Quantitative study of computing time of direct/iterative solver for MoM by GPU computing

Keisuke Konno[1a)], **Hajime Katsuda**[2], **Kei Yokokawa**[1], **Qiang Chen**[1], **Kunio Sawaya**[3], **and Qiaowei Yuan**[4]

[1] *Department of Communications Engineering, Graduate School of Engineering, Tohoku University*
6–6–05, Aramaki Aza Aoba Aoba-ku Sendai-Shi Miyagi 980–8579, Japan
[2] *Wireless Systems Innovation Laboratory, NTT Network Innovation Laboratories*
1–1, Hikarinooka Yokosuka-Shi Kanagawa 239–0847, Japan
[3] *New Industry Creation Hatchery Center, Tohoku University*
6–6–10, Aramaki Aza Aoba Aoba-ku Sendai-Shi Miyagi 980–8579, Japan
[4] *Sendai National College of Technology*
4–16–1, Ayashichuuou Aoba-ku Sendai-Shi Miyagi 989–3128, Japan
a) *konno@ecei.tohoku.ac.jp*

Abstract: Guidelines for reduction of computing time of direct/iterative solver for MoM on GPU computing are reviewed. Computing time of the direct/iterative solver with and without these guidelines is compared to show the computational efficiency of GPU computing quantitatively.

Keywords: MoM, GPU, direct solver, iterative solver
Classification: Antennas and Propagation

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.

© IEICE 2013
DOI: 10.1587/comex.2.359
Received June 10, 2013
Accepted July 29, 2013
Published August 20, 2013

[7] The Portland Group, "CUDA Fortran Programming Guide and Reference," ver.13.1, The Portland Group, Jan. 2013.
[8] M. Cwikla, J. Aronsson, and V. Okhmatovski, "Low-frequency MLFMA on graphics processors," *IEEE Antennas Wireless Propag. Lett.*, vol. 9, pp. 8–11, 2010.
[9] 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.

1 Introduction

Method of moments (MoM) is well-known as one of the effective techniques for the numerical analysis of antennas and scatterers [1]. For many years, various researches have been carried out for improvement of the MoM. One of the most important research topics for the MoM is the reduction of its computing time for large-scale problems. Because a conventional direct solver such as LU-decomposition has been used for solving the matrix equation of the MoM, the order of computing time of the MoM is $O(N^3)$ where N is the number of unknowns. Instead of the conventional direct solver, iterative solver such as conjugate gradient (CG) method has been proposed and its order of computing time is proportional to $N_{it}N^2$ where N_{it} is the number of iterations [2]. The computing time of the iterative solver can be smaller than that of the direct solver when $N_{it} < N$ but its solution includes error due to a tradeoff between computing time and accuracy.

In recent years, graphics processing unit (GPU) has been used as a powerful device to reduce computing time of the MoM [3]. Because the GPU consists of hundreds of processors, a part of numerical operations in the MoM can be carried out in parallel when the GPU is applied to the MoM. In previous researches, the GPU has been used to reduce the computing time of both direct/iterative solvers in the MoM [4, 5]. On the other hand, various guidelines for programming which are suitable for the architecture of the GPU has been proposed to improve the performance of the GPU [6, 7]. These guidelines have been applied to the iterative solvers such as MLFMA and Precorrected FFT [8, 9]. However, the effect of these guidelines to computing time of both direct/iterative solvers has not been evaluated and compared quantitatively. In addition, it has not been clarified that how each guideline makes an impact on direct/iterative solvers with different characteristics.

In this paper, at first, various guidelines are briefly reviewed. After that, these guidelines are applied to in-house direct/iterative solvers. 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 programs written in CUDA (compute unified device architecture) Fortran and GPU is NVIDIA Tesla C2075.

© IEICE 2013
DOI: 10.1587/comex.2.359
Received June 10, 2013
Accepted July 29, 2013
Published August 20, 2013

2 Review of guidelines for fast GPU computing

In references [6] or [7], it is recommended to keep following three guidelines in the CUDA programming to maximize the computing speed of the GPU.

1. Data transfer between CPU and GPU should be minimized.

In general, data transfer speed from the GPU to the CPU is relatively slow compared with the processing speed of numerical operation by the GPU. Therefore, data transfer between CPU and GPU should be avoided as long as possible to maximize the computing speed of the GPU.

2. Access to the global memory of the GPU should be coalesced.

Data transfer time from processors to the global memory in the GPU tends to be longer than computing time of processors. By using the cash and shared memory which have fast data transfer speed, the number of accesses from processors to the global memory in the GPU can be reduced and the computing speed of the GPU becomes faster. However, frequent access from processors to the global memory in the GPU still cannot be avoided in the numerical analysis of large-scale problems because the size of cache and shared memory in the GPU is limited and small (e.g. the total size of L1 cache and shared memory in Tesla C2075 is 64 KB. The size of L2 cache is 768 KB.). Coalesced memory access has been recommended to reduce the number of accesses to the global memory. When the memory access is coalesced, processors can receive a number of data from the global memory simultaneously and the number of accesses to the global memory can be reduced drastically. Therefore, 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 divided into many threads and a number of threads are grouped into a block. Here, thread means the minimum unit of operation, block means the assembly of threads. Threads in each block is assigned to processors in SM (streaming multiprocessors), where SM consists of many processors and memories. Because each SM in the GPU of Fermi architecture has 32 processors, the maximum number of threads which can be executed in parallel is also 32. 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].

3 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 MoM. Numerical analysis of a planar scatterer which is divided into a number

DOI: 10.1587/comex.2.359
Received June 10, 2013
Accepted July 29, 2013
Published August 20, 2013

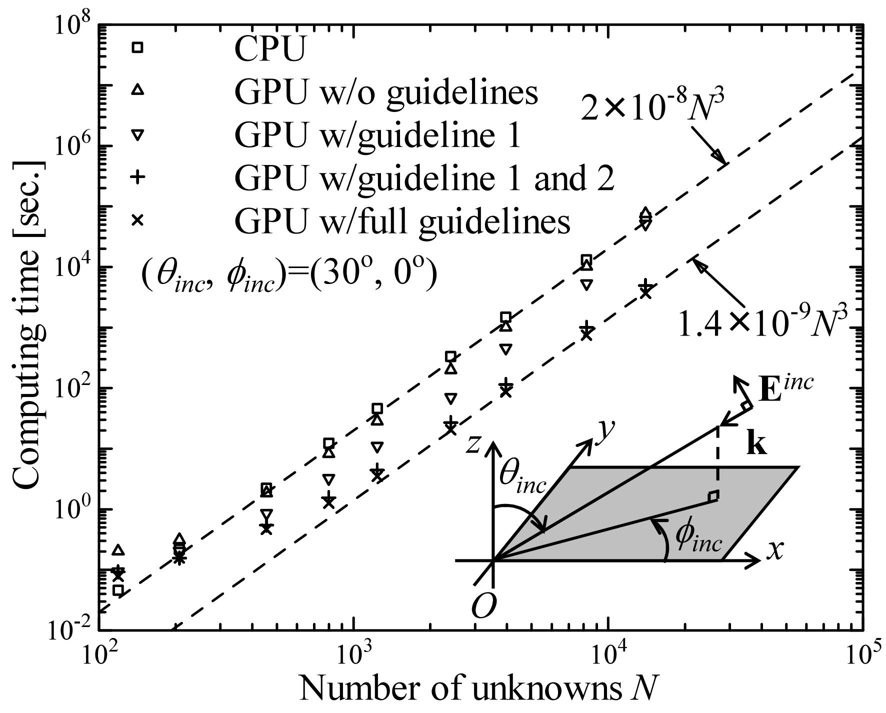

Fig. 1. Computing time of Gauss-Jordan method for numerical analysis of a planar scatterer with CPU and GPU.

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	$N_T = 96$	

of wire grid is carried out to evaluate the effect of the three guidelines. All numerical results in this paper are obtained by Intel Xeon CPU E5607 2.27 GHz without parallel programming and NVIDIA Tesla C2075 GPU. Double precision real and complex number are used in our in-house programs. All guidelines for direct/iterative solver for MoM are shown in Table I.

The computing time of Gauss-Jordan method is shown in Fig. 1. It is found that the computing time of the Gauss-Jordan method becomes so small by applying the three guidelines to the CUDA programs. On the other hand, the computing time of the Gauss-Jordan method w/o guidelines becomes large mainly 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 pivot search and sweeping out are executed by the GPU, Z matrix is modified and the modified Z matrix is required for the next operation which is executed by CPU. Therefore, in the Gauss-Jordan method w/o guideline 1, the modi-

© IEICE 2013
DOI: 10.1587/comex.2.359
Received June 10, 2013
Accepted July 29, 2013
Published August 20, 2013

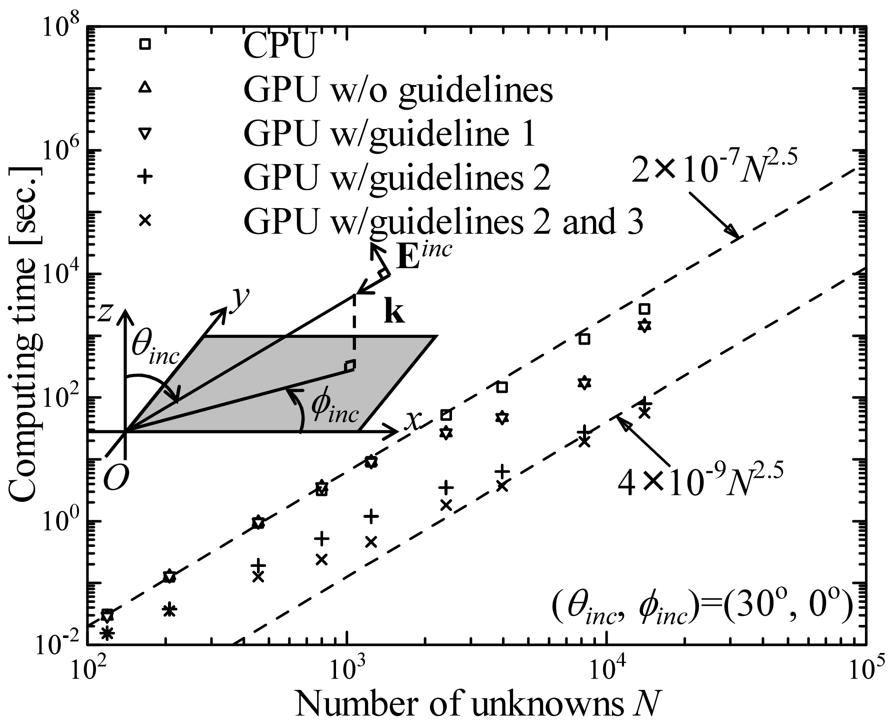

Fig. 2. Computing time of CG method for numerical analysis of a planar scatterer with CPU and GPU.

fied Z matrix whose size is $O(N^2)$ should be transferred from GPU to CPU every time the pivot search and sweeping out are finished. By applying the guideline 1 to the Gauss-Jordan method, such a frequent data transfer from GPU to CPU can be avoided and the computing time can be reduced as a result. For the Gauss-Jordan method, the guideline 2 is the most effective for reduction of the computing time because memory access of $O(N^2)$ 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 reduced by applying the guideline 3 to the Gauss-Jordan method.

The computing time of Conjugate gradient (CG) method is shown in Fig. 2. It is found that the guideline 1 is not effective for reduction of the computing time. When the CG method w/o guideline 1 is executed, the matrix-vector multiplication which requires $O(N^2)$ of computing time is only executed by the GPU. Unlike in the case of Gauss-Jordan method, the Z matrix remains the same from beginning to end of the CG 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 when the Z matrix is once 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 that for data transfer 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 for reduction of

© IEICE 2013
DOI: 10.1587/comex.2.359
Received June 10, 2013
Accepted July 29, 2013
Published August 20, 2013

the computing time. From Fig. 2, it is also found that about 20 $\sim$ 30% of computing time can be reduced by applying the guideline 3 to the CG method.

4 Conclusion

In this paper, three guidelines of direct/iterative solver for MoM by GPU computing were applied to these solvers. Numerical results showed that coalesced memory access (Guideline 2) is the most effective technique to reduce the computing time of these solvers. In addition, it was shown that the computing time of these solvers can be reduced when the number of threads per block is a multiple of 32 (Guideline 3). On the other hand, it was found that the minimization of the data transfer between CPU and GPU (Guideline 1) is only effective for reduction of computing time of the direct solver and not effective for the iterative solver.

Acknowledgments

Supercomputing resources at Cyberscience Center in Tohoku University were used in the process of this research.

© IEICE 2013
DOI: 10.1587/comex.2.359
Received June 10, 2013
Accepted July 29, 2013
Published August 20, 2013