Research Journal of Applied Sciences 9 (12): 1120-1123, 2014
ISSN: 1815-932X

Error Correction Method Results in a Multiplication of the Supercomputer

Nikolay Ivanovich Korsunov, Igor Sergeevich Konstantinov and
Aleksandr Aleksandrovich Nachetov
Belgorod State National Research University, Pobedy St. 85, 308015 Belgorod, Russia

Abstract: This study proposes a method for the detection and correction of errors in the implementation of the matrix multiplication supercomputers due to noise. The proposed method based on a comparison of the results of the two operations with using CUDA-technology. Operations are performed in parallel by the action of the same interference.

Key words: Supercomputer, detection and correction errors, matrix operations, CUDA-technology, Russia

INTRODUCTION

One of the major transformations used in supercomputers are matrix operations (DuBois *et al*., 2008; Russek and Wiatr, 2007). When the disturbance matrix operations are performed with errors and error detection and correction performance matrix transformations in supercomputers is an urgent task (Korsunov *et al*., 2013).

Known method to detect errors of arithmetic with arithmetic codes, based on a comparison of the selected module results of operations and control over the information bits (Teekaput and Chokchaitman, 2005). The method can detect errors in arithmetic operations provided unmistakable result of the operation of the check digit. This imposes limits on the duration of the interference and requires separation of run-time operations information and control results. Another disadvantage of this method is the lack of adjustment of the errors identified in the performance of each of the control of the operations of scalar quantities.

Another method of detecting errors (Farazmand and Tahoori, 2010) during conversion information is based on multiple overlapping and decided to exclude results with an error by the majority principle. Although, the method and allows you to adjust by selecting the mistake but as a method of error detection using arithmetic codes require error-free operation in most of the time intervals of multiplication are used to perform matrix multiplication 2k+1, k = 1, 2, 3, ..., L. The choice of k is associated with the duration of the disturbance. These developments significantly affect the performance of a supercomputer in the processing of information from the detection and correction of errors in operation noise (Cotroneo *et al*., 2013).

MATERIALS AND METHODS

This study proposes a method for the detection and correction of errors in the implementation of the matrix multiplication supercomputers due to noise. Using supercomputer involves the use of parallel computing. For parallel computing in the multiplication of the matrix A (size nxm) with elements a_{ij} and vector B (size m) represent multiplication by the following steps.

Step 1: Multiply each element of the jth columna$_{ij}$ to jth element b_j of the vector B. For parallel computation we obtain m-vectors with elements:

$$c_{ij} = a_{ij} \cdot b_j,\ j = \overline{1,m} \quad (1)$$

Step 2: Define the error of each of scalar multiplications (Eq. 1) and then adjust each of these multiplications. For error correction scalar multiplication we propose a method based on duplicate using one of the multiplier. Since, the operation (Eq. 1) is performed in parallel for each column of the matrix A independently, then these operations are performed autonomously and in parallel for all the columns of the matrix A. The result is an autonomous detection and correction of errors, introduced in the calculation of each element c_{ij} in accordance with the parallel execution in each jth column and detection operations of error correction. Autonomy error correction in each of the columns allows to perform parallel operations to correct errors multiplied in all columns:

$$j = \overline{1,n}$$

Step 3: Calculating the result of the conversion of vector C elements in the form:

Correponding Author: Nikolay Ivanovich Korsunov, Belgorod State National Research University, Pobedy St. 85, 308015 Belgorod, Russia

$$C_i = \sum_{j=1}^{n} c_{ij}$$

Performing matrix A multiplication by a scalar b_j and determining input values for each amendment for each value j in a_{ij}, calculated correction vector:

$$\delta = \alpha \cdot \delta c$$

Where:
α = α_{ij} with fixed values i, j
δ = Corrective amendment

When calculating the corrections for correcting parallel for each ith scalar multiplication (Eq. 1), due to the execution of the same transformations all processor elements, the amount of which is determined number of elements of the matrix column A_j, provided full load of processor elements during the time interval τ, required for operations.

In calculating, the relative amendments by one processing element and parallel computation of absolute amendments during time τ_1 uses one processing element. The remaining processing elements remain in standby mode. If other operations not provided, multiprocessor system is used inefficiently (Nikolov *et al.*, 2008).

RESULTS AND DISCUSSION

Main part: This study proposes a method for the detection and correction of errors in the implementation of the matrix multiplication supercomputers, due to noise and lead to the result:

$$C = AB + \Delta C \quad (2)$$

Where:
$A_{m\times n}$, $B_{n\times k}$ = The matrix of input operands
C_{nxk} = Matrix corresponding to the result works with an error matrix defined error ΔC

The proposed method is also based on a comparison of the results of the two operations, one of which is executed with operands A, B and the second with operands A, B_1 in accordance with Eq. 2, operations are performed in parallel by the action of the same interference.

Represent the result of the multiplication of matrices A, B in the form R_1, R_2...R_i...R_m where, Ri column vector obtained by multiplying the matrix A by the ith column of the vector B:

$$R_k = \begin{vmatrix} a_{11}a_{12}...a_{1j}...a_{1n} \\ a_{21}a_{22}...a_{2j}...a_{2n} \\ ..... \\ a_{m1}a_{m2}...a_{mj}...a_{mn} \end{vmatrix} \cdot \begin{vmatrix} b_{1k} \\ b_{2k} \\ ... \\ b_{nk} \end{vmatrix} = \sum_{j=1}^{n} A_j b_{jk} = \sum_{j=1}^{n} D_j \quad (3)$$

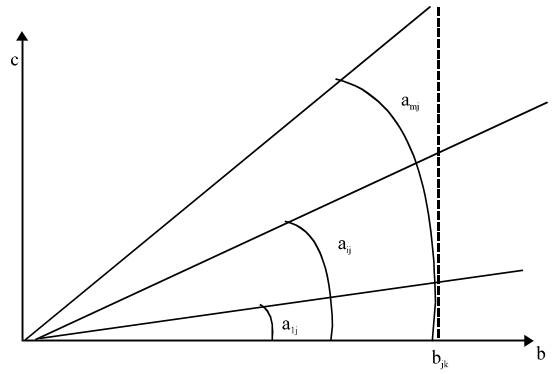

Fig. 1: Graphical representation of the proposed method

Where:

$$A_j = \begin{vmatrix} a_{1j} \\ a_{2j} \\ ... \\ a_{mk} \end{vmatrix}$$

scalar equal jth element of the kth column of the matrix B, $D_j = b_{jk}A_j$: matrix column. In accordance with (Eq. 2) the effect of noise can cause errors in D_j the vector multiplication by a scalar or vector addition to errors. Since, the number of multiplications of vectors by scalars significantly greater than the number of operations of vector addition, the greatest interest is the detection and correction of error multiplication of vectors by scalars.

In the proposed method are performed in parallel multiplication of vectors A_j by scalars b_{jk} which graphically represents the family of straight lines emanating from the same point at an angle determined by the corresponding element a_{ij} of the matrix A_j (Fig. 1). The result of multiplication a_{ij} by b_{jk} (Eq. 2) is represented as:

$$c_i = a_{ij}b_{jk}, i = \overline{1,m}$$

and for the values of i = r and i = s, respectively:

$$c_r = a_{rj}b_{jk}, c_s = a_{sj}b_{jk} \quad (4)$$

From Eq. 3 that:

$$c_s = \frac{a_{sj}}{a_{rj}} c_r \quad (5)$$

If c_r and c_s are calculated in parallel and at the same time of interference which leads to errors in the calculations c_r and c_s in accordance with Δc_r and Δc_s, linked by the relation:

$$\Delta c_s = \frac{a_{sj}}{a_{rj}} \Delta c_r \quad (6)$$

Taking $\Delta c_r/a_{rj}$ as a relative error δ_{cj}, the error of the inner product matrix elements D_j can be determined only by the relative error δ_{cj} of the scalar product in the jth column:

$$\delta_{ij} = \delta_{cj} a_{ij} \quad (7)$$

We define an error δ_{cj} multiplication of two scalar quantities:

$$c = ab$$

When is a graphical interpretation of the slope with respect to the x-axis b. Error calculation of the product leads to the deviation angle and the result:

$$c_1 = a_1 b_1 = c_1^* + \Delta c_1 \quad (8)$$

Calculate the product with the error:

$$c_2 = a_2 b_1 = c_2^* + \Delta c_2 \quad (9)$$

Since, Δc_2 and Δc_1 as well as c_2 and c_1 related by the same Eq. 4 and 5, then we require that the difference between Eq. 8 and 9 equal to a certain value $b = b_1+\Delta b$ of the expression:

$$c_2 - c_1 = b_1 + \Delta b \quad (10)$$

When taking into account (Eq. 4 and 5) it is easy to determine the relationship between a_2 and a_1 as:

$$a_2 = a_1 + 1 \quad (11)$$

And as Δb the error is reduced to the product of the factor b, then the error is determined by multiplying:

$$\Delta c = \Delta b \cdot a \quad (12)$$

Thus, the error Δb is the relative error of multiplication by a scalar b_{jk} matrix elements A_j. Hence, the multiplication algorithm jth column of the matrix by a scalar that represents j_k the item in the kth column of the matrix B is represented as a sequence of steps:

- Perform the multiplication of elements a_{ij} in the element b_{jk} for all i = 2, 3, ..., m
- Calculate $\Delta b = (c_2\text{-}c_1)\text{-}b_{jk} = \delta c_{ij}$
- Calculate multiplication $\delta c_{jk}\, a_{ij} = \Delta c_{ij}$
- Adjust the results of multiplying a_{ij}, i = 1, 2, ..., m on the value of Δc_{ij}

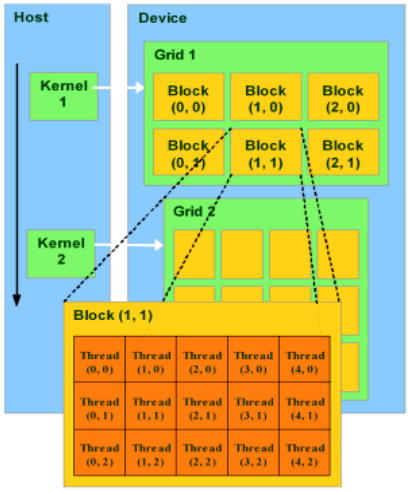

Fig. 2: The programming model for graphics processors

This sequence of steps to execute in parallel for all the columns of A and all columns of the matrix B.

The programming model for graphics processors (Fig. 2) the grouping of flows (NVIDIA, 2014). Streams are combined into blocks of threads-one-dimensional or two-dimensional grid flows interacting with each other via shared memory and synchronization points. Program (kernel) is executed on the grid thread blocks. At the same time, executed one mesh. Each unit can be one-, two-or three-dimensional shape and may be composed of 512 thread on the current hardware.

Blocks of threads are running in small groups called warp, size -32 threads. This is the minimum amount of data that can be processed in multiprocessors. And since it is not always convenient, CUDA allows you to work with blocks containing from 64-512 threads (Ding, 2014).

Grouping of blocks in the grid allows escape from the restrictions and apply the core of a larger number of streams in a single call. It helps when zooming. If insufficient GPU resources, it will perform consistently blocks. In the opposite case, the blocks can be executed in parallel which is important for optimal allocation of GPUs on different levels, ranging from mobile and integrated.

CUDA Memory Model in different byte-addressable opportunity, supporting both gather and scatter. Fairly large number of available registers for each stream processor, up to 1024 pieces. Access to them is very fast, they can be stored in 32 bit integers or floating point numbers (Dudnik *et al*., 2009).

Computational experiments were performed on high-performance computing system based on NVIDIA Tesla M2090, Belgorod State University using technology CUDA. The computing node which is the video card,

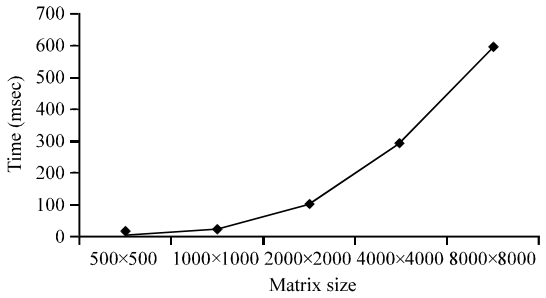

Fig. 3: Results of numerical experiments on the video card NVIDIA Tesla M2090

4-core Intel 5570 2.4 GHz RAM-24 GB. Computational experiments demonstrated in Fig. 3. Here, on the horizontal axis shows the dimensions of the matrices and the ordinate time in milliseconds.

The resulting graph can be described by the equation:

$$y = 0.08x - 47.3$$

Known methods for detecting and correcting errors represent a choice between speed and accuracy. Also, many of these different limitations are imposed. The proposed method is based on using CUDA-technology, showing good results. The method can not only detect the error but fix it in a short time.

CONCLUSION

Thus, the proposed method can quickly multiply matrices using a super computer and correct the result after noise impact.

ACKNOWLEDGEMENT

Research on this subject conducted as part of the a state contract No. 14.581.21.0003 Russian Ministry of Education.

REFERENCES

Cotroneo, D., F. Frattini, R. Natella and R. Pietrantuono, 2013. Performance degradation analysis of a supercomputer. Proceedings of the IEEE International Symposium on Software Reliability Engineering Workshops, November 4-7, 2013, Pasadena, CA., pp: 263-268.

Ding, C., 2014. CUDA tutorial. http://geco.mines.edu/tesla/cuda_tutorial_mio.

DuBois, D., A. DuBois, C. Connor and S. Poole, 2008. Sparse matrix-vector multiplication on a reconfigurable supercomputer. Proceedings of the 16th International Symposium on Field-Programmable Custom Computing Machines, April 14-15, 2008, Palo Alto, CA., pp: 239-247.

Dudnik, V., V.I. Kudryavtsev, T.M. Sereda, S.A. Us and M.V. Shestakov, 2009. Application of the opportunities of tool system Cuda for graphic processors programming in scientific and technical calculation tasks. Comput. Modell. Syst., 52: 159-165.

Farazmand, N. and M.B. Tahoori, 2010. Multiple fault diagnosis in crossbar nano-architectures. Proceedings of the IEEE European Test Symposium, May 24-28, 2010, Praha, pp: 94-99.

Korsunov, N., V. Mikhelev and A. Lomakin, 2013. Application of lattice neural networks for modeling of stationary physical fields. Proceedings of the 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems, September 12-14, 2013, Berlin, pp: 369-372.

NVIDIA., 2014. Parallel thread execution ISA version 4.1. CUDA Toolkit Documentation. http://docs.nvidia.com/cuda/parallel-thread-execution/.

Nikolov, H., T. Stefanov and E. Deprettere, 2008. Systematic and automated multiprocessor system design, programming and implementation. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 27: 542-555.

Russek, P. and K. Wiatr, 2007. Dedicated architecture for double precision matrix multiplication in supercomputing environment. Proceedings of the Design and Diagnostics of Electronic Circuits and Systems, April 11-13, 2007, Krakow, pp: 1-4.

Teekaput, P. and S. Chokchaitam, 2005. Secure embedded error detection arithmetic coding. Proceedings of the 3rd International Conference on Information Technology and Applications, July 4-7, 2005, Sydney, Australia, pp: 568-571.