Mathematical Problems of Computer Science 42, 121—126, 2014.

Benchmarking of GPU NVIDIA CUDA, CUBLAS and MAGMA Libraries Based on Matrix Multiplication Problem

Edita E. Gichunts

Institute for Informatics and Automation Problems of NAS RA
e-mail: editagich@ipia.sci.am

Abstract

Solving linear systems of equations is a fundamental problem in scientific computing. Many scientific computer applications need a high-performance matrix algebra. The major hardware developments always influenced the new developments in linear algebra libraries. Nowadays major chip manufacturers are developing next-generation microprocessor designs that integrate multicore CPU and GPU components [1]. The main aim is to benchmark CUBLAS and MAGMA libraries on matrix multiplication problem using the Tesla C1060 graphical processing unit.

Keywords: Parallel computing, Matrix algebra, Graphical processor.

1. Introduction

Today multi-core processor computers have become wide-spread. The software-hardware architecture CUDA which emerged with the advent of NVIDIA G80 chip allows performing of calculations using NVIDIA graphics processors. Nvidia Tesla is Nvidia's brand name for their products targeting stream processing and/or general purpose GPU. Products utilize GPUs from the G80 series onward. With 240 processor cores and a standard C compiler that simplifies application development, Tesla scales to solve the world's most important computing challenges-more quickly and accurately.

The technology CUDA which presents the NVIDIA software–hardware computational architecture is based on the extension of the C programming language. It provides an opportunity of using its memory for parallel computations. CUDA assists functioning of algorithms on GPU graphics processors. CUDA provides an effective transformation of data from the system to video-memory [2].

The project MAGMA is directed to the development of a library for the linear algebra tasks. It is similar to the LAPACK library, but it is targeted to the hybrid architecture, starting from multi-core processors and GPU systems. The investigation of MAGMA is based on the idea that the

solution of complex tasks in the hybrid environment has a potential to derive optimal programming solutions. Proceeding from this idea we apply the library to implementing the linear algebra algorithms on multi-core and graphics processor systems [3].
These days there are numerous works devoted to the investigation of parallel systems performance either on the clustering system or multi-core processor base.
The problem elaborated in this article is significantly important for producing an effective performance of optimal parallelization for the linear algebra tasks for the case of contemporary GPU NVIDIA capacities, envisioned for simple methods of programming as well as CUBLAS(CUDA Basic Linear Algebra Subroutines) and MAGMA software packages and multiplication of matrices with one point numbers .

2. The Problem Statement and Main Results

The efficiency of the matrix multiplication is elaborated. The calculations were implemented for the three cases. Only the graphics processor GPU (Tesla C1060, 1296.0 MHz clock, 4095.8 MB memory, capability 1.3) with involving the CUDA technology was used in the first case. The CUBLASlibrary of CUDA was used in the second case and the MAGMAlibrary - in the third one. The computational system is controlled with the Ubuntu 12.04.1 LTS operational system. The NVIDIA CUDA 4 with its libraries is loaded, the ATLAS library is installed, gcc-4.4, nvcc, gfortran-4.4 compilers are used. The MAGMA 1.3 library is installed. LAPACK library is required to install before compilation of MAGMA.

2.1 The First Case

Every program written on the NVIDIA CUDA irrespective its significance can be divided into a number of general steps. The GPU does not possess an access to the operative memory, hence, the programmer has to care about all the sources necessary for the kernel work which should be placed in the memory of the video-card. For this task three main functions, namely cudaMalloc, cudaMemcpy and cudaFree from CUDA SDK , are applied. It is worth to mention also that the function cudaMemcpy has an additional parameter, which denotes the direction in which the information loading (from CPU to GPU or the contrary) takes place. The process of configuration is easy and consists in indicating the size of the net and blocks. The main problem on the next step is the selection of the optimal balance between the number of blocks and their size. The NVIDIA suggests using blocks with 128 or 256 flows. The kernel is called like a conventional C function in the programming language. The only real difference is the requirement to indicate the predetermined sizes for the net and blocks during the call of the kernel. After completing the kernel implementation with applying the cudaMemcpy function it is necessary to copy back the results to operative memory with indicating the reverse direction of copying. In the way similar to the case of the memory flow interruption for any other C program it is necessary to release all the provided resources.
The functions provided by the kernel are written in a file with extension cu, which is compiled during the application of the program NVCC. In the result of compilation the program code is divided into two parts: the first one - supposed for performing on CPU, the second - containing codes PTX objects for the GPU. To define the data patters necessary for treatment the variables blockDim ¨ blockIdx which contain the size of the block and an index for the corresponding kernel in action are used.

The general way of a task solution on CUDA consists in the following:

1. Obtaining data for calculations.
2. Duplicate the data in GPU memory.
3. Perform calculations by means of GPU kernel function.
4. Duplicate the calculated data in the memory of CPU from the GPU.
5. Observe the results.
6. Release the resources.

2.2 The Second and Third Cases

The CUBLAS library is an implementation of the BLAS (A Basic Linear Algebra Subprograms) on top of the NVIDIA CUDA. It provides an input opportunity to computational resources of the graphics processor NVIDIA. The library is of the corresponding API level i.e., there is no direct interaction with the driver CUDA. CUBLAS is connected to one GPU and it is not connected in parallel with several GPU.
The package MAGMA R is a new collection of matrix class objects that implements procedures with matrix and linear operations. Operations are performed using the library MAGMA with the C sub-programs, which are optimized for parallel processing on the graphic card and multi-kernel processors. (http://icl.cs.utk.edu/magma/). It presents a GPU for the matrix algebra and multi-kernel architecture.

The general form of the task solution through MAGMA and CUBLAS libraries consists in the following:

- Formation of matrices and vectors in the memory of GPU,
- Allocating the data in the memory domain,
- Sequential call of the CUBLAS functions,
- Loading the results from the domain of GPU memory to the CPU.

CUBLAS and MAGMA contain auxiliary functions designed for the creation and deletion of objects, as well as for data registration and information extraction in the memory of GPU.
The plot on Fig. 1 shows the dynamics of change of time of implementation in dependence of the volume of input data N.

Examining the results we came to the conclusion that depending on data volume either CUBLAS or Magma library could have higher productivity. It is worth to mention that MAGMA is more effective in case of large data, such as, e.g., 1 gigabyte, but CUBLAS more frequently wins while working with data of 100 megabyte order. However, the productivity of GPUTesla C1060 which uses global memory is many times less in comparison with MAGMA and CUBLAS.

Two time scales were considered in the process of the performance estimating, namely the absolute time of computation and the data input/output time. For both MAGMA and CUBLAS libraries no increase in speed of the input and output was observed, resulting in that the increase in speed was observed only during operation performing, and since the computational time depends on the cube of the matrix sizes while the volume of input/output is a function of the matrix size squares.

Transferred matrices should be placed in GPU memory completely, which leads to the restriction of the total size of the two matrices with 4 GB.

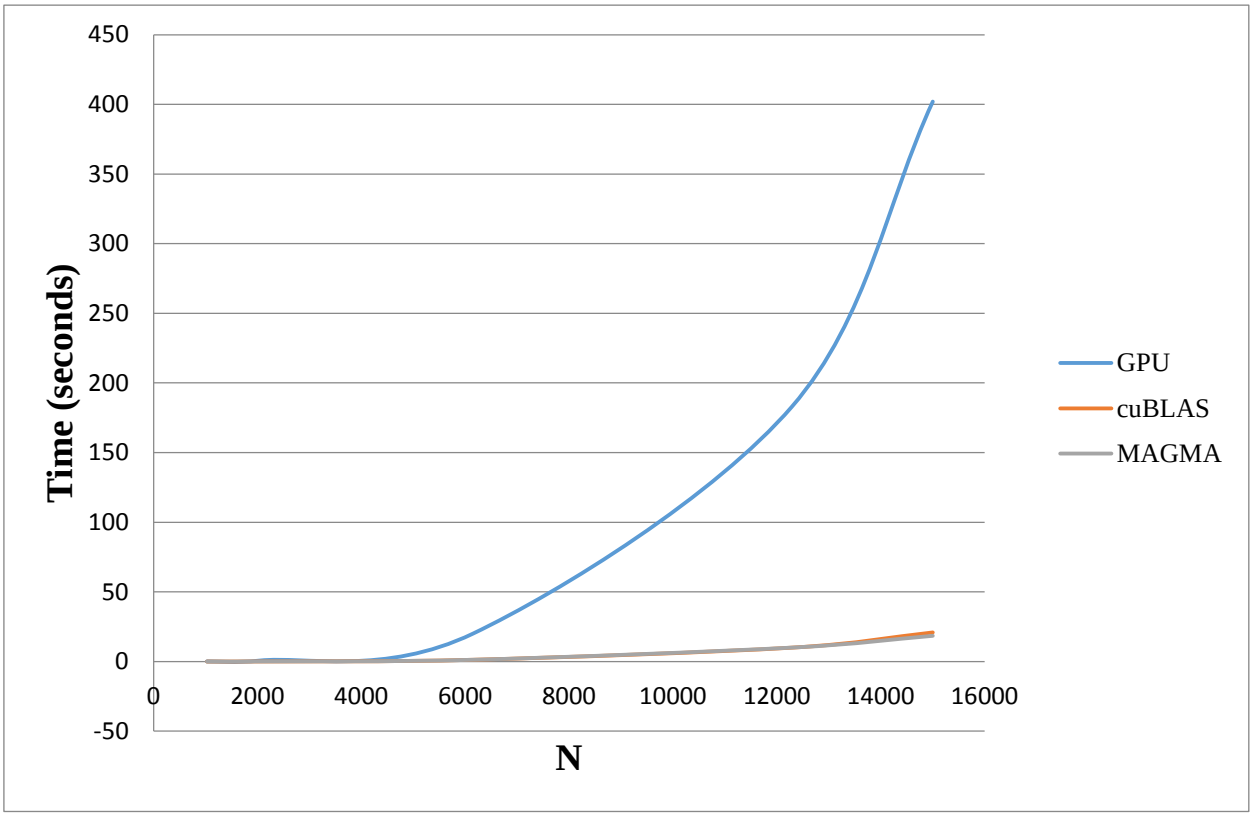

Fig. 1. The dynamics of change of time of implementation in dependence of the volume of input data N.

16000*16000*8 bytes make 2 GB, since two matrices are entered the bound for the data size is 4 GB. Thus, GPU Tesla C1060 works with two matrices of maximum size 16000*16000.

3. Related work

At present there are numerous studies on the productivity of parallel systems, both on the basis of cluster systems, and multi-core CPU and GPU NVIDIA. But there hasn't been carried out any detailed study of the capabilities of modern GPU NVIDIA applying a standard way of writing parallel programs, using the global memory GPU and also program libraries cuBLAS and MAGMA for solving matrix multiplication with single-precision numbers.

4. Conclusion

Several libraries have been installed during this study. Three cases are investigated. The productivity of graphics processor for the task of non-symmetric matrices is investigated.

To attain the maximal efficiency with the GPU NVIDIA CUDA it is necessary to use the libraries MAGMA and CUBLAS which in cases under investigation provide acceleration to 16-18 times more than in case of global memory using.

References

[1] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan and P. Hanrahan, "Larrabee: a many-core 86 architecture for visual computing", *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–15, 2008.

[2] J. Nickolls, I. Buck, M. Garland and K. Skadron,"Scalable parallel programming with CUDA", *Presentation by Christian Hansen Article Published in ACM Queue*, March, page 2, 2008.

[3] S.Tomov, R.Nath, H. Ltaief and J. Dongarra,"Dense linear algebra solvers for multicore with GPU accelerators", *Proc. of IPDPS'10*, Atlanta, GA, January 15, pp. 1—2. 2010.

Submitted 08.09.2011, accepted 26.11.2014.

GPU NVIDIA CUDA–ի, CUBLAS և MAGMA գրադարանների արտադրողականության գնահատականը մատրիցի բազմապատկման խնդրի լուծման ժամանակ

Է. Գիչունց

Ամփոփում

Գծային հավասարումների համակարգերի լուծումը հիմնարար խնդիր է գիտական հաշվարկներում: Բարձր արտադրողականության մատրիցային հանրահաշիվը պահանջվում է շատ գիտական կոմպյուտերային կիրառություններում: Հիմնական տեխնիկական զարգացումները միշտ ազդել են գծային հանրահաշվի գրադարանների նոր զարգացումների ստեղծման վրա: Ներկայումս խոշոր միկրո-չիպային արտադրողներ մշակում են նոր սերնդի չիպեր, որոնք ընդգրկում են բազմամիջուկային CPU և GPU բաղադրիչները [1]: Աշխատանքի գլխավոր նպատակն է CUBLAS և MAGMA գրադարանների թեստավորումը մատրիցի բազմապատկման խնդրի համար՝ Tesla C1060 գրաֆիկական պրոցեսորի օգտագործմամբ:

Оценка производительности GPUNVIDIACUDA, библиотек CUBLAS и MAGMA при решении задач матричного умножения

Э. Гичунц

Аннотация

Решение систем линейных уравнений является фундаментальной проблемой в научных вычислениях. Высоко-производительная матричная алгебра требуется во многих научных компьютерных приложениях. Основные аппаратные разработки всегда влияли на создание новых разработок библиотек линейной алгебры. В настоящее время главные производители микросхем разрабатывают схемы нового поколения, которые интегрируют компоненты с много-ядерными CPU и GPU [1]. Главная цель работы-это тестирование библиотек CUBLAS и MAGMA для задачи умножения матриц с использованием графического процессора TeslaC1060.