Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 1 (2012) 50 – 54

Procedia Technology

INSODE 2011

Investigation of the performance of LU decomposition method using CUDA

Caner Ozcan[a], Baha Sen[a]

[a]Karabuk University, Department of Computer Engineering, Karabuk, 78050, Turkey

Abstract

In recent years, parallel processing has been widely used in the computer industry. Software developers, have to deal with parallel computing platforms and technologies to provide novel and rich experiences. We present a novel algorithm to solve dense linear systems using Compute Unified Device Architecture (CUDA). High-level linear algebra operations require intensive computation. In this study Graphics Processing Units (GPU) accelerated implementation of LU linear algebra routine is implemented. LU decomposition is a decomposition of the form A=LU where A is a square matrix. The main idea of the LU decomposition is to record the steps used in Gaussian elimination on A in the places where the zero is produced. L and U are lower and upper triangular matrices respectively. This means that L has only zeros above the diagonal and U has only zeros below the diagonal. We have worked to increase performance with proper data representation and reducing row operations on GPU. Because of the high arithmetic throughput of GPUs, initial results from experiments promised a bright future for GPU computing. It has been shown useful for scientific computations. GPUs have high memory bandwidth and more floating point units as compared to the CPU. We have tried our study on different systems that have different GPUs and CPUs. The computation studies were also evaluated for different linear systems. When we compared the results obtained from both systems, a better performance was obtained with GPU computing. According to results, GPU computation approximately worked 3 times faster than the CPU computation. Our implementation provides significant performance improvement so we can easily use it to solve dense linear system.

Keywords: LU decomposition, dense linear systems, gpu computing, cuda

1. Introduction

With the rapid advancement in the world information industry, as a result of the problems with the widespread use of computer technology, develops personal sector workers who have forced to find quick solutions. Size of the data offered for the users increase each passing day and the need for computer systems capable of processing faster increases continuously. Processing speed is becoming crucial and offering services in shorter periods of time are provided with powerful computers by sharing the processing load. In addition, simulations of scientific and engineering problems require high calculation speed. For this reason, software developers, have to deal with parallel computing platforms and technologies to provide novel and rich experiences.

In comparison to the central processor's traditional data processing pipeline, performing general-purpose computations on a GPU is a new concept. In fact, the GPU itself is relatively new compared to the computing field at large. However, the idea of computing on graphics processors is not as new as you might believe. GPUs are optimized for fast rendering of geometric primitives for computer games and image generation. They are now

2212-0173 © 2012 Published by Elsevier Ltd.
doi:10.1016/j.protcy.2012.02.011

available in almost every desktop, laptop, game console and are becoming part of handheld devices. GPUs have high memory bandwidth and more floating point units as compared to the CPU [1]. CUDA programming model allows us to execute thousands of threads simultaneously on the GPU. CUDA organizes a parallel computation using the abstractions of threads, blocks and grids. The CUDA programming model guides the programmer to expose substantial fine-grained parallelism sufficient for utilizing massively multithreaded GPUs, while at the same time providing scalability across the broad spectrum of physical parallelism available in the range of GPU devices [2].

Many studies have been done on solving dense linear systems and LU decomposition using CUDA. In one study the fastest implementations of dense LU, QR and Cholesky factorizations running on a single or double NVIDIA GPUs is presented [3]. A novel GPU-based algorithm for solving dense linear systems is studied. It is reduced the problem to a series of rasterization problems and use appropriate data representations to match the blocked rasterization order and cache pre-fetch technology of a GPU [4]. The design and implementation of LU decomposition on the programmable GPU is presented. To study the performance behavior of modern GPUs, some implementation approaches in terms of loop, branch and vector processing have been developed and evaluated [5]. In another study CULA, a robust linear algebra library for computations in a hybrid CPU/GPU environment is presented. Results from factorizations such as LU decomposition, singular value decomposition and QR decomposition along with applications like system solution and least squares are studied [6]. In one study three blocked variants of the Cholesky and the LU factorizations using highly tuned implementations of BLAS on a G80 graphics processor and an Intel processor have been evaluated [7].

In this study we perform a novel algorithm to solve dense linear systems using CUDA. High-level linear algebra operations require intensive computation. GPU accelerated implementation of LU linear algebra routine is implemented. We have worked to increase performance with proper data representation and reducing row operations on GPU. We have tried our study on different systems that have different GPUs and CPUs. The computation studies were also evaluated for different linear systems. When we compared the results obtained from both systems, a better performance was obtained with GPU computing.

2. LU Decomposition Method

Linear systems, a mathematical model based on the use of the linear operator. Linear systems with typical characteristics have more simple features than non-linear systems in general. Linear equation systems have important applications in the field of signal processing, telecommunications and automatic control theory as a mathematical abstraction or idealization. In solving engineering problems with numerical methods a problem is often reduced to the problem of solving linear equation system and the solution of this equation system is performed in a way convenient and fast. For this purpose linear equation systems are solved by writing on matrix form. Matrix form representation of linear system equations are as follows:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix} x \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_N \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_N \end{Bmatrix} \quad (1)$$

In many applications of linear systems, we needs to solve Ax = b for many different vectors b. Gaussian elimination method is the most efficient and accurate way to solve a linear system. Most of time is spent on matrix A in the studies made with this method and size of matrix A is large. If we need to solve different systems with matrix A, we should avoid repeating the steps of Gaussian elimination on A for every different b. In linear algebra, LU decomposition (also called LU factorization) is a matrix decomposition which writes a matrix as the product of a lower triangular matrix and an upper triangular matrix. The product sometimes includes a permutation matrix as well. This decomposition is used in numerical analysis to solve systems of linear equations or calculate the determinant of a matrix. LU method is more complex but more efficient than Gauss method for solving an equation system [8].

In fact, the LU decomposition method is based on factorization of the matrix. However, matrices are not

allocated with random or arbitrary factors on this method and are separated upper triangular L and lower triangle U matrices of the factors.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} x \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & a_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \quad (2)$$

LU decomposition can be used to solve a system of equations and steps to solve a system using LU decomposition are as follows:

- Set up the equation Ax = b.
- Find LU decomposition for A. This will yield the equation (LU)x = b.
- Let y = Ux. Then solve the equation Ly = b for y.
- Take the values for y and solve the equation y = Ux for x. This will give the solution to the system Ax = b.

3. GPU Computing and Cuda

GPU computing is the use of a GPU to do general purpose scientific and engineering computing. The usage of CPU and GPU together in a heterogeneous co-processing computing model is called GPU computing model. CPU operates the sequential part of the program and GPU accelerates the computationally-extended non-sequential part of the program. According to the user, due to the usage of high performance of the GPU to increase it, the program runs faster than before.

Floating point performance of GPU has reached to teraflops level in recent years. Nvidia presents that not only floating point operations per second and chip bandwidth are better, but they also go on increasing at a lot faster rate on the GPU compared to CPU. GPU is developed into a highly parallel, multithreaded, many core processor with great computational power and very high memory bandwidth by taking into consideration the market demands, as given in Fig 1 [9]. The FLOPS achieved by GPU's have increased more quickly than that of the fastest and the bandwidth of the GPU has increased at a much faster rate than the chip bandwidth on the CPU. Therefore, our choice of using the GPU for parallelism enables us to construct large portions of our algorithm simultaneously and take advantage of the GPU's computational superiority. This allows us to achieve considerable speedup of solving dense linear systems.

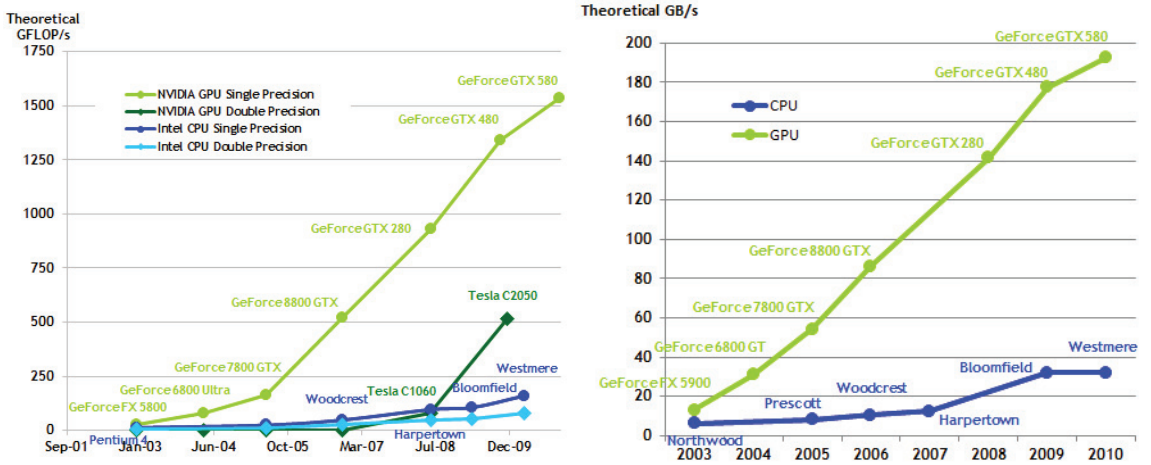

Fig. 1: (Left) The maximum number of FLOPS with CPU and GPU. (Right) Memory bandwidth for the CPU and GPU from 2003 to 2010.

Nvidia [9] changed the GPU radically and carried the computing world into a further level by introducing its new completely parallel architecture which is called CUDA. Hundreds of processor cores which runs together to work the data set in the program are included in the CUDA architecture. The success of GPUs has made the programming of the CUDA parallel model easy in the past few years. The application of this programming model has been modified by the program developers to get compute-intensive kernels and map them to the GPU. Then the CPU runs

the rest of the application. Not only rewriting the function to provide the parallelism in the function but also adding C keywords to move data to and from the GPU are used to map a function to the CPU. Tens of thousands threads are simultaneously launched simultaneously by the developer. The GPU hardware controls the threads and is responsible for the thread scheduling. CUDA, providing the parallel data processing capabilities of GPUs, is a hardware-software architecture. The task of managing the access to the GPU is done by the operating system's multitasking capability by a few CUDA and graphics applications running concurrently.

Each of the multiprocessors processes batches of blocks respectively. A block is processed by only one multiprocessor; therefore, very fast memory accesses is provided by the shared memory space which takes place in the on-chip shared memory. A group of threads, called a thread block, can work together efficiently by sharing data through shared memory and synchronizing their implementation in order to coordinate memory accesses. Certainly, the synchronization points can be specified in the kernel, in which threads in a block are not processed before reaching the synchronization point. Each thread is identified by the thread number within the block.

4. GPU – LU Construction

On designed study primarily how to perform parallel processing is designed for the calculation of LU. Determining how to distribute processors on ensuring performance for the L and U matrices is identified. Algorithm steps of LU_kernel () function is given below.

```
LU_kernel(matrix_L,matrix_U)
{
   initializing_variables;
   calculation_matrix_L;
   calculation_matrix_U;
}
```

LU_main () function is prepared in the following way for solution of linear equation system. Variable definitions and initializing were performed and the necessary codes for the solution of the equation system are written.

```
LU_main()
{
     initializing_variables;
     cuda_event_creation{&start, &stop};
     cuda_device_properties_check{&prop, 0};
     gpu_memory_allocation{&dev_L, &dev_U};
     timer_record{start, 0};
     cpu_to_gpu_memory_copy{{dev_U, U},{dev_L, L}};
     LU_kernel{dev_L, dev_U};
     gpu_to_cpu _memory_copy{{U, dev_U},{L, dev_L}};
     calculation_result_matrix_X;
     timer_record{stop, 0};
     gpu_memory_deallocation{dev_L, dev_U};
}
```

The program codes are written for the calculation of working time. Memory allocation for CPU and GPU is realized. Data are copied from CPU to GPU, LU_kernel() function is called and calculated values are copied this time GPU to CPU. After calculating L and U matrices, calculation of result vector X is performed. This operation performed in the main function because of parallelism is not possible to calculate result vector x. Finally, calculated matrices and vectors are shown on the screen at the end of the program.

Result and Conclusions

Performed algorithms were evaluated on the CPU and GPU for different linear systems. As a result, when both systems are compared, better performance with GPU computing is achieved. Our application has a significant performance increase on the solution of linear equation systems. In addition, new improvements on developed algorithm will provide better performance. Studies are firstly realized on core2duo computer with 16 cuda cores and performance results are given in Table 1. GPU provided performance increase on computation of result matrix at the rate of 431% compared to the CPU on a linear equation system consisting of 6 equations. When the system consisted of 12, 32, 64 equations, the rate was calculated as 346%, 219%, 177% respectively.

Table 1. GPU and CPU Performance of LU Decomposition Method on 16 Cuda Cores, Core2duo CPU

# of Equation	LU_CPU Min. (msec)	LU_CPU Max.(msec)	LU_CPU Av. (msec)	LU_GPU Min. (msec)	LU_GPU Max. (msec)	LU_GPU Av. (msec)	Performance Av. (%)
6	19	26	22	4,5	5,9	5,1	431
12	50	147	85,8	22,3	28,3	24,8	346
32	343	476	411	179	212	188	219
64	981	1111	1047	570	627	590	177

Studies are then realized on quadcore computer with 32 cuda cores and performance results given in Table 2. GPU provided performance increase on computation of result matrix at the rate of 275% compared to the CPU on a linear equation system consisting of 6 equations. When the system consisted of 12, 32, 64 equations, the rate was calculated as 183%, 210%, 206% respectively.

Table 2. GPU and CPU Performance of LU Decomposition Method on 32 Cuda Cores, Core2Quad CPU

# of Equation	LU_CPU Min. (msec)	LU_CPU Max.(msec)	LU_CPU Av. (msec)	LU_GPU Min. (msec)	LU_GPU Max. (msec)	LU_GPU Av. (msec)	Performance Av. (%)
6	7	7	7	1,4	3,7	2,55	275
12	16	16	16	7	10,7	8,72	183
32	94	141	126	51	70	60	210
64	390	531	469	188	259	228	206

As a result we implement GPU accelerated implementation of LU linear algebra routine in this study. We have worked to increase performance with proper data representation and reducing row operations on GPU. We have tried our study on different systems that have different GPUs and CPUs. The computation studies were also evaluated for different linear systems. When we compared the results obtained from both systems, a better performance was obtained with GPU computing. Our implementation provides significant performance improvement so we can easily use it to solve dense linear system.

References

1. Jason Sanders, Edward Kandrot, CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley, 2011.
2. M. Garland, S.L. Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, Parallel computing experiences with CUDA, IEEE Micro, 28 (4), 13-27, 2008.
3. V. Volkov and J. Demmel, LU, QR and Cholesky factorizations using vector capabilities of GPUs, Tech. Report UCB/EECS-2008-49, EECS Department, University of California, Berkeley, 2008.
4. N. Galoppo, N. Govindaraju, M. Henson, D. Manocha, LU-GPU: efficient algorithms for solving dense linear systems on graphics hardware, in: SC ’05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (Washington, DC, USA), IEEE Computer Society, p. 3, 2005.
5. Ino, F., Matsui, M., Hagihara, K., Performance study of LU decomposition on the programmable GPU, In: Proc. 12th Int’l Conf. High Performance Computing (HiPC’05), pp. 83–94, 18–21 December 2005
6. J. R. Humphrey, D. K. Price, K. E. Spagnoli, A. L. Paolini, and E. J. Kelmelis, CULA: hybrid GPU accelerated linear algebra routines, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 7705, April 2010.
7. S. Barrachina, M. Castillo, F. Igual, and R. Mayo adn E. S. Quintana-Ort´ı, Solving dense linear systems on graphics processors, Proceedings of the 14th International Euro-Par conference on Parallel Processing, August 26-29, 2008, Las Palmas de Gran Canaria, Spain
8. LU decomposition, Retrieved April 5, 2011 from the World Wide Web: http://en.wikipedia.org/wiki/LU_decomposition.
9. NVIDIA CUDA C Programming Guide, Version 4.0 available as http://developer.nvidia.com/nvidia-gpu-computing-documentation