Acceleration of Various Direct/Iterative Solvers for MoM by GPU and Its Computational Cost

Keisuke Konno, Qiang Chen
Department of Communications Engineering
Graduate School of Engineering, Tohoku University
Sendai, Miyagi, Japan
konno@ecei.tohoku.ac.jp

Hajime Katsuda
Wireless Systems Innovation Laboratory
NTT Network Innovation Laboratories
Yokosuka, Kanagawa, Japan

***Abstract*—Various guidelines for acceleration of MoM by GPU computing are summarized. Acceleration of direct/iterative solver for MoM by using GPU is realized. Quantitative study of computing time shows the performance of each guideline.**

I. Introduction

Method of moments (MoM) is well-known as one of the effective numerical techniques for the analysis of electric field integral equation (EFIE) [1]. Because the computing time of the MoM with conventional direct solver is $O(N^3)$ where N is the number of unknowns, various researches which aim to save the computing time of the MoM have been carried out. Iterative solver such as conjugate gradient (CG) method has been proposed to save the computing time of the MoM [2]. The order of computing time of the CG method is proportional to $N_{it}N^2$ where N_{it} is the number of iterations. The computing time of the iterative solver can be smaller than that of the direct solver when $N_{it} < N$. However, it is inevitable that the solution of the iterative solver includes error due to a tradeoff between computing time and accuracy.

Recently, graphics processing unit (GPU) has been used as a powerful device to save computing time of the MoM [3]. The GPU has hundreds of processors and numerical operations in the MoM can be performed in parallel when the GPU is applied to the MoM. In previous researches, the computing time of both direct/iterative solvers in the MoM has been saved by using the GPU [4], [5]. On the other hand, various programming guidelines which are suitable for the architecture of the GPU have been shown to enhance the performance of the GPU [6], [7]. In addition, how these guidelines affect the computing time of both direct/iterative solvers in the MoM has been clarified quantitatively [8].

In the reference [8], conventional direct/iterative solver (i.e. Gauss-Jordan method, CG method) has only been accelerated by these programming guidelines. On the other hand, as a faster direct/iterative solver, conjugate gradient fast multipole method (CG-FMM) [11] and characteristic basis function method (CBFM) [13] have been proposed, respectively. The effect of guidelines shown in [8] to computing time of both CG-FMM and CBFM has not been evaluated and compared quantitatively. In addition, it has not been clarified that how each guideline makes an impact on CG-FMM and CBFM which have different algorithm.

This paper is enhanced version of reference [8]. In this paper, at first, various programming guidelines for GPU computing are briefly reviewed. After that, these guidelines are applied to in-house code of Gauss-Jordan method, CG method, CG-FMM and CBFM. The same numerical example is solved by using these solvers with/without guidelines and its computing time is compared with each other. All numerical results in this paper are obtained from codes written in CUDA (compute unified device architecture) Fortran and GPU is NVIDIA Tesla C2075.

II. Review of guidelines for fast GPU computing

Following three guidelines have been shown to minimize the computing time of the GPU in [6] or [7].

1. Minimization of data transfer between CPU and GPU

In general, the processing speed of numerical operation by the GPU is much faster than that by the CPU. On the other hand, data transfer speed from the global memory of the GPU to that of the CPU is not so fast. Therefore, data transfer time can be dominant when frequent data transfer is carried out in the GPU computing. To maximize the computing speed of the GPU, it can be said that frequent data transfer between CPU and GPU should be avoided as long as possible.

2. Coalesced memory access to the global memory of the GPU

GPU memory consists of global memory, cash and shared memory. The cash and shared memory have fast data transfer speed compared with the global memory but the size of cash and shered memory is small and limited. On the other hand, the global memory has large size but its data transfer speed is slow compared with the cash and shared memory. Therefore, large $N \times N$ impedance matrix is stored in the global memory, when the large-scale problem of the MoM is carried out by the GPU. Because data transfer speed from processors to the global memory is slow compared with the processing speed of the GPU, the number of accesses to the global memory should be minimized in the MoM. Coalesced memory access has been recommended as a effective technique to reduce the number of accesses to the global memory. Processors can get a number of data from the global memory simultaneously and the number of accesses to the global memory can be reduced drastically, when the memory access is coalesced. Therefore, memory access to the global memory of the GPU should be coalesced to maximize the computing speed of the GPU.

3. The number of threads per block should be a multiple of 32 [6].

In the GPU computing, numerical operations are executed in parallel and assignment of numerical operations to processors is important to save computing time. The minimum unit of numerical operations executed by processors in GPU is called as thread and a number of threads is called as block. The number of threads per block which is assigned to processors in SM (streaming multiprocessors) can be controlled by users, where SM consists of many processors and memories. The maximum number of threads which can be executed in parallel is 32, because each SM in the GPU of Fermi architecture has 32 processors. Therefore, the number of threads per block should be a multiple of 32 to avoid wasting processors.

Due to the limitation of space, details of the above three guidelines are omitted here. For details of the above three guidelines, please refer to [6] or [7].

III. Direct/Iterative Matrix Solvers for MoM

A. Gauss-Jordan Method and CG Method

Gauss-Jordan method is conventional direct solver for MoM. The Gauss-Jordan method finds inverse matrix by pivot search and sweeping out. Because frequent access to the memory which stores impedance matrix is required for sweeping out, both memory access and data transfer should be minimized to save computing time.

On the other hand, CG method is one of the iterative solvers for MoM. The CG method updates approximate solution by iterative numerical operation including matrix-vector multiplication. Because matrix-vector multiplication is most computationally expensive part in the CG method, the optimum tuning for matrix-vector multiplication by using the guidelines is effective to save total computing time of the CG method.

B. CBFM

CBFM is one of the fast MoMs based on direct solver [13]. In the CBFM, the original large matrix equation is divided into block matrix equations. Characteristic basis function (CBF) is obtained as a solution of the block matrix equations and the original large matrix equation is reduced to smaller one by using the CBFs. The block matrix equation and the reduced matrix equation are solved by direct solver such as Gauss-Jordan method. Because the CBFM is based on the Gauss-Jordan method, the optimum tuning for the Gauss-Jordan method is also effective to save computing time of the CBFM.

C. CG-FMM

CG-FMM is one of the fast MoMs based on iterative solver [11], [12]. In the CG-FMM, CG method is combined with FMM which is based on the addition theorem of the scalar Green's function. By introducing the FMM, matrix-vector multiplication in the CG method can be carried out collectively. Because the CG-FMM is based on the CG method, the optimum tuning for the CG method is also effective to save computing time of the CG-FMM.

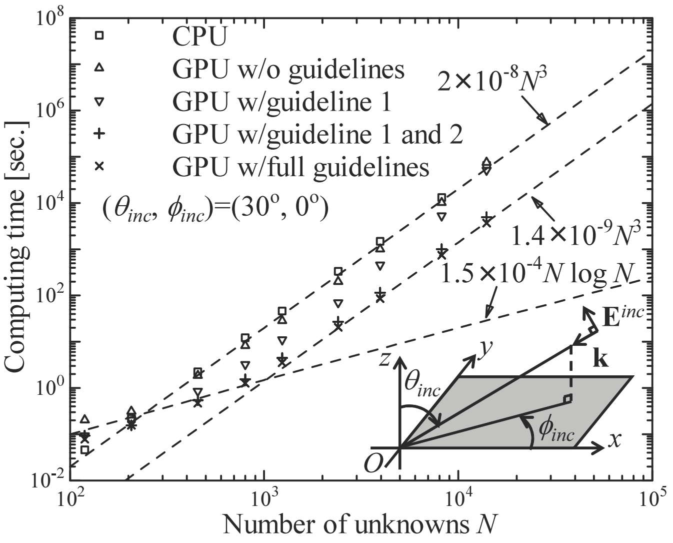

Fig. 1. Computing time of Gauss-Jordan method for numerical analysis of a planar scatterer with CPU and GPU.

TABLE I
Guidelines for direct/iterative solver.

Guidelines	Direct solver (Gauss-Jordan method)	Iterative solver (CG method)
w/guideline 1	All operations are executed by GPU	
w/guideline 2	Memory access to Z matrix is coalesced	
w/guideline 3	$N_T = 96$	

IV. Numerical Examples

In this section, the three guidelines for fast GPU computing which was reviewed in the previous section are applied to the CUDA programs of the direct/iterative solvers. Numerical analysis of a planar scatterer was carried out to evaluate the effect of the three guidelines. The planar scatterer was divided into a number of wire grid segments and Richmond's MoM was used for numerical analysis [15]. All results of numerical analysis in this paper were obtained by Intel Xeon 2.27 GHz CPU and NVIDIA Tesla C2075 GPU. Double precision real and complex number were used in our in-house programs. All guidelines for direct/iterative solver for MoM are shown in Table I.

A. Gauss-Jordan Method and CG Method

The computing time of Gauss-Jordan method is shown in Fig. 1. By applying the three guidelines to the CUDA programs, it is shown that the computing time of the Gauss-Jordan method becomes so small. On the other hand, the computing time of the Gauss-Jordan method w/o guidelines

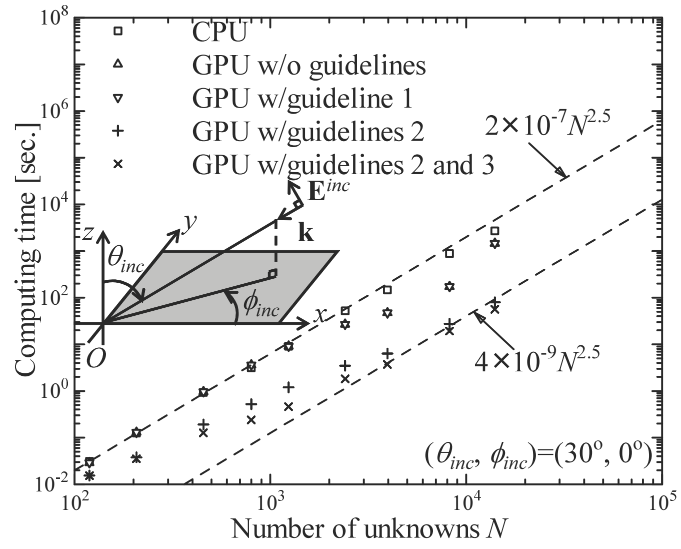

Fig. 2. Computing time of CG method for numerical analysis of a planar scatterer with CPU and GPU.

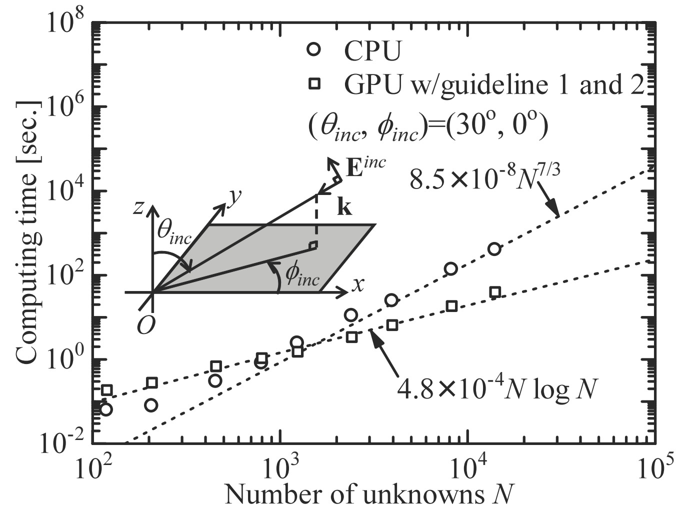

Fig. 3. Computing time of CBFM for numerical analysis of a planar scatterer with CPU and GPU.

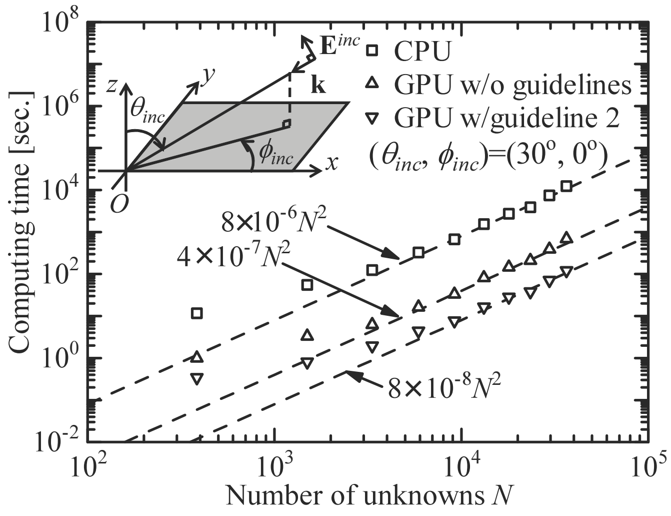

Fig. 4. Computing time of CG-FMM for numerical analysis of a planar scatterer with CPU and GPU.

becomes large due to frequent data transfer and inefficient memory access. When the Gauss-Jordan method w/o guideline 1 is executed, the pivot search and sweeping out which require $O(N^2)$ and $O(N^3)$ of computing time, respectively, are only executed by the GPU. After the both pivot search and sweeping out are executed by the GPU, Z matrix is modified and the modified Z matrix is required for the next numerical operation which is executed by CPU. Therefore, in the Gauss-Jordan method w/o guideline 1, the modified Z matrix whose size is $O(N^2)$ must be transferred from GPU to CPU every time the pivot search and sweeping out are finished. Data transfer time of $O(N^2)$ is large but can be avoided by applying the guideline 1. For the Gauss-Jordan method, the guideline 2 is the most effective to save the data transfer time because $O(N^2)$ memory access is required every time the pivot search and sweeping out are carried out. From Fig. 1, it is also confirmed that about 20 $\sim$ 30% of computing time can be saved by applying the guideline 3 to the CUDA program of the Gauss-Jordan method.

The computing time of Conjugate gradient (CG) method is shown in Fig. 2 and it can be said that the guideline 1 is not effective to save the computing time. When the CG method w/o guideline 1 is executed, the matrix-vector multiplication is only executed by the GPU. The Z matrix element remains the same from beginning to end in the CG method while the Z matrix element is modified by pivot search and sweeping out in the Gauss-Jordan method, Therefore, the maximum size of data which is transferred between CPU and GPU is at most $O(N)$ in the CG method w/o guideline 1 once the Z matrix element is transferred to the GPU. Because the computing time for matrix-vector multiplication in the CG method is $O(N^2)$, which is much larger than the data transfer time of $O(N)$ in the CG method, the guideline 1 has almost no effect on the computing time of the GPU. On the other hand, it is found that the computing time of the CG method becomes much smaller by applying the guidelines 2 and 3 to the CUDA programs. Because the memory access of $O(N^2)$ is required every time the matrix-vector multiplication is carried out, it can be said that the guideline 2 is most effective to save data transfer time. From Fig. 2, it is also found that about 20 $\sim$ 30% of computing time can be saved by applying the guideline 3 to the CUDA program of the CG method.

B. CBFM and CG-FMM

Here, the computing time of the CBFM and CG-FMM with guidelines is quantitatively evaluated. As shown in Fig. 1, it was found that the guideline 1 and 2 can save the computing time of Gauss-Jordan method greatly. Therefore, the guideline

1 and 2 are only applied to the Gauss-Jordan method used in the CBFM. On the other hand, according to the effects of guidelines in CG method shown in Fig. 2, it was shown that the guideline 2 can save the computing time of CG method greatly. Therefore, the computing time of the CG-FMM with the guideline 2 is quantitatively evaluated.

Fig. 3 shows the computing time of the CBFM. In the CBFM, it is known that the computing time of the CBFM becomes $O(N^{7/3})$ when the number of blocks $M = 0.9N^{1/3}$ [14]. As shown in Fig. 3, it is found that the computing time of the CBFM by using CPU is $O(N^{7/3})$. On the other hand, the computing time of the CBFM by using GPU is $O(N \log N)$. In the CBFM, the Gauss-Jordan method is used to solve block matrix equation and the size of block matrix is much smaller than that of the original matrix. From Fig. 1, it is found that the computing time of the Gauss-Jordan method becomes $O(N \log N)$ when the size of matrix is small. In this numerical example, the part of pivot search in the Gauss-Jordan method is tuned by the binary search tree algorithm and executed in parallel by GPU. Therefore, the computing time of the Gauss-Jordan method becomes $O(N \log N)$ when the number of processors in GPU is larger than the size of block matrix.

Fig. 4 shows computing time of the CG-FMM. It is well-known that the computational cost of the matrix-vector multiplication in the CG method can be reduced from $O(N^2)$ to $O(N^{1.5})$ by FMM. From the results of numerical analysis shown in Fig. 2 and 4, it is found that the order of the total computing time is reduced from $O(N^{2.5})$ to $O(N^2)$. From Fig. 4, it is found that the computing time of the CG-FMM with the guideline 2 is $1/5$ compared with that without guidelines. Therefore, it can be said that coalesced memory access is effective to save computing time of CG-FMM.

V. Conclusions

In this paper, three guidelines for GPU computing were applied to the direct/iterative solver for MoM. Results of numerical analysis showed that coalesced memory access (Guideline 2) is the most effective technique to save the computing time of these solvers. In addition, these guidelines were applied to the fast MoM such as the CBFM and CG-FMM. The computing time of the CBFM and CG-FMM with these guidelines were quantitatively investigated and summarized.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 25420394.

References

[1] R.F. Harrington, *Field Computation by Moment Methods*, New York, Macmillan, 1968.
[2] T.K. Sarkar, K.R. Siarkiewicz, and R.F. Stratton, "Survey of numerical methods for solution of large systems of linear equations for electromagnetic field problems," *IEEE Trans. Antennas Propag.*, vol.AP-29, no.6, pp.847-856, Nov. 1981.
[3] S. Peng and Z. Nie, "Acceleration of the method of moments calculations by using graphics processing units," *IEEE Trans. Antennas Propag.*, vol.56, no.7, pp.2130-2133, July 2008.
[4] E. Lezar and D.B. Davidson, "GPU-based LU decomposition for large method of moments problems," *Electron. Lett.*, vol.46, no.17, pp.1194-1196, Aug. 2010.
[5] D.D. Donno, A. Esposito, G. Monti, and L. Tarricone, "MPIE/MoM acceleration with a general-purpose graphics processing unit," IEEE Trans. Microw. Theory Tech., vol.60, no.9, pp.2693-2701, Sept. 2012.
[6] NVIDIA Corporation, "CUDA C Programming Guide," ver.5.0, NVIDIA Corp., Oct. 2012.
[7] The Portland Group, "CUDA Fortran Programming Guide and Reference," ver.13.1, The Portland Group, Jan. 2013.
[8] K. Konno, H. Katsuda, K. Yokokawa, Q. Chen, K. Sawaya, and Q. Yuan, "Quantitative study of computing time of direct/iterative solver for MoM by GPU computing, " IEICE Commun. Express, vol. 2, no. 8, pp. 359-364, 2013.
[9] M. Cwikla, J. Aronsson, and V. Okhmatovski, "Low-frequency MLFMA on graphics processors," *IEEE Antennas Wireless Propag. Lett.*, vol.9, pp.8-11, 2010.
[10] S. Peng and C.-F. Wang, "Precorrected-FFT Method on Graphics Processing Units," *IEEE Trans. Antennas Propag.*, vol.61, no.4, pp.2099-2107, April 2013.
[11] R. Coifuman, V. Rokhlin, and S. Wandzura, "The fast multipole method for the wave equation: A pedestrian prescription," IEEE Antennas Propag. Mag., vol.35, no.3, pp.7-12, June 1993.
[12] V. Rokhlin, "Rapid solution of integral equations of scattering theory in two dimensions," J. Comput. Phys., vol.86, no.2, pp.414-439, Feb. 1990.
[13] V.V.S. Prakash and R. Mittra, "Characteristic basis function method: A new technique for efficient solution of method of moments matrix equations," Microw. Opt. Technol. Lett., vol.36, no.2, pp.95-100, Janu. 2003.
[14] K. Konno, Q. Chen, K. Sawaya, and T. Sezai, "Optimization of block size for CBFM in MoM," IEEE Trans. Antennas Propag., vol.60, no.10, pp.4719-4724, Oct. 2012.
[15] J.H. Richmond and N.H. Geary, "Mutual impedance of nonplanar-skew sinusoidal dipoles," IEEE Trans. Antennas Propag., vol.AP-23, no.3, pp.412-414, May 1975.