Mathematical Problems of Computer Science 46, 59--65, 2016.

Performances of Methods for Solving a Linear System of Equations in the Architecture of GPU Accelerator in Case of Small Matrices

Hrachya V. Astsatryan and Edita E. Gichunts

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

Abstract

The algebraic operations with a large number of small matrices are a very important issue in science. The solutions for linear system of equations with LU factorization are specific of the mentioned operations that have numerous applications of algebraic operations with small matrices. In this work we consider the performances of methods for solving linear system of equations with batched LU factorization for small complex matrices on the graphic processor NVIDIA K40c.The versions with Partial Pivoting, without Pivoting and Random Butterfly Transformations of batched LU factorization for small matrices are presented and shown, which of these versions is the effective one, in which case we achieve high performance.

Keywords: GPU accelerator, MAGMA , linear algebra operation, LU factorization, small matrix, performance, batched computation, Random Butterfly Transformation.

1. Introduction

Numerous scientific computations require a number of small independent solutions to the problems. Since each individual problem size can fail to ensure the necessary parallelization for the given basic equipment, specifically in the case of accelerators, then these problems should be solved simultaneously so that to charge the device with the needed amount of work; hence the name of batched calculations came. The batched subprograms are designed to solve many problems independent of each other in a parallelization manner. Such an approach is required in a number of problems of various spheres, for example, in astrophysics [1], metabolic networks [2], quantum chemistry [3], high-order FEM schemes for hydrodynamics [4], direct-iterative preconditioned solvers [5], and some image [6] and signal processing [7].

As soon as a large number of heterogeneous systems with GPU accelerators appear, the linear algebra software is fully performed in small matrix operations. Graphic processors (GPUs)

are high-performance processors that are capable of making hundreds of floating point operations in parallel. Their large register files and the scratchpad memory of high performance are well suited for the model of streaming execution.

In this paper, via the versions of batched LU factorization, we focus on solving of linear equation systems that are processed on the graphic processor. It should be noted that the experiments were carried out for general complex matrices. New functions are considered in the case of small matrices that are new in the MAGMA 1.6.1 package [8] and are named as batched. In the previous article [9] the solving methods of linear equations via the versions of LU factorization are presented for large matrices as well as it is shown that the higher performances are achieved due to the solving method of Random Butterfly Transformation. In this paper we will present the same problem for small matrices where the batched subprograms of the MAGMA package are applied, and consider how high performance will be achieved for small matrices.

2. Implementation of Batched Subprograms

In the Batched problem each matrix represents a separate problem which is solved independently. The batched subprograms in MAGMA are called Lapack in accordance with the subprograms of the library. MAGMA 1.6.1 package contains four standard precision arithmetic of batched subprograms - single real, double real, single complex and double complex. In this paper we present the implementation of batched solutions of the solving methods of linear system of equations for small matrices for single complex and double complex precision. As in the case of large matrices, in small matrices as well the batched solutions of linear system of equations are implemented with Pivoting, without Pivoting and with Random Butterfly Transformations.

The batched solution of linear system of equations, where the batched LU factorization is performed with Partial Pivoting, is implemented through the MAGMA 1.6.1 library of the xgesv_batched.cpp function.

The solution consists of the following steps:

- A and B matrices are transferred from the CPU to the global memory of GPU.
- Batched LU factorization of the matrix A is performed through xgetrf_batched.cpp function of MAGMA library using a Partial Pivoting with row interchanges.
- A * X = B is solved through the function xgetrs_batched.cpp of MAGMA library.
- X derived solutions are transferred from the GPU to CPU.

xgetrf_batched.cpp consists of xgetf2_batched, xlaswp_rowparallel_batched, xtrsm_outofplace_batchedandxgemm subprograms.

a) xgetf2: DGETF2 computes an LU factorization of a general m-by-n matrix A using Partial Pivoting with row interchanges. The factorization has the form A = P * L * U, where P is a permutation matrix, L is a lower triangular with unit diagonal elements (lower trapezoidal if m > n), and U is an upper triangular (upper trapezoidal if m < n).

In each step of LU factorization, dgetf2 subprogram is used while factorizing the m x nb panel. It consists of three subprograms idamax, dswap and dscal of Level 1 BLAS library and dger subprogram of Level 2. The only approach to the implementation of dgetf2 subprogram lies in the fact that all the panels are loaded in the common memory of GPU and then perform all calculations before the results are sent to the main memory.

b) xlaswp: DLASWP performs a series of row interchanges on a general rectangular matrix. DLASWP performs a series of row interchanges on the matrix A. One row interchange is

initiated for each of rows K1 through K2 of A. To increase the digital stability, the transformation with Pivoting is required. dlaswp subprogram is used in case of the transformation with Pivoting, despite the fact that we get low performance. In any case, experiments show that the transformation with Pivoting is quite expensive for the batched problem.

c)xtrsm: DTRSM solves one of the matrix equations op(A)*X = alpha*B, or X*op(A) = alpha*B, where alpha is a scalar, X and B are m by n matrices, A is a unit, or a non-unit, an upper or lower triangular matrix and op(A) is one of op(A) = A or op(A) = A**T. The matrix X is overwritten on B.

d) xgemm: DGEMM performs one of the matrix-matrix operations C := alpha*op(A)*op(B) + beta*C, where op(X) is one of op(X) = X or op(X) = X**T, alpha and beta are scalars, and A, B and C are matrices, with op(A) an m by k matrix, op(B) a k by n matrix and C an m by n matrix. The aim of the batched LU factorization is to achieve batched dgemm high performance. A lot of effort has been concentrated for the optimization of dgemm subprogram. In particular, for our case, dgemm is not only used in the trailing matrix updates but also in the implementation of the triangular matrix solvers (xtrsm).

The batched solution of linear system of equations where the batched LU factorization is performed without Pivoting, is implemented through the MAGMA 1.6.1 library of xgesv_nopiv_batched.cpp and xgerbt¬_batched.cpp functions.

In case of xgesv_nopiv_batched.cpp function the solution sequence is as follows:

- A and B matrices are transferred from the CPU to the global memory of GPU.
- Batched LU factorization of the matrix A is performed through xgetrf_nopiv_batched.cpp function of MAGMA library without any Pivoting.
- A * X = B is solved through the function xgetrs_nopiv_batched.cpp of MAGMA library.
- X derived solutions are transferred from the GPU to CPU.

To implement xgetrf_nopiv_batched.cpp function, the xgetrf_ panel_nopiv_batched subprogram of MAGMA library, as well as xtrsm_ work_batched and xgemm_batched subprograms of Blas library are called.

To implement xgetrs_nopiv_batched.cpp function, only the xtrsm_outofplace_batched subprogram of Blas library is called.

The solution of linear system of equations, where the Random Butterfly Transformation is applied on A and B matrices, and the batched LU factorization is performed without Pivoting, is implemented via xgerbt_batched.cpp function of MAGMA 1.6.1 library.

This type of solution implementation is realized in the following sequence:

- We generate the random matrices U and V in packed storage on the CPU.
- The matrix A and the packed representation of U and V are sent from the host memory to the device memory.
- Randomization is performed on the GPU, updating A in the device memory.
- Perform Partial Random Butterfly Transformation on the GPU with magmablas_cprbt_batched () function.
- We compute U^Tbon the GPU, A_ry= U^Tbis solved on the GPU, followed by the solution x = Vy with magmablas_cprbt_mtv_batched () function.
- The solution is sent to the host memory.

3. Performance Results

The experiments were conducted on NVIDIA K40c GPU. The architecture of NVIDIA K40c consists of 2880 CUDA processor cores. It is endowed with much higher bandwidth 288 GB/s of message transfer between CPU and GPU, having 12 GB of global memory, GDDR5 memory interface, and CUDA C programming environment. The operation system of K40c is Ubuntu 14.04.2 LTS. MAGMA 1.6.1 package is installed. The code is compiled using the GNU gcc version 4.8, gfortran-4.8, g ++ - 4.8 and the nvcc version 7.0 with the optimization flag -O3 and linked with the Atlas Library.

Figures 1 and 2 show the performance results of batched functions of linear solution equations for small matrices for complex single precision and complex double precision, respectively.

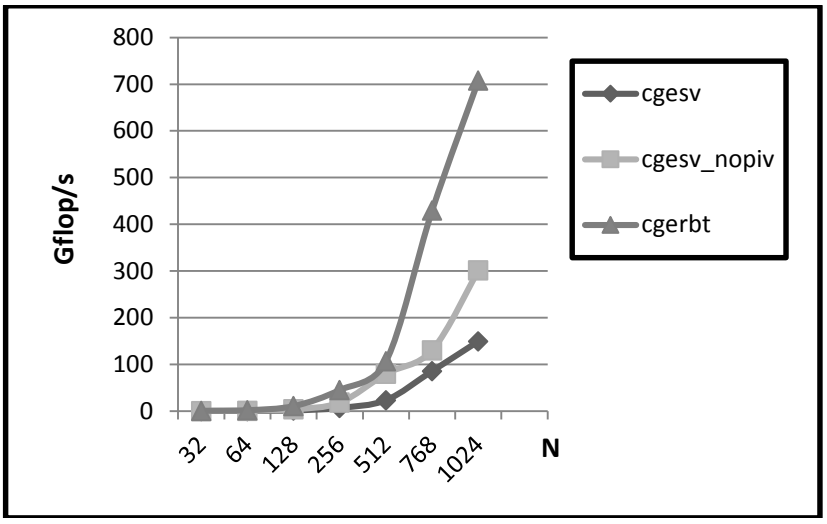

Fig. 1. Complex Single Precision.

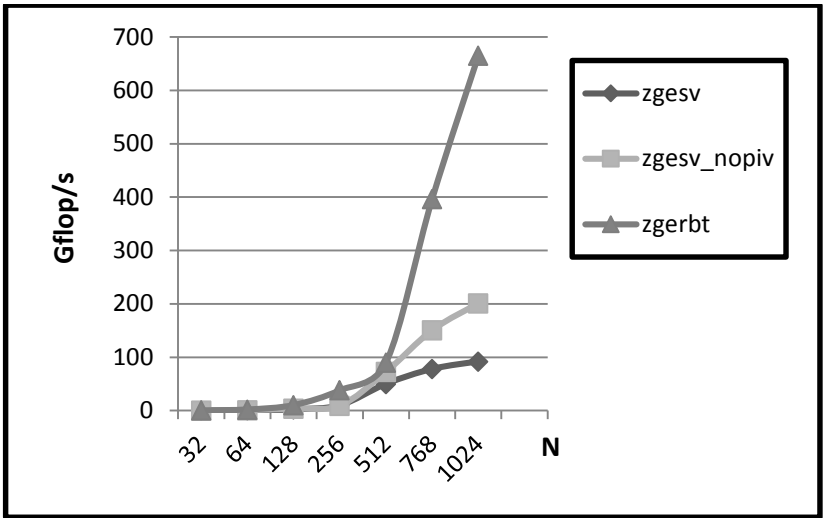

Fig. 2. Complex Double Precision.

In case of single precision the obtained results show that the performance of solutions determined by the LU factorization without Pivoting is several times higher than the solutions with Pivoting. Thus, for example, the performance of cgesv_nopiv_batched function without

Pivoting is twice higher than the performance of cgesv_batched function with Pivoting, and the performance of cgerbt_batched function with Random Butterfly Transformation without Pivoting is 4.5 times higher than the performance of cgesv_batched function and 2 times higher than the performance of cgesv_nopiv_batched function.

In case of double precision the obtained results show that the performance of zgesv_nopiv_batched function without Pivoting is twice higher than the performance of zgesv_batched function with Pivoting, and the performance of zgerbt_batched function with Random Butterfly Transformation without Pivoting is 6.5 times higher than the performance of zgesv_batched function and 3 times higher than the performance of zgesv_nopiv_batched function.

Also note that in case of single precision the performance is relatively higher than in the case of double precision for the corresponding functions with and without Pivoting.

4. Conclusion

We presented the methods for solving of linear system of equations on GPU accelerator for small matrices with and without Pivoting. For small matrices the batched functions of MAGMA 1.6.1. package have been applied which are new and the batched function with Random Butterfly Transformation was not applied in the tests. It is concluded from the experiment results that for small matrices a high performance in solutions of linear system of equations is achieved applying the method of Random Butterfly Transformation.

References

[1] O.E.B. Messer, J. A. Harris, S. Parete-Koon and M. A. Chertkow, “Multicore and accelerator development for a leadership-class stellar astrophysics code”, In *Proceedings of "PARA2012: State-of-the-Art in Scientific and Parallel Computing"*, 2012.

[2] J. C. Liao Khodayari, A. R. Zomorrodi and C. D. Maranas, “A kinetic model of Escherichia coli core metabolism satisfying multiple sets of mutant flux data”, *Metabolic engineering*, vol. 25, pp. 50–62, 2014.

[3] A. A. Auer, G. Baumgartner, D. E. Bernholdt, A. Bibireata,V. Choppella, D. Cociorva, X. Gao, R. Harrison, S. Krishnamoorthy, S. Krishnan, C.-C. Lam, Q. Luc, M. Nooijene,R. itzerf, J. Ramanujamg, P. Sadayappan and A. Sibiryakovc, “Automatic code generation for many-body electronic structure methods: the tensor contraction engine’, *Molecular Physics*, vol. 104, no. 2, pp. 211–228, 2006.

[4] T. Dong, V. Dobrev, T. Kolev, R.Rieben, S.Tomov and J. Dongarra, “A step towards energy efficient computing: Redesigning a hydrodynamicapplication on CPU-GPU”, In *IEEE 28th International Parallel Distributed ProcessingSymposium (IPDPS)*, 2014.

[5] Eun-Jin Im, K. Yelick and R. Vuduc, “Sparsity: Optimization frameworkfor sparse matrix kernels”, *Int. J. High Perform. Comput. Appl.*, vol. 18, no. 1, pp. 135–158, 2004.

[6] J. M. Molero, E. M. Garzón, I. García, E. S. Quintana-Ortí, and A. Plaza, “Poster: A batched Cholesky solver for local RX anomaly detection on GPUs”, PUMPS, 2013.

[7] M. J. Anderson, D. Sheffield and K. Keutzer, “A predictive model for solving smalllinear algebra problems in gpu registers’, In *IEEE 26th International Parallel DistributedProcessing Symposium (IPDPS)*, 2012.

[8] Matrix algebra on GPU and multicore architectures (MAGMA), MAGMA Release 1.6.1, 2015. Online. [Available]: http://icl.cs.utk.edu/magma/

[9] H. V. Astsatryan and E. E. Gichunts, “Performances of methods for solving a linear system of equations in the architecture of GPU accelerator”, *Transactions of IIAP NAS RA, Mathematical Problems of Computer Science*, vol. 45, pp. 44—52, 2016.

Submitted 02.08.2016, accepted 25.10.2016.

Փոքր մատրիցների դեպքում գծային հավասարումների համակարգի լուծման մեթոդների արտադրողականությունները GPU արագագործչի ճարտարապետությունում

Հ. Ասցատրյան և Է. Գիչունց

Ամփոփում

Գիտության մեջ շատ կարևոր խնդիր է հանդիսանում բազմաքանակ փոքր մատրիցների հետ կապված հանրահաշվական գործողությունները: Այդ գործողություններից առանձնահատուկ են LU վերլուծությամբ գծային հավասարումների համակարգի լուծումները, որոնք փոքր մատրիցների հետ հանրահաշվական գործողություններում բազմաթիվ կիրառություններ ունեն :Այս աշխատանքում դիտարկվում են կոմպլեքս փոքր մատրիցների համար batched LU վերլուծությամբ գծային հավասարումների համակարգի լուծման մեթոդների արտադրողականությունները NVIDIA K40c գրաֆիկական պրոցեսորի վրա: Փոքր մատրիցների համար ներկայացվում են batched LU վերլուծության պտույտով, առանց պտույտի և թիթեռնիկի պատահական ձևափոխությամբ տարբերակները և ցույց է տրվում, թե այդ տարբերակներից որն է արդյունավետը, որի դեպքում հասնում ենք բարձր արտադրողականության:

Производительности методов решения систем линейных уравнений в архитектуре GPU ускорителя в случае небольших матриц

Г. Асцатрян и Э. Гичунц

Аннотация

Алгебраические операции с большим числом малых матриц являются очень важными вопросами в науке. Из упомянутых операций специфическими являются решения для линейной системы уравнений с LU факторизацией, которые имеют многочисленные применения алгебраических операций с небольшими матрицами. В этой работе мы рассмотрим производительности методов для решения системы линейных уравнений с batched LU факторизацией для малых комплексных матриц на графическом процессоре NVIDIA K40c. Для небольших матриц представлены версии batched LU факторизации с поворотом, без поворота и со случайным преобразованием бабочки, а также показано, какой из этих вариантов является эффективным, и в каком случае мы достигаем высокой производительности.