Journal of
Information Technology & Software Engineering

Khirodkar, J Inform Tech Softw Eng 2016, 6:2
DOI: 10.4172/2165-7866.1000177

Review Open Access

A Survey on Optimization and Parallelization of Conjugate Gradient Solver

Khirodkar PP*

Department of Information Technology, Savitribai Phule University, Pune, India

***Corresponding author:** Department of Information Technology, Savitribai Phule University, Pune, India, Tel: 020 2569 6064; E-mail: puja.khirodkar@gmail.com

Rec date: Jan 20, 2016; **Acc date:** Apr 24, 2016; **Pub date:** Apr 30, 2016

Abstract

Conjugate Gradient Solver is a well-known iterative technique for solving sparse symmetric positive definite (SPD) systems of linear equations. The aim of this paper is to optimize and parallelize the currently available Conjugate Gradient Solver for OpenFOAM (Open source Field Operation and Manipulation) on GPU using CUDA which stands for Compute Unified Device Architecture, is a parallel computing platform and application programming interface (API) model created by NVIDIA. OpenFOAM is a C++ toolbox for development of customized numerical solvers of continuum mechanics problems, including Computational Fluid Dynamics. Existing Conjugate Gradient Solver can be optimized with the help of some techniques available for sparse matrix storage like Compressed Sparse Vecto (CSV).

Keywords: Iterative methods; Convergence; Sparse and very large systems; Linear systems; Parallel programming

Introduction

Conjugate Gradient method is the one of the most popular and well known iterative techniques for solving systems of linear equation which involves large sparse symmetric positive definite (SPD) matrices. A sparse matrix is a matrix which involves primarily large number of zero elements.

The general form of system of equation to solve is,

$$Ax = b$$

where, A is the symmetric positive definite matrix of order $m \times n$

m is the number of rows

n is the number of columns

b is the $n \times 1$ known right hand side vector.

x is the $m \times 1$ solution vector which we want to calculate.

Need of parallelization

One relevant purpose of using parallelism is to obtain the desirable reduction in the execution time, measured by the speedup metric. In the parallel computing context it is calculated as the ratio between the sequential execution time and the parallel execution time. Regarding only the number of processing elements (PE) executing a parallel program, it can be said the maximum speedup is equal to the number of PEs (linear speedup).

Related Work

The solution of large sparse linear systems, which are mainly arisen from the discretization of partial differential equations (PDEs), is an important problem in scientific computing. Krylov subspace methods such as Conjugate Gradient (CG) and Generalized Minimal Residual (GMRES) gain popularity in solving this problem because they require only matrix-vector products and a few vector operations per iteration, thus can be easily implemented on high-performance computers. Yao Chen et al suggested the two pre conditioners in order to accelerate convergence which is Incomplete Cholesky and Symmetric successive over-relaxation (SSOR)[1]. They also suggested level scheduling to increase multi-thread parallelism of sparse triangular solve on GPU. Level Scheduling splits the computation into two phases: an analysis phase which groups unknowns into different levels so that all unknowns of the same level can be determined simultaneously followed by a solve phase which solves the triangular system level by level. They analysed that the standard method of solving triangular system is forward/backward substitution which is serial processing and hard to parallelize on GPU.

Chevallier et al. [2] proved that the GPU speedup tends to increase with the problem size. Due to GPU setup and communication times, the speedup is greater than one typically for large problems[2]. From their experiments it clearly appears that unit-test models do not follow a simple relationship between speedup and the problem size, i.e., several values of speedup are found for a given value of the problem size where alternator was considered for experiment. So they then considered a rectangular rod for their experiment. Also they explained about OpenCL as its a standard for parallel computing consisting of a language (an extension of C), API, libraries and a runtime system. OpenCL is based on a platform model that divides a system into one host and one or several compute devices. Compute devices act as co-processors (e.g. GPUs) to the host (e.g. CPU). An OpenCL application is 1 executed in the host, which sends instructions, defined in special functions called kernels, to the device. A single host can manage multiple devices, even heterogeneous devices. OpenCL allows for creating contexts and queues in order to manage tasks being launched by the host in all attached devices.

Peixoto et al. [3] analyze the performance of the GPU using the incomplete Cholesky (IC) CG method (ICCG) and incomplete LU (ILU)preconditioned GMRES method [3].Viviane Cristine Silva presents an ICCG implementation architectures using domain

decomposition. They analysed two models where Model 1denotes the solution of the linear system of real unknowns, which models the 3-D elliptic problem, whereas Model 2stands for the solution of the complex linear system issued from the time-harmonic [3].

Guiming Wu et al. proposed a high performance architecture of Conjugate Gradient Solver on field programmable gate arrays(FPGAs),which can handle symmetric positive define systems of arbitrary size[4]. They suggested that we must partition sparse matrices to fit in the limited internal RAM assume that sparse matrix and vectors are stored in off-chip memory. It is different from some previous work, where only the internal RAM is used to store the matrix and vectors. In our design, a block of x is transferred to the internal RAM before computing, and then a block of A is streamed into our architecture to drive the computation. After all the blocks from the same row are finished, a block of y will be obtained and stored back to the off-chip memory.

et al.presents some obstacles to accelerate the preconditioned conjugate gradient (PCG) method on modern graphic processing units (GPUs) is presented and several techniques are proposed to enhance its performance over previous work independent of the GPU generation and the matrix sparsitypattern [5]. The proposed algorithm outperforms previous methods on both platforms. Unlike previous methods, which are not optimized for matrices with small number of non-zeros per row, the proposed optimizations, independent of the matrix sparsitypattern, are able to increaseconsiderably the performance for such matrices.

One of the paper is based on the implementation of five different algorithms on the GPU, and comparison of their performance with those of a CPU. The GPU-based solver was approximately eight times faster. Using CUDA, a 2D solver with multi-grid Full Approximation Scheme (FAS) used to accelerate convergence of all flow variables (u, v, and p). Steady-state calculations of driven cavity flow with 4096 x 4096 could be performed in a minute of GPU time [6,8].

Halfhill presented the real problem with parallel processing which is too many solutions. Finding the best one for a particular application isn't easy [9]. Choosing a processor architecture is only the first step and may be less important than the software-development tools, the amount of special programming required.

Proposed Work

For solving linear equation systems having large sparse matrices, the main time consuming part is matrix manipulation[7].This sparse matrix manipulations can be made quicker by the use of different sparse matrix storage formats such as

- Compressed Sparse Row format(CSR)
- Compressed Sparse Column format (CSC)
- Coordinate storage format(COO)

All of these techniques have some disadvantages [8]. To overcome them, the new technique is proposed i.e. Compressed Sparse Vector(CSV) which will be better if used than Compressed Sparse Row (CSR) format. There are many advantages of CSV over CSR format such as 1)Less Storage Volume, 2)Ease of Transpose Matrix Calculation, 3) High Speed,4)Broad Range for Storage Sparse Matrices [5].

Conclusion

This new method which we called it Compressed Sparse Vector (CSV) format, for storage of coefficient matrix A of linear system , has been based on row counting indexing, in CSV method, growing rate of indices values has been controlled by restarting indices after passing each non-zero element. Storage compaction in this new method will be better than other methods. Also, calculating of transpose of matrix A is very simple without any computation cost. Furthermore, we can conclude that application of CSV method for representing sparse matrices will not only reduce the storage volume of the compressed matrix, but also it increases the speed of the computers in practice. Also, using this method is suitable for dense sparse matrices, therefore, a broad range of sparse matrices could be compressed. Thus, since memory is an issue, the method's low storage requirements provide a means to tackle very large problems which would otherwise be out of reach.

Acknowledgement

I am indeed thankful to my internal guide Prof. P. P. Joshi and external guide Dr. Vikas Kumar for their able guidance and assistance to complete this paper. Im grateful for their valued support and faith on me. I extend my special thanks to Head of Department of Computer Engineering, Dr. Girish P. Potdar who extended the preparatory steps of this paper-work.

References

1. Chen Y, Zhao Y, Zhao W, Zhao L (2013) A Comparative Study of Preconditioners for GPU-Accelerated Conjugate Gradient Solver. 2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing.
2. Rodrigues AW, Chevallier L, Menach YL, Guyomarch F (2013) Test Harness on a Preconditioned Conjugate Gradient Solver on GPUs: An Efficiency Analysis. IEEE Transactions on Magnetics 49: 1729 - 1732.
3. Camargos AFP, Silva VC, Guichon JM, Munier G (2014) Efficient Parallel Preconditioned Conjugate Gradient Solver on GPU for FE Modeling of Electromagnetic Fields Highly Dissipative Media. IEEE Transactions on Magnetics 50.
4. Wu G, Xie X, Dou Y, Wang M (2013) High-Performance Architecture for the Conjugate Gradient Solver on FPGAs. IEEE Transactions on Circuits and Systems-Ii: Express Briefs 60.
5. Farzaneh A, Kheiri H, Shahmersi MA (2009) An Efficient Storage Format For Large Sparse Matrices. Commun Fac Sci Univ Ank Series A1 58: 1-10.
6. Ament M, Knittel G, Weiskopf D, Strasser W (2010) A parallel preconditioned conjugate gradient solver for the poisson problem on a multi-gpu platform. Parallel, Distributed and Network-Based Processing (PDP), 18th Euromicro International Conference, Pisa.
7. Bell N, Garland M (2008) Efficient sparse matrix-vector multiplication on CUDA. NVIDIA Corporation.
8. Saad Y (2003) Iterative Methods for Sparse Linear Systems. 2nd ed, New York, NY, USA.
9. Halfhill TR (2008) Parallel Processing With CUDA.