Using AMG to Accelerate Finite Differences by GPUs in Electromagnetic/Thermal Field Simulations

Christian Richter[1], Sebastian Schöps[2,3] and Markus Clemens[1], *Senior Member, IEEE*

[1] Bergische Universität Wuppertal, Chair of Electromagnetic Theory,
Rainer-Gruenter-Str. 21, D-42119 Wuppertal, Germany
[2] Technische Universitaet Darmstadt, Institut fuer Theorie Elektromagnetischer Felder,
Schlossgartenstr. 8, D-64289 Darmstadt, Germany
[3] Technische Universitaet Darmstadt, Graduate School of Computational Engineering,
Dolivostr. 15, D-64293 Darmstadt, Germany

***Abstract*—The simulation of coupled electromagnetic-/thermal-problems with high resolution requires efficient numerical schemes. High performance computing languages like CUDA help unlocking the massively parallel capabilities of graphics processor units (GPUs) to accelerate calculations. This reduces the time needed to solve real world problems. In this paper, the speed-up is discussed, which is obtained by using the new CUDA 5.0 on recent hardware and multiple GPUs for coupled time domain simulations with finite difference schemes. In particular, extended memory allows solving larger problems with more degrees of freedom without swapping. Promising speedups can be obtained by splitting up the workload and using more sophisticated GPU-optimized preconditioners to reduce the number iteration steps.**

***Index Terms*—SpMV, Conjugate Gradients, GPU, CUDA, Bio-heat, Finite Differences, Multiphysics**

I. Introduction

Many simulators for problems in the high-frequency-domain, use finite difference schemes for coupled multi-physics problems. This applies for example to electromagnetic field dosimetry problems and in particular to the coupled numerical simulation of the specific absorption rate (SAR) and to the thermal field distribution in biological materials and structures, [1], [2], [3]. As an example for a numerical application the SAR distributions in the head of the model 'Duke' is considered in this paper.

The calculation of the thermal distribution requires the solution of an implicit problem: for the calculation of the static temperature distribution a conjugate gradient (CG) solver can be used to solve the following linear Poisson-type system of equations based on the Pennes-equation [4].

In this paper we discuss the numerical techniques that are necessary to unlock the new CUDA 5.0 features for coupled electromagnetic/thermal field simulations.

A. State of the Art

In recent years the usage of GPUs (graphics processor units) was proposed to accelerate the solution process of large scale electromagnetic problems including high-resolution models. In particular the numerical linear algebra was addressed in the electromagnetic community, e.g. [5].

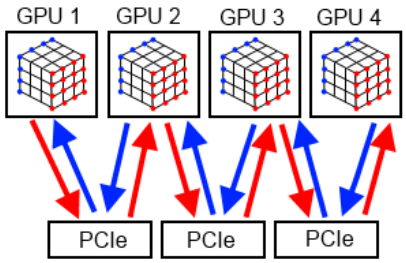

Fig. 1 Two step halo exchange, using the full PCIe's bandwidth, see [8]

In October 2012 Nvidia introduced CUDA 5.0. It comes with several new features, e.g., with '*dynamic parallism*' GPU kernels can dynamically create new kernels. Now, recursive sequences can be woven into existing codes. To make use of this promising feature very recent GPUs of the Kepler architecture, available since November 2012, are required.

Already in CUDA 4.0 some features for multi-GPU programming where introduced. With the *Unified Address Space (UAS)* the host and all attached GPUs share a common address space. Since CUDA 4.0 managing multiple GPUs from a single host is possible, while in earlier versions each GPU needed its own host thread. This was commonly realized using *Message Passing Interface (MPI)*. Interactions between two GPUs can be performed with *UAS Peer-To-Peer* via PCIe without involving the host. For CUDA 4.0 features GPUs of the Fermi or Kepler architecture are required.

II. Accelerating Electromagnetic-thermal Simulations

In [3] an implementation of a multi-physical code for electromagnetic and thermal field simulation was introduced. Recently [6] a GPU acceleration was presented. It uses the DIA-Format to take advantage of the shape of banded matrices caused by finite differences. Because of this storage format a significant speed-up can be realized compared to other storage formats like CRS. The reason for this is the more efficient storage order resulting in higher bandwidth utilizations.

For solving the linear systems with CG all matrices and vectors have to be uploaded to the GPU. Therefore, the global memory of the GPU limits the size of the problems that can be

TABLE 1
SPEEDUP FOR CG IN DOUBLE PRECISION, SEE FIG. 2, AND BANDWIDTH-RELATION FOR DIFFERENT DEVICES COMPARED TO AN I7 2.80 GHZ CPU, BENCHMARKED WITH [6]

Device	Speedup	Bandwidth [GB/s]	Bandwidth×t [GB]
Geforce GT540M	1.52	28.8	3846
Geforce GT640	1.47	28.5	3934
Geforce GTX285	8.40	159	3846
Tesla K20	12.08	209	3513

efficiently solved using the GPU. To overcome this impasse, a multi-GPU solver can be implemented. At best a speedup in the order of the number of GPUs used can be expected.

Communication between the GPUs normally reduces the speedup of an implementation. Two kinds of communication operations have to be looked at when parallelizing CG [7]

- For dot products the results of all GPUs are added up and broadcasted. This reduction can be done on the CPU and the result is uploaded to the GPUs. As modern GPUs of the Fermi- or Kepler-Architecture have at least one *copy* and one separate *calculation* engine, the time needed for CPU calculation and upload can be hidden behind other parallel calculations done on the GPUs.
- For sparse matrix-vector multiplications some elements of vectors residing on other GPUs are needed. These *halo*-elements [8] have to be exchanged among the GPUs. The UAS allows an efficient *peer-to-peer* exchange. First the last elements of each vector part are exchanged with their right neighbor, then the first ones with the left neighbor. This unlocks the full-duplex transfer rate of each PCIe bus of every GPU and requires only few additionally memory, see Fig. 1.

Tab. 1 shows the speed-up that can be obtained by CUDA 5.0 on current hardware including the new Tesla K20. Even though it's GK110 GPU is optimized for double-precision calculations, bandwidth remains the limiting factor for sparse matrix vector dominating the behavior of CG Solvers.

III. ADVANCED PRECONDITIONERS

Preconditioners are commonly used to improve convergence and reduce the required number of iteration steps of the CG method. In [6] a Jacobi preconditioner was employed. It combines the advantages of being easy to implement, showing good results in finite difference calculations and being optimally useable for the DIA-storage format and GPU memory access patterns. Fig. 2 shows the convergence of PCG versus CG using our multi-GPU in-house code.

Nevertheless, more sophisticated preconditioners promise higher speedups. Even if the single iteration step may take longer, the overall time could be compensated by step reduction. Especially at larger problems that can be solved on multiple GPUs, reducing the number of CG iteration steps may become a key feature.

With CUDA 5.0 (Kepler architecture) Nvidia introduced dynamic parallelism. This allows kernels to spawn new kernels and thus recursive code sequences on the GPU. With this feature a new closer look at multi-grid algorithms either as

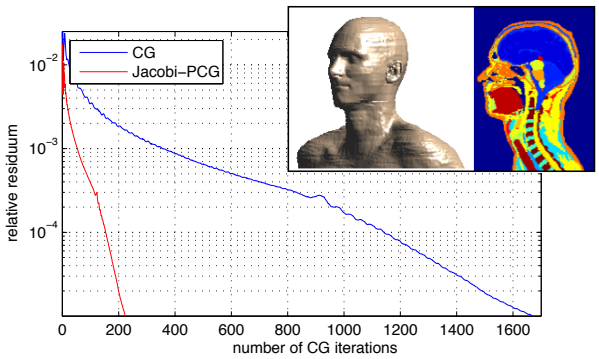

Fig. 2 Relative CG residuum in the simulation of the temperature distribution in a human head, see [6]; computed by the in-house multi-GPU code [3], performed sequentially on a Tesla K20.

stand alone solvers or as preconditioners must be taken. The possibilities of CUDA 5.0 for more complex preconditioners, e.g. [9], will be presented in the full paper.

IV. SUMMARY AND OUTLOOK

We have discussed recent advances in the development of a coupled electromagnetic/thermal FORTRAN code towards a CUDA 5.0 based code. In this digest we focused on the challenges of the multi-GPU implementation, e.g., regarding calculation time, maximum problem size and memory usage. We have shown first results of computations on Nvidia's new Tesla K20. In the full paper, different types of GPU-optimized preconditioners will be discussed regarding iteration steps, calculation time and bandwidth utilization.

Acknowledgements: The authors like to thank Nvidia for supporting this research by donating a Tesla K20 to the CUDA Research Center at the Bergische Universitaet Wuppertal. The second author is supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universitaet Darmstadt.

REFERENCES

[1] D. M. Sullivan, D. T. Borup, and O. P. Gandhi, "Use of the Finite-Difference Time-Domain method in calculating EM absorption in human tissues," *IEEE Trans. Bio. Eng.*, vol. BME-34, no. 2, pp. 148–157, 1987.
[2] J. Wang and O. Fujiwara, "FDTD computation of temperature rise in the human head for portable telephones," *IEEE Trans. Microw. Theor. Tech.*, vol. 47, no. 8, pp. 1528–1534, 1999.
[3] A. Bitz, Y. Zhou, A. E. Ouardi, and J. Streckert, "Occupational exposure at mobile communication base station antenna sites," *Frequenz*, vol. 63, no. 7-8, pp. 123–128, 2009.
[4] H. H. Pennes, "Analysis of tissue and arterial blood temperatures in the resting human forearm," *J. Appl. Physiol.*, vol. 1, pp. 93–122, 1948.
[5] M. Mehri Dehnavi, D. M. Fernández, and D. Giannacopoulos, "Finite-element sparse matrix vector multiplication on graphic processing units," *IEEE Trans. Magn.*, vol. 46, no. 8, pp. 2982–2985, 2010.
[6] C. Richter, S. Schöps, and M. Clemens, "GPU acceleration of finite differences in coupled electromagnetic/thermal simulations," *IEEE Trans. Magn.*, to be published.
[7] Y. Saad. "Krylov Subspace Methods on Supercomputers," *SIAM J. Sci. Stat. Comput.*, vol. 10, no. 6, p. 1200–1232, 1989.
[8] P. Micikevicius, "3D finite difference computation on GPUs using CUDA," *Proc. 2nd Workshop on General Purpose Processing on GPUs*, p. 79–84, 2009.
[9] M. Mehri Dehnavi, D. M. Fernández, and J.-L. Gaudiot, "Parallel sparse approximate inverse preconditioning on graphic processing units," *IEEE Trans. Parallel Distr. Syst.*, vol. PP, no. 99, p. 1–11, 2012.