Modeling system for GPU parallel tasks performance simulation

A.Y.Doroshenko [2,1], K.A. Zhereb[1], O.G.Beketov [1], M.V. Gnynjuk[2]

[1] *Institute of Software Systems NAS Ukraine, Glushkova Avenue 40, Kyiv, Ukraine*

[2]*National Technical University of Ukraine «Kyiv Polytechnic Institute»*

dor@isofts.kiev.ua, zhereb@gmail.com, beketov.oleksii@gmail.com, mgnyniuk@gmail.com

Abstract. *A flexible and extensible simulation tool architecture, called gpusim, is proposed for heterogeneous grid systems with graphics accelerators. The tool is based on open source Java framework GridSim. Checking for models adequacy and their initial investigation has been performed using known examples of parallel computation problems. The tool allows choosing the most optimal setting parameters in automatic mode and thus significantly reduces the time to obtain optimal implementation compared to the direct execution of the program.*

Key words

GPU, CUDA, simulation modeling

1 Introduction

Currently, the Grid systems are increasingly used in research and industrial applications. They allow to use existing heterogeneous parallel resources effectively and to solve large size problems that can't be resolved by separate concurrent facilities. However, grid systems are too complex for their creation and configuration, as well as for their programming. Selection of the optimal configurations for the structure of a Grid environment and for tasks that are performed on it is not reasonable to implement on real systems due to high costs and need to coordinate different Grid projects participants. So, current interest is the modeling problem that allows excluding the cost of creating, maintaining and using real Grid system to conduct experiments on the model suggesting that the results do not significantly different from the original ones.

The interest in graphics accelerators (GPUs) is ever growing due to their superior performance comparing to conventional processors (CPU), availability, and low energy consumption. However development of appropriate developer tools still is a problem. In particular we consider the problem of modeling parallel systems (clusters, grid systems) with heterogeneous components that contain both CPU and GPU.

In this paper flexible and scalable simulation environment architecture of heterogeneous grid systems, called *gpusim*, based on Java framework *GridSim* [8] is proposed. We describe a prototype tool for system modeling based on proposed architecture. The constructed models allow to select optimal values for parameters of parallel algorithm implementation. Two examples – matrix multiplication and N-body problem – are used to demonstrate sufficient precision of built models for large input data and ability to reduce optimal program development time significantly. In particular, the optimal value of the "number of threads per block" parameter of N-body problem has been found by simulation in 3 minutes while its confirmation by experiment needed 8 hours of computing [18].

Today research in grid systems simulation is conducting actively and a lot of tools are developed. Both analytical and simulation models of grid systems are developing and various operation aspects are being considered.

Analytical models allow obtaining general theoretical results for a wide range of possible grid environment settings although they ignore many real operation details. In particular, [1] describes the performance model of grid system that consists of separate components the performance of which is known. The analytical model allows to calculate the steady state of the system for a given input stream of tasks and separate units parameters. A similar approach is proposed in [2]. Interaction network between computing nodes is described taking into consideration the heterogeneity of nodes and connections. The model considers two levels of interaction: between grid nodes which are clusters and

within a single cluster. In [3] the computing network that contains both general purpose processors (CPU) and specialized reconfigurable processors (FPGA) is modeled. An analytical model describes the structure of individual components: pure CPU and FPGA nodes, as well as hybrid units that contain both CPU and FPGA wherein computing facilities and memory hierarchy in such a node are modeled.

A Grid modeling approach based on Petri nets has been proposed in [4]. This approach and its tools allows to examine the internal processes and interactions of Grid key nodes in detail but is difficult to tune and set input task and is not suitable for grid with complex or bulk internal structure.

Simulation models are used for a more precise measurement of the Grid systems performance with specific parameter values. They allow getting whichever simulation accuracy by taking into account all the necessary details. But in practice there is needed to achieve a balance between modeling accuracy and computational complexity.

Among the existing developments an attention should be paid for simulation model that was developed for the study of the effectiveness of resource scheduling techniques for different intensities of workflows in the Grid [5] and its software implementation GRID_Scheduler_Model. Homogeneous grid system is simulated and it needs refinement for heterogeneous grid model support but scalability of the implementation is not provided.

Currently there exist several tools (frameworks) that allow developers to create their own Grid models and carry out experiments. The review and comparative analysis of the most functional and stable frameworks MicroGrid, OptorSim, SimGrid, GridSim, Bricks is given in [6] and [7]. Based on the comparison we can conclude that the most convenient and functional is GridSim [8], a framework for Java programming language which uses SimJava library as a simulation kernel [9].

Problems of GPU program execution time modeling are also discussed in literature. In particular, a detailed analytical model [10], an empirical model of execution time and energy consumption [11] and performance of CPU and GPU comparison model [12] are proposed. Works [11, 12] are using the Ocelot emulator [13] to take into account the details of programs execution on the GPU. Such studies can achieve high simulation accuracy but require significant resources to execute the model. They also do not provide incorporation to the grid model environment.

This paper continues the research started in [14] and [18] in the direction of building a sufficiently simple model of program execution on the GPU which would allow to get the accuracy required to make important decisions on the load distribution in the grid environment.

2 Gpusim system architecture

Gpusim system is created by the extension of GridSim [8] to support graphics accelerators. To achieve scalability, flexibility and simplicity at the same time, the system is divided into several components (Figure 1).:

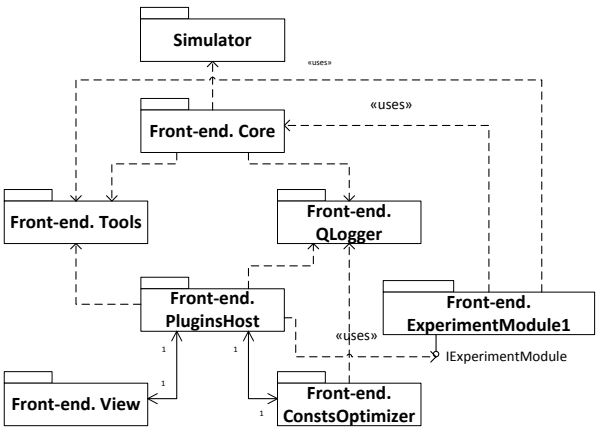

Fig. 1. Component diagram of gpusim

Gpusim simulator is a shell written in Java for the GridSim framework. Its main objective is to set up GridSim basing on the configuration file, to run the simulation, to collection and output statistics to the file.

Front-end component is written in C++ programming language using the Qt framework version 4.8.3 [15]. It includes QLogger login module, an instrumental kernel, execution kernel, constants optimization module, server plug-ins and a set of experimental modules.

Experimental module is a plug-in for the front-end which encapsulates within itself all the tools to work with a specific task: a graphical user interface for printing input/output data, configuration generator for simulator and statistics handler. The experimental module has a number of preset settings that are not specific to individual experiments but have a significant effect on the simulations results (these parameters are described in section 3.2). For example, currently developed a pilot module MatrixMultiply allows to study the problem of matrix multiplication using blocked algorithm for a parallel system model consisting of CPU and GPU where size of matrix block is essential for parallel program performance.

Plug-ins server is a virtual server that solves the problem of plug-ins loading and executing in the front-end component.

Instrumental core is a set of supporting functionality for working with data and other modules simplification.

QLogger login module encapsulates the login function and provides user friendly interface module for writing messages to the log file and to a console. It uses as a basis Qt-module QDebug mechanism of setup messages.

Executive kernel is a module whose primary goal is to provide other functional modules to work with the simulator, its configuring and statistics processing.

The module of constants optimization is designed to improve the accuracy of the simulation by automation of conducting series of simulations with different parameters of one of the experimental modules. This module generates a set of possible combinations of parameter values basing on a configuration file that contains the names of the parameters, their initial and final values, the amount and method of incrementing of each parameter, then gradually runs simulation with each of the sets of parameters and compares simulation results with the experiment results conducted on a real parallel system. If a result of the last measured performance simulation less differs from results of an experiment on the real system, then a set of predefined parameters corresponding to the last simulation is considered the best.

Gpusim experiment is a set of simulations each of which includes a configuration for the simulation and statistics generated as a result of his work.

Initially the plug-ins server loads required experimental module, then asks him for a widget that allows the end user to specify the input data for the experiment. Then plug-ins server asks the experiment module for the experiment that is based on the data entered by the end user. Experimental module delegates the part of the work to the executive core. Then the created experiment passes through the executing module to simulator for executing. Executing kernel asynchronously sends data on the progress of the experiment which are then displayed in a graphical interface.

At the end the simulator generates the output file, which is read by the executive core and transmitted through plug-ins server for processing to the experimental module. After processing the output data in an experimental module it asks for the widget that displays output and is being sent to the graphical interface.

3 Matrix multiplication problem

To check the accuracy and relevance of the developed simulation environment of heterogeneous Grid systems the matrix multiplication problem using blocked algorithm [14] has been selected. In [14] also an empirical model of the execution time of the blocked algorithm on GPU was constructed. This model has the following foundation:

$T_{gpu} = N^2 T_{st.global} + N^3 T_{ld.global}$, where $T_{ld.global}$ and $T_{st.global}$ are settings that determine time of downloading and saving data when working with a GPU global memory. These parameters are the part of the model description and need to be chosen for specific compute nodes experimentally.

Based on an empirical model of the execution time matrix multiplication problem on GPU [14] simulation experiments generator and statistics handler were developed, which are the part of the MatrixMultiply experimental module. The input parameters of the generator are the block size, minimum and maximum matrix size and the increment matrix size. Statistics handler receives output simulator data, converts them to display visually to the end user and compares the time of each of the simulations with experimental results on real parallel system.

The generator has a number of preset parameters that are not related directly to the experiment but impact the runtime of simulation, namely:

- the number of CPU and GPU processing elements, as well as their rankings in terms of MIPS (million instructions per second): cpuMachinePECount, cpuMashinePERating, gpuMashinePECount, gpuMashinePERating;
- connection bandwidth between resources and tasks distributors: resourceBaudRate, linkBaudRate;
- the cost of using Grid resources: resourceCostPerSec;

- the cost of memory work operations: loading and saving data: loadOperationCost, saveOperationCost.

These parameters depend on the internal Grid system structure and an appropriate model, and for further investigation of the model it is required to find their exact values using the constants optimization module.

To check the adequacy of the established model an experiment was carried out as described in [14] on a real parallel system consisting of a CPU Intel Core i7 3770k (4 Core, HT, 3.5 GHz, Smart Cache 8MB) and GPU GeForce 650 GTX (384 CUDA Cores, 1058 MHz, 86,4 GFLOPS FP64, 1024 MB).

The result of the experiment is the dependence of matrix multiplication runtime using blocked algorithm (taking in account data transfer) on the size of the matrix. Due to restrictions imposed by the Windows operating system on a GPU working time the results were obtained for a range of matrix sizes [16, 3760] with the increment of 16.

To adjust the preset experiment generator parameters the constants optimization module was used. To check model adequacy and accuracy of selection parameters an optimization module has been configured to work in a range of sizes of matrices [512, 1552] with the increment of 80.

First the preset parameters values in the first approximation were obtained. The value of each parameter was changed by one order and the value of other parameters remained unchanged. During the experiment it was found that the parameters cpuMachinePECount, cpuMachinePERating, resourceCostPerSec do little influence or no impact on the simulation at all, due to the fact that all the calculations are carried out on the GPU, and as a Grid resource is alone, then no matter what is its cost it is impossible to assign the task to another resource. It should also be mentioned that the resourceBaudRate and linkBaudRate parameters do not influence significantly on the simulation for values up to 1.0e-05. Data parameters values greater than 1.0e-05 have no effect on the simulation because of the small amount of data transferred between the elements of the Grid.

The experiment described above was repeated for the loadOperationCost and saveOperationCost parameters with more detailed range: [1e-04; 1e-03] and increment value 1e-04 as they have the greatest impact on the simulation. Then using the method of exhaustive search all the possible combinations of parameters with step 1e-06 (about 1000) near optimal values, in the first approach, more accurate values of preset parameters were obtained: loadOperationCost = 1.8e-4; saveOperationCost = 1.936e-3. These values correspond to the characteristics of real parallel system on which the calculations were made.

Fig. 2 shows graphs of real experiment execution time (dashed line) and simulation (solid line) and the value of relative error of simulation for a range of matrix size [512, 3760] with increment of 16 (for practically all relevant data that were obtained through implementation on a real resources).

As shown in the graphs the relative error is stabilized at large scale matrix (2000-3000 of elements) and takes the value of 3-4% that is acceptable for such systems.

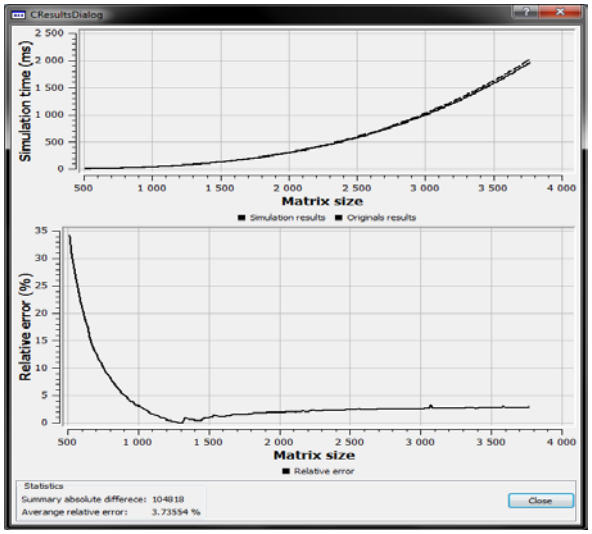

Fig. 2. Comparison of the real experiment execution time and the simulation for a range of matrix size [512, 3760] with increment of 16

During the model testing there were found two features. For the matrices of small sizes the difference between the results of simulation and real experiment is much larger than for the large ones. This is explained by the GridSim framework limitation: simulations have a duration threshold time not less than 2 ms. In addition it is difficult to measure the performance on GPU accurately for the matrix of a small size. Also for small sizes additional options that are not

described in the model begins to affect on the execution time. Therefore the models described, for example, in [5] are not suitable for small size matrices. It should be noted that the use of GPU for multiplication of matrices of small size is unjustified because of significant overhead for data transfer and initialization. Therefore for practically important cases this difference is not observed.

The second feature is the sharp increase of simulation time for certain values of the matrix size in the absence of a similar increase in the experimental data. This feature can be explained by an increase in overhead costs associated with the use of additional processing elements (computational kernel) of Grid resource.

4 N-body problem

We consider the problem of classical Newtonian mechanics, the evolution of a system of N particles with known masses in three-dimensional Euclidean space that interact in pairs under the influence of gravitational forces [16]. At the initial time the particles spatial coordinates and their velocities are given. The purpose is to predict the state of the system at the subsequent moments of time. The behavior of the system is described by the Cauchy problem of 2N first order differential equations. The N-body problem is not solvable analytically in a general case, thus an approximate integration numerical method of predictor-corrector type with time discretization using Hermite interpolation polynomials [17] was applied.

The parallel algorithm implementing this task involving GPU was constructed. Since the dependence of execution time on the size of the problem (i.e. the number of particles) is quadratic the current interest is the optimization of the parallel algorithm for large problems, by selecting the execution parameters in particular. One of the parameters that significantly influence the GPU execution time is a partition of the problem by threads and blocks of threads. In particular, the parameter that the developer can control is the number of threads per block (threads per block, TPB). According to the user manual CUDA v.5.0 the minimum and the maximum values of this parameter are 1 and 1024 respectively. It is recommended to use 256 threads per block in applications and set it as a static value in the program code [11]. However there exists a possibility of optimal choice of this parameter which increases the program effectiveness.

The model of the execution time of N-body problem was built depending on two parameters: the number of particles N and the number of threads per block TPB. Fig. 3 shows the comparison of the results of the model and experimental data. Using the constants optimization module the accurate values for the generator were found that achieve the smallest average relative error of 43%. As can be seen from the graphs the maximum relative error patterns were observed at small values of N and TPB. This feature can be explained by the fact that the model neglects the influence of the parameters of the algorithm and hardware-software environment on the execution time of the algorithm at small sizes of the input data. In this situation the influence of this effect is comparable to the amount of computations performed by the algorithm. However the greatest interest is of the large size of the input data.

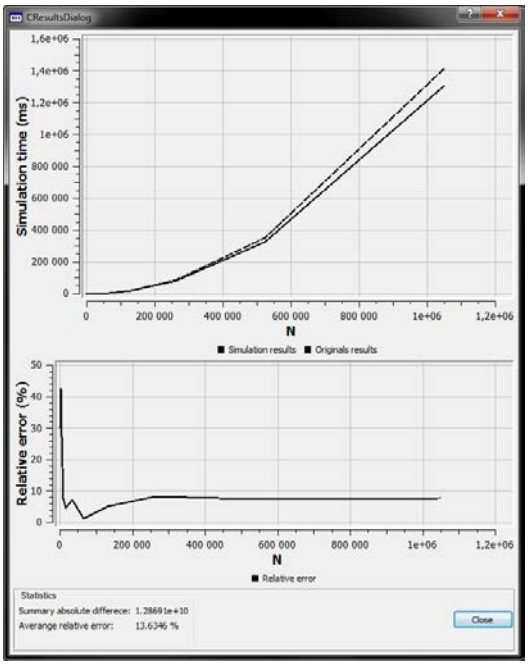

Fig. 3. Comparison of simulation results with experimental data of N-body problem for TPB = 256, N = [2^10, 2^20]

It should be mentioned that the main task of the model is to obtain the answer to the question "What number of threads per block should be used in the parallel algorithm to compute the gravitational interaction of the N-body problem?". In particular, even if the model does not accurately reproduce the execution time for individual values of N and TPB but can answer the posed question correctly it already is useful for selecting optimal parameter values.

The implemented model retains the character of the dependence of executing time on the TPB at constant N: for large N minimum execution time is achieved with larger TPB. In other words, the extremum points of the model and the experimental data do match. In addition, the model execution requires sufficiently less time than a real system: for N = 2^21 and TPB = [32, 1024] simulation lasts 2.5 minutes, the experiment on real system with such parameters takes 8 hours. The Fig. 4 shows the dependence of the runtime on the TPB parameter.

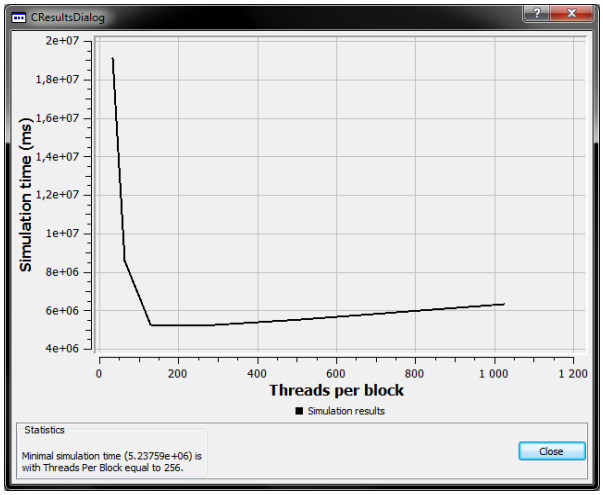

Fig. 4. The dependence of the execution time on TPB [32; 1024] for N = 2^21

Thus, the proposed model despite the presence of high relative error of 43% in a short time (3 minutes for N = 2^21, TPB = [32, 1024]) gives the correct value of the optimal parameter TPB and can be used in further studies.

5 Conclusions

In this paper the architecture of flexible, extensible and easy in use simulation environment of heterogeneous Grid systems based on GridSim simulation framework is proposed. The environment prototype based on the proposed architecture was implemented. The experimental verification of the developed prototype by comparing actual measurements and model execution time for matrix multiplication and N-body gravitational interaction problems was provided. The configuration model for the matrix multiplication problem was found by constants optimization module with a 3-4% simulation results deviation comparing to the results of the experiments on the real parallel CPU-GPU system. For the N-body problem the received error is sufficiently greater and reaches 43%, but it succeeded to determine the optimal value of the parameter. Simulation also allowed to significantly speed up the process of obtaining the optimal parameters values: for example, for N=2^21 the optimal value of TPB = 256 was obtained in 3 minutes, compared to 8 hours required for the experimental verification of this value.

Literature

[1] *Yongwei W., Likun L., Jiayin M., et al.* An analytical model for performance evaluation in a computational grid // Proceedings of the 2007 Asian technology information program's (ATIP's) 3rd workshop on High performance computing in China: solution approaches to impediments for high performance computing (CHINA HPC '07). – 2007. – P. 145–151.

[2] *Javadi B., Abawajy J.H., Akbari M.K.,et al.* Analytical Network Modeling of Heterogeneous Large-Scale Cluster Systems // IEEE International Conference on Cluster Computing. Barcelona, September 25–28, 2006. – P. 1–9.

[3] *Nadeem M.F., Ahmadi M., Nadeem M., et al.* Modeling and Simulation of Reconfigurable Processors in Grid Networks // Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs (RECONFIG '10). – 2007. – P. 226–231.

[4] *Shelestov A.Yu.* Approaches and simulation means of GRID-systems for satellite data processing // Problems in Programming. – 2008. – № 2–3. – P. 713–720 (in Russian).

[5] *Minukhin S.V., Znakhur S.V.* Investigation of efficiency of methods for planning resources for task flow of various intensity in Grid // Radio-electronic and computer systems. – 2012. – № 1 (53). – P. 165–171. (in Russian)

[6] *Petrenko A.I.* Computer modeling of Grid systems // Electronics and communication. – 2010. – № 5. – P. 40–48. (in Ukrainian).

[7] *Koren'kov V.V., Nechayevskii A.V.* Simulation packages for DATAGRID // System analysis in science and education. – 2009. – № 1. – P. 1–15. (in Russian).

[8] *Sulistio A., Cibej U., Venugopal S., et al.* A toolkit for modelling and simulating data Grids: an extension to GridSim // Concurrency and Computation: Practice & Experience. – 2008. – Vol. 20, N 13. – P. 1591–1609.

[9] *GridSim*: A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing http://www.buyya.com/gridsim/.

[10] *Baghsorkhi S.S., Delahaye M., Patel S.J., et al.* An adaptive performance modeling tool for GPU architectures // SIGPLAN Not. – 2010. – Vol. 45, N 5. – P. 105–114.

[11] *Hong S., Kim H.* An integrated GPU power and performance model // SIGARCH Comput. Archit. News. – 2010. – Vol. 38, N 3. – P. 280–289.

[12] *Kerr A., Diamos G., Yalamanchili S.* Modeling GPU-CPU workloads and systems // Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units. – 2010.– P. 31–42.

[13] *Kerr A., Diamos G., Yalamanchili S.* A characterization and analysis of PTX kernels // IEEE International Symposium on Workload Characterization (IISWC 2009). – 2009.– P. 3–12.

[14] *Zhereb K.A., Ihnatenko O.P.* Modeling matrix multiplication problem for GPU // Proc. Conf. "High Performance Computing " (HPC-UA 2012). Kyiv, 08–10 Oct. 2012. – P. 174–181. (in Ukrainian)

[15] *Qt* Developer Network http://qt-project.org/

[16] *Sverre J. Aarseth.* Gravitational N-body simulations. – Cambridge University Press, 2003. – 413 p.

[17] *Makino J., Aarseth S.J.* On a Hermite integrator with Ahmad-Cohen // Publications of the Astronomical Society of Japan – 1992. – Vol. 44, N 2. – P. 141–151.

[18] *A.Yu. Doroshenko, I.V. Okonsky, K.A. Zhereb, O.G. Beketov.* Using means of simulation for determining optimal parameters for running programs on GPU // Problems in Programming. – 2013. – № 2. – P. 23–31. (in Ukrainian).