Hindawi
International Journal of Antennas and Propagation
Volume 2017, Article ID 4309381, 10 pages
https://doi.org/10.1155/2017/4309381

Research Article

An Efficient GPU-Based Out-of-Core LU Solver of Parallel Higher-Order Method of Moments for Solving Airborne Array Problems

Zhongchao Lin, Yan Chen, Yu Zhang, Xunwang Zhao, and Huanhuan Zhang

School of Electronic Engineering, Xidian University, Xi'an, Shaanxi 710071, China

Correspondence should be addressed to Xunwang Zhao; xdzxw@126.com

Received 16 November 2016; Accepted 24 January 2017; Published 27 February 2017

Academic Editor: Claudio Gennarelli

The parallel higher-order method of moments (HoMoM) with a GPU accelerated out-of-core LU solver is presented for analysis of radiation characteristics of a 1000-element antenna array over a full-size airplane. A parallel framework involving MPI and CUDA is adopted to ensure that the procedures run on a hybrid CPU/GPU cluster. An efficient two-level out-of-core scheme is designed to break the bottleneck of both GPU memory and physical memory when solving electrically large and complex problems. To hide communication time between CPU and GPU, asynchronous communications are chosen to enable overlapping between communication and computation. For large problems that cannot fit in GPU memory or physical memory, the two-level out-of-core LU solver is able to achieve a speedup of about 1.6x over the traditional out-of-core LU solver based on a highly optimized math library.

1. Introduction

The method of moments (MoM) can provide highly accurate results for a wide variety of complex electromagnetic (EM) problems [1, 2]. However, it requires a lot of memory and calculation time when solving large dense matrix equations using the lower/upper (LU) decomposition based direct solver. Because of these restrictions, the application of MoM for simulation of extremely large problems, such as an airborne array, is seriously limited. To break these difficulties, researchers have been employing several approaches. One is the fast algorithms based on MoM, for example, the fast multipole method (FMM) [3, 4], multilevel fast multipole algorithm (MLFMA) [5, 6], and adaptive integral method (AIM) [7]. Generally speaking, these approaches reduce the memory requirement and the computation complexity compared with MoM and get more accurate results compared with the hybridization of MoM with high frequency methods [8, 9]. However, the fast algorithms may not be suitable for the ill-conditioned equations because of the use of iterative solvers, which may converge very slowly for models including complex structures and various materials. An alternative approach is to employ the higher-order basis based integral equation solver, which can greatly reduce the number of unknowns compared to use of the traditional piecewise basis functions, such as the Rao-Wilton-Glisson basis functions (RWGs) [2, 10]. Meanwhile, the out-of-core technique of the higher-order MoM (HoMoM) has been adopted to improve the capability of MoM to deal with larger complex EM problems, which is limited only by the capacity of hard-disk storage [2].

Along with the latest developments on computer technology, the use of high performance computing and graphics processing units (GPUs) acceleration techniques can significantly improve the computing speed. Early GPUs were mainly for image processing, until the compute unified device architecture (CUDA) programming was proposed in 2006 [11]. Then the GPUs were usually utilized to improve the performance of the EM methods, such as MoM and its fast algorithms [12–15] and the Finite Difference Time Domain (FDTD) [16, 17]. However, there is a serious imbalance between GPU memory and the storage required by MoM for solving electrically large problems.

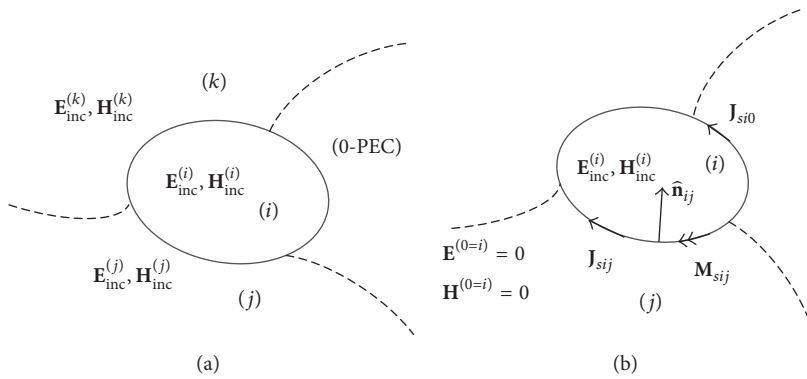

Figure 1: Surface equivalence theorem: (a) the original problem and (b) the equivalent problem for region i.

In [18, 19], an out-of-core scheme between GPU memory and CPU memory (RAM) was presented to break the limitation of the GPU memory and to improve the capability of the parallel LU solver. Compared with a single core, a speedup of 15 can be obtained. In [20], an efficient out-of-GPU memory scheme for solving matrix equations generated by MoM was presented, and it also obtains a good speedup on a single CPU/GPU platform. In [21, 22], an out-of-core scheme between RAM and hard-disk drives (HDD) using RWGs and higher-order basis functions (HOBs) was adopted to break the limitation of RAM and improve the capability of parallel MoM to solve larger EM problems. In [23], the parallel MoM was utilized to solve a full-size airplane with nearly one million unknowns using the out-of-core LU solver on a high performance cluster with 1 TB RAM and 16 TB hard-disk space. In [24], a GPU-based parallel out-of-core LU solver of HoMoM is presented, but the procedure was developed on a single CPU/GPU platform.

High performance computing and GPU acceleration are an effective way to improve the computational capability of the MoM. They were analyzed for the parallel HoMoM in our previous studies [25–29]. In this paper, an efficient GPU-based out-of-core parallel LU solver of the HoMoM using message passing interface (MPI), which runs on a high performance cluster with multiple CPU/GPU computing nodes, is presented to solve the complex EM problems. The radiation patterns of a wire antenna array with about 1000 elements over an airplane are simulated to demonstrate the acceleration performance of the GPU-based solver. The proposed solver has two distinguishing characteristics as follows: (1) an efficient two-level out-of-core scheme is presented to overcome the restriction of the RAM and GPU memory; (2) an overlapping scheme based on the asynchronous data transfer technique and CUDA streams is adopted to hide the communication time between CPUs and GPU cards.

2. Higher-Order Method of Moments

The Poggio-Miller-Chang-Harrington-Wu (PMCHW) integral equations [2, 30, 31] are employed for modeling metallic and dielectric structures, and higher-order polynomials are used as the basis functions, which will be briefly reviewed.

2.1. Integral Equations. Consider that a model consists of $n+1$ types of materials, each of which is assigned to a region. In the ith region ($i = 1, \ldots, n$), the permittivity and the permeability are $\varepsilon^{(i)}$ and $\mu^{(i)}$, and the incident electric and magnetic fields are $\mathbf{E}_{\text{inc}}^{(i)}$ and $\mathbf{H}_{\text{inc}}^{(i)}$, respectively, as shown in Figure 1(a). Note that region 0 is reserved for perfect electric conductors (PECs), in which the electromagnetic field is zero. According to the equivalence theorem, the PMCHW formulation [2, 26] at the boundary surface between regions i and j is expressed as

$$\hat{\mathbf{n}}_{ij} \times \left[\sum_{\substack{k=0 \\ k \neq i}}^{n} \mathbf{E}_{\text{scat}}^{(i)} \left(\mathbf{J}_{\text{sik}}, \mathbf{M}_{\text{sik}} \right) - \sum_{\substack{k=0 \\ k \neq j}}^{n} \mathbf{E}_{\text{scat}}^{(j)} \left(\mathbf{J}_{sjk}, \mathbf{M}_{sjk} \right) \right] = -\hat{\mathbf{n}}_{ij} \times \left[\mathbf{E}_{\text{inc}}^{(i)} - \mathbf{E}_{\text{inc}}^{(j)} \right], \tag{1}$$

$$\hat{\mathbf{n}}_{ij} \times \left[\sum_{\substack{k=0 \\ k \neq i}}^{n} \mathbf{H}_{\text{scat}}^{(i)} \left(\mathbf{J}_{\text{sik}}, \mathbf{M}_{\text{sik}} \right) - \sum_{\substack{k=0 \\ k \neq j}}^{n} \mathbf{H}_{\text{scat}}^{(j)} \left(\mathbf{J}_{sjk}, \mathbf{M}_{sjk} \right) \right] = -\hat{\mathbf{n}}_{ij} \times \left[\mathbf{H}_{\text{inc}}^{(i)} - \mathbf{H}_{\text{inc}}^{(j)} \right], \tag{2}$$

where, as shown in Figure 1(b), $\hat{\mathbf{n}}_{ij}$ is the unit normal vector; $\mathbf{E}_{\text{scat}}^{(i)}$ and $\mathbf{H}_{\text{scat}}^{(i)}$ are the scattered electric and magnetic fields in region i, which can be computed by using the L and K integral operators [2]; $\mathbf{J}_{\text{sik}}$ and $\mathbf{M}_{\text{sik}}$ are the electric and magnetic currents on the boundary surface between regions i and k. In region j, the fields and currents are similarly represented. On boundary surfaces of PECs, the magnetic currents vanish, and (1) degenerates into the electric field integral equation (EFIE) [32].

2.2. Higher-Order Basis Functions. The boundary surfaces are discretized into bilinear quadrilateral patches, over which

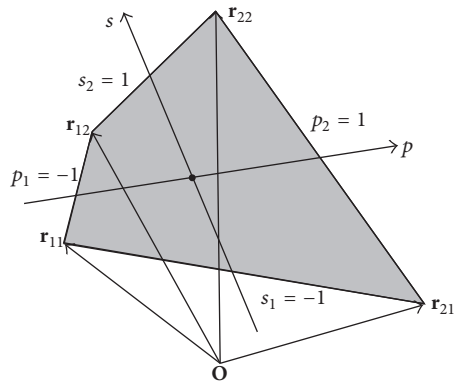

Figure 2: Bilinear quadrilateral patch with its local coordinates p and s.

the higher-order basis functions (HOBs) are defined in the polynomial form [2],

$$\mathbf{F}_{ij}(p,s) = \frac{\boldsymbol{\alpha}_s}{\left|\boldsymbol{\alpha}_p \times \boldsymbol{\alpha}_s\right|} p^i s^j,$$
$$\boldsymbol{\alpha}_p = \frac{\partial \mathbf{r}(p,s)}{\partial p}, \tag{3}$$
$$\boldsymbol{\alpha}_s = \frac{\partial \mathbf{r}(p,s)}{\partial s},$$

where p and s are local coordinates of the patch, i and j are orders of polynomials, and $\mathbf{r}(p,s)$ is the parametric equation of the bilinear patch, which is illustrated in Figure 2.

The polynomial orders can be adjusted according to the electrical size of the geometric element. The orders become higher as the element increases larger. The electrical size of the geometric element can be as large as two wavelengths. Compared with the widely used RWG basis functions, the use of HOBs can reduce the number of unknowns by a factor of 5–10 and thus requires much less computational amount and memory. This kind of basis functions is also applicable for wire structures modeled by truncated cones.

3. GPU-Based Out-of-Core LU Solver

The parallel LU solver and the out-of-core scheme between CPU memory and HDD are briefly introduced firstly. Then the GPU-based two-level out-of-core scheme is discussed in detail. Finally, an overlapping scheme is designed to save the data transfer time between GPU memory and CPU memory.

3.1. Parallel LU Solver. The LU solver is a numerical accurate method in solving matrix equations. There are two decomposition algorithms: left-looking and right-looking algorithms, where the left-looking algorithm requires less I/O operation between RAM and HDD [2]. For the parallel LU solver, the data should be distributed to multiple processes in a certain way and a load balanced approach called two-dimensional (2D) block-cyclic matrix distribution, which is similar to the ScaLAPACK, is adopted.

As shown in Figure 3, consider a matrix $\mathbf{A}$ with 9×9 elements, which is distributed using a 2×2-size blocks to 6 processes in a 2×3 process grid using 2D cyclic boundary condition. The outermost numbers denote the row and column indices of the process coordinates. Figure 3(b) shows the matrix $\mathbf{A}$ divided into 2×2 -sized blocks A_{mn} (m, $n = 1, 2, 3, 4, 5$), and Figure 3(c) shows the rearrangement of A_{mn} in (a) and (b) for each process.

The left-looking algorithm is mainly considered in this paper, and there are two primary phases in the left-looking LU solver with pivoting neglected: the active block column updates and its factorization. When it is factorized, next block column will become the active column. And this process is repeated until the LU decomposition is completed. Note that the block columns (slabs) to the left of the active column are required in the update process, as denoted by gray color in Figure 4(a).

In Figure 4(a), the green part is called an active block column and the gray part has been factorized already. The data dependence of the active block column on its left block columns is illustrated by red arrows in Figure 4. When the RAM cannot store the whole matrix, an efficient solution is to use the storage of the HDD, as shown in Figure 4(b).

3.2. Out-of-Core Scheme between CPU Memory and HDD. In the case where the RAM cannot meet the storage requirement, the whole matrix should be decomposed into a set of smaller matrices or slabs, and the size of which is determined by the RAM. In the process of the out-of-core LU decomposition, each slab is read into the RAM from the HDD in order, and then the LU decomposition is started. The decomposed results of each slab are written back to the HDD when its factorization is done. The solver then proceeds with the next slab until the whole matrix is decomposed, as shown in Figure 5.

From Figure 5 we can see that CPUs start computing after the data are transferred from the HDD to RAM. In this phase, L1 and L2 cache are used frequently to help the CPUs exchange data with RAM, and there are no I/O operations to the HDD. The right-looking algorithm [2] is adopted in this phase, because it has the larger computation destiny than the left-looking algorithm. In contrast to the left-looking algorithm, the active block column depends on its right block columns in the right-looking algorithm, as shown in Figure 6.

There is no doubt that the performance of the CPU-HDD out-of-core procedures is reduced by the data-movement between the HDD and CPU RAM. To accelerate the parallel out-of-core MoM, GPUs are used to implement the LU decomposition procedures.

3.3. GPU-Based Two-Level Out-of-Core Scheme. To implement the GPU-based two-level out-of-core scheme, the multilevel storage architecture should be defined, as shown in Figure 7, where the dashed line frame represents the CPU and the GPU card, respectively.

The data should be transferred to GPU to implement computing tasks. When the GPU memory cannot store the data, an efficient out-of-core scheme between RAM and GPU

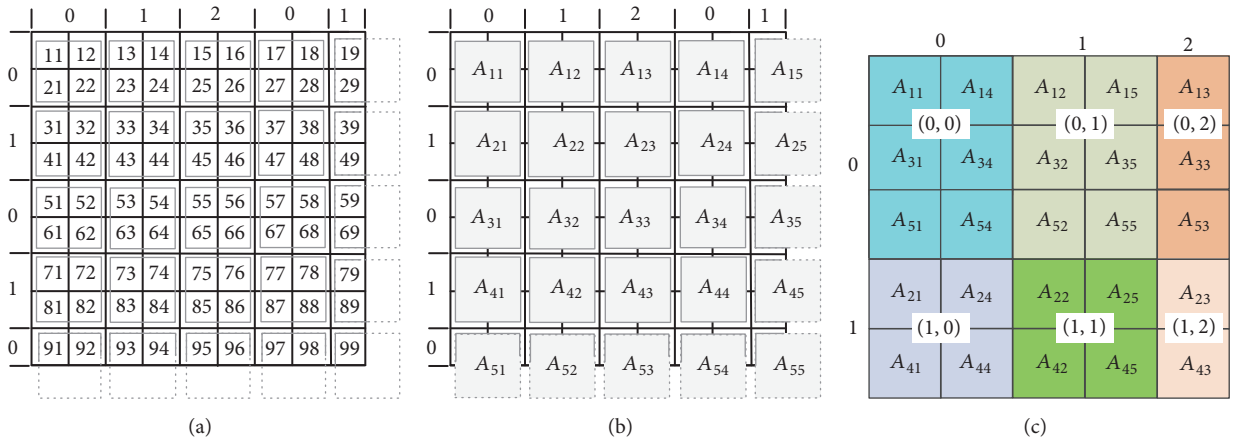

FIGURE 3: 2D block-cyclic distribution scheme: (a) matrix **A** divided into 2 × 2-sized blocks, (b) the matrix **A** is consisting of 5 × 5 blocks, and (c) the rearrangement of A_{mn} in (b) for each process.

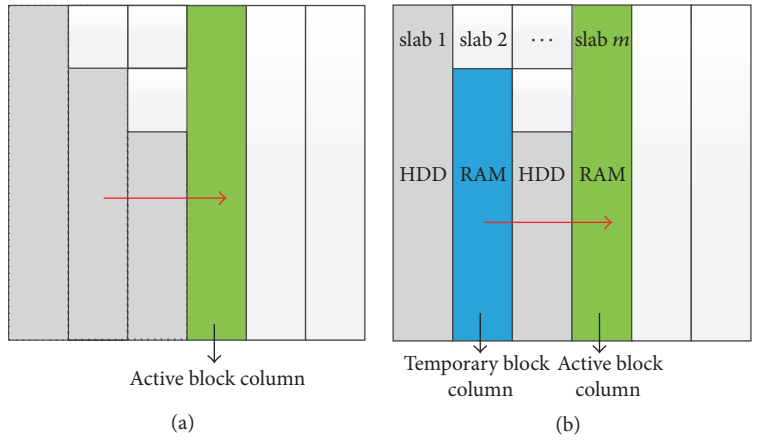

FIGURE 4: The left-looking LU algorithm: (a) the data dependence in the left-looking algorithm and (b) the out-of-core scheme implemented slab by slab.

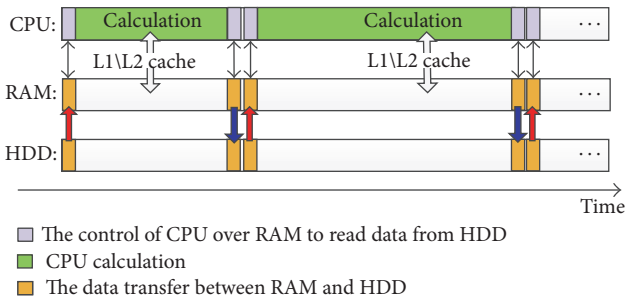

FIGURE 5: Operation sequence of the out-of-core scheme between RAM and HDD.

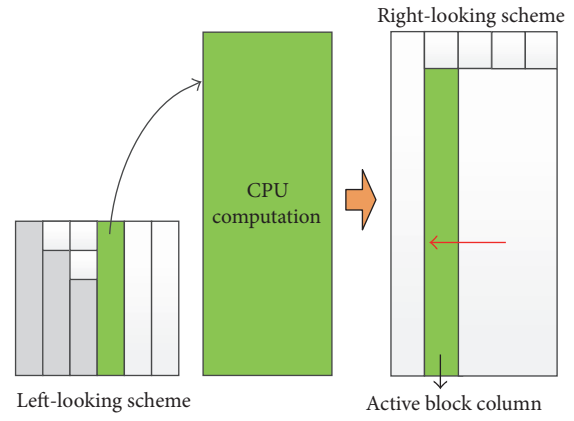

FIGURE 6: The data dependence in right-looking algorithm.

memory must be utilized, which is similar to the out-of-core scheme between RAM and HDD. As shown in Figure 8, the block column in RAM (active block column in Figure 4) should be divided into finer block columns.

Combine the RAM to HDD and GPU memory to CPU out-of-core schemes in a certain way as shown in Figure 9, which is the software and hardware architecture of parallel out-of-core left-looking algorithm. To make one GPU card used by several CPU cores at the same time, the GPU context technique [33] is used. In Figure 9, each node has two CPU

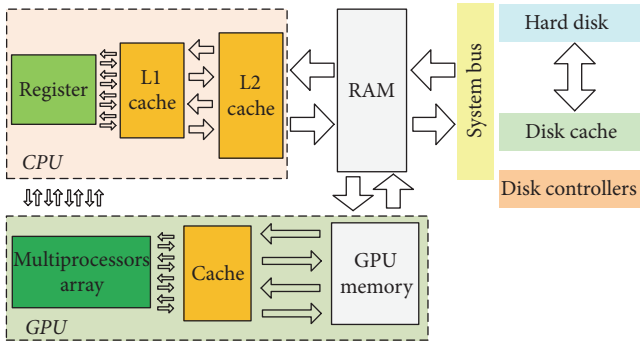

Figure 7: Multilevel storage architecture of a CPU/GPU platform.

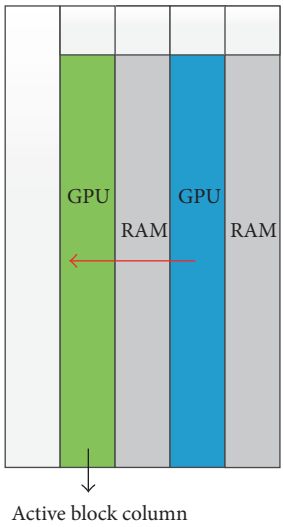

Figure 8: Data transfer of the GPU-CPU out-of-core scheme.

cores and one GPU card. The context technique is used to partition one GPU card into two virtual GPU cards. And one MPI process is assigned to one CPU core and one virtual GPU card. Each MPI process can use its own GPU resources to accelerate the computing tasks.

In Figure 9, the CPUs first read data from the HDD to the RAM and then upload them into GPU, then GPU and CPUs start computation, and finally all the data will be written to the HDD.

Figure 10 is the implementation scheme of the GPU-based out-of-core LU solver of HoMoM. The left column of Figure 10(a) is the out-of-core left-looking algorithm for the LU solver, where the load represents the data transferred from RAM to HDD.

The most time-consuming functions of the algorithm are FSub^{LL} and $\mathrm{LU}_{\mathrm{blk}}^{RL}$, which are given in the right column of Figure 10(a) and the left column of Figure 10(b), respectively. FSub^{RL} is the hot spot of FSub^{LL}, and $\mathrm{GPUFSub}^{\mathrm{GEMM}}$ is the hot spot of $\mathrm{LU}_{\mathrm{blk}}^{RL}$. FSub^{RL} is given in the right column of Figure 10(b), where $\mathrm{GPUFSub}^{\mathrm{GEMM}}$ is the hot spot of FSub^{RL}. Obviously, $\mathrm{GPUFSub}^{\mathrm{GEMM}}$ is the hot spot of the whole process of the LU solver, which is the matrix-matrix multiplication function in the GPU-based out-of-core LU solver.

3.4. Overlapping of Communication and Computation. In the GPU-based out-of-core scheme, communication can be hidden by computation by using the CUDA asynchronous technology and CUDA stream technology [34, 35]. As shown in Figure 9, each MPI process opens several CUDA streams on GPU context. The CUDA stream is similar to a CPU pipeline operation queue. The overlapping scheme consists of three kinds of operations: data transfer from RAM to GPU memory, GPUs calculation, and data transfer from GPU memory to RAM, which are performed by different hardware units. The operations in different CUDA streams can be executed asynchronously, and the data transfer and calculation can avoid resource confliction.

The working order of the three operations in different CUDA streams is shown in Figure 11. From the figure, one can see that the communication is hidden by computation.

4. Numerical Results and Discussion

The computational platform used in the following examples is a hybrid CPU/GPU cluster, which is equipped with 2 compute nodes connected by 1000 Mbps network cards. Each node has two six-core Intel Xeon E5-2620 2.0 GHz CPUs, 64 GB RAM, and one Tesla K20c GPU card with 4.6 GB memory available. The parallel hybrid CPU/GPU codes are programmed by hybrid languages based on MPI and OpenMP techniques. In this paper, the performance of the GPU-based parallel out-of-core LU solver is compared with the CPU version.

4.1. Performance Tuning. A new parameter cuRatio is introduced to demonstrate the task ratio of GPUs calculation. To obtain high acceleration performance, we firstly tune the cuRatio value. Note that the speed of the CPU and GPU card is different, so the optimal value of cuRatio should be testing to ensure the optimal performance of GPU-based out-of-core parallel LU solver.

The test is carried out on a single CPU/GPU node. In the calculation, one GPU card should be partitioned into twelve virtual GPU cards, and each virtual GPU card contains 1/12 resources of the whole GPU card.

Because the matrix equation solving contributes more than 80% of the computation time for a large scale dense complex matrix, the matrix equation solving time is mainly considered in this paper. A problem involving 67,552 unknowns is simulated. The cuRatio values are chosen from 0.0 to 1.0, and the step size is 0.05. The testing results are shown in Figure 12. From the results, one can see that the optimal value of cuRatio is 0.75.

4.2. Correctness of the Implementation. The accuracy of the proposed GPU-based out-of-core HoMoM is validated in this section. As shown in Figure 13, an 11 × 11 microstrip patch phased array, which is housed in a 520 mm × 580 mm × 7 mm cavity in a ground plane, is considered. The parameters of substrate are $\varepsilon_r = 2.67$ and $\mu_r = 1.0$. The dimensions of the

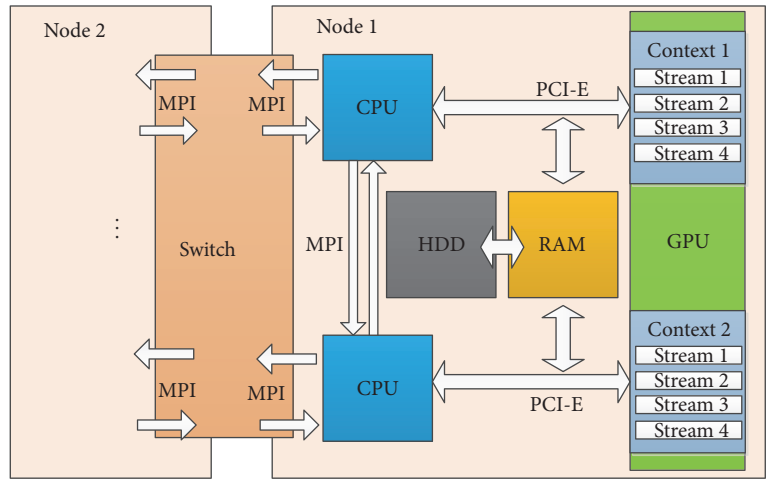

FIGURE 9: Software and hardware architecture of parallel out-of-core left-looking scheme.

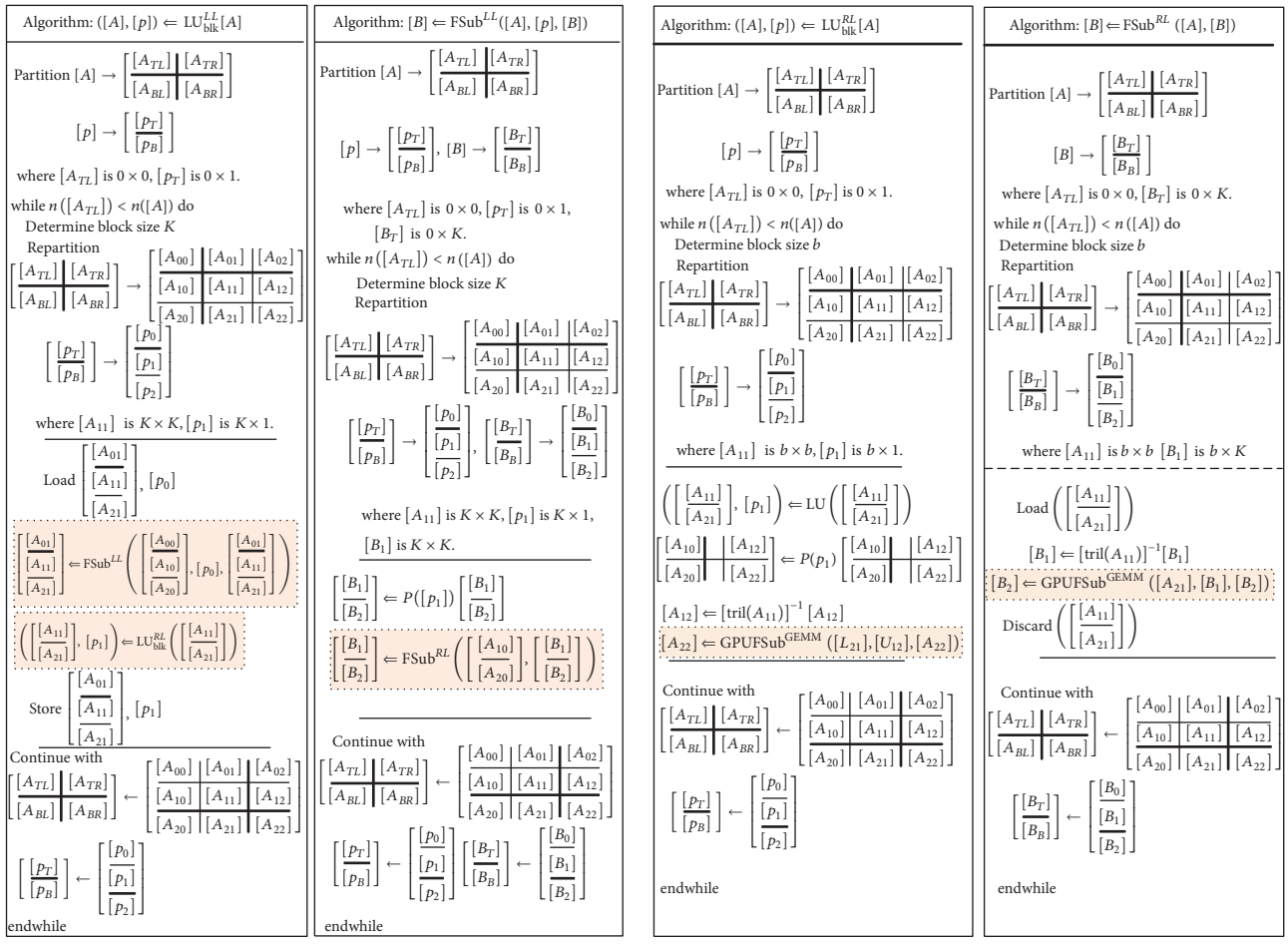

FIGURE 10: The scheme of the GPU-based out-of-core LU solver.

element are 30 mm × 35.6 mm, and the gaps between any two neighboring elements are 14.0 mm along both directions. The entire array is discretized into 121 wires for the feeds and 6,490 bilinear patches for the microstrip surfaces, corresponding to a total of 14,956 unknowns. Figure 14 shows the gain patterns of microstrip patch phased array, and the results from the proposed method and the RWG MoM in FEKO agree with each other very well.

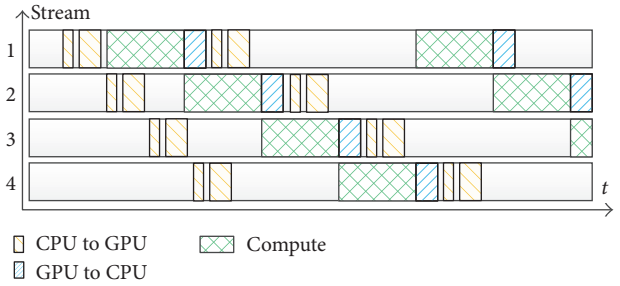

Figure 11: Overlapping between communication and computation on GPU context.

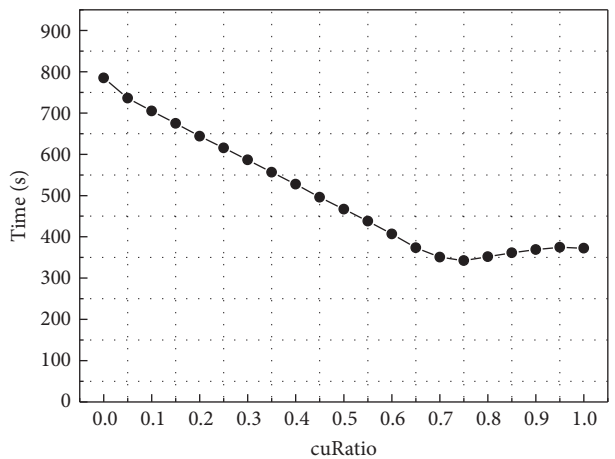

Figure 12: Matrix solving time associated with the cuRatio value.

Table 1: Computational parameters for the array.

Number of unknowns	Memory (GB)	Computational resource	Solving time (s)	Speedup
12,166	**2.21**	24 CPU cores	52.68	1
		24 CPU cores and 2 GPUs	22.81	**2.309**

4.3. Performance Analysis for Metallic Structures. Here we present the acceleration performance of the GPU-based out-of-core HoMoM for an airborne wire antenna array. The accuracy of the CPU version of the parallel out-of-core HoMoM has been validated through comparison with measurement and MoM code with RWGs in [2].

Consider a wire antenna array with 72 × 14 elements, as shown in Figure 15. The dimensions of the wire array are 10.8 m × 2.9 m and the distances between any two neighboring elements are 150 mm and 50 mm along the length and width directions, respectively. The amplitude at the feed of the array is designed by a −35 dB Taylor distribution in *xoy* plane. The operation frequency is 1.0 GHz and the number of unknowns is 12,166. The three-dimensional (3D) gain pattern is given in Figure 16 and the 2D gain patterns are given in Figure 17. The 2D gain patterns computed by CPU alone are also given for comparison. The side-lobe level in *xoy* plane is below −35 dB, which meets the design target.

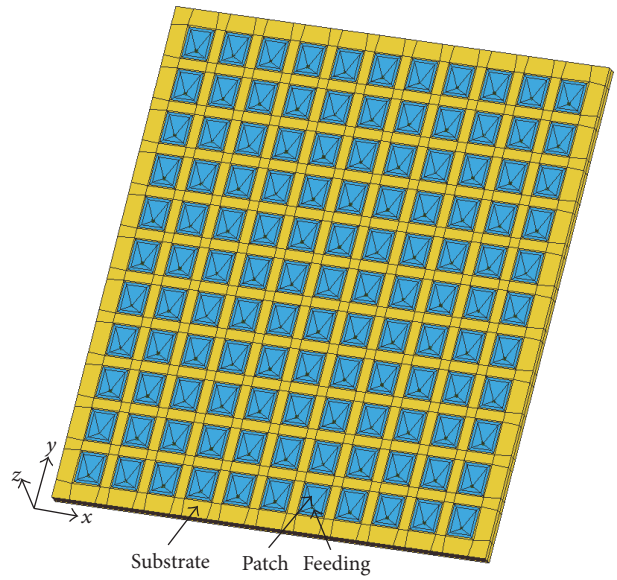

Figure 13: Model of the microstrip patch phased array.

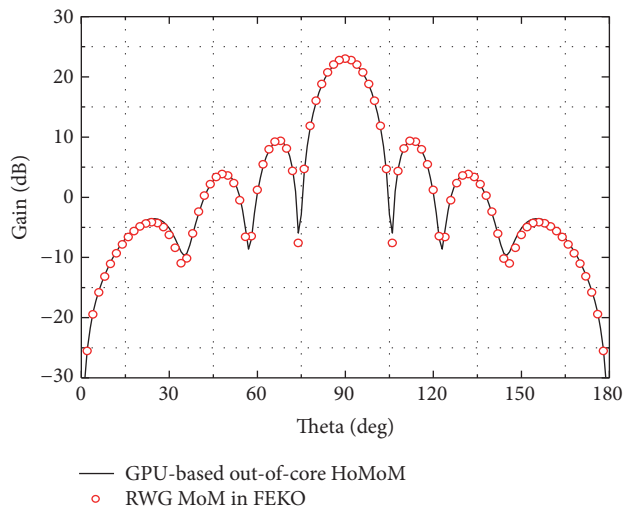

Figure 14: 2D results of the microstrip phased array in *xoz* plane.

From the comparisons one can see that the results using the parallel out-of-core HoMoM and the GPU-based out-of-core HoMoM agree with each other very well. The calculation parameters are given in Table 1, and the time in Table 1 is matrix equation solving time. The memory column shows the memory requirements of double complex dense matrices. In this simulation, there is no effect of the out-of-core scheme, because the GPU memory is enough to deal with this problem. According to Table 1, take the 24 CPU cores as a reference, the speedup of the hybrid CPU/GPU version is 2.309. But compared with the sequential HoMoM, a speedup of over 55 times can be obtained using the GPU-based version.

Install the 72 × 14 wire antenna array over an airplane, as shown in Figure 18. The dimensions of the airplane are 36 m × 40 m × 10.5 m. The simulation frequency is also 1.0 GHz and

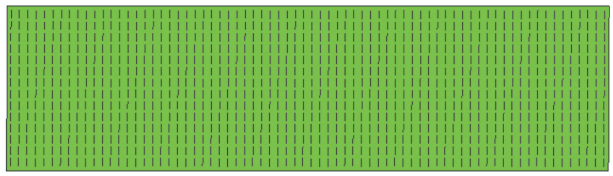

Figure 15: Model of the wire antenna array.

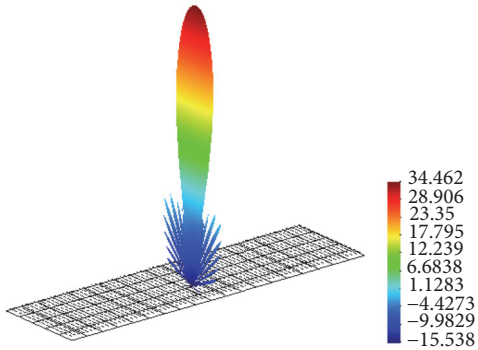

Figure 16: 3D gain patterns of the wire array.

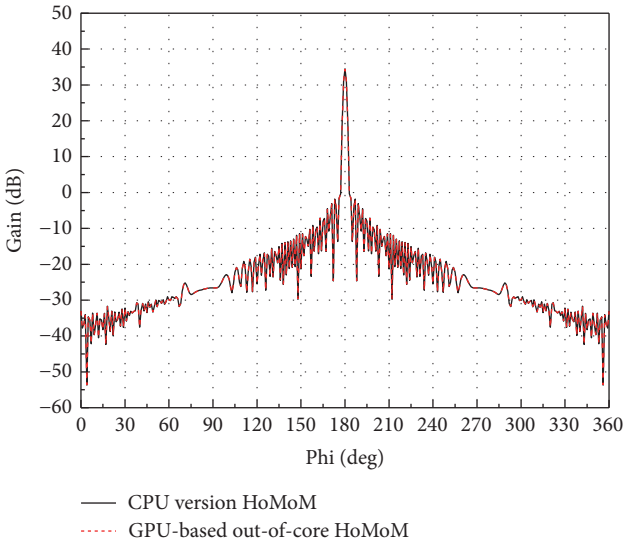

Figure 17: 2D results of the wire array in *xoy* plane.

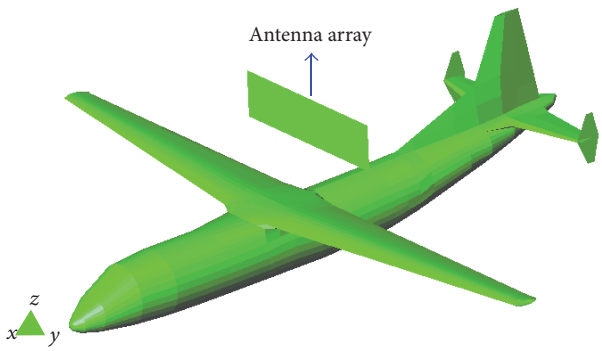

Figure 18: Model of airborne array.

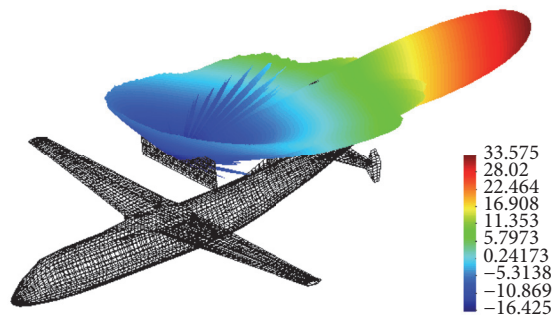

Figure 19: 3D results of the airborne wire antenna array.

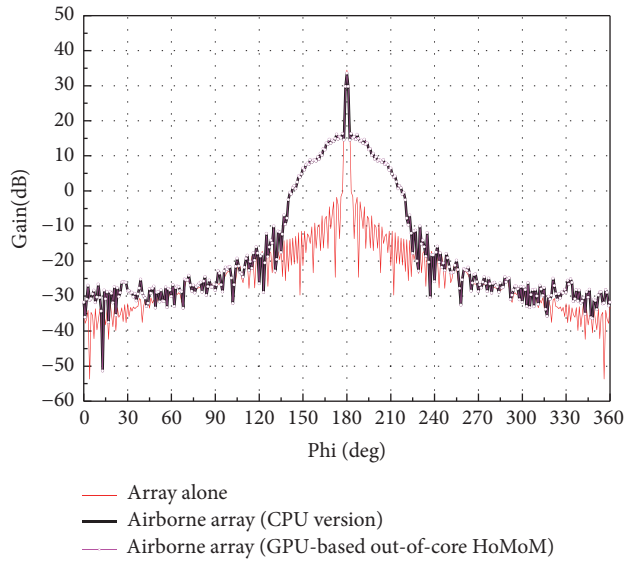

Figure 20: 2D results of the airborne wire antenna array in *xoy* plane.

the number of unknowns is 259,128. Here this airborne problem is simulated using two approaches, the CPU version out-of-core HoMoM and the GPU-based out-of-core HoMoM, to demonstrate the acceleration performance. The 3D and 2D gain patterns obtained by the proposed GPU-based out-of-core algorithm are shown in Figures 19 and 20, in which the corresponding results of the array alone are also given.

From the comparison one can see that the results using parallel out-of-core HoMoM and GPU-based out-of-core HoMoM agree with each other very well. Meanwhile, we can see that the mainlobe region of gain patterns (i.e., 178°~182° in the *xoy* plane) of the airborne wire antenna array shows slight changes, while the side-lobe region of gain patterns (i.e., 0°~45°, 135°~225°, and 315°~360° in the *xoy* plane) seriously deteriorates compared with the result of the array alone. When the main beam is towards the tail fin of the airplane, its gain patterns get much worse mainly due to the reflection effect of the airplane tail fin.

The computing resources and solving time are given in Table 2. According to Table 2, the memory requirement is about 1000 GB, but the available RAM and GPU memory are

Table 2: Computational parameters for the array installed on an airplane.

Number of unknowns	Memory (GB)	Computational resource	Solving time (s)	Speedup
259,128	**1000.573**	24 CPU cores	455893.93	1
		24 CPU cores and 2 GPUs	283050.01	**1.611**

less than 140 GB. It represents how the proposed GPU-based solver breaks the restriction of the RAM and GPU memory. Meanwhile, taking the 24 CPU cores as a reference, a speedup of 1.611 can be obtained using the GPU-base version. And compared with the sequential HoMoM, the speedup of the GPU-based version is over 38 times.

5. Conclusion

Using GPU-based two-level out-of-core scheme can solve larger EM problems at faster speed. In this paper, an airborne array problem with 259,128 unknowns is solved on a hybrid CPU/GPU cluster with about 10 GB GPU memory, 128 GB RAM, and 1 TB storages of HDD, and a speedup of 1.6 can be obtained compared with the parallel CPU version. The technique presented in this paper can also be used to accelerate the solution of radiation patterns from on-board antenna systems including complex antennas and large platforms.

Competing Interests

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

Acknowledgments

This work was supported in part by the program of International S&T Cooperation (2016YFE0121600), by the program for New Century Excellent Talents in University of China (NCET-13-0949), by the National High Technology Research and Development Program of China (863 Program) (2014AA01A302 and 2012AA01A308), by the NSFC (61301069), and by the Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase).

References

[1] R. F. Harrington, *Field computation by moment methods*, Wiley-IEEE Press, 1993.

[2] 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.

[3] N. Engheta, W. D. Murphy, V. Rokhlin, and M. S. Vassiliou, "The fast multipole method (FMM) for electromagnetic scattering problems," *Institute of Electrical and Electronics Engineers. Transactions on Antennas and Propagation*, vol. 40, no. 6, pp. 634–641, 1992.

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

[5] 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.

[6] 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.

[7] 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.

[8] Y. Zhang, X.-W. Zhao, D. G. Donoro, S.-W. Ting, and T. K. Sarkar, "Parallelized hybrid method with higher-order MoM and PO for analysis of phased array antennas on electrically large platforms," *Institute of Electrical and Electronics Engineers. Transactions on Antennas and Propagation*, vol. 58, no. 12, pp. 4110–4115, 2010.

[9] X.-W. Zhao, Y. Zhang, H.-W. Zhang et al., "Parallel MoM-PO method with out-of-core technique for analysis of complex arrays on electrically large platforms," *Progress in Electromagnetics Research*, vol. 108, pp. 1–21, 2010.

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

[11] NVIDIA Corporation, "NVIDIA GeForce 8800 GPU architecture overview," Tech. Brief TB-02787-001_v0.9, Santa Clara, Calif, USA, 2006.

[12] 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.

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

[14] S. Peng and C.-F. Wang, "Precorrected-FFT method on graphics processing units," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 4, pp. 2099–2107, 2013.

[15] 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.

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

[17] N. Kinayman, "Parallel programming with GPUs: parallel programming using graphics processing units with numerical examples for microwave engineering," *IEEE Microwave Magazine*, vol. 14, no. 4, pp. 102–115, 2013.

[18] E. Lezar and D. B. Davidson, "GPU-based LU decomposition for large method of moments problems," *Electronics Letters*, vol. 46, no. 17, pp. 1194–1196, 2010.

[19] E. D'Azevedo and J. C. Hill, "Parallel LU factorization on GPU cluster," *Procedia Computer Science*, vol. 9, pp. 67–75, 2012.

[20] T. Topa, "Efficient out-of-GPU memory strategies for solving matrix equation generated by method of moments," *Electronics Letters*, vol. 51, no. 19, pp. 1542–1544, 2015.

[21] Y. Zhang, M. Taylor, T. K. Sarkar, H. G. Moon, and M. Yuan, "Solving large complex problems using a higher-order basis: parallel in-core and out-of-core integral-equation solvers," *IEEE Antennas and Propagation Magazine*, vol. 50, no. 4, pp. 13–30, 2008.

[22] Y. Zhang, M. Taylor, T. K. Sarkar et al., "Parallel in-core and out-of-core solution of electrically large problems using the RWG basis functions," *IEEE Antennas and Propagation Magazine*, vol. 50, no. 5, pp. 84–94, 2008.

[23] Y. Zhang, T. K. Sarkar, M. Taylor, and H. Moon, "Solving MoM problems with million level unknowns using a parallel out-of-core solver on a high performance cluster," in *Proceedings of the IEEE Antennas and Propagation Society International Symposium (APSURSI '09)*, June 2009.

[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," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 11, pp. 5634–5646, 2014.

[25] Y. Chen, Z. Lin, Y. Zhang, S. Jiang, and X. Zhao, "Parallel out-of-core higher-order method of moments accelerated by graphics processing units," in *Proceedings of the IEEE Antennas and Propagation Society International Symposium (APS '15)*, pp. 1674–1675, British Columbia, Canada, July 2015.

[26] Y. Zhang, Z.-C. Lin, Z.-N. Yang, W.-H. Ge, X.-W. Zhao, and H. Zhao, "Simulations of airborne phased array using parallel MoM," in *Proceedings of the IET International Radar Conference 2013*, April 2013.

[27] Y. Zhang, Z. Lin, X. Zhao, and T. K. Sarkar, "Performance of a massively parallel higher-order method of moments code using thousands of CPUs and its applications," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 12, pp. 6317–6324, 2014.

[28] Z. Lin, Y. Chen, Y. Zhang, S. Jiang, and X. Zhao, "Solution of EM problems using hybrid parallel CPU/GPU implementation of higher-order MoM," in *Proceedings of the IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, 2015.

[29] Z. Lin, Y. Zhang, S. Jiang, X. Zhao, and J. Mo, "Simulation of airborne antenna array layout problems using parallel higher-order MoM," *International Journal of Antennas and Propagation*, vol. 2014, Article ID 985367, 11 pages, 2014.

[30] P. Ylä-Oijala, M. Taskinen, and S. Järvenpää, "Analysis of surface integral equations in electromagnetic scattering and radiation problems," *Engineering Analysis with Boundary Elements*, vol. 32, no. 3, pp. 196–209, 2008.

[31] R. F. Harrington, "Boundary integral formulations for homogeneous material bodies," *Journal of Electromagnetic Waves and Applications*, vol. 3, no. 1, pp. 1–15, 1989.

[32] J. L. Volakis and K. Sertel, *Integral Equation Methods for Electromagnetics*, Institution of Engineering and Technology, 2012.

[33] NVIDIA Corporation, *CUDA API Reference Manual Version 5.0*, NVIDIA Corporation, 2012.

[34] NVIDIA Corporation, *CUDA C Programming Guide*, PG-02829-001_v5.0, NVIDIA Corporation, Santa Clara, Calif, USA, 2012.

[35] NVIDIA Corporation, *CUBLAS LIBRARY User Guide v5.0*, NVIDIA Corporation, Santa Clara, Calif, USA, 2012.