforms, such as PC-cluster with MPI and multicore CPU workstation with OpenMP. The methodology of the parallelization based on a single PC platform is the massive use of threads while sharing the same memory. The bottleneck appears immediately as the explosive growth of communication load among threads. Therefore, the theoretical speedup when using multiple processors can be predicted by the famous Amdahl's Law [13].

Table 1: Detailed parameters of GPU platform.

Parameters	Fermi GF104	Kepler GK104	Kepler GK110
Compute ability	2.1	3.0	3.5
Threads/warp	32	32	32
Max warps/multiprocessor	48	64	64
Max threads/multiprocessor	1536	2048	2048
Max thread Blocks/Multiprocessor	8	16	16
32-bit registers/multiprocessor	32768	65536	65536
Max registers/thread	63	63	255
Max threads/thread block	1024	1024	1024
Max X-grid dimension	$2^{16}-1$	$2^{32}-1$	$2^{32}-1$
Dynamic parallelism	×	×	√
Used in numerical algorithms	—	CG-FFT	FMM; ACA

Thanks to the upgrading in hardware architecture of GPU, it is possible to allocate much more transistors devoted to data processing, that is, arithmetic logic unit (ALU), rather than data caching or flow control [14]. This makes the GPU most involved in highly parallel, multiprocess, and many-core computing with very high memory bandwidth. The combination of GPU and CPU can be realized by the concept of heterogeneous computing, which includes other popular hardware modules, that is, field-programmable gate array (FPGA) and digital signal processing (DSP). The programming combination between GPU and CPU can fully depend on the easy-to-use language CUDA-C/PTX. To the authors' knowledge, the GPU/CPU heterogeneous platform is the most popular choice especially in CEM, such as the GPU-based FDTD [15], MLFMM [16–19], AIM [20], P-FFT [21], MoM [22, 23], and higher-order MoM [24]. In 2013, an impressive implementation of MLFMM by OpenMP-CUDA was realized on Fermi architecture (NVIDIA Tesla C2050), which achieved much higher performance than those before [16].

In this paper, our recent works on high performance computing based on GPU/CPU heterogeneous platform are introduced with the following:

(1) A novel realization of OpenMP-CUDA-MLFMM method for EM scattering problem, in which the near-field matrix filling and the sparse matrix-vector production (MVP) are optimized, together with a warp-level parallel scheme for aggregation/disaggregation and the use of texture memory for 2D local interpolation/anterpolation

(2) A developed ACA method for EM scattering problem, in which the near-field matrix filling realizes a 100% efficiency enhancement and a thread-block-level parallel and a register-reusable schemes are applied to matrix compression and far-field MVP, respectively; also, a double-buffer technique for double precision is proposed to further enhance the efficiency of MVP computation [25]

(3) A deeply optimized CG-FFT method for planar multilayered structure, in which the hardware instruction "*__shufl_down*" instead of shared memory was firstly used in the summation of MVP [26] and the current distribution of a large-scale flat lens was obtained efficiently.

It is worth emphasizing that the domain decomposition method (DDM) is not considered in this paper, by which means all the optimization comes from arithmetic programming. Corresponding to numerical examples, the detailed parameters of GPU platform can be found in Table 1, from which one can also discover the changes in computing ability brought by hardware's upgrading. The paper is organized as follows. With a brief introduction in numerical algorithms for MLFMM, ACA, and CG-FFT, Section 2 introduces the proposed and optimized methods from the programming point of view. Section 3 deals with the analysis of fruitful numerical examples. Section 4 presents the conclusions. Some technical terms used in this paper relating to NVIDIA GPU architectures and CUDA-C language, such as thread, threadblock, grid, warp, kernel, instruction, streaming multiprocessor (SMX), and stream processor (SP), are essential for understanding and can be found in [14].

2. Numerical Analysis of MLFMM, ACA, and CG-FFT

2.1. Principle of MLFMM and Its Optimization on GPU. As pointed out in [2], either the electric-field integral equation (EFIE) or the magnetic-field integral equation (MFIE) encounters the interior resonance at certain frequencies if exterior medium is lossless. The combined-field integral equation (CFIE) is always used to eliminate the problem, in which the matrix element can be calculated by

$$Z_{ij} = \alpha\eta \left\langle \vec{f}_i, K\left[\vec{f}_j\right]\right\rangle + (1-\alpha)\left\langle \vec{f}_i, L\left[\vec{f}_j\right]\right\rangle, \quad (1)$$

where the EFIE and MFIE are presented as

$$\left\langle \vec{f}_i, L\left[\vec{f}_j\right]\right\rangle = jk\eta \int_{S_i} \vec{f}_i\left(\vec{r}_i\right)\cdot ds \int_{S_j}\left(\bar{\bar{I}} + \frac{1}{k^2}\nabla\nabla\right)G\left(\vec{r}_i, \vec{r}_j'\right) \cdot \vec{f}_j\left(\vec{r}_j'\right) ds',$$

$$\left\langle \vec{f}_i, K\left[\vec{f}_j\right]\right\rangle = -\int_{S_i}\left(\vec{f}_i\left(\vec{r}_i\right)\times \hat{n}_i\right)\cdot ds \cdot \int_{S_j} \nabla G\left(\vec{r}_i, \vec{r}_j'\right)\times \vec{f}_j\left(\vec{r}_j'\right) ds'. \quad (2)$$

By using the addition theorem of Green's function, formula (1) can be rewritten in the form named as "aggregation–translation–disaggregation" for EFIE and MFIE:

$$\left\langle \vec{f}_i, D\left[\vec{f}_j\right]\right\rangle = \int_E d^2\hat{k}\vec{V}^{S,P}_{m_l',j}\left(\hat{k}\right)\cdot \alpha_{m_l m_l'}\left(\hat{k}\right)\cdot \vec{V}^{F,P}_{m_l',i}\left(\hat{k}\right),$$

$$\vec{V}^{S,D}_{m_l',j}\left(\hat{k}\right) = \int_{S_j} ds'\left(\bar{\bar{I}} - \hat{k}\hat{k}\right)\cdot \vec{f}_j\left(\vec{r}_j'\right)e^{-j\vec{k}\cdot\vec{r}_{m_l'j}},$$

$$\alpha_{m_l m_l'}\left(\hat{k}\right) = \frac{jk\eta}{4\pi}T_B\left(\hat{k}, \vec{r}_{m_l m_l'}\right), \quad (3)$$

$$\vec{V}^{F,D}_{m_l,i}\left(\hat{k}\right) = \int_{S_i} ds\left(\bar{\bar{I}} - \hat{k}\hat{k}\right)\cdot \vec{f}_i\left(\vec{r}_i\right)e^{-j\vec{k}\cdot\vec{r}_{im_l}},$$

$$T_B\left(\hat{k}, \vec{r}\right) \doteq \frac{k}{4\pi}\sum_{l=0}^{B}(-j)^{l+1}\cdot(2l+1)\,h_l^{(2)}(kr)\,P_l\left(\hat{k}\cdot\vec{r}\right),$$

where $D = K$ or L, $h_l^{(2)}(\cdot)$, and $P_l(\cdot)$ stand for the l-order Hankel function of the second kind and the l-order Legendre function, respectively. T_B is the truncation length and related to the diameter of the cubic in lth layer and the numerical accuracy.

Since the generation of near-field matrix is a highly intensive work in computation of MLFMM, the implementation of GPU or multi-GPU can significantly speedup the procedure even with RWG-to-RWG rather than triangle-to-triangle scheme. All the RWG functions are stored as two copies, namely, testing chain and basis chain, and each thread on GPU deals with one or more nonzero matrix elements by accessing corresponding data from the chains. This scheme can easily extend to multi-GPU case by splitting the whole task into subtasks. For example, if there are 2 GPUs, namely, GPU-0 and GPU-1, the chain can be split as $\{f_i^t\}_{i=0}^{N/2-1}$ and $\{f_i^t\}_{i=N/2}^{N}$, which makes each GPU has a $N/2 \times N$ sparse matrix to deal with. Considering the asynchronous launching scheme of kernel functions in CUDA, the control returns to CPU after launch immediately, which means a duty waste for CPU when a thread is working on GPU. Therefore, it is necessary to establish a heterogeneous scheme for coordination between CPU and GPU. For this reason, each near matrix is further divided into two parts with N_G and N_C rows, respectively, as shown in Figure 1. All the tasks can be executed concurrently while satisfying the asynchronous mechanism of GPU/CPU heterogeneous platform. This kind of subdivisions can accelerate the sparse matrix-vector productions (SpMVP). The determination of N_G and N_C cannot be easily derived from mathematics; however, the empirical value can be obtained from numerical simulation and curve fitting, in which the optimum solution can be refined by parameter scanning, as shown in Figure 2. Considering the coexistence of multicore CPU and multi-GPU, n_C and n_G stand for the number of cores in CPU and number of GPU, respectively. The sampling points were obtained from sphere scatterings with radius changing from one to six wavelengths. As a result, for the case of $n_C = 4$ and $n_G = 2$, the optimum solution of $N_C/(N/2)$ equals 0.016 and 0.083 for single and double precision, respectively.

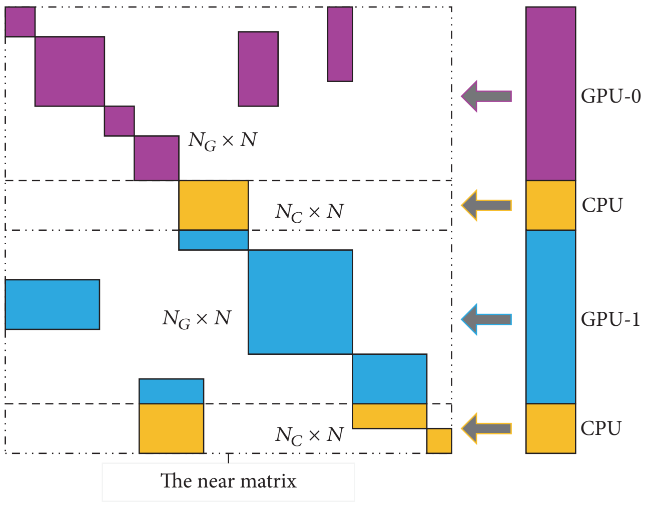
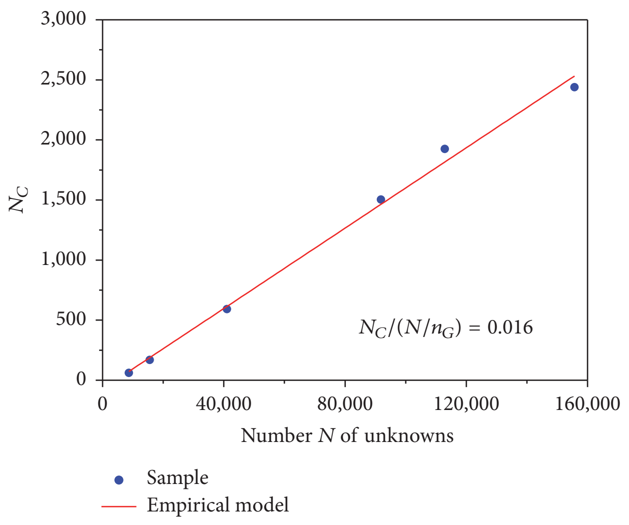
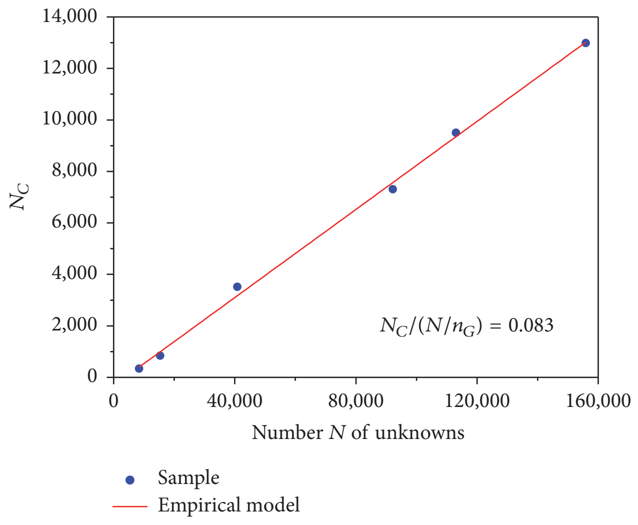


Figure 1: The subdivisions of the near matrix and the task assignments.

The implementation of aggregation and disaggregation at finest level on GPU was proposed by means of allocating a thread to each spectrum point [16]. To increase the utilization of GPU on Kepler architecture (GK110) further, which has a maximum value of 32 threads in one warp, a smart scheme is designed with two steps. Firstly, all the cubes are grouped into each GPU evenly. Secondly, one warp is assigned to a certain cube, in which one thread is assigned to one or more RWG functions immediately. Taking the advantage of the hardware architecture in Kepler GK110, the shuffle instruction "*__shulf()*" can efficiently calculate the reductive summation in register rather than shared memory [14]. Meanwhile, all the 32 threads in a warp can read the data from constant memory according to a certain spectrum point through the read-only cache by using the instruction "*__ldg()*" or "__restrict__".



(a) Single precision



(b) Double precision

FIGURE 2: N_C as a function of N/n_G with $n_G = 2$.

Unlike the strategy of thread-based task assignment proposed for aggregation and disaggregation at coarser level [16], we make a further step in data storage by using the four times larger texture memory in Kepler than Fermi. Since the local inter-/anterpolation accesses neighboring data frequently, it had better store data in a certain pattern form looking like the geometric topology in texture memory. As is pointed out in [27], the texture cache is optimized for 2D spatial locality. Therefore, the threads in one warp reading close-set texture addresses can acquire hardware acceleration and achieve the best performance. With texture memory, the process from coarser level towards finest level can also be executed efficiently. In detail, after all the cubes at child level are grouped evenly in every device (i.e., GPU), a thread will be assigned to give a cube for a spectrum point at this level. By fetching the data from global memory, the information can be organized into a 3D CUDA array that is bound to the layer of texture memory. Then, the cube receives data disaggregated from its parent level by applying local anterpolation within texture memory. All this kind of tasks for cubes at child level can be executed concurrently and all the layers of texture memory assigned can be flushed for the subsequent requests.

2.2. Principle of ACA and Its Optimization on GPU. The ACA is a purely algebraic algorithm [8, 9], aiming at independence from the integral kernel, that is, Green's function, which makes it different from MLFMM. The main idea of ACA is based on the compressing of a rank deficient system, which corresponds to the submatrix representing interactions between two well-separated groups of RWGs. Therefore, the MVP of this kind of submatrices can be efficiently calculated by using the factorial forms [9]. In ACA, the MVP is divided into near- and far-field interactions, which leads to the same data structure as that in MLFMM, that is, the Octree structure, and forms a hierarchical representation of far-field submatrices. Hereby we use $Z^l_{m,n}$ to denote the $m \times n$ submatrix for interaction between group m and group n at the lth-level with indicating a well-separated situation. Then, it can be approximated as

$$Z^l_{m,n} \approx \widetilde{Z}^l_{m,n} = U^l_{m,k} V^l_{k,n}, \tag{4}$$

$$k \approx \operatorname{rank}\left(U^l_{m,n}\right) < \min\{m, n\} \tag{5}$$

which satisfies

$$\left\| Z^l_{m,n} - U^l_{m,k} V^l_{k,n} \right\|_F < \varepsilon_{\text{ACA}} \left\| Z^l_{m,n} \right\|_F, \tag{6}$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix and ε_{ACA} is a given threshold. The workflow of ACA can be depicted as in the following steps with initial value of $k = 1$ and $i_k = 1$.

Step 1. Generate $v^{(k)}$ as the ith row of $Z^l_{m,n}$ and update it.

$$\left(v^{(k)}\right)_j \Longleftarrow \left(v^{(k)}\right)_j - \sum_{i=1}^{k-1} \left(u^{(i)}\right)_k \left(v^{(i)}\right)_j. \tag{7}$$

Step 2. If $(v^{(k)})_j = 0, \forall j$, then $i_k \Leftarrow i_k + 1$ and go back to Step 1; otherwise, normalize the vector with its maximum element,

$$\left|\left(v^{(k)}\right)_{s_k}\right| = \max_j \left\{\left|\left(v^{(k)}\right)_j\right|\right\},$$
$$\left(v^{(k)}\right)_j \Longleftarrow \frac{\left(v^{(k)}\right)_j}{\left(v^{(k)}\right)_{s_k}}, \quad \forall j. \tag{8}$$

Step 3. Generate $u^{(k)}$ as the s_kth row of $Z^l_{m,n}$ and update it,

$$\left(u^{(k)}\right)_j \Longleftarrow \left(u^{(k)}\right)_j - \sum_{i=1}^{k-1} \left(u^{(i)}\right)_j \left(v^{(i)}\right)_{s_k}. \tag{9}$$

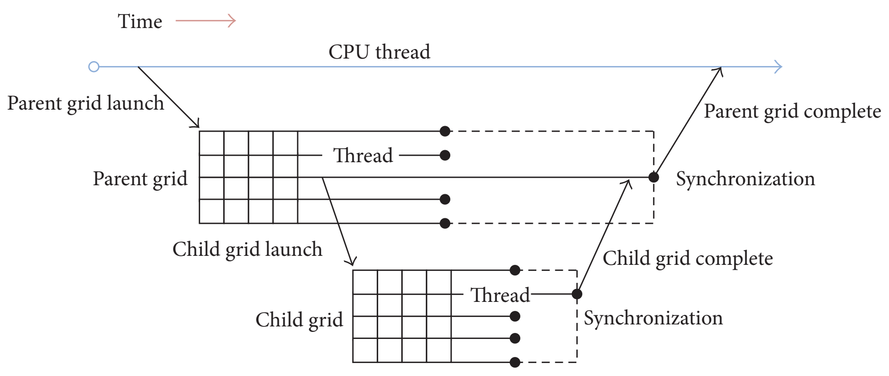


FIGURE 3: Flowchart of dynamic parallelism.

Step 4. Generate $U^l_{m,k}$ and $V^l_{k,n}$, where

$$U^l_{m,k} = \left[u^{(1)}, \ldots, u^{(k)}\right], \quad V^l_{k,n} = \begin{bmatrix} v^{(1)} \\ \vdots \\ v^{(k)} \end{bmatrix}. \tag{10}$$

Step 5. The feedback system will keep updating by $k \Leftarrow k+1$ and $i_k \Leftarrow i_k + 1$ until the condition in (5) is satisfied.

Obviously, it is convenient to employ both threadgrid and threadblock to take charge of one dimension. The total number of threads in each block is selected as 256, by means of 8 warps. The RWG functions, no matter what it is seen as, basis or test function, can be stored in shared memory during the generation of factorial form of a submatrix. Meanwhile, the submatrices U and V are stored in column- and row-major order, respectively. For the task assignment, 2^l warps perform the ACA calculation in one submatrix at the $(L-l)$th level. However, this arrangement has no worry about insufficient blocks or logic error for $l = 0, 1, 2, 3$ until $l = 4$, because there is no synchronization mechanism between different threadblocks. In this paper, we realize one threadblock to one submatrix even at $(l \geq 4)$ by launching a child CUDA kernel function during the generation of new row or column vector [25]. The number of threads enabled in this so-called child kernel is equal to the length of the row or column vector and they calculate only one entry of the vector. It is worth emphasizing that an explicit synchronization between parent and child kernels should be adjusted in advance, which ensures that all the tasks assigned to child kernel are ultimately completed before logic and data operates on parent kernel. This treatment can be realized by the specific "dynamic parallelism" [28] in Kepler GK110 (Table 1) and a vivid flowchart is shown in Figure 3.

After the ACA operation, the MoM matrix is converted into an H-matrix that greatly reduces the storage and speeds up the MVPs. The upcoming procedure is to calculate massive

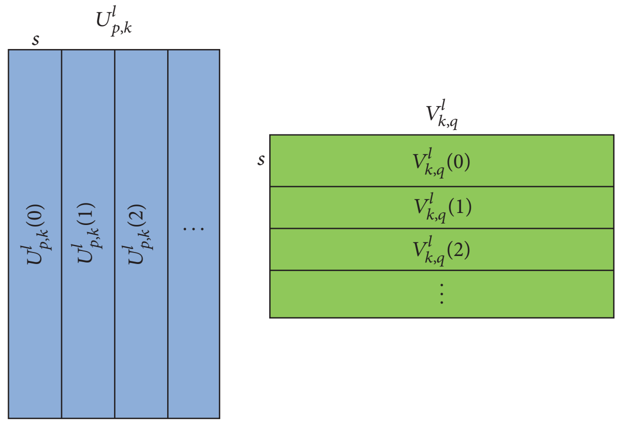


FIGURE 4: The segmentation scheme.

MVPs, which is still time consuming even though the H-matrix is used. In this paper, the batched matrix-vector productions (bMVP) are considered to deal with the massive computation [25]. The main idea of bMVP is to apply the 2D form of threadblock with dim3(pth, gridDimy) to calculate all the MVPs. The column in U and the row in V are performed concurrently, such that all segments (column or row) are with the same size except the last part, as shown in Figure 4. The result of $y = V^l_{k,n}(i)x$ is stored in the shared memory as an intermediate buffer and then the result of $z = U^l_{m,k}(i)y$ is accumulated into the result by reductive summation in shared memory. If we let s be equal to 256, there will be 64 threads in a group with the same ID in y-direction to do multiplication and the vector x can be reused in registers. Therefore, each threadblock can deal with 8 rows of V matrix and at most 32 threadblocks (gridDimy = 32) are needed to calculate the MVP.

2.3. Principle of CG-FFT, Multilayered Green's Functions, and Its Optimization on GPU.

The CG-FFT [11] is one of the most classical methods with iterative procedure and many applications have been found in different areas such as algebra, signal processing, computational geometry, and CEM, especially in

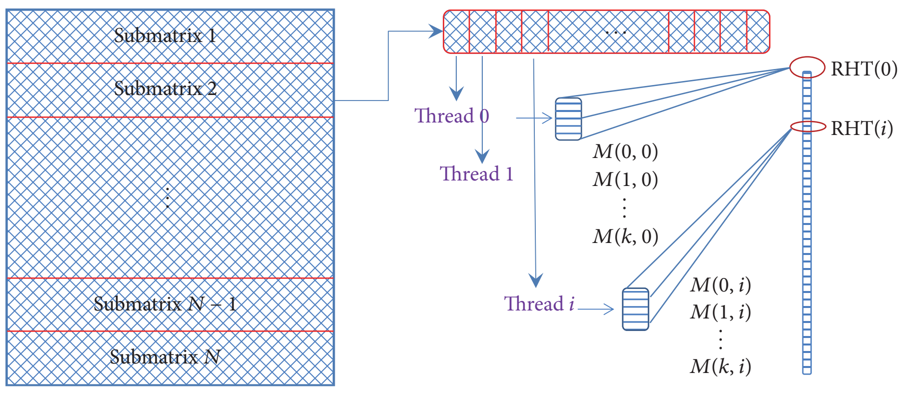


FIGURE 5: Flowchart of matrix partition for threads assignment.

MoM based algorithms [29]. The pseudocode of traditional CG-FFT can be depicted as

```
$x = x_0$,    initial value
$r_0 = b - Ax_0$,
$p_0 = s_0 = A^H r_0$,
$\gamma_0 = \|s_0\|_2^2$
for $k = 0, 1, 2$, until convergence
$q_k = Ap_k$,    MVP
$\alpha_k = \gamma_k / \|q_k\|_2^2$,
$x_{k+1} = x_k + \alpha_k p_k$,
$r_{k+1} = r_k - \alpha_k q_k$,
$s_{k+1} = A^H r_{k+1}$,    MVP
$\gamma_{k+1} = \|s_{k+1}\|_2^2$,
$\beta_k = \gamma_{k+1} / \gamma_k$,
$p_{k+1} = s_{k+1} + \beta_k p_k$,
```

where A^H is the conjugate transpose of matrix A and k stands for the iteration round. Obviously, the MVP operation dominates the most computation source and needs to be optimized. As we mentioned before, after a multiprocessor is assigned one or more threadblocks to execute, it will divide them into warps and each warp will be scheduled by a warp scheduler. The full efficiency can be realized when all 32 threads of a warp agree on their execution path. Considering the Kepler GK110 platform, the maximum threadblocks in one stream multiprocessor are 16 and each SMX contains 192 ALUs. An optimized parallelization can be realized by the so-called "single instruction multiple thread (SIMT)" [14]. The flowchart of this optimized CG-FFT on GPU is depicted in Figure 5, in which the "RHT" stands for right-hand term in MoM matrix equation.

As the matrix is partitioned into N submatrices with at least $(N - 1)$ of them having the same size, each submatrix can also be partitioned into pieces and each piece occupies a certain thread. Just like the realization in OpenMP-CUDA-MLFMM, the hardware instruction "*__shufl*()" is used instead of the shared memory to do reductive summation. This makes the performance even more efficient than the famous CUBLAS library. That is because the shuffle instruction is more efficient and reduces the usage of shared memory with increasing the occupancy.

A brief review of dyadic Green's functions in planar multilayered medium structure is also necessary in this part. The most important one is the mixed potential integral equation (MPIE), based on which the MoM can be established [30–35].

$$\vec{E} = -j\omega\mu_0 \left\langle \overline{\overline{G}}^A ; \vec{J} \right\rangle + \frac{1}{j\omega\varepsilon_0} \nabla \left(\left\langle G^\Phi ; \nabla' \cdot \vec{J} \right\rangle \right), \quad (11)$$

where the dyadic Green's functions can be presented as

$$\overline{\overline{G}}^A = \begin{bmatrix} G_{xx}^A & 0 & G_{xz}^A \\ 0 & G_{xx}^A & G_{yz}^A \\ G_{zx}^A & G_{zy}^A & G_{zz}^A \end{bmatrix}. \quad (12)$$

One of the most efficient ways to evaluate the Green's functions through Sommerfeld Integral (SI) from the spectral to spatial domain is the discrete complex image method (DCIM). According to the analysis in [30], the DCIM is suitable for near-field region ($\rho \le 0.05\lambda$) by calculating three terms, namely, the quasi-static term ($G_{\text{Q-S}}$), the surface wave term (G_{SWP}), and the complex image term (G_{CIM}), as shown below.

$$G(\rho) = \frac{A}{4\pi} \left(G_{\text{Q-S}} + G_{\text{SWP}} + G_{\text{CIM}} \right), \quad (13a)$$

$$G_{\text{Q-S}} = G|_{\rho\to\infty} \cdot \frac{e^{-jk_0\rho}}{\rho}, \quad (13b)$$

$$G_{\text{SWP}} = -2\pi j \sum_{i}^{N_{\text{SWP}}} \text{Res}_i H_0^{(2)} \left(k_{\rho i}\rho \right) k_{\rho i}, \quad (13c)$$

$$G_{\text{CIM}} = \sum_{i=1}^{N_c} a_i \frac{e^{-jk_0 r_i}}{r_i}, \quad (13d)$$

where $H_0^{(2)}$ stand for the zero-order Hankel function of the second kind and Res_i stands for the residues of surface wave poles on the complex spectrum plane. For far-field region ($\rho \geq 0.05\lambda$), both the surface and leaky wave should be considered, together with a branch cut integral contribution, as shown below.

$$\int_{SIP} (\bullet)\, dk_\rho = \int_\Gamma (\bullet)\, dk_\rho - 2\pi j \cdot \left(\sum_i^{N_{\text{SWP}}} R_{k_{\rho i}} + \sum_j^{N_{\text{LWP}}} R_{k_{\rho j}} \right), \tag{14a}$$

$$\int_\Gamma (\bullet)\, dk_\rho = \int_\Gamma \left[\widetilde{G}^+ (k_\rho) + \widetilde{G}^- (k_\rho) \right] \cdot H_n^{(2)} (k_\rho \rho)\, k_\rho dk_\rho, \tag{14b}$$

where SIP stands for the Sommerfeld integral path, N_{SWP} and N_{LWP} stand for the number of extracted surface and leaky wave poles, respectively, Γ stands for the integral path along the branch cut, and $\widetilde{G}^+$ and $\widetilde{G}^-$ are the spectral Green's functions along the top and bottom Riemann sheets of the branch cut, respectively [30]. With accurate evaluation of all the components in dyadic Green's functions, the MPIE can be solved by MoM with RWG functions, in which the matrix element is calculated as

$$\begin{aligned} Z_{mn} = {} & k_0^2 \sum_{n=1}^{N} I_n \int_{T_m} \int_{T_n} \vec{f}_n (\vec{r}\,') \cdot \overline{\overline{G}}^A (\vec{r} \mid \vec{r}\,') \cdot \\ & \cdot \vec{f}_m (\vec{r})\, ds' ds + \sum_{n=1}^{N} I_n \\ & \cdot \int_{T_m} \nabla \left(\int_{T_n} G^\Phi (\vec{r} \mid \vec{r}\,') \left(\nabla_S' \cdot \vec{f}_n (\vec{r}\,') \right) ds' \right) \\ & \cdot \vec{f}_m (\vec{r})\, ds, \end{aligned} \tag{15}$$

where $\vec{f}_n$ and $\vec{f}_m$ stand for the basis and testing RWG functions, respectively. The right-hand term of the MoM matrix equation will be different from radiating, transmitting to scattering cases.

3. Numerical Examples

To verify the efficiency enhancement of the proposed optimization based on GPU/CPU heterogeneous platform, this section will demonstrate fruitful numerical examples, respectively, corresponding to the OpenMP-CUDA-MLFMM, the GPU-based ACA, and the deeply optimized CG-FFT. All the programming codes are compiled in CUDA-C environment, and the hardware configuration can be found in Table 2. Similar to Section 2, the numerical results are also separated into 3 groups. The correctness of proposed methods with MLFMM and ACA have been verified by the very good agreement between bistatic RCS and standard Mie solutions of PEC spheres, which will not be demonstrated in this section.

Table 2: Configurations of GPU/CPU platform of numerical examples.

Parameters	MLFMM	ACA	CG-FFT
CPU	Intel	Intel	Intel Xeon
Processor #	i5 4570	i5 4570	E5420
# of cores	4	4	4
Base frequency (GHz)	3.20	3.20	2.50
GPU	NVIDIA	NVIDIA	NVIDIA
Architecture	GK110	GK110	GK104
CUDA cores	2304	2304	960
Base clock (MHz)	863	863	980
Memory Config (MB)	3072	3072	2048
Memory BW (GB/s)	288	288	144
CUDA version	6.0	7.0 RC	4.2
# of GPU	2	2	1

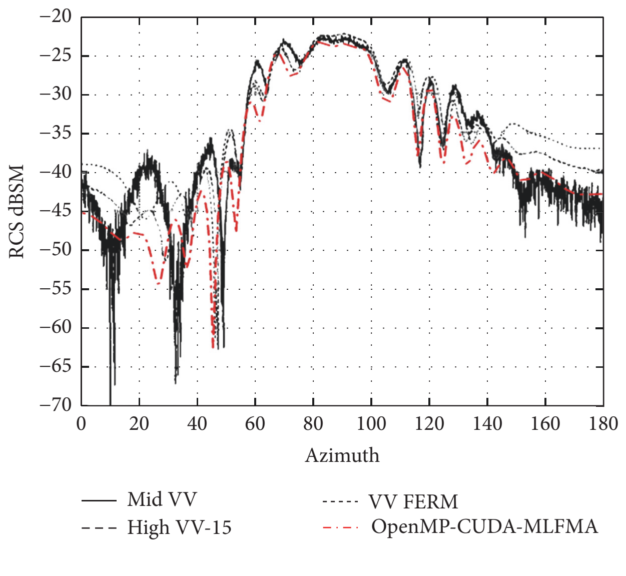
Table 3: Variables definition of time consumption.

Variables	Description
t_M	Time for near matrix generation
t_S	Time for corresponding SpMVP
t_{AF}	Time for aggregation at finest level
t_{DF}	Time for disaggregation at finest level
t_{AS}	Time for aggregation towards coarsest level
t_{DS}	Time for disaggregation towards finest level

3.1. Numerical Results of OpenMP-CUDA-MLFMM.

3.1. Numerical Results of OpenMP-CUDA-MLFMM. As a benchmark result to verify the robustness of algorithms, the scattering from the PEC NASA Almond at 7 GHz for VV polarization and that from the PEC NASA Ogive at 9 GHz for HH polarization are considered. These two PEC surfaces are discretized into refined triangles with edge length of about 0.1λ, which produce 8416 and 15516 RWG unknowns, respectively. The monostatic RCS curves with the incident angle as argument are obtained by the proposed method, as shown in Figure 6. From the comparison with measurement results, very good agreement can be found. Note that the interpretations labeled as "mid VV," "high VV," and "VV FERM" refer to Figure 5 in [36], and those labeled as "HH," "VV," "HH Cicero," and "VV Cicero" refer to Figure 9 in [36]. To demonstrate the efficiency performance of the proposed OpenMP-CUDA-MLFMM, we need to define some variables to record time consumption in different procedures, as shown in Table 3.

By sampling on different number of unknowns, an approximate curve of $(t_M; t_S)$ and $(t_{\text{AF}} + t_{\text{DF}}; t_{\text{AS}} + t_{\text{DS}})$ can be fitted in a broad range, as shown in Figure 7. In can be clearly seen that the proposed method has higher efficiency than that reported in [16] based on the same Kepler GK110 architecture. It is worth pointing out that the translation is not included because both the proposed method and that in [16] have almost the same performance in this procedure.



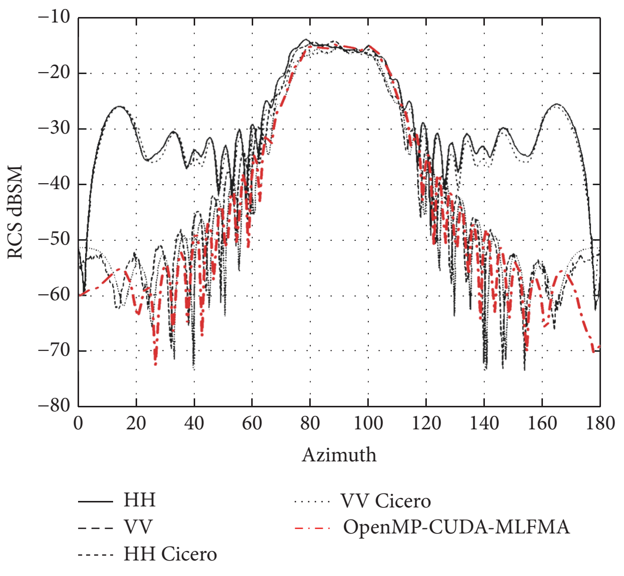


FIGURE 6: (a) The monostatic RCS at 7 GHz for VV polarization. (b) The monostatic RCS at 9 GHz for HH polarization.

3.2. Numerical Results of GPU-Based ACA. To verify the robustness of the proposed GPU-based ACA, the bistatic RCS from a rotor with four blades and a depressed cylindrical object are considered, as shown in Figure 8. Very good agreement can be seen between the proposed method and the traditional MoM on CPU. These two PEC surfaces are discretized into refined triangles with edge length of about 0.1λ, which produce 6636 and 4817 RWG unknowns, respectively.

To demonstrate the efficiency performance of the proposed GPU-based ACA, the scattering from a PEC sphere with radius of $n\lambda$ ($n = 1, 2, 3$, and 4) is considered. The number of generated RWG unknowns is 4774, 18468, 40950, and 72120, respectively. In this paper, the threshold in formula (6) is selected as $\varepsilon_{\mathrm{ACA}} = 10^{-3}$. The efficiency comparison between GPU and CPU platform in both the bMVP operations and the whole ACA scheme with single and double precision are shown in Figure 9. It can be seen that the speedup ratio of matrix compression by using ACA can achieve about 50~100 and 25~50 for single and double precision, respectively. Meanwhile, the speedup ratio of bMVP can achieve about 10~30 and 6~17 for single and double precision, respectively.

3.3. Numerical Results of Deeply Optimized CG-FFT. To validate the accuracy and efficiency of the proposed method, the quasiperiodical structures of flat lens on substrate ($\varepsilon_r = 2.2$, $\mu_r = 1.0$) with a copper ground ($\sigma = 5.98 \times 10^7$ S/m) are considered, which consist of 15×15 and 33×33 Jerusalem crosses and generate 19561 and 85849 RWG unknowns, respectively, as shown in Figure 10. The flat lenses are illuminated by an orthogonal incident plane wave with polarization in the x direction, where the incident information are $\theta = \pi$, $\varphi = 0$, $E_{0x} = 1$, and $E_{0y} = E_{0z} = 0$.

Since the MVP operation is the most time consuming part in CG-FFT, direct optimization from arithmetic programming point of view is a challenging topic and should be tested before applying to complicated structure. Therefore, both square and nonsquare matrices MVP are tested, from which a significant speedup ratio is observed comparing to the famous CUBLAS, as shown in Tables 4 and 5. For this experiment on arithmetic realization, the GPU occupancy of CUBLAS and that of the proposed method is 25% and 75%, respectively, while the usage of shared memory is 288 B and 10240 B, respectively. This can be seen as a great enhancement in hardware performance, which makes a speedup ratio of 1.4 in average on single GPU.

As is mentioned in Section 2, the right-hand term in MoM of multilayered structure is different from radiating, transmitting, and scattering cases. With an orthogonal incident wave, the right-hand term can be calculated as below.

$$V_m = -j\omega\varepsilon_0 \int_{T_m} \overrightarrow{E}^{\mathrm{inc}}\left(\overrightarrow{r}\right) \cdot \overrightarrow{f}_m\left(\overrightarrow{r}\right) ds,$$
$$\overrightarrow{E}^{\mathrm{inc}}(\rho) = \overrightarrow{E}_0 e^{j(k_x x + k_y y + k_z d)} + \widetilde{R} e^{j(k_x x + k_y y - k_z d)}, \tag{16}$$

where the calculation of reflection coefficient refers to [37] and d in the exponential term is the thickness of substrate. The relative error of iteration in CG-FFT for the case with 15×15 elements is shown in Figure 11 and the current distribution in centre of the case with 33×33 elements is shown in Figure 12. To enhance the efficiency further, we have realized the fast calculation of multilayered Green's functions

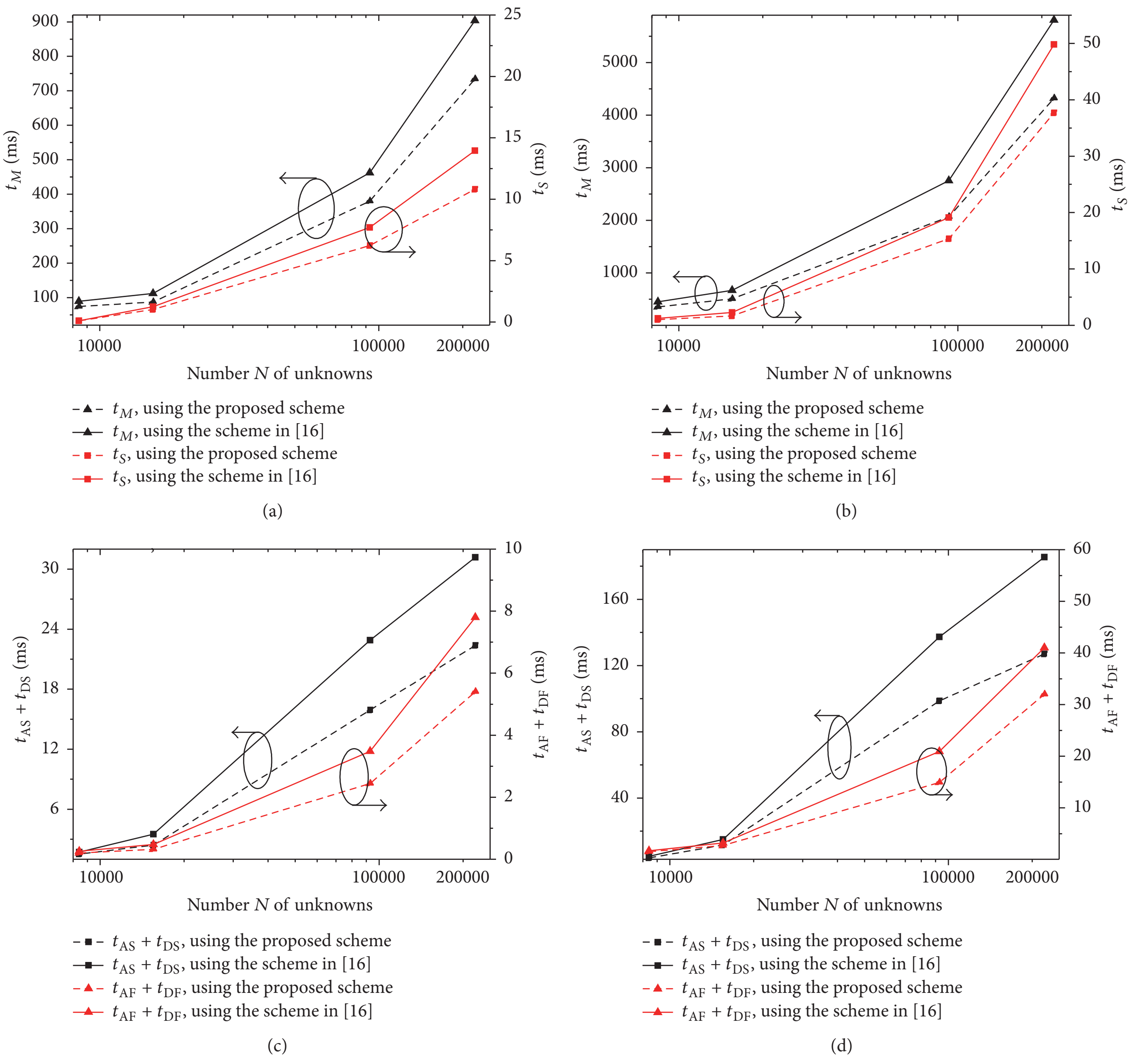
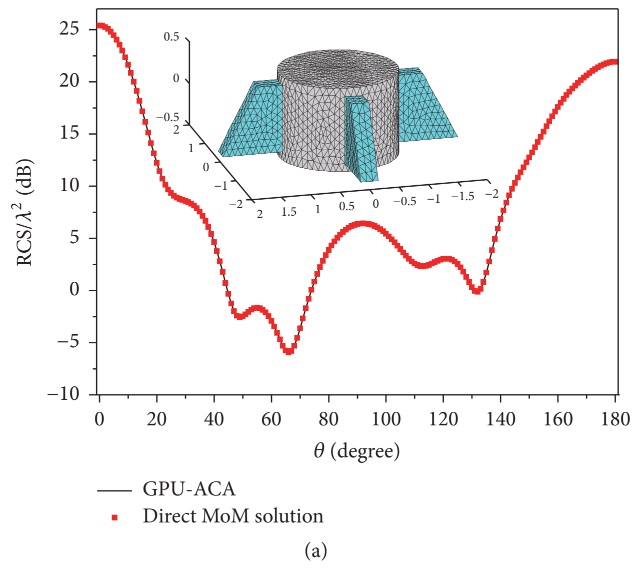


FIGURE 7: t_M and t_S as function of N with (a) single and (b) double precision; $t_{AF} + t_{DF}$ and $t_{AS} + t_{DS}$ as function of N with (c) single and (d) double precision.

on GPU/CPU heterogeneous platform [38], which will be implemented to the MPIE-MoM in future work.

4. Conclusions

In this paper, our recent works on high performance computing based on GPU/CPU heterogeneous platform and its application to EM scattering problems and planar multilayered medium structures are introduced. There are three typical applications analyzed, including a novel implementation of OpenMP-CUDA-MLFMM, a developed GPU-based ACA method, and a deeply optimized CG-FFT method. Comparing to the MLFMM, the ACA method starts to accelerate computing with Green's functions; however, it is still worth mentioning that the data storage strategy of ACA needs to be further developed. This paper is to explain the coordination between GPU and CPU on a heterogeneous platform. In addition, the methodology of optimization through hardware instructions is also adopted. Following the trends of both parallel and distributed computing, the OpenMP-CUDA-MLFMM and ACA are realized on a 2-GPU platform. With fruitful numerical examples for the three kinds of algorithms, we can find obvious enhancement in efficiency. As the development in hardware world, it



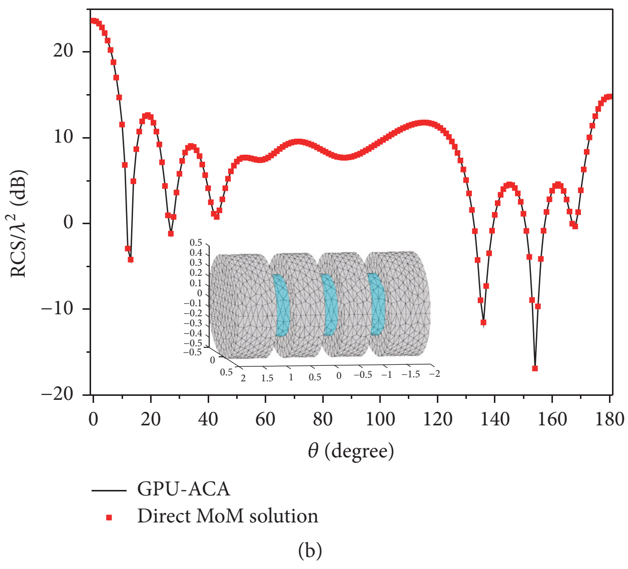
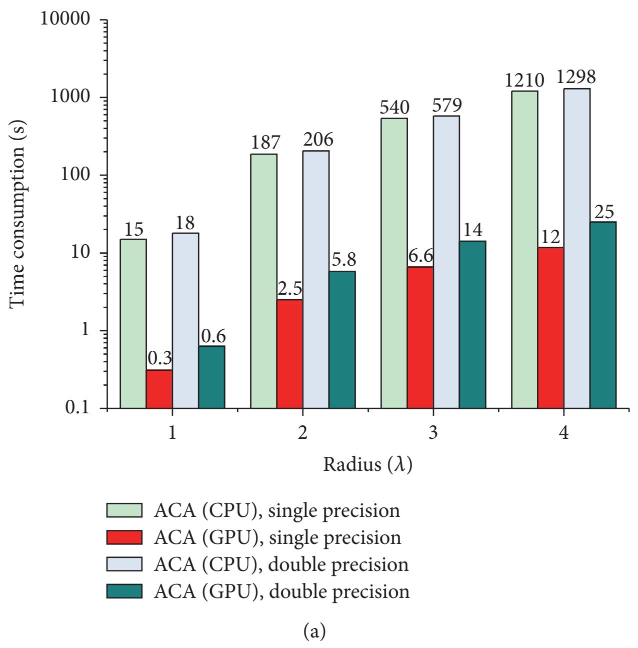


FIGURE 8: Bistatic RCS of (a) 4 blades rotor and (b) depressed cylindrical object.



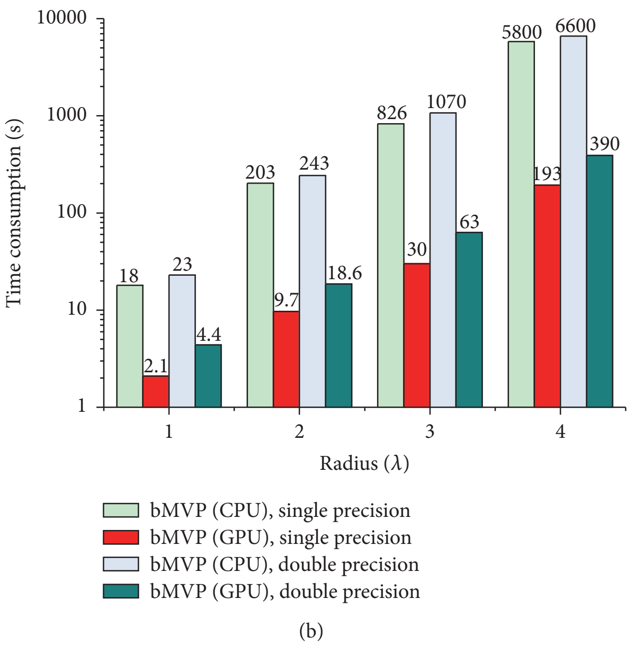


FIGURE 9: Efficiency comparison of (a) ACA and (b) bMVP.

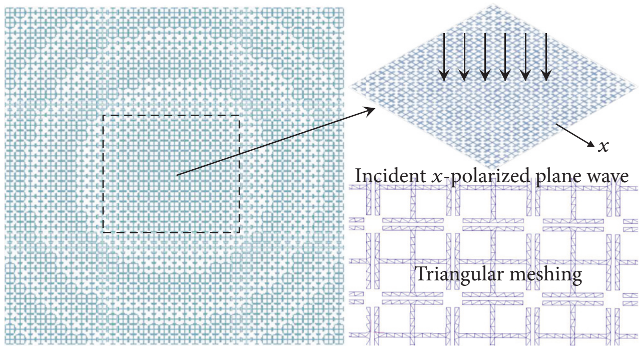


FIGURE 10: Geometry of flat lens on substrate and its meshing scheme.

Table 4: Comparison in MVP for square matrix.

Matrix dimension	CUBLAS (ms)	This paper (ms)
4,000	1.58	1.12
5,000	2.50	1.77
6,000	3.43	2.54
7,000	4.63	3.46
8,000	5.65	4.07
9,000	7.17	5.19
10,000	8.83	6.48
11,000	10.75	7.94
12,000	13.28	9.53

Table 5: Comparison in MVP for nonsquare matrix.

Matrix dimension	CUBLAS (ms)	This paper (ms)
1,700 × 9,000	1.69	1.10
1,800 × 9,000	1.58	1.17
1,900 × 9,000	1.71	1.22
2,900 × 15,000	4.28	2.95
3,000 × 15,000	4.00	2.98
3,100 × 15,000	4.69	3.16
4,900 × 25,000	11.48	8.24
5,000 × 25,000	11.19	8.44
5,200 × 25,000	11.29	8.72

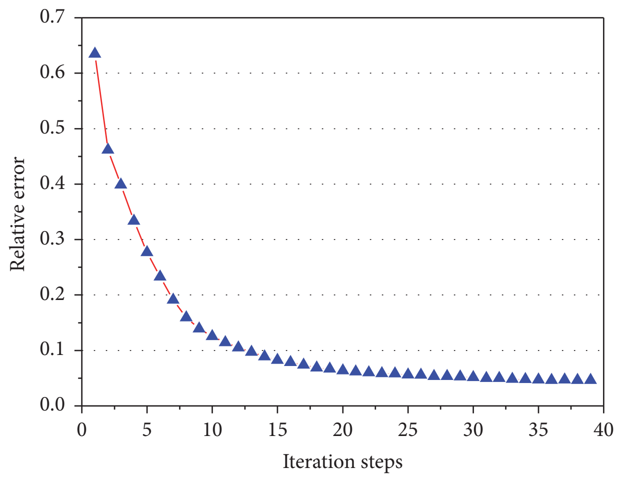


Figure 11: Relative error of iterations of the flat lens with 15 × 15 elements.

is worthy keeping an eye on different kinds of modules, including GPU, FPGA, and DSP. Meanwhile, it is also worthy keeping one on investigating and understanding the operating mechanism of heterogeneous computing.

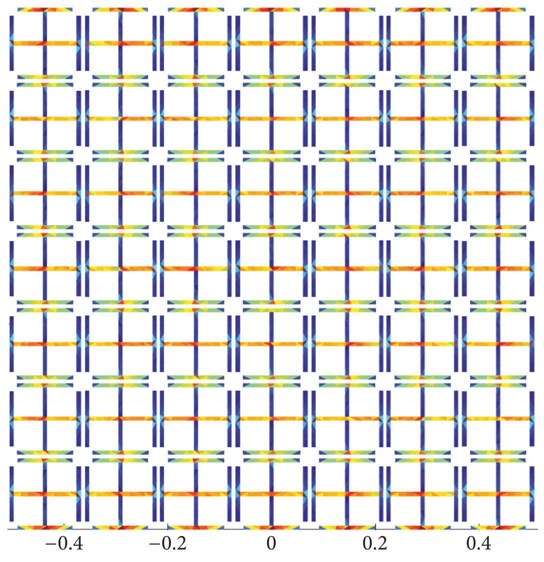


Figure 12: Current distribution in centre of the flat lens with 33 × 33 elements.

Disclosure

Zhe Song is now with Eindhoven University of Technology, TUE, The Netherlands.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the NSFC under Grants 61601121, 61401092, and 61302019.

References

[1] W. C. Chew, J. M. Jin, E. Michielssen, and J. M. Song, *Fast and Efficient Algorithm in Computational Electromagnetics*, Artech House, Boston, USA, 2001.

[2] 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*, vol. 10, no. 1, pp. 14–19, 1995.

[3] 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.

[4] C.-F. Wang, F. Ling, J. Song, and J.-M. Jin, “Adaptive integral solution of combined field integral equation,” *Microwave and Optical Technology Letters*, vol. 19, no. 5, pp. 321–328, 1998.

[5] J. R. Phillips and J. K. White, “A Precorrected-FFT method for electrostatic analysis of complicated 3-D structures,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1059–1072, 1997.

[6] X.-C. Nie, N. Yuan, and L.-W. Li, “Precorrected-FFT algorithm for solving combined field integral equations in electromagnetic

scattering," *Journal of Electromagnetic Waves and Applications*, vol. 16, no. 8, pp. 1171–1187, 2002.

[7] S. M. Seo and J.-F. Lee, "A fast IE-FFT algorithm for solving PEC scattering problems," *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1476–1479, 2005.

[8] W. Leipzig, "A sparse matrix arithmetic based on H-matrices. Part I: introduction to Hmatrices," *Computing*, vol. 62, no. 2, pp. 89–108, 1999.

[9] W. Leipzig and S. Saarbrjcken, "Adaptive low-rank approximation of collocation matrices," *Computing*, vol. 70, no. 1, pp. 1–24, 2003.

[10] A. Schroder, H.-D. Bruns, and C. Schuster, "Fast evaluation of electromagnetic fields using a parallelized adaptive cross approximation," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 5, pp. 2818–2822, 2014.

[11] M. F. Càtedra and E. Gago, "Spectral Domain Analysis of Conducting Patches of Arbitrary Geometry in Multilayer Media Using the CG-FFT Method," *IEEE Transactions on Antennas and Propagation*, vol. 38, no. 10, pp. 1530–1536, 1990.

[12] P. Pacheco, *An Introduction to Parallel Programming*, Elsevier, 2011.

[13] D. P. Rodgers, "Improvements in multiprocessor system design," *ACM SIGARCH Computer Architecture News*, vol. 13, no. 3, pp. 225–231, 1985.

[14] NVIDIA corp., *CUDA, C Programming Guide Version 5.5*, 2013.

[15] D. De Donno, A. Esposito, L. Tarricone, and L. Catarinucci, "Introduction to GPU computing and CUDA programming: a case study on FDTD," *IEEE Antennas and Propagation Magazine*, vol. 52, no. 3, pp. 116–122, 2010.

[16] J. Guan, S. Yan, and J.-M. Jin, "An OpenMP-CUDA implementation of multilevel fast multipole algorithm for electromagnetic simulation on multi-GPU computing systems," *Institute of Electrical and Electronics Engineers. Transactions on Antennas and Propagation*, vol. 61, no. 7, pp. 3607–3616, 2013.

[17] N. A. Gumerov and R. Duraiswami, "Fast multipole methods on graphics processors," *Journal of Computational Physics*, vol. 227, no. 18, pp. 8290–8313, 2008.

[18] K. Xu, D. Z. Ding, Z. H. Fan, and R. S. Chen, "Multilevel fast multipole algorithm enhanced by GPU parallel technique for electromagnetic scattering problems," *Microwave and Optical Technology Letters*, vol. 52, no. 3, pp. 502–507, 2010.

[19] M. Cwikla, J. Aronsson, and V. Okhmatovski, "Low-frequency MLFMA on graphics processors," *IEEE Antennas and Wireless Propagation Letters*, vol. 9, pp. 8–11, 2010.

[20] S. Li, R. Chang, A. Boag, and V. Lomakin, "Fast electromagnetic integral-equation solvers on graphics processing units," *IEEE Antennas and Propagation Magazine*, vol. 54, no. 5, pp. 71–87, 2012.

[21] S. Peng and C.-F. Wang, "Precorrected-FFT method on graphics processing units," *Institute of Electrical and Electronics Engineers. Transactions on Antennas and Propagation*, vol. 61, no. 4, part 2, pp. 2099–2107, 2013.

[22] S. Peng and Z. Nie, "Acceleration of the method of moments calculations by using graphics processing units," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 7, pp. 2130–2133, 2008.

[23] B. M. Kolundzija and D. P. Zoric, "Efficient evaluation of MoM matrix elements using CPU and/or GPU," in *Proceedings of the 6th European Conference on Antennas and Propagation, EuCAP 2012*, pp. 702–706, cze, March 2012.

[24] X. Mu, H.-X. Zhou, K. Chen, and W. Hong, "Higher order method of moments with a parallel out-of-core LU solver on GPU/CPU platform," *Institute of Electrical and Electronics Engineers. Transactions on Antennas and Propagation*, vol. 62, no. 11, pp. 5634–5646, 2014.

[25] X. Mu, H.-X. Zhou, Z. Song, W.-B. Kong, and W. Hong, "An implementation of the ACA on GPU platform," in *Proceedings of the Asia-Pacific Microwave Conference, APMC 2015*, chn, December 2015.

[26] Z. Song, Y. Zhang, X. Mu, H.-X. Zhou, and W. Hong, "Full wave analysis of millimeter wave quasi-periodical structure for antenna applications by method of moments and its conjugate gradient solution on GPU/CPU platform," in *Proceedings of the 2015 1st IEEE International Conference on Computational Electromagnetics, ICCEM 2015*, pp. 41-42, hkg, February 2015.

[27] NVIDIA corp., *NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110*, 2013.

[28] J. Marco and M. Taufer, "Performance impact of dynamic parallelism on different clustering algorithms on GPUs," in *Proceedings of NVIDIA GPU Tech Conf*, 2014.

[29] C.-F. Wang, F. Ling, and J.-M. Jin, "A Fast Full-Wave Analysis of Scattering and Radiation from Large Finite Arrays of Microstrip Antennas," *IEEE Transactions on Antennas and Propagation*, vol. 46, no. 10, pp. 1467–1474, 1998.

[30] Z. Song, H.-X. Zhou, K.-L. Zheng, J. Hu, W.-D. Li, and W. Hong, "Accurate evaluation of green's functions for a lossy layered medium by fast extraction of surface-and leaky-wave modes," *IEEE Antennas and Propagation Magazine*, vol. 55, no. 1, pp. 92–102, 2013.

[31] K. A. Michalski and J. R. Mosig, "Multilayered media Green's functions in integral equation formulations," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 508–519, 1997.

[32] W. C. Chew, *Waves and Fields in Inhomogeneous Media, ser. Electromagn. Waves Piscataway*, Waves Piscataway, IEEE Press, New York, USA, 1995.

[33] R. E. Collin, *Field Theory of Guided Waves*, Mc-Graw Hill, New York, NY, USA, 1960.

[34] D. G. Fang, J. J. Yang, and G. Y. Delisle, "Discrete image theory for horizontal electric dipoles in a multilayered medium," *IEE Proceedings H: Microwaves, Antennas and Propagation*, vol. 135, no. 5, pp. 297–303, 1988.

[35] Y. L. Chow, J. J. Yang, and G. E. Howard, "A Closed-Form Spatial Green's Function for the Thick Microstrip Substrate," *IEEE Transactions on Microwave Theory and Techniques*, vol. 39, no. 3, pp. 588–592, 1991.

[36] A. C. Woo, M. J. Schuh, M. L. Sanders, H. T. G. Wang, and J. L. Volakis, "Benchmark Radar Targets for the Validation of Computational Electromagnetics Programs," *IEEE Antennas and Propagation Magazine*, vol. 35, no. 1, pp. 84–89, 1993.

[37] D. Vande Ginste, E. Michielssen, F. Olyslager, and D. De Zutter, "An efficient perfectly matched layer based multilevel fast multipole algorithm for large planar microwave structures," *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 5, pp. 1538–1548, 2006.

[38] Z. Song, "Fast and Accurate Evaluation of Multilayered Green's Functions by Extracting Surface and Leaky Wave Poles Based on GPU/CPU Heterogeneous Platform," in *Proceedings of the IEEE Int. Workshop on Electromagnetics. (iWEM2017)*, London, UK, 2014.