Efficient Preconditioned Conjugate Gradient Parallelization on GPU

A. F. P. Camargos[1, 2], V. C. Silva[1]
[1] Universidade de São Paulo - Escola Politécnica
Av. Prof. Luciano Gualberto, T. 3, nº 158, 05508-900, São Paulo, Brasil
[2] Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais (IFMG) – Departamento de Engenharia
Rua Padre Alberico, nº 440, 35570-000, Formiga, Minas Gerais, Brasil
anaflavia@ifmg.edu.br, vivianecs@usp.br

***Abstract*—We present a performance analysis of a parallel implementation of both conjugate gradient and preconditioned conjugate gradient solvers using graphic processing units with CUDA parallel programming model. The solvers were optimized for a fast solution of sparse systems of equations arising from Finite Element Analysis (FEA) of electromagnetic phenomena. The preconditioners were Incomplete Cholesky factorization and Incomplete LU factorization. Results show that the speedup factor for the incomplete Cholesky decomposition was above 3 compared to the CPU implementation.**

***Index Terms*—Linear systems, Finite Element Methods, Parallel Processing, Performance Analysis, Graphic Processing Unit (GPU).**

I. Introduction

Emerging many-core platforms yield enormous raw processing power, in the form of massive parallelism [1]-[3]-[8]. Owing to both complexity and large scale of the problems arising in the Finite Element Analysis of electromagnetic phenomena, GPUs have recently been applied in iterative methods for solving sparse linear systems, especially in the sparse matrix-vector multiplication (SpMV), which represents their dominant computational cost [3]. This product is computed directly and affects the memory requirement and algorithm speed.

The conjugate gradient (CG) method is commonly used in iterative algorithm for solving sparse symmetric positive definite linear systems. However, in this method many iterations are required to reach a certain specified accuracy. A technique to accelerate the convergence is preconditioning the matrix, which improves its conditioning [6]. Two widespread preconditioning techniques are incomplete Cholesky factorization (IC) and incomplete LU factorization (ILU), which are often used along with the CG method in FEA of electromagnetic phenomena.

These preconditioners require extra storage for processing of triangular systems in the forward and backward substitution [4]-[6]. Due to its nature, these mechanisms are known to be the bottleneck of every sequential implementation of such preconditioners along with SpMV. However, these steps can be parallelized on graphics hardware, and are applied at each iteration of the sparse linear systems.

Therefore, the aim of this paper is twofold. First, we will focus on implementations of both CG and preconditioned conjugate gradient (PCG) methods, which are optimized for GPUs with CUDA parallel programming model. We will then compare their performance. Second, we will examine the impact of parallelization of forward and backward substitution in the PCG method. Here, we will show that a proper implementation of the algorithm can still provide significant speedups despite the inherently sequential nature of the two most well-known preconditioners used in the solution of sparse linear system by iterative algorithms.

II. Related Work

As mentioned previously, preconditioning is necessary for faster convergence of iterative methods, which can be accomplished using algorithms such as the IC an ILU factorizations. These factorizations decompose the coefficient matrix into two triangular matrices, which can then be solved by forward and backward substitution.

After the advent of NVIDIA CUDA, GPUs have drawn much more attention for sparse linear algebra and the solution of sparse linear systems [8]. There are several works which attempts to optimize the PCG method [8]-[2]-[5]. The first authors analyze the performance of the GPU using ICCG method and incomplete LU factorization preconditioned GMRES method. However, in the first case, the speedup was up to 3 for one case. The second one presents an ICCG implementation optimized on both multi-core and GPU architectures by using domain decomposition. In [5], the authors implemented a variable preconditioned Krylov subspace method with mixed precision on GPU for solving the linear system obtained from the edge element.

Other authors [9] presented some PCG implementations for GPU using CUDA, with both Jacobi and SSOR preconditioners, whereas in [7] the preconditioning matrix was obtained from an approximate inverse derived from the SSOR preconditioner.

Implementations of both Incomplete LU and Cholesky preconditioned iterative methods in GPU are presented in [4]. The author used these preconditioners with Bi-Conjugate Gradient Stabilized and Conjugate Gradient iterative methods, respectively. The performance achieved in this case was a speedup of 2 at most.

III. Methodology and Results

In this work, we analyze the solution of a system arising from the discretization of the Laplace equation in a 3D domain. The FEA was applied to compute the characteristics of a grounding system in steady state. The preconditioned matrix is factorized using both IC and ILU preconditioners [6].

The following methods of accessing and manipulating entries of the matrix were implemented: Coordinate Format (COO) and Compressed Sparse Row Format (CSR) [3]. The tests were executed on an Intel Core i5 CPU with 3.2 GHz, 16GB of RAM and a GPU NVidia Geforce GT 240 with a total of 12 multi-processors and 96 cores running at 0.8 GHz and 54.4 GB/sec memory bandwidth.

The total execution times (including element integration, assembling and CG solving) obtained for five different implementations are shown in Table I in order to compare their computational performance. Each test case, with two mesh sizes (coarse and fine), was run five times with a precision of 1e-6 for the CG solution and PCG solution. The values represent the average values of the five runs.

The first and second columns refer to the approaches executed in CPU, whereas the third, fourth and fifth ones were run in the GPU. In the first column, the implementation uses sparse data structure (SDS), whereas in columns two to five the CSR format was employed. In the second column, the COO format was used, as well. The substantial improvement in the computational performance with the use of GPU parallel solving is self-evident from the Table I.

TABLE I
TOTAL EXECUTION TIME IN SECONDS

Mesh - NNZ*	**CPU**		**GPU**		
	CG – SDS	**CSR – SpMV**	**CG - CSR**	**ILUCG - CSR**	**ICCG - CSR**
coarse – 325,992	30701.84	32.41	12.20	10.88	10.79
fine – 682,689	142203.52	92.53	35.82	35.78	30.67

* Matrix non-null entries

The first and second implementations are sequential and were executed in the CPU using CG for solving sparse linear systems, with a SDS. The CSR–SpMV, however, was developed using compressed sparse row format in the sparse matrix-vector. Notice that the time to solve CSR–SpMV using compressed format was faster than CG–SDS.

The other methods were executed in the GPU with CUDA parallel programming model. For each iteration of the ICCG and ILUCG, it is necessary to perform one sparse matrix-vector multiplication and two triangular solutions, which corresponds to the most costly step in the solution process. In the pure CG implementation, this triangular solution is not necessary.

For the ICCG algorithm, the processing of the triangular systems in the forward and backward substitution consumes around 26% of the total time in the GPU. Nevertheless, the total time to solve the system without preconditioning is superior, namely, 10.79 seconds for the ICCG-CSR implementation against 12.20 seconds for the CG-CSR implementation.

The results of the numerical experiments in the Table I were used to obtain the speedup, shown in Fig.1. The speedup with respect to the sequential CSR–SpMV method. It can be seen that the ICCG implementation is three times faster than CSR-SpMV method and also faster than ILUCG. The number of iterations with the two meshes was nearly the same (105 - 111).

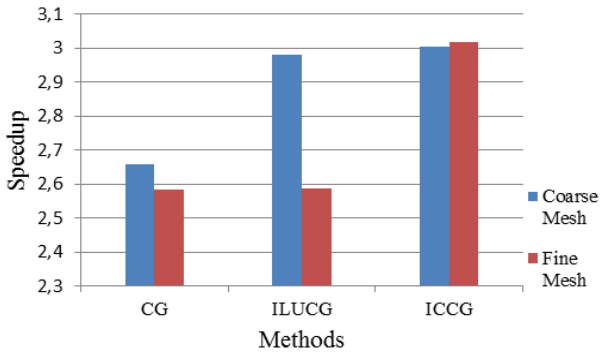

Fig. 1. Speedup with respect to CSR-SpMV

Furthermore, the speedup of the preconditioned solvers was better when compared to the pure CG algorithm.

IV. CONCLUSIONS

We present a comparative analysis of the performance of both CG and PCG algorithms implemented in GPU using CUDA. Despite the forward and backward substitution steps that occur in PCG algorithms, their GPU implementations still perform better than pure CG versions, thanks to the significant reduction in the number of iterations in the former. The speedup we achieved was similar to that reported in [8], nevertheless, the total execution time that we present also includes element integration and assembling of the global system.

ACKNOWLEDGEMENT

The authors would like to acknowledge CNPQ (Conselho Nacional de Desenvolvimento Científico e Tecnológico).

REFERENCES

[1] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol.96, nº.5, pp. 879-899, 2008.

[2] H. Moghnieh, D. A. Lowther, "Understanding the efficiency of parallel incomplete Cholesky preconditioners on the performance of ICCG solvers for multi-core and GPU systems," *IEEE 14th Biennial Conference on Electromagnetic Field Computation (CEFC)*, Jan. 2010.

[3] N. Bell and M. Garland. "Efficient sparse matrix-vector multiplication on CUDA," NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, Dec. 2008.

[4] M. Naumov, "Incomplete-LU and Cholesky preconditioned iterative methods using CUSPARSE and CUBLAS," NVIDIA Corporation, Jun. 2011. Available from: https://developer.nvidia.com.

[5] S. Ikuno, Y. Kawaguchi, N. Fujita, T. Itoh, S. Nakata, K. Watanabe, "Iterative solver for linear system obtained by edge element: variable preconditioned method with mixed precision on GPU," *IEEE Transactions on Magnetics*, vol. 48, n° 2, Feb. 2012.

[6] Y. Saad, *Iterative Methods for Sparse Linear Systems*, New York: PWS Publishing, 2nd ed., 2003, pp. 275-369.

[7] R. Helfenstein, J. Koko, "Parallel preconditioned conjugate gradient algorithm on GPU", *Journal of Computational and Applied Mathematics*, Elsevier, vol. 236, issue 15, pp. 3584–3590, Sep. 2012.

[8] R. Li, Y. Saad, "GPU-accelerated preconditioned iterative linear solvers," Available from: http://citeseerx.ist.psu.edu/index.

[9] M. Ament, G. Knittel, D. Weiskopf, W. Straßer, "A parallel preconditioned conjugate gradient solver for the Poisson problem on a multi-GPU platform," *IEEE 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 583-592, Feb. 2010.